Statistical Analysis Techniques for Logic and Memory Circuits

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Qunzeng Liu

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

July, 2010

Acknowledgements

First of all, my academic advisor Prof. Sachin S. Sapatnekar, deserves my ultimate gratitude toward finishing this thesis. His superb vision and broad range of knowledge have always amazed me. His direction is clear, his guidance enlightening and he encourages me to believe in myself and continue when I meet with difficulties. More importantly, I learned from him the problem solving skills, the absolute attention to details and the attitude to make every effort to do high quality work. It is my great honor to be his student and to do research with him. Every discussion with him has been fruitful and rewarding. I will always look up to him and try to follow his steps.

I would also like to thank Prof. Chris H. Kim for sharing his knowledge about circuit design and memory operations. He is always modest yet super smart, and his suggestions are practical and down to earth.

Thanks are also due to other professors on my committee, Prof. Bazargan, Prof. Riedel and Prof. Zhai. Thank you for reading through the thesis and providing valuable suggestions.

Special thanks to Dr. Singhee from IBM and Prof. Rutenbar from Carnegie Mellon University for providing their source code of the statistical blockade work so that we can compare our work with theirs.

Dedication

To my parents, my sister, and my lovely niece.

Abstract

Process variations have become increasingly important as feature sizes enter the sub-100nm regime and continue to shrink. Both logic and memory circuits have seen their performance impacted due to these variations. It is increasingly difficult to ensure that the circuit manufactured is in accordance with the expectation of designers through simulation. For logic circuits, general statistical static timing analysis (SSTA) techniques have emerged to calculate the probability density function (PDF) of the circuit delay. However, in many situations post-silicon tuning is needed to further improve the yield. For memory circuits, embedded DRAM (eDRAM) is beginning to replace SRAM as the on-die cache choice in order to keep the scaling trend. Although techniques exist for statistical analysis of SRAM, detailed analysis of eDRAM has not been developed prior to this thesis.

In this thesis, we provide techniques to aid statistical analysis for both logic and memory circuits. Our contribution in the logic circuits area is to provide robust and reliable, yet efficient post-silicon statistical delay prediction techniques for estimating the circuit delay, to replace the traditional critical path replica method that can generate large errors due to process variations during the manufacturing process. We solve this problem from both the analysis perspective and the synthesis perspective. For the analysis problem, we assume that we are given a set of test structures built on chip, and try to get the delay information of the original circuit through measurement of these test structures. For the synthesis problem, we automatically build a representative critical path which maximally correlate with the original circuit delay. Both of these approaches are derived using variation aware formula and use SSTA as sub-steps. They capture the delay variation of the original circuit better than the traditional critical path replica approach and eliminates the need to perform full chip testing for the post-silicon tuning purpose.

In response to the growing interest in using eDRAM-based memories as on-die cache, in the memory analysis area we provide the first statistical analysis approach for the cell voltage of eDRAM. We not only calculate the main body of the PDF for the cell

voltage, but also specifically look at the tail of this PDF which is more important to ensure quality design due to the highly repetitive nature of the memory systems.

We demonstrate the accuracy and efficiency of our methods by comparing them with Monte Carlo simulations.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Feature sizes in VLSI design have been shrinking for several decades, and are currently in the tens of nanometers. In this regime, variations in the process and operating conditions play a critical role in determining circuit performance, and must be taken into consideration during the design process in order to ensure that a circuit meets its specifications over its entire lifetime. These variations can arise due to process shifts, environmental effects, and circuit aging.

Examples of parameters affected by process variations include the gate length, gate width, oxide thickness, and the dopant concentration. In general, these process variations can be classified as inter-die variations and intra-die variations. Inter-die variations are fluctuations in process parameters from chip to chip, while intra-die variations are the variations among different elements within a single die. Some, but not all, intra-die variations may show the property of spatial correlation, which implies that the process parameters associated with transistors or wires that are close to each other are more likely to vary in a similar way than those of transistors or wires that are far away from each other.

Environmental variations are caused by effects such as changes in the on-chip temperature and supply voltage variations, while aging variations may be attributed to effects such as negative bias temperature instability (NBTI), time-dependent dielectric breakdown (TDDB) in the gate oxide, and hot carrier injection (HCI) in transistors, and electromigration in wires. Typically, environmental and aging variations are worst-cased, while process variations are handled statistically. The rationale is that process

variations are baked into a circuit after it is manufactured: if the manufactured part fails to meet its specifications, it can be discarded: the economic impact can be determined statistically by the manufacturing yield. However, environmental and aging effects are time-dependent and kick in during the life of the die, and therefore must be worst-cased.

Figure 1.1 shows the experiment result done by Intel to illustrate the impacts of the oxide thickness variations on processor performances such as normalized leakage and normalized power. The experiment use a 10% variation in oxide thickness and $100nm$ BPTM technology. It is shown that even at this older technology node, the current can have a 15× difference. Currently chip designers are using feature sizes as small as $28nm$ or even $22nm$, which makes process variations a more prominent issue and the correlation between pre-silicon and post-silicon performance evaluations becomes even weaker.

These variations affect all aspects of VLSI design including logic and memory circuits. In this thesis, we contribute to the variation aware design tools community by providing post-silicon statistical delay analysis techniques for logic circuits, as well as statistical analysis of cell voltage for memory circuits.

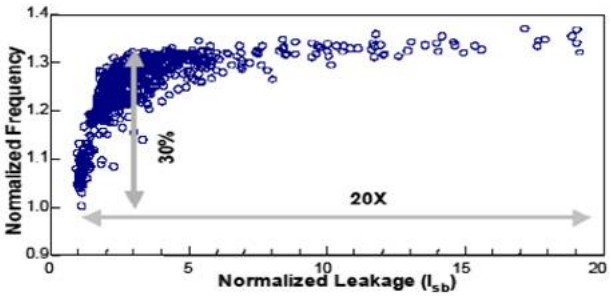


Figure 1.1: Impact of process variations, courtesy [1]

1.1 Variations in Logic Circuits

1.1.1 Pre-Silicon Analysis Under Process Variations

The process variations introduced above pose great challenges to analyzing the timing behavior of a logic circuit because traditional static timing analysis (STA) for one set

of process parameters is obvious not good enough as the process parameters change during the fabrication. One way to tackle this is to use corner-based static timing analysis (STA). However, the number of combinations for different corners for each parameter grows exponentially with increasing the number of process parameters. To exhaust every combination is clearly not an option. On the other hand, if we choose to only focus on the worst corner of each parameter, the arrival time calculation may be overly pessimistic [2]. Besides, the process variations complicates both the arrival time and the setup time and hold time constraints.

To overcome this problem, the concept of design for variability [3] was proposed. In the area of timing, statistical static timing analysis (SSTA) has been proposed as an alternative timing analysis engine to STA. Instead of trying to obtain different arrival numbers for different sets of process parameters, as is done in the corner-based SSTA method, SSTA tries to get a full probability density function (PDF) for the worst case delay of the arrival time of a combinational block. From the PDF of the worst case arrival time, designers can set their own cut off threshold according to the yield requirement. For examples, the 99.9% point of the PDF can be used for timing optimization. The problems associated with this technique include, but are not limited to, statistical delay characterizations for each gate and wire, the propagation of statistical delay information from primary inputs (or launching flip-flops) to primary outputs (or capturing flip-flops, as well as affected clock distribution network and timing constraints.

Existing SSTA engines generally fall into two categories: path-based [4, 5, 6] and block-based [7, 8, 9]. Path-based methods have the advantage that they do not need to perform the max operation until reaching the primary outputs (or the launching flip-flop). The false path problem can also be taken care of in this method. However, the number of paths grows exponentially with the circuit size. Block-based methods, on the other hand, inspired by the critical path method (CPM) widely used in STA [2], use a PERT-like traversal to propagate the distribution of the arrival times toward the primary outputs based on the order of a topological sorting, which brings down the time complexity to be linear with respect to the number of edges in the directional acyclic graph (DAG) representing the circuit, assuming that calculation of the distribution of the delay for each gate and wire are constant under appropriate characterizations.

Early work on SSTA using analytical methods and closed forms makes assumptions

that the process parameters are Gaussian distributed and the delay models are linear with respect to those parameter variations. More recent work deals with non-Gaussian parameter variations [10], non-linear delay models [11], and both [12, 13]. Most of the existing SSTA work deals with the arrival time of a combinational block. In addition, other research wolves problems such as interconnect modeling [14, 15], clock skew [16], latch modeling [17], pipelining and timing constraint concerns [18], and PDF evaluation techniques [19].

With the aid of SSTA tools, designers can optimize the circuit before it is fabricated, for example using the sizing technique proposed in [20], in the expectation that it will meet the timing requirements after the fabrication. In other words, SSTA is a presilicon analysis technique used to determine the range of performance (delay or power) variations over a large population of dies. As timing and power are always tradeoffs for a design, statistical power analysis is equally interesting and work abounds in the literature [21, 22, 23, 24] . While power issues are not explicitly addressed in this thesis, the approaches here can also be extended for statistical power measurement.

1.1.2 Post-Silicon Tuning Under Process Variations

Pre-silicon analysis and optimization can greatly improve yield. However, due to fluctuations that may not be taken into full account, the yield may still not be satisfactory after the circuit is manufactured. A complementary role, at this stage, is played by post-silicon diagnosis, which is typically directed toward determining the performance of an individual fabricated chip based on measurements on that specific chip. This procedure provides particular information that can be used to perform post-silicon optimizations to make a fabricated part meet its specifications. Because presilicon analysis has to be generally applicable to the entire population of manufactured chips, the statistical analysis that it provides shows a relatively large standard deviation for the delay. On the other hand, post-silicon procedures, which are tailored to individual chips, can be expected to provide more specific information. Since tester time is generally prohibitively expensive, it is necessary to derive the maximum possible information about the circuit delay for each chip manufactured through the fewest post-silicon measurements.

A use case scenario for post-silicon analysis in the realm of post-silicon tuning is adaptive body bias (ABB) [25, 26, 27]. ABB is a post-silicon method that determines

the appropriate level of body bias to be applied to a die to influence its performance characteristics. ABB is typically a coarse-grained optimization, both in terms of the granularity at which it can be applied (typically on a per-well basis) as well as in terms of the granularity of the voltage levels that may be applied (typically, the separation between ABB levels is 50 to 100 mV). Current ABB techniques use a critical path replica to predict the delay of the fabricated chip, and use this to feed a phase detector and a counter, whose output is then used to generate the requisite body bias value, as is shown by Figure 1.2 in [25].

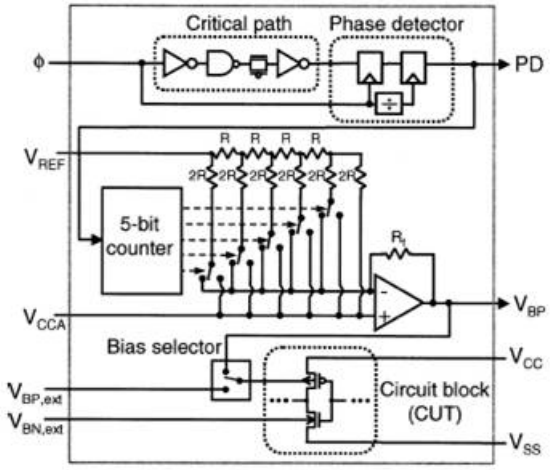


Figure 1.2: Implementation diagram of adaptive body bias (ABB), courtesy [25]

Another post-silicon optimization technique uses adaptive voltage scaling [28, 29]. In [28], a delay synthesizer, composed of three delay elements, is used to synthesize a critical path as part of a dynamic voltage and frequency management system. However, the control signals of the synthesizer is chosen arbitrarily and therefore it is not able to adapt to a changing critical path as a result of process variations. In [29], the authors compensate this problem using a pre-characterized look up table (LUT) to store logic speed and interconnect speed inside different process bins. A logic and interconnect speed monitor is then used as an input to select through the LUT control signals to program a critical path. The block diagram is drawn in Figure 1.3.

In the previous literature, the interaction between presilicon analysis and post-silicon measurements has been addressed in several ways. In [30], post-silicon measurements

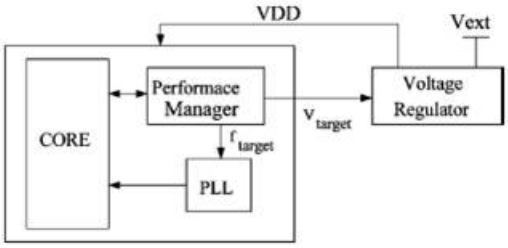


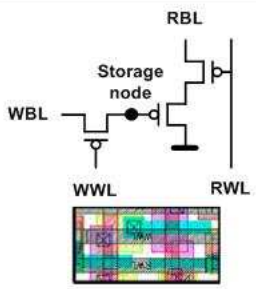
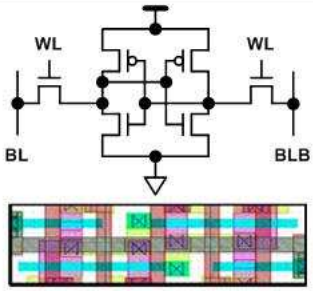
Figure 1.3: Implementation diagram of adaptive voltage scaling (AVS), courtesy [29]

are used to learn a more accurate spatial correlation model to refine the SSTA framework. A path-based methodology is proposed in [31] to correlate post-silicon test data to presilicon timing analysis. In [32], a statistical gate sizing approach is presented to optimize the binning yield. The work is extended to simultaneously consider the presence of post-silicon-tunable clock tree and statistical gate sizing in [33]. Post-silicon debug methods and their interaction with circuit design are discussed in [34]. A joint design-time and post-silicon tuning procedure is described in [35]. In [36], a critical path monitor is built to monitor the critical path of the circuit as well as measuring process variations. A path selection methodology is proposed in [37] to monitor unexpected post-silicon systematic timing effects. However, none of these approaches can be used to provide better post-silicon delay prediction results for ABB or AVS.

1.2 Variations in Memory Circuits

Memory performance is a critical bottleneck in the performance of high-performance designs, and improvements on this front are typically obtained through further scaling of the memory cells. Current microprocessors usually employ SRAMs as on-die cache memory because of their high speed, low power, and logic process compatibility. However, conventional SRAM arrays consist of six transistors (6T) for each cell, and face significant read and write stability problems as feature sizes continue to shrink. Moreover, in nanoscale technologies, process variations play a very significant role, and the mismatches between neighboring devices are inversely proportional to the square root of the area [38], driving the need for low-area solutions.

Embedded DRAM (eDRAM) cells present a promising alternative to replace conventional 6T SRAM cells for on-die caches. eDRAMs can use a standard CMOS process and can be built on the same die as the processor, as against conventional DRAMs. An eDRAM cell can be implemented with less than three transistors (3T), plus appropriate low-overhead control circuits in the read path. Apart from reduced area and "keep-alive" power, eDRAMs provide the advantage of wide read/write margins: a critical problem in scaled 6T SRAMs has been the fighting between read/write devices and the cross-coupled latch, but this is entirely avoided by eDRAMs.

	3T EDRAM	6T SRAM
*Cell schematic and layout		
Features	Small, logic compatible	Fast, logic compatible
Issues	Short retention time	Large size, low noise margin
**Cell size (ratio)	0.54x1.02= 0.551μm^2 (1.0X)	0.575x2.05= 1.178μm^2 (2.14X)
***RWL-BL delay (0.9V, Δ=100mV)	1.07ns	0.40ns
Retention time	110μsec (measured)	-
Static power	Only the refresh power	Large due to transistor leakage

* PMOS cells for low I_{gate}, ** 65nm logic design rule, *** Monte-carlo 3σ simulation results

Figure 1.4: Comparisons between a 6T SRAM cell and a 3T eDRAM cell.

Figure 1.4 shows a brief comparison of a 3T eDRAM cell with a 6T SRAM cell. The eDRAM structure we consider is based on a "gain cell" topology that uses standard CMOS structures, as against alternatives that use trench or MIM capacitor processes. A number of successful eDRAM test chips have been reported [39, 40, 41]. For example, IBM's powerPC chips will be using eDRAMs consisting of one transistor and one capacitor for L2 caches [42, 43]. Therefore using eDRAM as on-die cache is not only

desirable, but also practical.

However, unlike SRAM, eDRAM needs to be refreshed after a certain time due to the degradation of the cell voltage. Process variations further complicate the issue of leakage and data retention. Under these variations, the cell voltage of each eDRAM cell at a given time is not a specific number, but is at a certain value only with a certain probability. In other words, within an eDRAM array, some cells experience more serious signal loss than others.

While techniques exist for statistical analysis of SRAMs [44, 45, 46, 47, 48], there is currently no existing work for eDRAM specific statistical analysis such as the cell voltage analysis, which is one of the problems we address in this thesis.

1.3 Our Contributions

The critical path replica approach, used by current adaptive body bias techniques, assumes that one critical path on a chip is an adequate reflection of on-chip variations. In general, there will be multiple potential critical paths even within a single combinational block, and there will be a large number of combinational blocks in a within-die region. Choosing a single critical path as representative of all of these variations is impractical and inaccurate.

In the adaptive voltage scaling work, the authors use simplified circuitry for the speed monitor, consisting of only one logic dominated element and one interconnect dominated element, and assume that the results are generally applicable to all parts of the circuit. In the presence of significant within-die variations, this assumption becomes invalid. Moreover, the approach requires substantial memory components even for process bins of a very coarse resolution, and is not scalable to fine grids.

To address the variations problem in logic circuits, in this thesis, we will provide smarter methodologies to aid these post-silicon optimization techniques. In order for proper post-silicon timing optimization to be executed, we have to get an approximate value, or a small range of the circuit delay after it is fabricated. As we illustrated, the critical path replica is obviously not a good choice. Therefore we introduce two novel approaches. In the first approach, we first build a few ring oscillators on chip at different locations. After the circuit is manufactured, we measure the delays of these

ring oscillators, and use some statistical calculations to get the delay information of the original circuit. The result of this approach is a narrow, die-specific conditional PDF of the circuit delay. In the second approach, we provide methods to build a Representative Critical Path (RCP) on chip. After the circuit is fabricated, measurements of this RCP along with a simple calculation would give the delay of the fabricated circuit, despite the parameter variations having occurred during the manufacturing process.

To address the variations problem in memory circuits, the final part of the thesis is devoted to statistical analysis for the cell voltage of eDRAM. We first develop analytical models for all leakage components contributing to the cell voltage degradation, then we feed these into the APEX PDF evaluation technique [19] to get the PDF of the distribution up to the 99% point. Because the designers are usually more interested in the far tails of the distribution for memory systems, we borrow ideas from the extreme value theory (EVT) [49] to specifically fit the tail of the distribution to an exponential distribution.

The rest of the thesis is organized as follows: Chapter 2 briefly introduces the motivation and the development of statistical static timing analysis (SSTA). The SSTA framework used by this work is highlighted. Chapter 3 presents our proposed scalable technique of post-silicon delay analysis by making use of a few test structures on chip. Methods for synthesizing RCP are discussed in Chapter 4. Chapter 6 concludes the thesis.

Chapter 2

Statistical Static Timing Analysis

In this chapter, we will introduce the motivation and basic theory of statistical static timing analysis (SSTA). Specifically, we will focus on the parameterized block-based SSTA framework with Gaussian parameters and a linear delay model. This framework is used in later chapters, and makes an essential part of the thesis.

2.1 Introduction

For feature sizes in the tens of nanometers, it is widely accepted that design tools must take into account parameter variations during manufacturing. These considerations are important during both circuit analysis and optimization, and are essential to ensure adequate manufacturing yield. As is described in Chapter 1, parameter variations can be classified into two categories: intra-die variations and inter-die variations. Inter-die variations correspond to parameter fluctuations from one chip to another, while intra-die variations are defined as the variations among different locations within a single die. Intra-die variations of some parameters have been observed to be spatially correlated, i.e., the parameters of transistors or wires that are placed close to each other on a die are more likely to vary in a similar way than those of transistors or wires that are far away from each other. For example, among the process parameters for a transistor, the variations of channel length L and transistor width W are seen to have such spatial correlation structure, while parameter variations such as the dopant concentration N_A and the oxide thickness T_{ox} are generally considered not to be spatially correlated.

There are also environmental variations such as temperature and supply voltage.

Process parameter variations have resulted in significant challenges to the conventional corner-based timing analysis paradigm, and statistical static timing analysis (SSTA) has been proposed as an alternative [4,8,9,10,11,50,51,52,53,54,55,56,57,58]. The idea of SSTA is that instead of computing the delay of the circuit as a specific number, a probability density function (PDF) of the circuit delay is determined. Designers may use the full distribution, or the 3σ point of the PDF, to estimate and optimize timing.

There are various kinds of SSTA techniques in the literature. In this thesis, they are roughly categorized as follows.

- Path-based SSTA vs. block-based SSTA

  Path-based methods [5,6] try to find the PDF of the circuit delay on a path-by-path basis, and in the end perform a single "max" operation to find the PDF of the worst case delay of the circuit. Path-based methods are only applicable to circuits with a small number of paths. In modern VLSI circuits, the number of paths is exponential in the number of gates, and under process variations, any path can be potentially the most critical one after the circuit is manufactured. Therefore this approach is too time consuming in that case. However, path-based approaches can provide path specific diagnosis and it also makes considering false path much easier [4].

  Block-based methods [7,8,9] perform a PERT-like traversal based on topological sorting and process each gate only once during the propagation of the delay distribution. Therefore theoretically these methods are much faster. However, to get path specific information from block-based SSTA is more complex than for critical path method (CPM) based STA. Thesse methods also suffer from the limitations built into topological STA methods.

- SSTA using continuous PDFs vs. using discrete PDFs

  Using a continuous PDF sometimes provides the benefit that one can model the PDF as a closed-form function, and makes computation much easier [53,54,55,56]. However, these techniques usually have to make assumptions about the PDF of the delay which are not always realistic.

SSTA techniques that use discrete PDFs [4, 57, 58] don not have to make assumptions about the shape of the delay PDF, thus making them more general. However, the sum of two delay variables requires calculating the discrete convolution of the two PDFs, and the number of terms can increase exponentially after repeated convolution operations. In [7], a piecewise linear model of CDF is used to simplify the calculation.

- Gaussian vs. non-Gassuan parameter variations and linear vs. non-linear delay models for SSTA

  Early SSTA work [8, 54] assumes that the process parameter variations are Gaussian and the delay for each gate can be approximated by a first-order Taylor series expansion. While these assumptions are reasonable for many situations, they are known to suffer from two limitations. First, some process parameters are known to be non-Gaussian: for example, the via resistances exhibit an asymmetric PDF and the dopant concentration NA is better modeled as a Poisson distribution. Second, linear delay models are only accurate when the process variations are small. Various extensions of this model have been proposed to address these cases. Larger variations call for higher-order approximations of the delay model. Therefore work has been done to address a linear delay model with non-Gaussian parameter variations [10], a quadratic delay model with Gaussian parameter variations [11], and a non-linear delay model with non-Gaussian parameter variations [12, 13]. Most of these methods require computationally intensive numerical techniques and the time complexity is not as good as the Gaussian and linear case.

Several representative techniques for SSTA will be introduced in the following sections.

2.2 SSTA Using Bounding Methods

As is in STA, SSTA views the circuit as a directed acyclic (DAG) timing graph. The difference is now we are propagating PDFs instead of delay values. The PDFs for series-connected edges with a single fanin can be processed by a convolution of the PDFs of the individual edges, and for parallel-connected edges, the CDF of their maximum may

be computed by taking the product of the CDFs of the incoming edges. However, reconvergent subgraphs do not belong to either case. The work of [59,60] uses bounding methods and selective enumeration to solve the problem. An upper bound of the timing CDF for a reconvergent subgraph is provided by ignoring the structural correlation, and a lower bound of the CDFs for two dependent arrival times is found by the envelope of their CDFs using min operator on the original graph. Heuristic methods are developed to determine whether selective enumeration is needed.

2.3 Parameterized Block-Based SSTA Considering Spatial Correlations

The bounding methods ignore spatial correlations of the process variations. Spatial correlation means parameters of devices and wires that are close to each other are more likely to vary the same way than devices and wires that are far away on the same chip. Figure 2.1 shows an example die. The process parameters of a and c are more likely to vary the same way than process parameters of a and d due to spatial correlations.

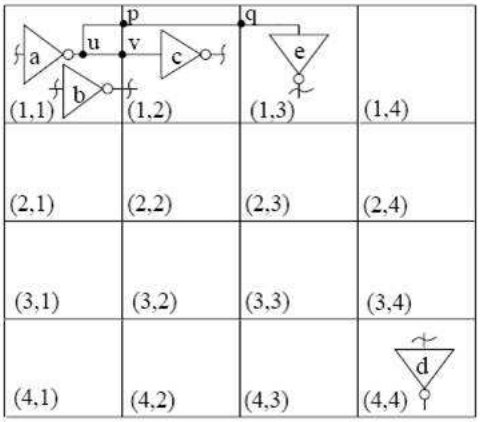


Figure 2.1: Spatial correlations, courtesy [8]

Experiments show that ignoring spatial correlations would generate large errors for the final PDF of the circuit delay [8]. In regard to this, efficient statistical timing analysis tools have been developed. A grid-based spatial correlation model is widely

used for SSTA techniques considering spatial correlations. As is shown in Figure 2.1, in this model, perfect correlations are assumed among devices in the same grid, such as a and b. High correlations are assigned among devices in close grids such as a and c. For devices in far-away grids such as a and d, very low or zero correlations can be assigned. It is also assumed that there is no correlation among different kind of parameter variations. A covariance matrix of size $n \times n$ can be generated for each spatially correlated parameter variation, with n being the number of grids used in the model. It is also noted that not all parameter variations are correlated.

Such models and appropriate assumptions enable parameterized block-based SSTA [8,9] is made possible to take into consideration both spatial and structural correlations. As is proposed in [8], Gaussian-distributed correlated variations can be orthogonalized using a technique called principal component analysis (PCA), which takes the covariance matrix of the parameter variations as input, and generates the coefficients for a set of independent principal components (PCs) for each parameter variation. In other words, each parameter variation can be represented by a linear combination of the same set of independent principal components. Under the assumption that all parameter variations are Gaussian-distributed, and delay models for each circuit element can be approximated by a first order Taylor serious expansion, a canonical form of the delay variable can be generated for each gate and wire. This canonical form, as shown in Equation (2.1), for each delay variable d, includes the nominal value μ, and a set of independent PCs p_i with their corresponding coefficients a_i generated by PCA. Uncorrelated variations are captured by a single independent random variable R [61].

$$d = \mu + \sum_{i=1}^{m} a_i p_i + R = \mu + \mathbf{a}^T \mathbf{p} + R. \tag{2.1}$$

The random variable p_i corresponds to the ith principal component, and is normally distributed, with zero mean and unit variance; principal components p_i and p_j for $i \neq j$ are uncorrelated by definition, stemming from the property of PCA. The parameter a_i is the first order coefficient of the delay variable with respect to p_i. For simplicity, we refer to $\mathbf{p} = \begin{bmatrix} p_1 & p_2 & \cdots & p_m \end{bmatrix}^T \in \mathbf{R}^m$ as the *PC vector* and $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix}^T \in \mathbf{R}^m$ as the *coefficient vector* for the circuit.

Equation (2.1) is general enough to incorporate both inter-die and intra-die variations. For a spatially correlated parameter, the inter-die variation can be taken into account by adding a value σ^2_{inter}, the variance of inter-die parameter variation, to all entries of the covariance matrix of the intra-die variations of that parameter before performing PCA. The uncorrelated component R accounts for contributions from both the inter-die and intra-die variations of those independent parameters.

The SSTA technique then propagates this canonical form from the primary inputs (or outputs of the launching registers) to the primary outputs (or inputs of the capturing registers) of the circuit using a PERT-like traversal, in the same fashion of traditional STA. During the propagation, the *sum* operation is straightforward because the sum of two Gaussian random variables are still Gaussian. The *max* operation, on the other hand, needs some approximations to maintain the canonical form because the maximum of two Gaussian variables is not strictly Gaussian. Therefore techniques in [62] are borrowed to solve the problem. In the end, the delay of whole circuit is also of the canonical form, and its PDF can be easily obtained. The limitation of this approach is that structural correlations of spatially uncorrelated parameters are not considered. However, this approach successfully incorporates both inter-die and intra-die variations, as well as both spatial and structural correlations, which is a big step forward. In the remaining chapters of this thesis, this parameterized block-based SSTA technique is heavily used to develop our techniques.

2.4 Other Techniques Related to SSTA

In this section some other techniques of interest related to the SSTA areas are discussed. They complement with the SSTA technique introduced in Section 2.3 well and completes the SSTA framework.

2.4.1 Incremental SSTA

Timing analysis usually serves as a step in the inner loop of physical synthesis and can be called millions of times in response to changes made in the design. To make this practical, incremental timing analysis algorithms are essential. The work in [9] uses a similar SSTA framework as in Section 2.3, but provides many interesting points,

including making the algorithm incremental by taking advantage of both *level-limiting* and *dominance-limiting* properties.

2.4.2 SSTA for latch-based design

Latch-based designs are popular for high performance circuits because of the high performance, low power and area savings of latches [63] as compared to edge-triggered flip-flops. For latch-based designs, a signal is allowed to have a delay larger than the clock period without incurring incorrect data propagation due to the time borrowing (also called cycle stealing) property of latch. Under process variations, the advantage of latch-based design is even more pronounced [64]. However, this poses challenges for timing analysis because they can no longer be carried out only on the separated combinational blocks due to the fact that the delay in one pipeline stage depends on the delays in the previous pipeline stage. Most existing SSTA techniques such as the one in Section 2.3 deals with only the arrival time for combinational blocks. While the extension to edge-triggered sequential designs are straightforward, for latch-based designs it is less obvious. In lieu to this, the work in [18] proposes an SSTA technique to evaluate the probability that a given latch-based pipeline design violates the timing constraints under process variations. The work in [65] takes a step further by not only considering data delay variations but also clock skew variations. A new latch model specifically tailored for SSTA is introduced in [17].

2.4.3 The APEX PDF evaluation technique

APEX [19] is not strictly an SSTA technique. However, it provides a useful tool to evaluate the delay PDF if the moments the delay random variable are known or can be easily computed, and proves very useful in our statistical analysis of eDRAM work presented in Chapter 5. The basic idea of the APEX approach is to approximate the PDF of a delay random variable by an impulse response of order M.

$$h(t) = \begin{cases} \Sigma_{i=1}^{M} r_i e^{q_i t} & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{2.2}$$

The residues r_i and the poles q_i can be determined by matching the *time moments* of Equation (2.2) and the moments of delay d. The kth time moment is defined as the

kth statistical moment multiplied by $\frac{(-1)^k}{k!}$. Assume $q_i < 0$, the kth time moment of Equation (2.2) can be derived as a closed form of r_i, q_i.

$$m_{t,k} = \frac{(-1)^k}{k!} \int_{-\infty}^{\infty} t^k h(t) dt = -\Sigma_{i=1}^{M} \frac{r_i}{q_i^{k+1}}. \tag{2.3}$$

On the other hand, if the delay random variable is of the quadratic form

$$d = d_0 + \mathbf{B}^T \mathbf{\Delta Y} + \mathbf{\Delta Y}^T \mathbf{A} \mathbf{\Delta Y} \tag{2.4}$$

where $d_0 \in R$ is constant, $\mathbf{B} \in R^M$ contains the first order coefficients and $\mathbf{A} \in R^{N \times N}$ contains the second order coefficients, then the time moments of the delay variable can be calculated, using the binomial moment evaluation method in [19], as m_i for the ith time moment. Random variables contained in $\mathbf{\Delta Y}$ are assumed to be Gaussian and independent. As is discussed in Section 2.3, correlated Gaussian random variables can be orthogonalized using PCA.

There are $2M$ unknowns in Equation (2.2), by matching the first $2M$ of these two set of moments, the following system of nonlinear equations are obtained.

$$\begin{aligned} -\left(\frac{r_1}{q_1} + \frac{r_2}{q_2} + \cdots + \frac{r_M}{q_M}\right) &= m_0 = 1 \\ -\left(\frac{r_1}{q_1^2} + \frac{r_2}{q_2^2} + \cdots + \frac{r_M}{q_M^2}\right) &= m_1 \\ &\vdots \\ -\left(\frac{r_1}{q_1^{2M}} + \frac{r_2}{q_2^{2M}} + \cdots + \frac{r_M}{q_M^{2M}}\right) &= m_{2M-1} \end{aligned} \tag{2.5}$$

These equations can be solved using the method provided in [66]. After we get r_i and q_i, we can get the approximate PDF of d, $h(t)$. Proper shifting of the PDF before the calculation should be taken care of because of limitations of the impulse response form.

The corresponding CDF can be approximated by the step response

$$s(t) = \begin{cases} \Sigma_{i=1}^{M} \frac{r_i}{q_i} \left(e^{q_i t} - 1\right) & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{2.6}$$

It is noted that the impulse response of Equation (2.2) is only nonzero for $t \geq 0$, and is close to the origin. However, in practice the PDF can be nonzero for $t < 0$, or the main body of the PDF can be very far from the origin. In such cases, appropriate

shifting [19] of the PDF before evaluation is needed. On the other hand, if the best case instead of the worst case is desired for the PDF, a reverse evaluation scheme should be applied [19].

We would like to point out that all techniques introduced in this section can be implemented on top of the parameterized block-based SSTA technique in Section 2.3 to form a more complete SSTA framework. For example, it can be modified to be incremental, can serve as the arrival time calculation part in a latch-based design, and in [10], the authors employed the APEX method, used independent component analysis (ICA) to replace PCA as the pre-processing step because ICA can handle non-Gaussian random variables, and extended the SSTA technique to take into consideration non-Gaussian parameter variations.

The SSTA framework in Section 2.3 is extensively used in Chapter 3 and Chapter 4, and the APEX technique is employed in Chapter 5.

2.5 Taking Advantage of Spatial Correlations

Spatial correlations of parameter variations have been considered as a challenge in pre-silicon techniques such as SSTA until the arrival of parameterized methods. SSTA methods ignoring spatial correlations usually results in a PDF with smaller variance than the realistic case, as is shown in [8], because ignoring spatial correlations can result in unrealistic cancellations among different random variables. However, from the perspective of post-silicon analysis, which we will discuss in detail in the remainder of this thesis, the presence of spatial correlations can be exploited to generate delay information of a particular chip based on some kind of test structure built on it, whether it be a set of ring oscillators (RO) discussed in Chapter 3, or a representative critical path (RCP) discussed in Chapter 4. More specifically, because of spatial correlations, the parameter variations for the test structure on a chip are correlated with those of the gates near them. For the specific case where only inter-die variations are seen, and no intra-die variations exist, or in other words, the parameter variations are fully correlated spatially, then the parameter variations of a test structure anywhere on a chip are identical to those of the original circuit to be tested. The presence of intra-die variations creates some challenges: the parameter variations of the test structure may

now be correlated with, but not identical to, those in the original circuit. In such a case, a test structures may not reveal the characteristics of the whole original chip, but it can reveal some characteristics for the devices nearby. In order to get desired information of the delay of the original circuit, we either have to use a number of test structures as in Chapter 3, or we can create a dedicated critical path on our own as in Chapter 4.

Chapter 3

Post-Silicon Statistical Delay Analysis

While SSTA has an important role to play in the process of circuit analysis and optimization, it is equally important to develop die-specific delay prediction techniques using post-silicon measurements. In this chapter, we present a novel method for post-silicon delay analysis. We gather data from a small number of on-chip test structures, and combine this information with presilicon statistical timing analysis to obtain narrow, die-specific, timing probability density function (PDF). Experimental results show that for the benchmark suite being considered, taking all parameter variations into consideration, our approach can obtain a PDF whose standard deviation is 79.0% smaller, on average, than the statistical timing analysis result. The accuracy of the method defined by our metric is 99.6% compared to Monte Carlo simulation. The approach is scalable to smaller test structure overheads and can still produce acceptable results.

3.1 Introduction

As is discussed in Chapter 1, SSTA is a presilicon analysis technique used to determine the range of performance (delay or power) variations over a large population of dies. Because it has to be generally applicable to the entire population of manufactured chips, its result will show a relatively large standard deviation for the delay. On the other hand, post-silicon diagnosis, directed toward determining the performance of an

individual fabricated chip based on measurements on that chip, can be expected to provide more specific information. However, full chip testing is not an option to do this since tester time is generally prohibitively expensive. Efficiency in post-silicon measurement demands the need to get maximum information based on as few measurements as possible.

In previous literature, the interaction between presilicon analysis and post-silicon measurements has been addressed in several ways. In [30], post-silicon measurements are used to learn a more accurate spatial correlation model, which is fed back to the analysis stage to refine the statistical timing analysis framework. In [31], a path-based methodology is used for correlating post-silicon test data to presilicon timing analysis. In [32], a statistical gate sizing approach is studied to optimize the binning yield. Post-silicon debug methods and their interaction with circuit design are discussed in [34].

The method that we present in this chapter differs from these in terms of its goals. Our approach forms a framework for post-silicon statistical delay prediction: the role of this step is seated between presilicon SSTA and post-silicon full chip testing. We combine the results of presilicon SSTA for the circuit with the result of a small number of post-silicon measurements on an individual manufactured die to estimate the delay of that particular die.

Given the *original circuit* whose delay is to be estimated, the primary idea is to determine information from specific on-chip *test structures* to narrow the range of the performance distribution substantially; for purposes of illustration, we will consider delay to be the performance metric in this work. In particular, we gather information from a small set of test structures such as ring oscillators (ROs), distributed over the area of the chip, to capture the variations of spatially correlated parameters over the die. The physical sizes of the test structures are small enough that it is safe to assume that they can be incorporated into the circuit using reserved space that may be left for buffer insertion, decap insertion, etc. without significantly perturbing the layout. To illustrate the main idea, we show a die in Figure 3.1, whose area is gridded[1] into spatial correlation regions. Figure 3.1(a) and 3.1(b) show two cases where test structures are inserted on the die: the two differ only in the number and the locations of these test

[1] For simplicity, we will assume in this example that the spatial correlation regions for all parameters are the same, although the idea is valid, albeit with an uglier picture, if this is not the case.

structures. Figure 3.2 shows a sample test structure consisting of a 3-stage RO; however, in practice, the number of stages in this structure may be larger, and these trade-offs are explored in Section 3.6. The data gathered from the test structures in Figures 3.1(a) and 3.1(b) are used in this chapter to determine a new PDF for the delay of the original circuit, conditioned on this data. This has significantly smaller variance than the result of SSTA, as is illustrated in Figure 3.3; detailed experimental results are available in Section 3.7.

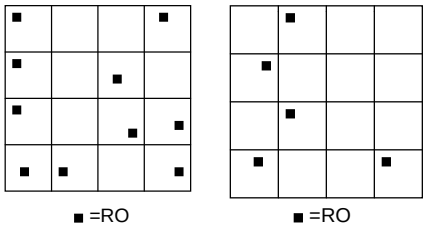


Figure 3.1: Two different placements of test structures under the grid spatial correlation model.

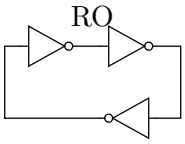


Figure 3.2: An example of a test structure: A three-stage ring oscillator.

The plots in Figure 3.3 may be interpreted as follows. When no test structures are used and no post-silicon measurements are performed, the PDF of the original circuit is the same as that computed by SSTA. When 5 ROs are used, a tighter spread is seen for the PDF, and the mean shifts towards the actual frequency for the die. This spread becomes tighter still when 10 ROs are used. In other words, as the number of test structures is increased, more information can be derived about variations on the die, and its delay PDF can be predicted with greater confidence: the standard deviation of the PDF from SSTA is always an upper bound on the standard deviation of this new delay PDF. In other words, by using more or fewer test structures, the approach is

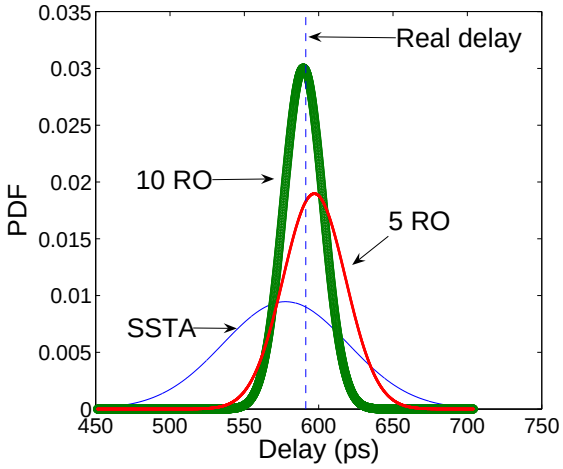


Figure 3.3: Reduced-variance PDFs, obtained from statistical delay prediction, using data gathered from the test structures in Figure 3.1.

scalable in terms of statistical confidence.

As is opposed to the critical path replica approach described in 1 which is generally not applicable when a large amount of intra-die variations are present, our approach implicitly considers the effects of all paths in a circuit (without enumerating them, of course), and provides a PDF that concretely takes spatially correlated and uncorrelated parameters into account to narrow the variance of the sample, and has no preconceived notions, prior to fabrication, as to which path will be critical. The 3σ or 6σ point of this PDF may be used to determine the correct body bias value that compensates for process variations. Temperature variations may be compensated for separately using temperature sensors, for example, as in [67].

The remainder of this chapter is organized as follows. Section 3.2 abstracts the physical problem into a mathematical formulation. Next, Sections 3.3 through 3.5 introduce our approach in detail and outline its limitations. Section 3.6 then discusses the impact of changing the number of stages in the RO test structures on the quality of the results. Experimental results are shown in Section 3.7, followed by concluding remarks in Section 3.8.

3.2 Problem Formulation

We assume that the circuit undergoes SSTA prior to manufacturing, and that the random variable that represents the maximum delay of the original circuit is d. Further, if the number of test structures placed on the chip is n, we define a *delay vector* $\mathbf{d}_t = \begin{bmatrix} d_{t,1} & d_{t,2} & \cdots & d_{t,n} \end{bmatrix}^T$ for the test structures, where $d_{t,i}$ is the random variable (over all manufactured chips) corresponding to the delay of the i^{th} test structure.

For a particular fabricated die, the delay of the original circuit and the test structures correspond, respectively, to one sample of the underlying process parameters, which results in a specific value of d and of $\mathbf{d}_t$. After manufacturing, measurements are performed on the test structures to determine the sample of $\mathbf{d}_t$, which we call the *result vector* $\mathbf{d}_r = \begin{bmatrix} d_{r,1} & d_{r,2} & \cdots & d_{r,n} \end{bmatrix}^T$. This corresponds to a small set of measurements that can be performed rapidly. The objective of our work is to develop techniques that permit these measurements to be used to predict the corresponding sample of d on the same die. In other words, we define the problem of post-silicon statistical delay prediction as finding the conditional PDF given by $f(d|\mathbf{d}_t = \mathbf{d}_r)$.

In the ideal case, given enough test structures, we can estimate the delay of the original circuit with very little variance by measuring these test structures. However, practical constraints limit the overhead of the added test structures (such as area, power, and test time) so that the number of these structures cannot be arbitrarily large. Moreover, as stated in Section 2.5, our method is made possible by spatial correlations of parameter variations at different locations. However, the variations in some parameters, such as T_{ox} and N_A, are widely believed to show no spatial correlation structure at all. Test structures are inherently not capable of capturing any such variations in the original circuit (beyond the overall statistics that are available to the SSTA engine): these parameters can vary from one device to the next, and thus, variations in the test circuit are totally independent of any variations in the original circuit, but even under these limitations, any method that can narrow down the variational range of the original circuit through a few test measurements is of immense practical use.

We develop a method that robustly accounts for the aforementioned limitations by providing a conditional PDF of the delay of the original circuit with insufficient number of test structures and/or purely random variations. In the case when the original circuit

delay can actually be computed as a fixed value, the conditional PDF is an impulse function with mean equal to the delay of the original circuit and zero variance. The variance becomes larger with fewer test structures, and shows a graceful degradation in this regard. We include all of these in a single generalized framework and automatically take each case into consideration.

3.3 Statistical Delay Prediction

3.3.1 SSTA Revisited

We use the SSTA technique provided in Section 2.3. The m PCs affect the statistical distribution of both the original circuit and the test structures on the same chip, and the canonical form for the delay of the original circuit is rewritten as:

$$d = \mu + \sum_{i=1}^{m} a_i p_i + R = \mu + \mathbf{a}^T \mathbf{p} + R, \tag{3.1}$$

where all notations are the same as defined for Equation (2.1) in Section 2.3. Here we use them for the original circuit.

In a similar manner, the delay of the i^{th} of the n test structures can also be represented in the canonical form as:

$$d_{t,i} = \mu_{t,i} + \mathbf{a}_{t,i}^T \mathbf{p} + R_{t,i}. \tag{3.2}$$

The meanings of all variables are inherited from Equation (3.1).

We define $\boldsymbol{\mu}_t \in \mathbf{R}^n$ as the *mean vector*, $\mathbf{R}_t \in \mathbf{R}^n$ as the *independent parameter vector*, and $\mathbf{A}_t \in \mathbf{R}^{m \times n}$ as the *coefficient matrix* of the test structures, respectively, where

$$\begin{aligned} \boldsymbol{\mu}_t &= \begin{bmatrix} \mu_{t,1} & \mu_{t,2} & \cdots & \mu_{t,n} \end{bmatrix}^T \\ \mathbf{R}_t &= \begin{bmatrix} R_{t,1} & R_{t,2} & \cdots & R_{t,n} \end{bmatrix}^T \\ \mathbf{A}_t &= \begin{bmatrix} \mathbf{a}_{t,1} & \mathbf{a}_{t,2} & \cdots & \mathbf{a}_{t,n} \end{bmatrix}. \end{aligned} \tag{3.3}$$

We can then stack the delay equations of all of the test structures into a matrix form.

$$\mathbf{d}_t = \boldsymbol{\mu}_t + \mathbf{A}_t^T \mathbf{p} + \mathbf{R}_t \tag{3.4}$$

where $\mathbf{d}_t$ is defined in Section 3.2.

To illustrate the procedure more clearly and in an easier way, we will first assume, in the remainder of this section and in Section 3.4, that the spatially uncorrelated parameters can be ignored, i.e., $R = 0$ and $\mathbf{R_t} = \mathbf{0}$. We will relax this assumption later in Section 3.5, and introduce the extension of the method to include those parameters.

The variance of the Gaussian variable d and the covariance matrix of the multivariate normal variable $\mathbf{d}_t$ can be conveniently calculated as:

$$\sigma^2 = \mathbf{a}^T\mathbf{a} \tag{3.5a}$$

$$\mathbf{\Sigma}_t = \mathbf{A}_t^T\mathbf{A}_t. \tag{3.5b}$$

3.3.2 Conditional PDF Evaluation

The objective of our approach is to find the conditional PDF of the delay, d, of the original circuit, given the vector of delay values, $\mathbf{d}_r$. The values of $\mathbf{d}_r$ are measured from the test structures after the circuit is manufactured, corresponding to one set of samples of $\mathbf{d}_t$. We first introduce a theorem below; a sketch of the proof of the theorem can be found in [68].

Theorem 3.3.1 *Consider a Gaussian-distributed vector* $\begin{bmatrix}\mathbf{X}_1\\ \mathbf{X}_2\end{bmatrix}$ *with mean* $\boldsymbol{\mu}$ *and a non-singular covariance matrix* $\mathbf{\Sigma}$*. Let us define* $\mathbf{X}_1 \sim N(\boldsymbol{\mu}_1, \mathbf{\Sigma}_{11})$*,* $\mathbf{X}_2 \sim N(\boldsymbol{\mu}_2, \mathbf{\Sigma}_{22})$*. If* $\boldsymbol{\mu}$ *and* $\mathbf{\Sigma}$ *are partitioned as follows,*

$$\boldsymbol{\mu} = \begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix} \text{ and } \mathbf{\Sigma} = \begin{bmatrix}\mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12}\\ \mathbf{\Sigma}_{21} & \mathbf{\Sigma}_{22}\end{bmatrix}, \tag{3.6}$$

then the distribution of $\mathbf{X}_1$ *conditional on* $\mathbf{X}_2 = \mathbf{x}$ *is multivariate normal, and its mean and covariance matrix are given by*

$$\mathbf{X}_1|(\mathbf{X}_2 = \mathbf{x}) \sim N(\bar{\boldsymbol{\mu}}, \bar{\mathbf{\Sigma}}) \tag{3.7a}$$

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\mu}_1 + \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}(\mathbf{x} - \boldsymbol{\mu}_2) \tag{3.7b}$$

$$\bar{\mathbf{\Sigma}} = \mathbf{\Sigma}_{11} - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}\mathbf{\Sigma}_{21}. \tag{3.7c}$$

It can be shown that our problem can be mapped directly to the theorem. We refer to $\mathbf{X}_1$ the *original subspace*, and $\mathbf{X}_2$ the *test subspace*. By stacking d and $\mathbf{d}_t$ together,

a new vector $\mathbf{d}_{all} = \left[d\ \mathbf{d}_t^T\right]^T$ is formed, with the original subspace containing only one variable d and the test subspace containing the vector $\mathbf{d}_t$. The random vector $\mathbf{d}_{all}$ is multivariate Gaussian-distributed, with its mean and covariance matrix given by:

$$\boldsymbol{\mu}_{all} = \begin{bmatrix} \mu \\ \boldsymbol{\mu}_t \end{bmatrix} \text{ and } \boldsymbol{\Sigma}_{all} = \begin{bmatrix} \sigma^2 & \mathbf{a}^T\mathbf{A}_t \\ \mathbf{A}_t^T\mathbf{a} & \boldsymbol{\Sigma}_t \end{bmatrix}. \tag{3.8}$$

We may then apply the result of Theorem 3.3.1 to obtain the conditional PDF of d, given the delay information from the test structures. We know the conditional distribution of d is Gaussian, and its mean and variance can be obtained as:

$$\text{PDF}(d_{cond}) = \text{PDF}\left(d|(\mathbf{d}_t = \mathbf{d}_r)\right) \sim N(\bar{\mu}, \bar{\sigma}^2) \tag{3.9a}$$

$$\bar{\mu} = \mu + \mathbf{a}^T\mathbf{A}_t\boldsymbol{\Sigma}_t^{-1}(\mathbf{d}_r - \boldsymbol{\mu}_t) \tag{3.9b}$$

$$\bar{\sigma}^2 = \sigma^2 - \mathbf{a}^T\mathbf{A}_t\boldsymbol{\Sigma}_t^{-1}\mathbf{A}_t^T\mathbf{a}. \tag{3.9c}$$

3.3.3 Interpretation of the Conditional PDF

In this section, we analyze the information provided by the equations that represent the conditional PDF. From equations (3.9b) and (3.9c), we conclude that while the conditional mean of the original circuit is adjusted making use of the result vector $\mathbf{d}_r$, the conditional variance is *independent* of the measured delay values, $\mathbf{d}_r$.

Examining Equation (3.9c) more closely, we see that for a given circuit, the variance of its delay before measuring the test structures, σ^2, and the coefficient vector $\mathbf{a}$ are fixed and can be obtained from SSTA. The only variable that is affected by the test mechanism is the coefficient matrix of the test structures, $\mathbf{A}_t$, which also impacts $\boldsymbol{\Sigma}_t$. Therefore, the value of the conditional variance can be modified by adjusting the matrix $\mathbf{A}_t$. We know that $\mathbf{A}_t$ is the coefficient matrix formed by the sensitivities with respect to the principal components of the test structures. The size of $\mathbf{A}_t$ is determined by the number of test structures on the chip, and the entry values of $\mathbf{A}_t$ is related to the type of the test structures and their locations on the chip. Therefore if we use the same type of test structures on the circuit, then by varying their number and locations, we can modify the matrix $\mathbf{A}_t$, hence adjust the value of the conditional variance. Intuitively, this implies that the value of the conditional variance depends on how many test structures we have, and how well the test structures are distributed, in the sense of capturing spatial correlations between variables.

In our problem, $\mathbf{A}_t^T \in \mathbf{R}^{n \times m}$, where n is the number of test structures on chip, and m is the number of principal components. In the grid-based spatial correlation model, a large circuit usually has many grids, hence many principal components, whereas the number of test structures we can put on chip is limited by several factors mentioned in Section 3.2. Therefore n is usually less than m. Theorem 3.3.1 assumes that $\mathbf{\Sigma}_t = \mathbf{A}_t^T \mathbf{A}_t$ is of full rank and has an inverse, which means $\mathbf{A}_t^T$ must have full row rank. Detailed discussion about the ranks of $\mathbf{A}_t^T$ and $\mathbf{\Sigma}_t$ can be found in Section 3.4. For the present, we will assume that $\mathbf{A}_t^T$ is of full row rank.

Based on this assumption, consider the special case when $m = n$; in other words, that the number of test structures is identical to the number of PCA components. Intuitively, this means that we have independent data points that can predict the value of each of these components. In this case, $\mathbf{A}_t$ is a square matrix with full rank and has an inverse $\mathbf{A}_t^{-1}$. Substituting $\mathbf{\Sigma}_t^{-1} = (\mathbf{A}_t^T \mathbf{A}_t)^{-1} = \mathbf{A}_t^{-1}(\mathbf{A}_t^T)^{-1}$ into Equation (3.9b),

$$\begin{aligned} \bar{\mu} &= \mu + \mathbf{a}^T \mathbf{A}_t \mathbf{\Sigma}_t^{-1} (\mathbf{d}_r - \boldsymbol{\mu}_t) \\ &= \mu + \mathbf{a}^T (\mathbf{A}_t^T)^{-1} (\mathbf{d}_r - \boldsymbol{\mu}_t). \end{aligned} \tag{3.10}$$

It is interesting to note that the term $(\mathbf{A}_t^T)^{-1}(\mathbf{d}_r - \boldsymbol{\mu}_t)$ is the solution of the linear equations

$$\mathbf{d}_t = \boldsymbol{\mu}_t + \mathbf{A}_t^T \mathbf{p} = \mathbf{d}_r \tag{3.11}$$

with $\mathbf{p}$ as the set of unknowns. Therefore, Equation (3.10) is equivalent to first solving $\mathbf{p}$ from linear equations (3.11), then substituting its value into Equation (2.1) (with uncorrelated parameters disregarded for now) to find d. We can see that in this case,

$$\begin{aligned} \bar{\sigma}^2 &= \sigma^2 - \mathbf{a}^T \mathbf{A}_t \mathbf{\Sigma}_t^{-1} \mathbf{A}_t^T \mathbf{a} \\ &= \sigma^2 - \mathbf{a}^T \mathbf{A}_t \mathbf{A}_t^{-1} (\mathbf{A}_t^T)^{-1} \mathbf{A}_t^T \mathbf{a} \\ &= \sigma^2 - \mathbf{a}^T \mathbf{a} \\ &= 0. \end{aligned} \tag{3.12}$$

Thus the derived PDF is an impulse function with the mean equal to the original circuit delay and the variance equal to zero, and Equation (3.9) automatically takes the special case of $m = n$ into consideration.

We end this section by pointing out that an equivalent way of looking at the problem is to first stack the PC vector $\mathbf{p}$ and the delay vector $\mathbf{d}_t$ together, referring to $\mathbf{p}$ as the

original subspace, and $\mathbf{d}_t$ as the test subspace. From this, we obtain the conditional distribution of $\mathbf{p}$, using Theorem 3.3.1, as:

$$\text{PDF}(\mathbf{p}_{cond}) = \text{PDF}\left(\mathbf{p}|(\mathbf{d}_t = \mathbf{d}_r)\right) \sim N(\bar{\boldsymbol{\mu}}_{\mathbf{p}}, \bar{\boldsymbol{\Sigma}}_{\mathbf{p}}) \tag{3.13a}$$

$$\bar{\boldsymbol{\mu}}_{\mathbf{p}} = \mathbf{A}_t \boldsymbol{\Sigma}_t^{-1}(\mathbf{d}_r - \boldsymbol{\mu}_t) \tag{3.13b}$$

$$\bar{\boldsymbol{\Sigma}}_{\mathbf{p}} = \mathbf{I} - \mathbf{A}_t \boldsymbol{\Sigma}_t^{-1} \mathbf{A}_t^T \tag{3.13c}$$

where $\mathbf{I}$ represents the identity matrix, which is the unconditional covariance matrix of $\mathbf{p}$. The result (3.13) tells us that given the condition $\mathbf{d}_t = \mathbf{d}_r$, the mean and covariance matrix of $\mathbf{p}_{cond}$ are no longer $\mathbf{0}$ and $\mathbf{I}$. In other words, the entries in $\mathbf{p}_{cond}$ can no longer be perceived as principal components. Due to the linear relationship between $\mathbf{p}_{cond}$ and the process parameter variations, we are in fact gaining information on the parameter variations inside each grid.

According to Theorem 3.3.1, $\mathbf{p}_{cond}$ remains Gaussian distributed. Because d_{cond} has a linear relationship with $\mathbf{p}_{cond}$, d_{cond} is also Gaussian-distributed. Since $\mathbf{a}$ is fixed for a given circuit, the conditional mean and variance of d can be calculated as:

$$\begin{aligned}
\bar{\mu} &= \mu + \mathbf{a}^T E(\mathbf{p}_{cond}) = \mu + \mathbf{a}^T \mathbf{A}_t \boldsymbol{\Sigma}_t^{-1}(\mathbf{d}_r - \boldsymbol{\mu}_t) \\
\bar{\sigma}^2 &= E(\mu + \mathbf{a}^T \mathbf{p}_{cond} - (\mu + \mathbf{a}^T \bar{\boldsymbol{\mu}}_{\mathbf{p}}))^2 \\
&= \mathbf{a}^T E((\mathbf{p}_{cond} - \bar{\boldsymbol{\mu}}_{\mathbf{p}})(\mathbf{p}_{cond} - \bar{\boldsymbol{\mu}}_{\mathbf{p}})^T)\mathbf{a} \\
&= \mathbf{a}^T(\mathbf{I} - \mathbf{A}_t \boldsymbol{\Sigma}_t^{-1} \mathbf{A}_t^T)\mathbf{a} \\
&= \sigma^2 - \mathbf{a}^T \mathbf{A}_t \boldsymbol{\Sigma}_t^{-1} \mathbf{A}_t^T \mathbf{a}
\end{aligned} \tag{3.14}$$

Not surprisingly, this end result is exactly the same as (3.9). However, dividing the derivation into two steps, as we have done here, provides additional insight into the problem.

3.4 Locally Redundant but Globally Insufficient Test Structures

In practice, correlation matrices tend to be sparse since the spatial density of correlation goes up to a limited radius. As a consequence, it is found that a number of entries of

each row of A_t^T are zero for typical correlation matrices. For such a scenario, it is possible that we place too many test structures that collectively capture only a small portion of PCs, with the coefficients of other PCs being all zeros. In other words, in some portion of the chip, the number of test structures may exceed the number of PCs with nonzero coefficients, but overall there are not enough test structures to actually compute the delay of the original circuit. We refer to this as a *locally redundant but globally insufficient* problem.

We show below that in such a scenario $\mathbf{\Sigma}_t$ would be rank deficient. While this problem can be overcome by appropriate placement of the test structures, the placement of these structures is beyond the scope of this chapter: we assume that this has been done by the designer, and that it is provided as an input to our problem. Instead, we provide a general solution to take the locally redundant but globally insufficient problem into consideration during the evaluation of the conditional distribution. Our approach groups the redundant equations together and use a least-squares approach to capture the information. With locally redundant but globally insufficient test structures, the matrix $\mathbf{A}_t^T$ has the following structure after grouping all the zero coefficients together:

$$\mathbf{A}_t^T = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{0} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} \tag{3.15}$$

where $\mathbf{B}_{11} \in \mathbf{R}^{s \times q}$, with s being the number of test structures that have all-zero coefficients for the last $n - q$ principal components, and $s > q$, which means we have locally redundant test structures for these q principal components. Since we have prohibited two test structures with the same configurations from being placed in one grid, $\mathbf{B}_{11}$ must be of full column rank with rank q. Therefore, the maximum rank of $\mathbf{A}_t^T$ is $q + n - s$, less than n, so $\mathbf{\Sigma}_t$ also has a rank less than n and is singular. In this case, Equation (3.11) can be divided into two sets of equations:

$$\mathbf{B}_{11}\mathbf{p}_u = \mathbf{d}_{r,u} - \mu_{r,u} \tag{3.16}$$

$$\mathbf{B}_{21}\mathbf{p}_u + \mathbf{B}_{22}\mathbf{p}_v = \mathbf{d}_{r,v} - \mu_{r,v} \tag{3.17}$$

where $\mathbf{p}_u$, $\mathbf{p}_v$, $\mathbf{d}_{r,u}$, $\mathbf{d}_{r,v}$, $\mu_{r,u}$, $\mu_{r,v}$ are sub-vectors of the PC vector $\mathbf{p}$, the result vector $\mathbf{d}_r$, and the mean vector μ_t, correspondingly. Note that $\mathbf{B}_{11}$ is not square, and Equation (3.16) is an over-determined system. This can be solved in several ways, and we take

the least-squares solution as its equivalence.

$$\bar{\mathbf{p}}_u = (\mathbf{B}_{11}^T \mathbf{B}_{11})^{-1} \mathbf{B}_{11}^T \left(\mathbf{d}_{r,u} - \mu_{r,u}\right) \tag{3.18}$$

Under conditions (3.18) as well as (3.17), the conditional PDF of d can be computed as follows.

$$\begin{aligned} & \mathrm{PDF}(d_{cond}) = \mathrm{PDF}(d|\mathbf{d}_t = \mathbf{d}_r) \\ = \quad & \mathrm{PDF}(d|\mathbf{p}_u = \bar{\mathbf{p}}_u, \mathbf{B}_{21}\bar{\mathbf{p}}_u + \mathbf{B}_{22}\mathbf{p}_v = \mathbf{d}_{r,v} - \mu_{r,v}) \end{aligned} \tag{3.19}$$

This step is safe because Equation (3.16) does not provide any information for $\mathbf{p}_v$. The statistical properties of $\mathbf{p}_v$ have not been changed, meaning they can still act as PCs. Assume $\mathbf{a}_u$ is the sub-vector of $\mathbf{a}$ corresponding to $\mathbf{p}_u$, and $\mathbf{a}_v$ is the sub-vector corresponding to $\mathbf{p}_v$, then $d = \mu + \mathbf{a}_u^T \bar{\mathbf{p}}_u + \mathbf{a}_v^T \mathbf{p}_v$. The mean, variance of d and $\mathbf{B}_{21}\bar{\mathbf{p}}_u + \mathbf{B}_{22}\mathbf{p}_v$, and their covariance can be easily updated. The same technique introduced in Section 3.3 can be applied to calculate the final conditional PDF of d.

Special cases include when $q = m$, in which case we can compute all the PCs and the delay of the original circuit by applying least-squares approach to the whole system, and when $s = n$, in which case we cannot get any information on $\mathbf{p}_v$ and they will still be uncorrelated Gaussian with zero mean and unit variance in the end.

3.5 Spatially Uncorrelated Parameters

In Section 3.3, we had developed a theory for determining the conditional distribution of the delay, d, of the original circuit, under the data vector, $\mathbf{d}_r$, provided by the test structures. This derivation neglected the random variables R and $\mathbf{R}_t$ in the canonical form of Equation (2.1) and (3.4), corresponding to spatially uncorrelated variations.

We now extend this theory to include such effects, which may arise due to parameters such as T_{ox} and N_A that can take on a different and spatially uncorrelated value for each transistor in the layout. While these parameters can show both inter-die and intra-die variations, because the inter-die variation of each such parameter can be regarded as a PC and easily incorporated in the procedure of Section 3.3, we hereby focus on

the intra-die variations of these parameters, i.e., the purely random part. Thus, R is the random variable generated by merging the intra-die variations for each gate during traversal of the whole circuit [61], with mean 0 and variance σ_R^2. Considering this effect, the variance of the original circuit is adjusted to be

$$\sigma'^2 = \mathbf{a}^T\mathbf{a} + \sigma_R^2. \tag{3.20}$$

The covariance matrix of the test structures must also be updated as follows:

$$\mathbf{\Sigma}_t' = \mathbf{A}_t^T\mathbf{A}_t + diag[\sigma_{R_{t,1}}^2, \sigma_{R_{t,2}}^2, \cdots, \sigma_{R_{t,n}}^2]. \tag{3.21}$$

The same kind of technique from Section 3.3 can still be applied. However, in this case, due to the diagonal matrix added to $\mathbf{\Sigma}_t$, $\bar{\sigma}$ is never equal to zero, meaning that we can never compute the actual delay of the original circuit, which is a fundamental limitation of any testing-based diagnosis method. Any such strategy is naturally limited to spatially correlated parameters. The values of uncorrelated parameters in the original circuit cannot be accurately replicated in the test structures: these values may change from one device to the next, and therefore, their values in a test structure cannot perfectly capture their values in the original circuit.

3.6 Changing the Number of Stages in the ROs

In Section 3.5, it is shown that spatially uncorrelated parameter variations impose a challenge for our method, since it is physically impossible for a test structure to capture uncorrelated variations. However, it is possible to dilute the effects of uncorrelated variations, and to overcome this problem, an intuitive idea is to increase the number of stages of the RO test structures.

The essential idea of increasing the number of stages is that it leaves the spatially correlated variations unchanged: since each RO is small and lies within a spatial correlation grid, all spatially correlated parameters that affect its delay show identical variations. However, variations for spatially uncorrelated parameters may be in opposite directions and thus increasing the number of stages increases the likelihood of cancellations, implying that spatially uncorrelated parameters are likely to become relatively less important. In other words, this implies that the delay of each RO as a variable will be more correlated to the delay of the original circuit.

On the other hand, while increasing the number of stages of the ROs increases the correlation coefficient between the delays of the RO and the original circuit, it also makes the delays of the ROs more correlated with each other. This suggests that the RO test structures may collectively yield less independent information about the variations.

There is a clear trade-off here, and in this section, we illustrate the above qualitative argument from a more rigid, mathematical perspective, and present it in a quantitative way. We will show in Section 3.7 that for our implementation, increasing the number of stages does indeed yield better estimations of the post-silicon delay.

As stated in Section 3.3, the delay of the original circuit can be written in the canonical form of Equation (2.1). We rewrite the equation below.

$$d = \mu + \sum_{i=1}^{m} a_i p_i + R = \mu + \mathbf{a}^T \mathbf{p} + R. \tag{3.22}$$

Similarly, the delay of RO i can be written in the form of Equation (3.2), which is

$$d_{t,i} = \mu_{t,i} + \mathbf{a}_{t,i}^T \mathbf{p} + R_{t,i}. \tag{3.23}$$

First, if we assume that there is only one RO i on the chip, Equation (3.9c) becomes

$$\bar{\sigma}^2 = \sigma^2 - \frac{\mathbf{a}^T \mathbf{a}_{t,i} \mathbf{a}_{t,i}^T \mathbf{a}}{\sigma_{t,i}^2} = \sigma^2 \left(1 - \rho_i^2\right). \tag{3.24}$$

where ρ_i is the correlation coefficient between the delay of the original circuit and the delay of RO i. It is obvious that in this case, the result only depends on ρ_i.

Second, we explain how the number of stages affects the value of ρ_i, so that we can observe clearly how the number of stages affects our results. Let us assume that RO i has k stages, and for purposes of illustration, we will assume that each stage of the RO is identical, with a canonical delay of the form $\alpha_i + \sum_{j=1}^{m} \gamma_{ij} p_j + \zeta_i = \alpha_i + \mathbf{\Gamma}_i \mathbf{p} + \zeta_i$. The half-period of RO i, which is a surrogate for its delay, is therefore given by

$$d_{t,i} = k\alpha_i + k\mathbf{\Gamma}_i \mathbf{p} + \sqrt{k}\zeta_i \tag{3.25}$$

From Equation (3.21) in Section 3.5, the variance of the delay of RO i can be written as

$$\sigma_{t,i}^2 = k^2 \mathbf{\Gamma}_i^T \mathbf{\Gamma}_i + k\zeta_i^2. \tag{3.26}$$

The correlation coefficient between RO i and the original circuit can thus can be calculated from the relation:

$$\rho_i^2 = \frac{k^2 \mathbf{a}^T \mathbf{\Gamma}_i \mathbf{\Gamma}_i^T \mathbf{a}}{\sigma^2 \left(k^2 \mathbf{\Gamma}_i^T \mathbf{\Gamma}_i + k\zeta_i^2\right)} = \frac{\mathbf{a}^T \mathbf{\Gamma}_i \mathbf{\Gamma}_i^T \mathbf{a}}{\sigma^2 \left(\mathbf{\Gamma}_i^T \mathbf{\Gamma}_i + \frac{1}{k}\sigma_r^2\right)}. \tag{3.27}$$

It is easy to see that as k increases, the correlation coefficient between RO i and the original circuit increases, implying that the conditional variance of the delay of the original circuit decreases. Therefore, we have more specific information about the delay of the original circuit. This is in accordance with the intuition that increasing the number of stages in the RO helps in reducing the effect of the spatially uncorrelated parameters.

Third, we illustrate the fact that as the number of stages increases, the ROs can become more correlated with each other and might not give as much information collectively. To see this, we consider the delays of two ROs $d_{t,1}$ and $d_{t,2}$. If we assume that each has k stages, then

$$d_{t,1} = k\alpha_1 + k\mathbf{\Gamma}_1^T \mathbf{p} + \sqrt{k}\zeta_1 \tag{3.28}$$

$$d_{t,2} = k\alpha_2 + k\mathbf{\Gamma}_2^T \mathbf{p} + \sqrt{k}\zeta_2. \tag{3.29}$$

The correlation coefficient between the two can be calculated as

$$\begin{aligned} \rho_{1,2} &= \frac{k^2 \mathbf{\Gamma}_1^T \mathbf{\Gamma}_2}{\left(k^2 \mathbf{\Gamma}_1^T \mathbf{\Gamma}_1 + k\zeta_1^2\right)\left(k^2 \mathbf{\Gamma}_2^T \mathbf{\Gamma}_2 + k\zeta_2^2\right)} \\ &= \frac{\mathbf{\Gamma}_1^T \mathbf{\Gamma}_2}{\left(\mathbf{\Gamma}_1^T \mathbf{\Gamma}_1 + \frac{1}{k}\zeta_1^2\right)\left(\mathbf{\Gamma}_2^T \mathbf{\Gamma}_2 + \frac{1}{k}\zeta_2^2\right)}. \end{aligned} \tag{3.30}$$

It is easily observed that as k increases, the correlation coefficient between the delays of the two ROs increases.

The conditional variance of the delay of the original circuit can be calculated based on the testing results of the delays of these two ROs, using Equation (3.9c), as

$$\begin{aligned} \bar{\sigma}^2 &= \sigma^2 - \mathbf{a}^T \begin{bmatrix} \mathbf{a}_{t,1} & \mathbf{a}_{t,2} \end{bmatrix} \begin{bmatrix} \sigma_{t,1}^2 & \mathbf{a}_{t,1}^T \mathbf{a}_{t,2} \\ \mathbf{a}_{t,2}^T \mathbf{a}_{t,1} & \sigma_{t,2}^2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{a}_{t,1}^T \\ \mathbf{a}_{t,2}^T \end{bmatrix} \mathbf{a} \\ &= \sigma^2 \left(1 - \frac{\rho_1^2 + \rho_2^2 - 2\rho_1\rho_2\rho_{1,2}}{1 - \rho_{1,2}^2}\right) \end{aligned} \tag{3.31}$$

This result confirms our intuition that the conditional variance of the delay of the original circuit is not only dependent upon the correlation coefficient between the delay of the original circuit and the delay of each RO (ρ_1,ρ_2), but also dependent upon the correlation coefficient between the two ROs ($\rho_{1,2}$).

To see the effect of k on the conditional variance more clearly, we write the above equation as

$$\bar{\sigma}^2 = \sigma^2 - \frac{C_1^2\left(V_2 + \frac{1}{k}\zeta_2^2\right) - C_2^2\left(V_1 + \frac{1}{k}\zeta_1^2\right) + 2C_1C_2\mathbf{\Gamma}_1^T\mathbf{\Gamma}_2}{\left(V_2 + \frac{1}{k}\zeta_2^2\right)\left(V_1 + \frac{1}{k}\zeta_1^2\right) - \left(\mathbf{\Gamma}_1^T\mathbf{\Gamma}_2\right)^2} \tag{3.32}$$

where $C_i = \mathbf{a}^T\mathbf{\Gamma}_i$ and $V_i = \mathbf{\Gamma}_i^T\mathbf{\Gamma}_i$ are not dependent on k. As k increases, both the numerator and the denominator decrease, the function is not guaranteed to be monotonic with respect to k. Therefore theoretically increasing the number of stages does not necessarily reduce the conditional variance of the delay of the original circuit we can get. We demonstrate in Section 3.7 that for the practical test cases that we study, the results lie within a monotone decreasing region with respect to k.

3.7 Experimental Results

We summarize the proposed post-silicon statistical delay prediction approach as Algorithm 1.

Algorithm 1 Post-silicon statistical delay prediction.

1: Perform SSTA on both the original circuit and the test structures to determine μ, $\mathbf{a}$, $\boldsymbol{\mu}_t$, $\mathbf{A}_t$, and σ_R, $\sigma_{R_{t,1}}, \cdots, \sigma_{R_{t,n}}$.
2: After fabrication, test the delay of the test structures on-chip to obtain $\mathbf{d}_r$.
3: Compute the conditional mean $\bar{\mu}$ and variance $\bar{\sigma}^2$ for the original circuit using the expressions in Equation (3.9).

We use the software package *MinnSSTA* [8] to perform SSTA, and use Monte Carlo methods to test our approach. The original circuits correspond to the ISCAS89 benchmark suite, and each test structure is assumed to be a RO. Specifically, the RO used in our experiments has three stages for the first set of experiments and five stages thereafter. Section 3.6, combined with simulation later in this section, shows that increasing the number of stages can compensate for the effects of spatially uncorrelated parameter variations in practice.

A grid-based spatial correlation model [6] is used to compute the covariance matrix for each spatially correlated parameter. Under this model, if the number of grids is G, and the number of spatially correlated parameters being considered is P, then the total number of principal components is no more than $P \cdot G$. The parameters that are considered as sources of spatially correlated variations include the effective channel length L, the transistor width W, the interconnect width W_{int}, the interconnect thickness T_{int} and the inter-layer dielectric H_{ILD}. The dopant concentration, N_A, is regarded as the source of spatially uncorrelated variations. For interconnects, instead of two metal tiers used in [69], we use four metal tiers (corresponding to two horizontal and two vertical layers). Parameters of different metal tiers are assumed to be uncorrelated.

Table 3.1 lists the level of parameter variations assumed in this work. The process parameters are Gaussian-distributed, and their mean and 3σ values are shown in the table. For each parameter, half of the variational contribution is assumed to be from inter-die variations and half from intra-die variations. We assume this variation model is accurate in our simulation. In practice the model should be tailored according to manufacturing data.

Table 3.1: Parameters used in the experiments.

	L (nm)	W (nm)	W_{int} (nm)	T_{int} (nm)	H_{ILD} (nm)	N_A nmos/pmos (10^{17}cm^{-3})
μ	60.0	150.0	150.0	500.0	300.0	9.7/10.04
3σ	12.0	22.5	30.0	75.0	45.0	1.45

In the *first set* of experiments, only one variation is taken into consideration in the Monte Carlo analysis: in this case, we consider the effective channel length L, which we observe to be the dominant component of intra-die variations. Under the grid-based correlation model, there will only be G independent variation sources in this case, and by providing G test structures, we can use the techniques in Section 3.3 to calculate the delay of the original circuit.

The result is shown as a scatter plot in Figure 3.4. The method is applied to 1000 chips: we simulate this by performing 1000 Monte Carlo simulations on each benchmark,

each corresponding to a different set of parameter values. For each of these values, we compute the deterministic delays of the test structures[2] and the original circuit: we use the former as inputs to our approach, and compare the delay from our statistical delay prediction method with the latter. The fact that all of the points lie closely around the $y = x$ line indicates that the circuit delays predicted by our approach matches very well with the Monte Carlo simulation results.

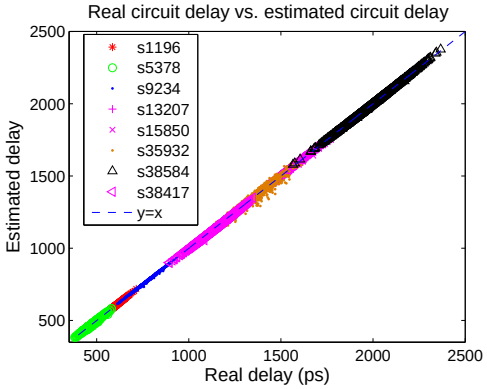


Figure 3.4: The scatter plot: real circuit delay vs. predicted circuit delay.

The precise testing error for each benchmark is listed in Table 3.2. If we denote the delay of the original circuit at a sample point as d_{orig} and the delay of the original circuit, as predicted by our statistical delay prediction approach, as d_{pred}, the test error for each simulation is defined as $\frac{|d_{orig}-d_{pred}|}{d_{orig}} \times 100\%$. The second column of the table shows the average test error, based on all 1000 sample points, which indicates the overall aggregate accuracy: this is seen to be well below 1% in almost all cases. The third column shows the maximum deviation from the mean value of statistical timing over all 1000 sample points, as a fraction of the mean. The test error at this point is shown in the fourth column of the table. These two columns indicate that the results are accurate even when the sampled delay is very different from the mean value.

Note that in theory, according to the discussion in Section 3.3, when one test structure is placed in each variational grid, the prediction should be perfect. However, some

[2] Because of the way in which these values are computed in our experimental setup, variations in the test structure delays are only caused by random variations. In practice, the measured test structure delays will consist of deterministic variations, random variations, and measurement noise. It is assumed here that standard methods can be used to filter out the effects of the first and the third factor.

Table 3.2: Test errors considering only variations in L.

Benchmark	Average Error	Maximum Deviation (% of mean)	Error at Maximum Deviation
s1196	0.18%	24.2%	0.20%
s5378	0.58%	25.7%	0.02%
s9234	0.35%	22.7%	0.50%
s13207	0.09%	25.2%	0.51%
s15850	0.25%	26.1%	0.47%
s35932	1.31%	22.4%	1.01%
s38584	0.10%	27.5%	0.69%
s38417	0.09%	27.4%	0.58%

inaccuracies creep in during SSTA, primarily due to the error in approximating the *max* operation in SSTA, during which the the distribution of the maximum of two Gaussians, which is a non-Gaussian, is approximated as a Gaussian to maintain the invariant. For circuits like s35932, which show the largest average error among this set, of under 2%, the canonical form (2.1) is not perfectly accurate in modeling the circuit delay. Note that our experimental setup is based on simulation, and does not include any measurement noise.

For the unoptimized ISCAS89 benchmark suite, one or a small number of critical paths tend to dominate the circuit, which is unrealistic. However, s35932 is an exception and thus is used to compare our approach with the critical path replica approach currently used in ABB. We assume that in the critical path approach the whole critical path for the nominal design can be perfectly replicated, and compare the delay of that path and the delay of the whole circuit during the Monte Carlo simulation. It is observed that the critical path replica can show a maximum error of 15.5%, while our approach has a maximum error of 6.92%, an improvement of more than 50%. The average error of critical path replica for this circuit is 1.92%, also significantly larger than our result of 1.31%.

To show the confidence scalability of our approach, in the *second set* of experiments, we consider cases in which the number of test structures is insufficient to completely predict the delay of the original circuit. In this experiment, different numbers of test

Table 3.3: Prediction results with insufficient number of test structures (considering L): case 1 and case 2 are distinguished by the number of ROs available for each circuit.

Benchmark			SSTA Results		Case 1			Case 2		
Name	#Cells	#Grids	μ(ps)	σ(ps)	#RO	Avg. $\bar{\sigma}(ps)$	Avg. $\sigma_{reduction}$	#RO	Avg. $\bar{\sigma}(ps)$	Avg. $\sigma_{reduction}$
s1196	547	16	577.06	35.32	10	6.48	81.64%	5	11.97	66.1%
s5378	2958	16	475.97	29.84	10	5.96	80.02%	5	10.77	63.9%
s9234	5825	16	775.36	51.51	10	9.50	81.55%	5	18.85	63.4%
s13207	8260	256	1399.8	92.81	150	9.63	89.62%	60	18.56	80.0%
s15850	10369	256	1573.7	100.48	150	8.25	91.79%	60	16.88	83.2%
s35932	17793	256	1359.5	82.17	150	11.08	86.52%	60	27.69	66.3%
s38584	20705	256	1994.0	120.83	150	16.54	86.31%	60	29.96	75.2%
s38417	23815	256	1139.8	76.38	150	9.40	87.69%	60	17.87	76.6%

structures are implanted on the die. Specifically, for circuits divided into 16 grids, we investigate Case 1, when 10 test structures and Case 2, when 5 test structures are available.

For circuits where the die is divided into 256 grids, Case 1 corresponds to a die with 150 test structures, and Case 2 to 60 test structures. To show how much more information than SSTA we get from the test structures, we define $\sigma_{reduction}$ as $\frac{\sigma-\bar{\sigma}}{\sigma} \times 100\%$ which is independent of the test results but is dependent on how the available test structures are placed on the chip. To be as general as possible, we perform 1000 random selections of the grids to put test structures in. The μ, σ of the original circuit, obtained from SSTA, and the average $\bar{\sigma}$, $\sigma_{reduction}$ of the statistical delay prediction approach for both cases, over the 1000 selections, are listed in Table 3.3 for each benchmark circuit. It is observed that there is a trade-off between test structure overhead and $\sigma_{reduction}$.

Figure 3.5 shows the predicted delay distribution for a typical sample of the circuit s38417, the largest circuit in the benchmark suite. Each curve in the circuit corresponds to a different number of test structures, and it is clearly seen that even when the number of test structures is less than G, a sharp PDF of the original circuit delay can still be obtained using our method, with a variance much smaller than provided by SSTA. The trade-off between the number of test structures and the reduction in the standard deviation can also be observed clearly. For this particular die, while SSTA can only assert that it can meet a 1400 ps delay requirement, using 150 test structures we can say with more than 99.7% confidence that the fabricated chip meets a 1040 ps delay

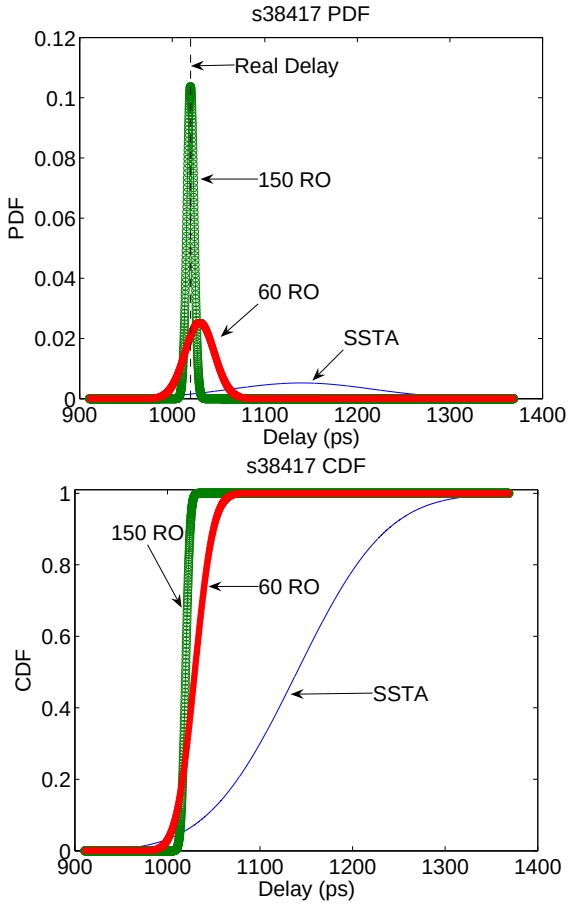


Figure 3.5: PDF and CDF with insufficient number of test structures for circuit s38417 (considering L).

requirement, and using 60 test structures we can say with such confidence that it can meet a 1080 ps delay requirement.

In our *third set* of experiments, we consider the most general case in which all parameter variations are included. While the first two sets of experiments provided general insight into our method, this third set shows the result of applying it to real circuits under the full set of parameter variations listed in Table 3.1. In Case I of this set of experiments, the number of test structures is equal to the number of grids. The values of $\bar{\sigma}$ and $\sigma_{reduction}$ are fixed in this case. Case II and Case III are set up the same way as in Case 1 and Case 2, respectively, of the second set of experiments described

Table 3.4: Prediction results considering all parameter variations: case I, case II and case III are distinguished by the number of ROs.

Benchmark	SSTA Results		Case I		Case II			Case III		
	μ (ps)	σ (ps)	$\bar{\sigma}$ (ps)	$\sigma_{reduction}$	Avg. $\bar{\sigma}$ (ps)	$\sigma_{reduction}$ Avg.	$\sigma_{reduction}$ Min.	Avg. $\bar{\sigma}$ (ps)	$\sigma_{reduction}$ Avg.	$\sigma_{reduction}$ Min.
s1196	577.42	45.61	11.32	75.2%	12.67	72.2%	65.3%	15.20	66.7%	58.4%
s5378	475.65	37.24	6.35	82.9%	7.69	79.4%	71.4%	10.28	72.4%	59.8%
s9234	776.79	62.63	9.17	85.4%	12.20	80.5%	66.7%	17.21	72.5%	56.2%
s13207	1404.25	109.41	20.90	80.9%	22.97	79.0%	74.6%	27.13	75.2%	66.5%
s15850	1579.73	119.45	19.59	83.6%	21.09	82.3%	79.4%	24.69	79.3%	73.7%
s35932	1371.55	98.45	24.75	74.9%	27.11	72.5%	67.7%	30.69	68.8%	63.9%
s38584	2011.62	147.46	39.47	73.2%	43.16	70.7%	64.7%	48.77	66.9%	60.8%
s38417	1146.56	89.84	22.01	75.5%	24.09	73.2%	67.2%	28.17	68.6%	57.3%

earlier. The μ, σ of each benchmark circuit obtained by SSTA, the $\bar{\sigma}$, $\sigma_{reduction}$ for Case I, the average $\bar{\sigma}$ and average $\sigma_{reduction}$ for Case II and Case III obtained from the post-silicon statistical delay prediction are listed in Table 3.4. In order to get an idea on what the result would be like if a really bad set of grids are selected to put test structures in, in this table we also show the minimum (Min.) $\sigma_{reduction}$ over the 1000 random selections for each circuit in Case II and Case III. The distribution plot for this set of experiment is similar to that in Figure 3.5, and the conditional PDFs of one particular sample of the circuit s1196 for Case II and Case III are shown in Section 3.1 as Figure 3.3, with the SSTA PDF as a comparison. Note that the conditional PDF obtained by our approach would be even sharper for Case I.

The reduction in the standard deviation is only able to demonstrate that our predicted delay is within a certain range. To see whether the prediction is reasonable and accurate, in our third set of experiments, we also perform the following Monte Carlo simulations. In Case I of this experiment, because in each grid we have one RO, we just perform one thousand Monte Carlo simulations based on this structure. In Case II and Case III, however, the number of ROs is smaller than the number of grids. Therefore we use five randomly selected sets of grids to put ROs in, and for each set of grids, we perform 1000 Monte Carlo simulations, which means totally we have 5000 Monte Carlo simulations for each circuit of Case II and Case III. While each Monte Carlo simulation generates a specific delay number, our prediction result is a conditional distribution of

the delay. Therefore if the Monte Carlo result falls within $\pm 3\bar{\sigma}$ of the predicted distribution, then we call the result a *hit*. Otherwise, we call it a *miss*. The *hit rate* of our prediction for a circuit is then defined as the number of hits divided by the total number of Monte Carlo simulations. We show the hit rates for each circuit in Table 3.5.

Table 3.5: Hit rates considering all parameter variations: case I, case II and case III are distinguished by different number of ROs available for each circuit.

Benchmark	Hit Rate		
	Case I	Case II	Case III
s1196	100.0%	99.9%	99.9%
s5378	99.8%	99.7%	99.9%
s9234	100.0%	99.9%	99.9%
s13207	99.9%	100.0%	100.0%
s15850	99.9%	99.9%	99.9%
s35932	97.2%	97.7%	98.7%
s38584	100.0%	99.9%	99.8%
s38417	99.9%	100.0%	99.9%

It is observed that most hit rates are above 99.9%.

Now we show that for the ISCAS89 benchmark circuits and our experimental setup, increasing the number of stages can compensate for the effect of spatially uncorrelated parameter variations and give us more specific information about the circuit delay after fabrication. We assume that each grid contains an RO, and for each RO, every stage has the same timing characteristics. Therefore we can use the coefficients and the spatially uncorrelated variable calculated for a 5-stage RO to derive the corresponding coefficients and spatially uncorrelated variable for a unit stage of that RO. Based on these timing characteristics of one unit stage, the timing characteristics of a RO with any number of stages can be calculated. This procedure is repeated for each of the ROs on chip. For each circuit we draw a curve, with the y axis being the conditional variance of the delay of the original circuit we can get from our approach, and the x axis being the number of unit stages we have for every ring oscillator built on this circuit. This plot shows that for our set of benchmarks, as the number of stages increases, the conditional variance we can get becomes progressively smaller. Sample results of the circuits s13207 and s5378 are shown in Figure 3.6. It is easily observed that the curves are monotonically

decreasing. The results are similar for all other circuits in the benchmark set.

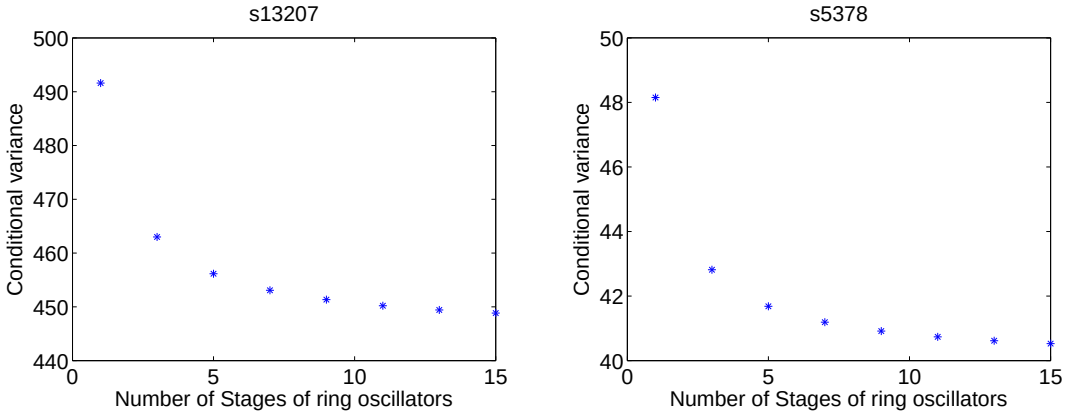


Figure 3.6: Conditional variance of the delay of the original circuit with respect to the number of stages of ROs.

Table 3.6: Runtime results

Circuit	s1196	s5378	s9234	s13207	s15850	s35932	s38584	s38417
Runtime (sec)	5.68×10^{-4}	5.70×10^{-4}	5.96×10^{-4}	0.39	0.39	0.35	0.37	0.68

Finally we provide runtime results for our approach. It is easily observed that our algorithm can be divided into two parts, separated by the physical measurements of the delays of the ring oscillators. The first part is SSTA, and because the framework is similar to [8], the readers are referred to that paper for a runtime estimate. The runtime for the second part, which is conditional PDF evaluation, is listed in Table 3.6. The experiments are run on a Linux PC with a 2.0GHz CPU and 256MB memory. The results we show here are for Case I of Table 3.4, where the matrix Σ_t in Section 3.2 is the largest of all three cases. It is shown that for all the benchmark circuits, the runtime is less than one second.

3.8 Conclusion

In this chapter, a general framework for the post-silicon statical delay prediction approach is proposed, using SSTA and a conditional PDF evaluation method, making use

of test data from RO test structures. We would like to point out that in cases where the circuit is dominated by a single critical path (this is not often the case, since most circuits are timing-optimized, which implies that there are numerous near-critical paths), it may be beneficial to use a critical path replica instead of our ring oscillator based scheme. The critical path replica can also be viewed as a type of test structure, which means that after determining the nominal critical path, we can replicate it, perform SSTA on this path, and calculate the conditional variance of the original circuit delay, given that the delay of this path is known. If a circuit is highly dominated by this path, then the conditional variance would be small. We then can compare the conditional variance calculated in this way with the conditional variance calculated by our approach.

Depending on which variance is smaller, we can choose the appropriate approach and start building the circuit embedded with the proper test structure. This choice can be made entirely through presilicon analysis. The variances of the conditional PDFs for the two possible test structures (a set of RO measurements, or a critical path replica) may be computed using Equation (3.9c). Note that (3.9c) provides results that are independent of measurement data, and hence depending on which structure has the smaller covariance, we can choose an appropriate test structure.

Chapter 4

Representative Critical Path Synthesis

Several approaches to post-silicon adaptation require feedback from a replica of the nominal critical path, whose variations are intended to reflect those of the entire circuit after manufacturing. For realistic circuits, where the number of critical paths can be large, the notion of using a single critical path is too simplistic. This chapter overcomes this problem by introducing the idea of synthesizing a representative critical path (RCP), which captures these complexities of the variations. We first prove that the requirement on the RCP is that it should be highly correlated with the circuit delay. Next, we present two novel algorithms to automatically build the RCP. Our experimental results demonstrate that over a number of samples of manufactured circuits, the delay of the RCP captures the worst case delay of the manufactured circuit. The average prediction error of all circuits is shown to be below 2.8% for both approaches, as compared to errors of as large as 5.8% over the conventional method that uses a replica of the nominal critical path. For both our approach and the critical path replica method, it is essential to guard-band the prediction to ensure pessimism: on average our approach requires a guard band 31% smaller than for the critical path replica method.

4.1 Introduction

In this chapter, we focus on post-silicon tuning methods that require replicating the critical path of a circuit. Such techniques include adaptive body bias (ABB) [25,26,27] or adaptive supply voltage (ASV) [28,29]. The ABB approach that is used in [25,26,27] employs a replica of the critical path at nominal parameter values (we call this the nominal critical path), whose delay is rapidly measured and used to determine the optimal adaptation of the body bias levels of the transistors in the combinational block. However, as is stated in Chapter 1, this has obvious problems: first, it is likely that a large circuit will have more than a single critical path, and second, a nominal critical path may have different sensitivities to the parameters than other near-critical paths, and thus may not be representative. We quantitatively illustrate this problem in our experimental results. The ASV technique in [28, 29], on the other hand, use a speed monitor consisting of only one logic dominated element and one interconnect dominated element, and assume that the speed tested for this simplified cicuitry is generally applicable to all parts of the circuit. In the presence of significant within-die variations, this assumption becomes invalid. Moreover, the approach requires substantial extra memory even for process bins of a very coarse resolution, and is not scalable to fine grids.

From an analysis perspective, Chapter 3 uses a number of on-chip ring oscillators, presumably provided by designers, to capture the parameter variations of the original circuit. However, this approach requires measurements for hundreds of ring oscillators for a circuit with reasonable size and does not provide an explicit critical path. From a synthesis perspective, a less costly test structure, preferably a representative critical path, is desired.

In this chapter, we propose a new way of thinking about the problem. We automatically build an on-chip test structure that captures the effects of parameter variations on all critical paths, so that a measurement on this test structure provides us a reliable prediction of the actual delay of the circuit, with minimal error, for all manufactured die. The key idea is to synthesize a test structure whose delay can reliably predict the maximum delay of the circuit, under across-die as well as within-die variations. In doing so, we take advantage of the property of spatial correlation between parameter variations to build this structure and determine the physical locations of its elements.

The test structure that we create, which we refer to as the *representative critical path* (RCP), is typically different from the critical path at nominal values of the process parameters. In particular, a measurement on the RCP provides the worst-case delay of the whole circuit, while the nominal critical path is only valid under no parameter variations, or very small variations. Since the RCP is an on-chip test structure, it can easily be used within existing post-silicon tuning schemes, e.g., by replacing the nominal critical path in the schemes in [25, 26, 27]. While our method accurately captures any correlated variations, it suffers from one limitation that is common to any on-chip test structure: it cannot capture the effects of spatially uncorrelated variations, because by definition, there is no relationship between those parameter variations of a test structure and those in the rest of the circuit. To the best of our knowledge, this work is the first effort that synthesizes a critical path in the statistical sense. The physical size of the RCP is small enough that it is safe to assume that it can be incorporated into the circuit (using reserved space that may be left for buffer insertion, decap insertion, etc.) without significantly perturbing the layout.

The remainder of the chapter is organized as follows. Section 4.2 introduces the background of the problem and formulates the problem mathematically. Next, Section 4.3 illustrates the detailed algorithms of our approach. Experimental results are provided in Section 4.4, and Section 4.5 concludes the chapter.

4.2 Problem Formulation

Similar to Chapter 3, in this chapter we also use the grid-based model discussed in Section 2.5 of Chapter 2 to model spatially correlated parameter variations. Our overall approach can be summarized as follows. We have a circuit whose delay can be represented as a random variable, d_c. Using the method presented in this chapter, we build the RCP whose delay can be represented by another random variable, d_p. After the circuit is manufactured, we measure the delay of the RCP, and find that it equals d_{pr}. In other words, d_{pr} corresponds to one sample of d_p for a particular set of parameter values. From this measured value of d_{pr}, we will infer the value, d_{cr}, of d_c for this sample, i.e., corresponding to this particular set of parameter values.

To mathematically simplify the situation and explore the relationship between the

variables d_c and d_p, we assume that all parameter variations are Gaussian-distributed, and the delay of both the circuit and the critical path can be approximated by an affine function of those parameter variations. These functions can be obtained by performing SSTA using existing techniques [8], and the end results of d_c and d_p can be represented by Gaussian-distributed PDFs.

Let $d_c \sim N(\mu_c, \sigma_c)$, $d_p \sim N(\mu_p, \sigma_p)$, and let the correlation coefficient of d_c and d_p be ρ. Then we know that the joint PDF of d_c and d_p is

$$f(d_c = d_{cr}, d_p = d_{pr}) = \frac{1}{2\pi\sigma_c\sigma_p\sqrt{1-\rho^2}} e^{C_1}, \tag{4.1}$$

where
$C_1 = -\frac{1}{2(1-\rho^2)}\left(\frac{(d_{cr}-\mu_c)^2}{\sigma_c^2} + \frac{(d_{pr}-\mu_p)^2}{\sigma_p^2} - \frac{2\rho(d_{cr}-\mu_c)(d_{pr}-\mu_p)}{\sigma_c\sigma_p}\right)$.
Using basic statistical theory, the conditional PDF of $d_c = d_{cr}$, given the condition $d_p = d_{pr}$, can be derived to have the following expression.

$$f(d_c = d_{cr} | d_p = d_{pr}) = \frac{f(d_{cr}, d_{pr})}{f(d_{pr})} = \frac{1}{2\pi\sigma_c\sqrt{1-\rho^2}} e^{C_2}, \tag{4.2}$$

where
$C_2 = -\frac{1}{2\sigma_c^2(1-\rho^2)}\left(d_{cr} - \left(\mu_c + \frac{\rho\sigma_c}{\sigma_p}(d_{pr}-\mu_p)\right)\right)^2$.
Therefore the conditional distribution of d_{cr} is a Gaussian with mean $\mu_c + \frac{\rho\sigma_c}{\sigma_p}(d_{pr}-\mu_p)$ and variance $\sigma_c^2(1-\rho^2)$.

The result of this conditional distribution can be used in various ways. For example, we can provide the entire conditional distribution as the output of this procedure, as in [69]. On the other hand, if the conditional variance can be made sufficiently small, we can be more specific and directly use the mean of the conditional distribution as the predicted value of the delay of the circuit, and the variance may be interpreted as the mean square error of infinite samples.

An alternative view, from a least squares perspective, is that it is desirable to minimize the variance, so that the mean is an estimate of the circuit delay with the smallest mean square error. For the term representing the variance of the conditional distribution, $\sigma_c^2(1-\rho^2)$, σ_c is fixed since the original circuit must remain undisturbed, implying that the variance of the conditional distribution is dependent only on ρ.

In other words, minimizing the variance is therefore equivalent to maximizing ρ. This formal result is also an intuitive one: the RCP should satisfy the property that

the correlation of its delay with that of the original circuit is maximized. Hence, our focus is on developing an efficient algorithm to build such an RCP, with the objective of maximizing the correlation coefficient.

4.3 Generation of the Critical Path

4.3.1 Background

As mentioned in Section 4.2, it is important to represent the variables d_c and d_p as affine functions with respect to the parameter variations. To achieve this, we will employ previously developed SSTA techniques. As in Chapter 3, we use the parameterized SSTA procedure introduced in Section 2.3 of Chapter 2 to obtain d_c as an affine function in the canonical form. We will show that this canonical form, in which the variables in the affine function consist of the m PCs and the independent parameter, makes the calculation of the correlation coefficient ρ defined in Section 4.2 much easier.

The canonical expression for d_c, which is in similar fashion with Equation (2.1), is shown below for clarity:

$$d_c = \mu_c + \sum_{i=1}^{m} a_i p_i = \mu_c + \mathbf{a}^T \mathbf{p} + R_c, \tag{4.3}$$

where d_c, μ_c are defined in Section 4.2, and all other notations are defined as in Equation (2.1) of Chapter 2, for the original circuit.

Performing SSTA on the RCP yields another delay expression in canonical form:

$$d_p = \mu_p + \sum_{i=1}^{m} b_i p_i = \mu_p + \mathbf{b}^T \mathbf{p} + R_p \tag{4.4}$$

where d_p, μ_p are defined in Section 4.2, and $p_i, b_i, \mathbf{b}, \mathbf{p}, R_p$ are all inherited from Equation (4.3).

4.3.2 Finding the Correlation Coefficient: Computation and Intuition

Since the original circuit and RCP are on the same chip, the values of the principal components for a given manufactured part are identical for both, and therefore their delays are correlated. In the manufactured part, any alteration in the PCs affects both

the original circuit and the RCP, and if the RCP can be constructed to be highly correlated with the original circuit, the circuit delay can be estimated to a good degree of accuracy. In the extreme case where the correlation coefficient $\rho = 1$, the circuit delay, d_c, can be exactly recovered from d_p; however, as we will show later, this is not a realistic expectation.

The correlation coefficient, ρ, can easily be computed as

$$\rho = \frac{\mathbf{a}^T\mathbf{b}}{\sigma_c\sigma_p} \tag{4.5}$$

where $\sigma_c = \sqrt{\mathbf{a}^T\mathbf{a} + \sigma_{R_c}^2}$ and $\sigma_p = \sqrt{\mathbf{b}^T\mathbf{b} + \sigma_{R_p}^2}$. An important point to note is that ρ depends only on the coefficients of the PCs for both the circuit and the critical path and their independent terms, and not on their means.

As discussed in Section 4.2, the mean of the conditional distribution $f\left(d_c = d_{cr} | d_p = d_{pr}\right)$, which can be used as an estimate of the original circuit delay, is:

$$\bar{\mu} = \mu_c + \frac{\rho\sigma_c}{\sigma_p}\left(d_{pr} - \mu_p\right) = \mu_c + \frac{\mathbf{a}^T\mathbf{b}}{\sigma_p^2}\left(d_{pr} - \mu_p\right). \tag{4.6}$$

The variance, which is also the mean square error of the circuit delay estimated using the above expression, for infinite samples, is $\sigma_c^2\left(1-\rho^2\right)$. Our goal is to build a critical path with the largest possible ρ.

Our theory assumes that the effects of systematic variations can be ignored, and we will show, at the end of Section 4.4, that this is a reasonable assumption. However, it is also possible to extend the theory to handle systematic variations in parameters that can be controlled through design: for a fully characterized type of systematic variation, we can compensate for it by choosing a shifted nominal value for the parameter.

It is also useful to provide an intuitive understanding of the ideas above. If we were to achieve our goal of setting $\rho = 1$, this would imply that

$$\rho = \frac{\mathbf{a}^T\mathbf{b}}{\sigma_c\sigma_p} = 1$$

This means that

$$\sum_{i=1}^{m} \bar{a}_i\bar{b}_i = 1$$

where $\bar{a}_i = a_i/\sigma_c$, and $\bar{b}_i = b_i/\sigma_p$. Note that $\bar{a}_i$ and $\bar{b}_i$ correspond to the entries of the normalized $\mathbf{a}$ and $\mathbf{b}$ vectors, respectively.

This may be achieved if $\bar{a}_i = \alpha\bar{b}_i$, i.e., $b_i = a_i/\alpha \ \forall \ 1 \leq i \leq m$, where $\alpha = \sigma_p/\sigma_c$. In other words, all of the PCA parameters for the original circuit and the RCP are identical, within a scaling factor of α. This could be achieved if the sensitivities of the delays of d_c and d_p to all process parameter variations are identical, within a fixed scaling factor.

The key issue here is that it is not essential for the delays d_c and d_p to be identical. In fact, the circuit delay and the RCP delay may have very different nominal values, since the nominal delays, μ_c or μ_p, never enter into Equation (4.5). All that matters is that the variations in the RCP should closely track those of the original circuit. This observation provides us with a significant amount of flexibility with respect to the critical path replica method, which attempts to exactly mimic all properties of the maximum delay of the original circuit, including the nominal delay.

4.3.3 Generating the Representative Critical Path

Next, we propose three methods for generating the RCP. The first is based on sizing gates on an arbitrarily chosen nominal critical path, while the second synthesizes the RCP from scratch using cells from the standard cell library, while the third is a combination of the two methods.

Method I: Critical Path Generation Based on Nominal Critical Path Sizing

As described in Section 4.1, the nominal critical path is usually not a good candidate to capture the worst case delay of the circuit over all reasonable parameter variations. However, there is intuitive appeal to the argument that variations along the nominal critical path have some relationship to the variations in the circuit. Based on this idea, our first approach begins with setting the RCP to a replica of the nominal critical path, and then modifies transistor sizes on this path so that the sized replica reflects, as far as possible, the variation of the delay of the manufactured circuit. The objective of this modification is to meet the criteria described in Section 4.2, in order to ensure that the RCP closely tracks the delay of the critical path in the manufactured circuit.

For an optimized circuit, it is very likely that there are multiple nominal critical (or near-critical) paths with similar worst-case delays at nominal parameter values. To make our approach as general as possible, we pick the one nominal critical path that has

the maximum worst-case delay at nominal process parameter values, even if its delay is only larger than a few other paths by a small margin. If there are multiple such paths, we arbitrarily pick one of them. We show in Section 4.4 that even with this relaxed initial choice, after the optimizations presented in this section, our method can produce very good results.

The problem can be formulated as a nonlinear programming problem as listed below:

$$\text{maximize} \quad \rho = \frac{\mathbf{a}^T \mathbf{b}(\mathbf{w})}{\sqrt{\mathbf{a}^T \mathbf{a} + \sigma^2_{R_c}} \sqrt{\mathbf{b}(\mathbf{w})^T \mathbf{b}(\mathbf{w}) + \sigma^2_{R_p}(\mathbf{w})}} \tag{4.7}$$

$$\text{s.t.} \quad \mathbf{w} \in \mathbf{Z}^n$$

$$\mathbf{w}_{min} \leq \mathbf{w} \leq \mathbf{w}_{max} \tag{4.8}$$

The objective function above is the correlation coefficient, ρ, between d_p and d_c, as defined by Equation (4.5), which depends on $\mathbf{a}$, $\mathbf{b}$, and $\sigma^2_{R_p}$. The values of the latter two quantities are both influenced by the transistor widths, which are allowed to take on any values between some user-specified minimum and maximum values.

Algorithm 2 illustrates our procedure for building the RCP under this approach. We begin with the nominal critical path of the circuit, chosen as described above, and replicate it to achieve an initial version of the RCP. Note that this is similar to the critical path replica method described in [25], and guarantees our method is at least as good as that approach. This critical path is then refined by iteratively sizing the gates on the path, using a greedy algorithm, in such a way that its correlation with the original circuit delay is maximized.

The first step of this approach involves performing conventional STA on the circuit to identify a nominal critical path, which is picked as the initial version of the RCP. Next, we perform SSTA on the circuit to obtain the PDF of the circuit delay variable, d_c, in the canonical form. This analysis provides us with the coefficients of the PCs in the circuit delay expression, namely, the vector $\mathbf{a}$ of Equation (4.3), as well as the independent term. We repeat this procedure for the initial RCP, to obtain the coefficients of the PCs in the expression for the delay, d_p, of the RCP. Based on these two canonical forms, we can compute the correlation coefficient, ρ^0, between the two delay expressions.

The iterative procedure updates the sizes of gates on the current RCP, using a TILOS-like criterion [70], with one modification: while TILOS will only upsize the gates, we also allow for the gates to be downsized. The rationale is that TILOS, for transistor

Algorithm 2 Variation-aware critical path generation based on sizing.

1: Perform deterministic STA on the original circuit and find the maximum delay path as the initial RCP. If there is more than one such path, arbitrarily pick any one.
2: Perform SSTA on the original circuit to find the PC coefficients corresponding to the vector **a** and the variance of the independent term.
3: Perform SSTA on the initial RCP to find its PC coefficients and the variance of its independent term. Calculate the correlation coefficient ρ^0 between the delay variables of the original circuit and the initial RCP.
4: $k = 1$
5: **while** (1) **do**
6: **for** each gate i on the critical path **do**
7: If not violating the maximum size constraint, bump up the size by multiplying it by a factor $F > 1$, keeping all other gate sizes unchanged from iteration $k-1$
8: Compute $\rho^k_{u,i}$, the correlation coefficient for this modified RCP with the original circuit. Change the size of the gate back to its size in iteration $k-1$.
9: If not violating the minimum size constraint, size down the gate by multiplying the size by the factor $1/F$, keeping all other gate sizes unchanged from iteration $k-1$
10: Compute $\rho^k_{d,i}$ as the correlation coefficient for the modified RCP by sizing gate i down. Change the size of the gate back to its size in iteration $k-1$.
11: **end for**
12: Choose j such that $\rho^k_{u,j}$ or $\rho^k_{d,j}$ is the largest among all correlation coefficients, and set ρ^k to be this correlation coefficient.
13: **if** $\rho^k > \rho^{(k-1)}$ **then**
14: Set the RCP to be the RCP from iteration $k-1$, except that the size of gate j is sized up or down by the factor F.
15: **else**
16: break
17: **end if**
18: **end while**

sizing, begins with the minimum-sized circuit; in contrast, our approach begins with the sized nominal critical path, with the intention that since this configuration lies within the solution space for the RCP, the final RCP is guaranteed to be no worse than the nominal critical path.

In the k^{th} iteration, we process each gate on the RCP one by one. As an example, for the gate i, we examine the case of multiplying its current size by a constant factor, F or $1/F$, to, respectively, up-size or down-size the gate, while leaving all other gate sizes identical to iteration $k-1$. We perform SSTA on this modified RCP to obtain the new coefficients for the PCs corresponding to this change, and calculate the new correlation coefficient, $\rho_{u,i}^{k}$ and $\rho_{d,i}^{k}$. We apply this procedure to all gates on the RCP during each iteration, and over all of these possibilities, we greedily choose to up-size or down-size the gate j whose size update provides the maximum improvement in the correlation coefficient. We then update the RCP by perturbing the size of the gate j, and set the value of ρ^k to the improved correlation coefficient. We repeat this procedure until no improvement in the correlation coefficient is possible, or until the sizes of gates in the RCP become too large.

We can save on the computation time by exploiting the fact that the RCP is a single path, and that SSTA on this path only involves sum operations and no max operations. When the size of a gate is changed, the delays of most gates on the critical path are left unchanged. Therefore, it is sufficient to only perform SSTA on the few gates and wires that are directly affected by the perturbation, instead of the entire path. However, we still must walk through the whole path to find the gate with the maximum improvement. If the number of gates of a nominal critical path is bounded by s, and the sizing procedure takes K iterations, then the run time of Algorithm 2 is $O(Ks)$.

The final RCP is built on-chip, and after manufacturing, its delay is measured. Using Equation (4.6) in Section 4.3.1, we may then predict the delay of the original circuit.

As mentioned at the beginning of this section, a significant advantage of this approach is that by choosing the nominal critical path as the starting point for the RCP, and refining the RCP iteratively to improve its correlation with the circuit delay, this approach is guaranteed to do no worse than one that uses the unmodified nominal critical path, e.g., in [25, 26, 27]. For a circuit that is dominated by a single critical path, this method is guaranteed to find that dominating path, e.g., the optimal solution.

The primary drawback of this method is also related to the fact that the starting point for the RCP is the nominal critical path. This fixes the structure of the path and the types of gates that are located on it, and this limits the flexibility of the solution. Our current solution inherits its transformations in each iteration from the TILOS algorithm, and changes the sizes of gates in the circuit. However, in principle, the idea could also be used to consider changes, in each iteration, not only to the sizes but also to the functionality of the gates on the RCP by choosing elements from a standard cell library, so that the delay of the modified RCP (with appropriately excited side-inputs) shows improved correlations with the circuit delay. Another possible enhancement could be to select the nominal critical path with the highest initial correlation coefficient with the circuit delay, instead of choosing this path arbitrarily. These extensions may be considered in future work, but Section 4.4 shows that even without them, our approach still produces good results.

Method II: Critical Path Generation Using Standard Cells

The second approach that we explore in this work builds the RCP from scratch, using cells from the standard cell library that is used to build the circuit. In principle, the problem of forming a path that optimally connects these cells together to ensure high correlation with d_c can be formulated as an integer nonlinear programming problem, where the number of variables corresponds to the number of library cells, and the objective function is the correlation between the statistical delay distribution, d_p, of an RCP with n stages of logic composed of these cells, where a stage is defined as a gate together with the interconnect that it drives, and d_c.

The integer nonlinear programming formulation is listed below:

$$\begin{aligned} \text{maximize} \quad & \rho = \frac{\mathbf{a}^T \mathbf{b}(\mathbf{N}_s)}{\sqrt{\mathbf{a}^T\mathbf{a}+\sigma_{R_c}^2}\sqrt{\mathbf{b}(\mathbf{N}_s)^T\mathbf{b}(\mathbf{N}_s)+\sigma_{R_p}^2(\mathbf{N}_s)}} \\ \text{s.t.} \quad & \mathbf{N}_s \in \mathbf{Z}^n \\ & \mathbf{e}^T\mathbf{N}_s \leq s \\ & \mathbf{b} = \Sigma_{i=1}^n N_{si}\mathbf{b}_i \\ & \sigma_{R_p}^2 = \Sigma_{i=1}^n N_{si}\sigma_{R_{pi}}^2 \end{aligned} \tag{4.9}$$

The objective function above is the correlation coefficient, ρ, between d_p and d_c, as

defined by Equation (4.5). The variable n represents the number of possibilities for each stage of the RCP, and the vector $\mathbf{N}_s = [N_{s1}, N_{s2}, \cdots, N_{sn}]^T$, where N_{si} is the number of occurrences of i in the RCP.

The first constraint states the obvious fact that each element of $\mathbf{N}_s$ must be one of the allowable possibilities. In the second constraint, $\mathbf{e} = [1, 1, \cdots, 1]^T$, so that the constraint performs the function of placing an upper bound on the total number of stages in the RCP. For the purposes of this computation, $\mathbf{a}$ and $\sigma^2_{R_c}$ come from the canonical form of the circuit delay, d_c, and are constant. The values of $\mathbf{b}$ and $\sigma^2_{R_p}$ are functions of $\mathbf{N}_s$, where the mapping corresponds to performing SSTA on the RCP to find the vector of PC coefficients $\mathbf{b}$ and the variance of the independent term R_p in the canonical form. The terms $\mathbf{b}_i, 1 \leq i \leq n$, are the PC coefficients corresponding to each stage of the RCP, and the R_{pi}s correspond to the independent terms, so that $\mathbf{b}$ and $\sigma^2_{R_p}$ are related to $\mathbf{N}_s$ through the last two constraints.

Since Equation (4.9) does not map on to any tractable problem that we are aware of, we propose an incremental greedy algorithm, described in Algorithm 3, which is simpler and more intuitive than any exact solution of the integer nonlinear program. While this algorithm is not provably optimal, it is practical in terms of its computational cost. We recall that the goal of our problem is to make the correlation coefficient between d_c and d_p as large as possible. The algorithm begins by performing SSTA on the original circuit to determine d_c.

Algorithm 3 Critical path generation using standard cells.

1: Initialize the RCP P to be the initial load INV.
2: Perform SSTA on the original circuit to find d_c in canonical form, and also compute the canonical form for the delay of each of the $p \times q$ choices for the current stage.
3: Calculate the load L^{k-1} presented by the $(k-1)$-stage RCP computed so far.
4: With L^{k-1} as the load, perform SSTA on the $p \times q$ choices for stage k.
5: Statistically add the canonical expressions for the delays of each of the $p \times q$ choices with the canonical form for the delay of the partial RCP computed so far, P. Calculate the correlation coefficient between the summed delays and the delay of the original circuit for each case.
6: Select the choice that produces the largest correlation coefficient as stage k in path P.
7: Go to Step 3.

During each iteration, the RCP is constructed stage by stage. If we have p types of standard gates, and q types of metal wires, then in each iteration we have $p \times q$ choices for the stage to be added. For an RCP with m stages, to find the optimal solution corresponds to a search space of $(p \times q)^m$. Instead, our method greedily chooses one of the $p \times q$ choices at each stage that maximizes the correlation of the partial RCP constructed so far with d_c, thereby substantially reducing the computation involved. In practice, we control the complexity even further by using the minimum driving strength gate of each functionality in the library, rather than considering all driving strengths for all gates.

The approach begins at the end of the critical path. We assume that the path drives a measurement device such as a flip-flop, and the part of the device that acts as a load for the critical path is an inverter INV . Therefore, for the first iteration, this inverter is taken as the load, and it corresponds to a known load for the previous stage, which will be added in the next iteration.

In iteration k, we consider appending each of the $p \times q$ choices to the partial RCP from iteration $k-1$, and perform SSTA for all of these choices to obtain the coefficients for the PCs, and the correlation with d_c, using Equation (4.5). The choice that produces the largest correlation coefficient is chosen to be added to the critical path. The load presented by this choice is then calculated for the next selection procedure, and the process is repeated. During the process of building the RCP, there may be cases where a wire on the RCP crosses the boundary between two correlation grids: if so, the current gate and the one it drives belong to two different grids, and the wire connecting them must be split into two parts to perform the SSTA. The maximum number of stages used in the RCP is a user-specified parameter. During our iterations, we keep a record of the correlation coefficient after adding each stage. Once all stages up to the maximum number are added, we find the maximum correlation coefficient saved and eliminate stages added beyond that point.

A complementary issue for this algorithm is related to determining the physical layout of each stage. To simplify the search space, we assume that the RCP moves monotonically: for example, the signal direction on all horizontal wires between stages must be the same, and the same is true of signal directions on all vertical wires. Because of symmetry of the spatial correlation profile and hence the PCA results, we only choose

the starting points to be from the bottom grids of the die. For a given starting point, the path would move towards the right and upper directions of the circuit.

It should be noted that systematic variations would affect the sensitivities of the parameter values, causing PC coefficients of identical cells at symmetric locations to become different. However, because systematic variations can be precharacterized before statistical analysis by a change of nominal values at different locations, we show in Section 4.4 that a reasonable disturbance of the nominal values would not significantly affect the final results. The procedure continues until the number of stages in the RCP reaches a prespecified maximum, or when the monotonic path reaches the end of the layout.

If the number of stages of the RCP is bounded by s and the number of starting points that we try for the RCP is ω, the runtime of Method II is $O(\omega pqs)$, because at each of the s stages, we have $p \times q$ choices. In comparison to Method I, if the bound of maximum number of stages for each method is comparable, then the comparison between K and $\omega \times p \times q$ determines which method has the longer asymptotic run time.

This approach has the advantage of not being tied to a specific critical path, and is likely to be particularly useful when the number of critical paths is large. However, for a circuit with one dominant critical path, this method may not be as successful as the first method, since it is not guided by that path in the first place.

Method III: Combination of the Two Methods

As stated above, each of the above two methods has its strengths and weaknesses. If the circuit is likely to be dominated by the nominal critical path, it is likely that Method I will outperform Method II; moreover, by construction, one can guarantee that Method I will do no worse than the critical path replica method. However, the structure of the RCP from Method I is also closely tied to that of the nominal critical path, limiting its ability to search the design space, and Method II provides improved flexibility in this respect, although it loses the guarantee of doing no worse than the critical path replica method.

We can combine the two methods discussed above in different ways to obtain potentially better results than either individual method. Many combinations are possible. For example, we could first build the RCP by sizing the nominal critical path using Method

I, and then add additional stages from the standard cell library to it using Method II. However, the number of stages may become too large in this case. The approach used in this paper combines the methods by first building the RCP from scratch using Method II, setting the size of each gate to be at its minimal value. Next, we update the sizes of the gates on this RCP using Method I to further improve the result. The procedure is listed as Algorithm 4 below for completeness.

Algorithm 4 Critical path generation using the combined method.

1: Build the initial RCP, P, using Method II. All gates are at their minimal size.
2: Perform the TILOS-like sizing of Method I on P

4.4 Experimental Results

We demonstrate the effectiveness of the approaches presented in this paper, and compare it with the *critical path replica (CPR)* approach, which represents the conventional approach to solving the problem. As in Chapter 3, our experiments are shown on the ISCAS89 benchmark suite and we use 90nm technology and the related constants are extracted from PTM model. The netlists are first sized using our implementation of TILOS: this ensures that the circuits are realistic and have a reasonable number of critical paths. The circuits are placed using Capo [71], and global routing is then performed to route all of the nets in the circuits. The cell library contains the following functionalities: NOT, 2-input NAND, 3-input NAND, 4-input NAND, 2-input NOR, 3-input NOR, and 4-input NOR.

Similar to Chapter 3, the variational model uses the hierarchical grid model in [6] to compute the covariance matrix for each spatially correlated parameter. The parameters that are considered as sources of variations include the effective channel length L, the transistor width W, the interconnect width W_{int}, the interconnect thickness T_{int} and the inter-layer dielectric H_{ILD}. The width W is the minimum width of every gate before the TILOS sizing. We use two layers of metal, and take the parameters for different layers of metal to be independent. The parameters are Gaussian-distributed, and their mean and 3σ values are shown in Table 4.1. As in many previous works on variational analysis, we assume that for each parameter, half of the variational contribution is assumed to

be from die-to-die (D2D) variations and half from within-die (WID) variations. We use *MinnSSTA* [8] to perform SSTA, in order to obtain the PCA coefficients for d_c. All programs are run on a Linux PC with a 2.0GHz CPU and 256MB memory.

Table 4.1: Parameters used in the experiments.

	L (nm)	W (nm)	W_{int} (nm)	T_{int} (nm)	H_{ILD} (nm)
μ	60.0	150.0	150.0	500.0	300.0
3σ	12.0	22.5	30.0	75.0	45.0

We first show the results of the algorithm that corresponds to Method I, described in Section 4.3.3, synthesizing the RCP by modifying a nominal critical path of the original circuit. The initial sizes of the gates are their sizes after timing optimization. We only show the results of the larger circuits, since these are more realistic, less likely to be dominated by a small number of critical paths, and are large enough to allow significant within-die (WID) variations. Of these, circuit s9234 is smaller than the others, and is divided into 16 spatial correlation grids, while all other circuits are divided into 256 grids.

In our implementation of Method I, we do not consider congestion issues. We assume both the CPR method and Method I replicate the nominal critical path, including the interconnects, to provide a fair comparison. In practice, Method I can route the replicated nominal critical path in the same way as any of the prior CPR methods reported in the prior literature.

We use a set of Monte Carlo simulations to evaluate the RCP. For each circuit being considered, we perform 10,000 Monte Carlo simulations, where each sample corresponds to a manufactured die. For each Monte Carlo sample, we compute the delay of the RCP, the delay of the original circuit, and the delay of the nominal critical path that may be used in a CPR method, as in [25, 26, 27]. The delay of the RCP is then used to compute the circuit delay using Equation (4.6) in Section 4.3.1, which corresponds to the mean of the conditional distribution in Equation (4.2).

Figure 4.1 illustrates the idea of the conditional PDF, described in Section 4.2, based on a sample of the Monte Carlo simulations for circuit s9234. The lower curve shown

with a solid line is the result of SSTA, and the circled points represent the conditional PDF obtained using Equation (4.2). The mean of this conditional distribution is indicated using a dashed line, and this is used as the estimate of the true circuit delay. The figure uses a solid vertical line to display the true circuit delay, and it can be seen that the two lines are very close (note that the plot does not start at the origin, and distance between these two lines is exaggerated in the figure).

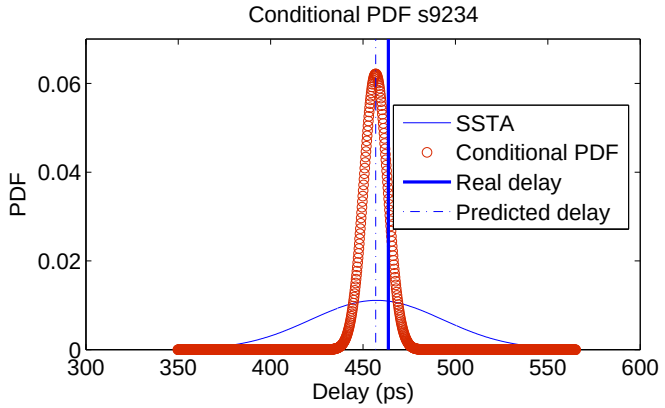


Figure 4.1: Conditional PDF of s9234.

Since the probability of the true circuit delay being beyond the 3σ value of the conditional PDF is very low, the smaller the conditional variance, the smaller the errors. It is worth recalling that the conditional variance is given by $\sigma_c^2(1-\rho^2)$, each term of which is a constant for a specific RCP. Therefore, this variance is exactly the same for each die (corresponding to each sample of the Monte Carlo simulation), and if the RCP heuristic maximizes ρ as intended, we minimize this variance.

In our experiments, we compare the computed circuit delay, called the predicted delay, d_{predic}, with the true circuit delay of the circuit, referred to as the true delay, d_{true}. We define the prediction error as

$$\frac{|d_{true} - d_{predic}|}{d_{true}} \times 100\%. \tag{4.10}$$

In order to maximize yield, we must add a *guard band* for the predicted delay values to ensure that the predictions are pessimistic. Therefore in this set results we also compare the guard band needed to make 99% of the delay predictions pessimistic for both Method I and the CPR method, respectively.

The results of the comparisons are presented in Table 4.2, where the rows are listed in increasing order of the size of the benchmark circuit. For Method I as well as the CPR Method, we show the average error and maximum error over all samples of the Monte Carlo simulation. All of the average errors of our approach are below 3% and both the average errors and maximum errors are significant improvements compared to the CPR method. The guard bands required by the two methods are listed in the last two columns. The guard band for Method I for each circuit is observed to be much smaller than the CPR method, and the advantage of Method I becomes particularly noticeable for the larger circuits.

Table 4.2: A comparison between Method I and the CPR Method.

Circuit	Average error		Maximum error		Guard band (ps)	
	Method I	CPR	Method I	CPR	Method I	CPR
s9234	1.58%	2.84%	10.50%	14.93%	28.5	43.7
s13207	0.52%	1.07%	5.67%	6.61%	18.3	26.9
s15850	1.00%	2.15%	7.70%	10.88%	36.6	57.6
s35932	2.35%	5.77%	12.53%	21.46%	33.2	58.9
s38584	1.79%	3.23%	11.44%	17.89%	43.8	72.3
s38417	2.77%	5.24%	13.87%	21.22%	53.5	84.1

Table 4.3: Conditional standard deviation, number of stages for RCP, and CPU time of Method I.

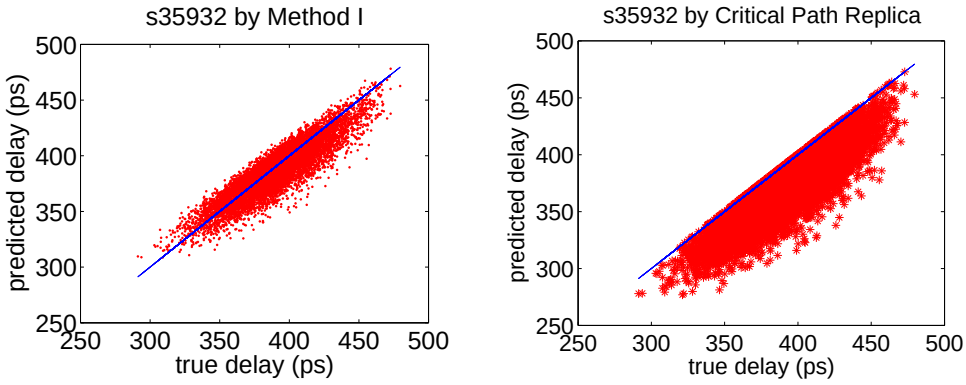
Circuit	Avg $\frac{\sigma_{cond}}{\mu_{cond}}$	Max $\frac{\sigma_{cond}}{\mu_{cond}}$	Number of stages	CPU time
s9234	1.40%	1.84%	67	1.46m
s13207	1.06%	1.41%	71	10.98m
s15850	1.30%	1.74%	96	20.92m
s35932	2.51%	3.18%	36	8.23m
s38584	2.11%	2.68%	66	13.95m
s38417	3.12%	3.95%	41	3.88m

The conditional variance derived in Section 4.2 defines the confidence of our estimate. Therefore we show the conditional standard deviation σ_{cond} as a percentage of the

conditional mean μ_{cond} in Table 4.3. Because μ_{cond} is different for each sample, we list the average and maximum $\frac{\sigma_{cond}}{\mu_{cond}}$ over all samples for each circuit. In order to provide more information about the RCP that we generate, we also show the number of stages for each RCP in the table. In this case, the number of stages for each RCP is the same as the nominal critical path for that circuit. The last column of the table shows the CPU time required by Method I for these benchmarks. The run time of Method I ranges from around one to twenty minutes. The conditional standard deviation is typically below 3% of the conditional mean on average.

For a visual interpretation of the performance of Method I, we draw scatter plots of the results for circuit s35932 in Figure 4.2(a) for Method I, and in Figure 4.2(b) for the CPR. The horizontal axis of both figures is the delay of the original circuit for a sample of the Monte Carlo simulation. The vertical axis of Figure 4.2(a) is the delay predicted by our method, while the vertical axis of Figure 4.2(b) is the delay of the nominal critical path, used by the CPR method. The ideal result is represented by the $(x = y)$ axis, shown using a solid line. It is easily seen that for the CPR method, the delay of the CPR is either equal to the true delay (when it is indeed the critical path of the Monte Carlo sample) or smaller (when another path becomes more critical, under manufacturing variations). On the other hand, for Method I, all points cluster closer to the $(x = y)$ line, an indicator that the method produces accurate results. The delay predicted by our approach can be larger or smaller than the circuit delay, but the errors are small. Note that neither Method I nor the CPR Method is guaranteed to be pessimistic, but such a consideration can be enforced by the addition of a guard band that corresponds to the largest error. Clearly, Method I can be seen to have the advantage of the smaller guard band in these experiments.

Our second set of experiments implements the algorithm corresponding to Method II, presented in Section 4.3.3. The maximum number of stages that we allow for the RCP for each circuit is 50, comparable to most nominal critical paths for the circuits in our benchmark suite. We use 7 standard cells at each stage, and 2 metal layers; therefore we have 14 choices for each stage. As in Method I, we do not consider congestion issues here and assume that the CPR method can perfectly replicate the nominal critical path. In practice, congestion considerations can be incorporated issues by assigning a penalty to congested areas when selecting wire directions. The setup of the Monte Carlo



(a) (b)

Figure 4.2: The scatter plot: (a) true circuit delay vs. predicted circuit delay by Method I and (b) true circuit delay vs. predicted circuit delay using the CPR method.

simulations is similar to the first set of experiments, and the corresponding errors and guard bands are shown in Table 4.4. Since this Monte Carlo simulation is conducted separately from that in Table 4.2, there are minor differences in the CPR error in these two tables, even though both tables use the same CPR as a basis for comparison. The average and maximum $\frac{\sigma_{cond}}{\mu_{cond}}$, the number of stages for each RCP, as well as the run times are shown in Table 4.5. The advantage of Method II, again, increases with the size of the circuit.

Table 4.4: A comparison between Method II and the CPR Method.

Circuit	Average error		Maximum error		Guard band (ps)	
	Method II	CPR	Method II	CPR	Method II	CPR
s9234	1.98%	2.84%	10.57%	15.15%	31.4	44.0
s13207	1.51%	1.06%	8.51%	7.22%	35.3	26.5
s15850	1.73%	2.14%	9.22%	10.97%	45.4	56.9
s35932	2.27%	5.80%	13.91%	21.34%	32.3	59.9
s38584	2.11%	3.29%	10.89%	17.12%	43.0	72.1
s38417	2.28%	5.27%	12.01%	22.88%	42.4	84.2

It is observed that for almost all cases, the average and maximum errors for Method

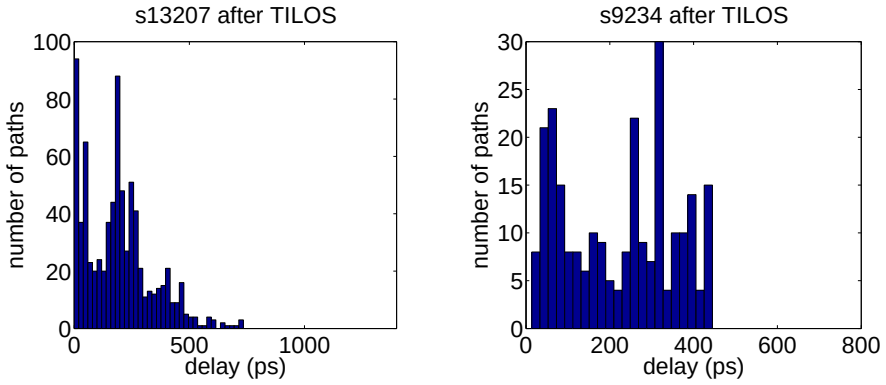
Table 4.5: Conditional standard deviation, number of stages for RCP, and CPU time of Method II.

Circuit	Avg $\frac{\sigma_{cond}}{\mu_{cond}}$	Max $\frac{\sigma_{cond}}{\mu_{cond}}$	Number of stages	CPU time
s9234	2.18%	2.79%	49	0.1s
s13207	1.75%	2.31%	30	15.7s
s15850	1.88%	2.45%	50	15.1s
s35932	2.19%	2.81%	50	16.7s
s38584	2.14%	2.73%	50	18.6s
s38417	2.13%	2.77%	50	15.5s

II are better than those for the CPR method. An exception to this is circuit s13207, which is dominated by a small number of critical paths, even after sizing using TILOS. We illustrate this using the path delay histogram in Figure 4.3(a), which aggregates the delays of paths in the sized circuit into bins, and shows the number of paths that fall into each bin. In this case, it is easily seen that the number of near-critical paths is small. In contrast, Figure 4.3(b) shows the same kind of histogram for circuit s9234, which is a more typical representative among the remaining benchmarks: in this case it is seen that a much larger number of paths is near-critical, and likely to become critical in the manufactured circuit, due to the presence of variations.

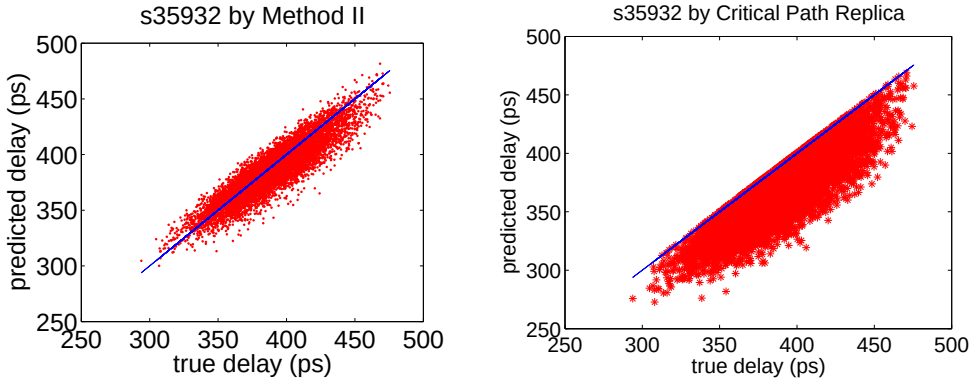
Under the scenario where the number of near-critical paths is small, it is not surprising that Method II does not perform as well as a CPR. First, as pointed out in Section 4.3.3, Method II does not take advantage of any information about the structure of the original circuit, and is handicapped in such a case. Moreover, the unsized circuit s13207 was strongly dominated by a single critical path before TILOS sizing; after sizing, the optimized near-critical paths are relatively insensitive to parameter variations, meaning even if one of these becomes more critical than the nominal critical path on a manufactured die, it is likely to have more or less the same delay.

We also show scatter plots for both Method II and CPR in this case, in Figure 4.4(a) and Figure 4.4(b), respectively. The figures are very similar in nature to those for Method I, and similar conclusions can be drawn. In comparing Methods I and II by examining the numbers in Tables 4.2 and 4.4, it appears that there is no clear winner,



(a) (b)

Figure 4.3: Histograms of path delays of (a) s13207 and (b) s9234 after TILOS optimization.



(a) (b)

Figure 4.4: The scatter plot: (a) true circuit delay vs. predicted delay by Method II and (b) true circuit delay vs. predicted delay using the CPR method.

though Method II seems to show an advantage for the largest circuits, s35932 and s38417. With our limited number of choices for each stage of the RCP, referring to discussions about run time in Section 4.3.3, it is not surprising that Method II is faster in terms of CPU time, as shown in Table 4.5. The algorithm finishes within a few seconds for all of the benchmark circuits.

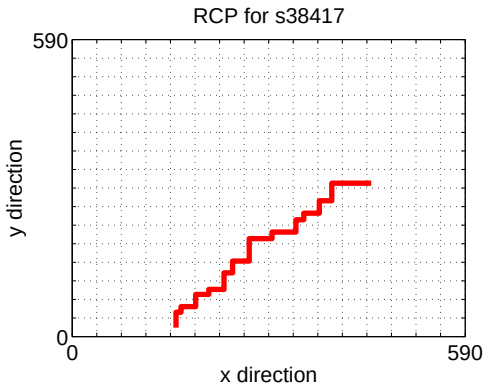


Figure 4.5: The RCP created by Method II for circuit s38417.

Next we show the location of the RCP built for circuit s38417 using Method II on the chip in Figure 4.5. The figure shows the die for the circuit. The size of the die is determined by our placement and routing procedure, and the dashed lines indicate the spatial correlation grids. The solid bold lines are the wires of the critical path. The figure shows that the critical path grows in a monotonic direction and it starts from one of the grids at the bottom of the chip, both due to the layout heuristics discussed in Section 4.3.3.

In order to gain more insight into the trend of improvement of the correlation coefficients, Figure 4.6 shows the correlation coefficient of Method II after each stage is added for one starting point: beyond a certain number of iterations, the marginal improvement flattens out. A similar trend is seen for Method I.

Our third set of experiments show the results of Method III, which is a combination of Method I and Method II. We first build an RCP from scratch using Method II, and then refine this RCP by the iterative sizing technique employed by Method I. Considering that Method II uses minimum-sized standard cells to restrict its search space, in the combined method, we allow the cells to be sized up. In Table 4.6, we compare the

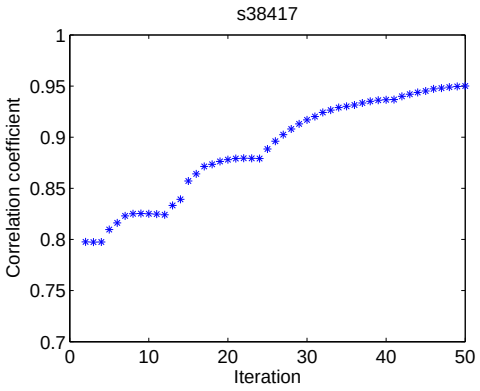


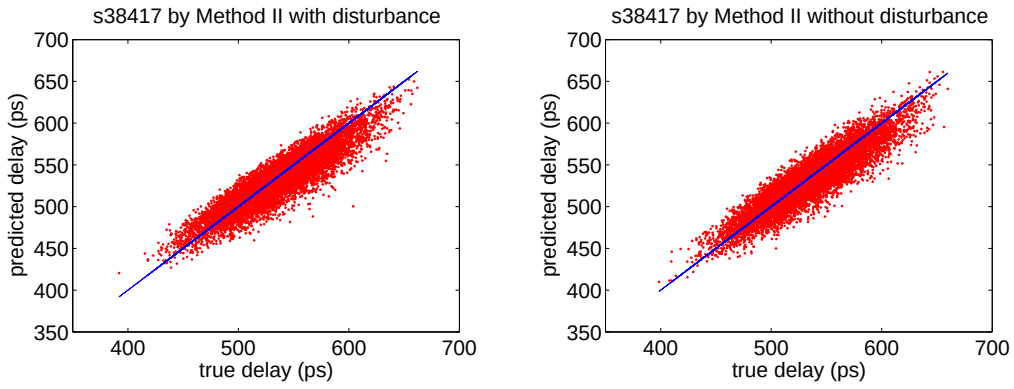
Figure 4.6: Trend of correlation coefficient after each iteration.

correlation coefficients of the delay of the built RCP using Method II and using Method III. The correlation coefficients are calculated using Equation (4.5) and indicate of how closely the RCP can track the original circuit delay. The average and maximum errors, the guard band needed to ensure 99% of the predictions pessimistic, as well as the run time of the combined method are also listed in the table. It is observed that the sizing does indeed improve the results of Method II.

Table 4.6: Results of the Method III and its comparison with Method II.

Circuit	ρ		Method III			
	Method II	Method III	Avg. error	Max. error	Guard band	CPU time
s9234	0.9732	0.9821	1.68%	7.64%	28.9ps	2.44s
s13207	0.9719	0.9750	1.38%	5.55%	31.2ps	16.97s
s15850	0.9613	0.9737	1.46%	7.2%	39.5ps	22.52s
s35932	0.9464	0.9492	2.26%	11.98%	34.1ps	20.38s
s38584	0.9493	0.9609	1.89%	9.13%	43.0ps	28.84s
s38417	0.9505	0.9526	2.17%	9.12%	37.4ps	15.06s

Finally, we experimentally demonstrate that our assumption of neglecting systematic variations is reasonable. We demonstrate this on Method II, and show that a reasonable change in the nominal parameter values of the RCP cells due to systematic variations would not affect the final results by much. This justifies our heuristic to only choose



(a) (b)

Figure 4.7: Scatter plots of s38417 with and without nominal value disturbance for the RCP, to model systematic variations.

the starting point of the RCP at the bottom of the die.

The experiment proceeds as follows: after the RCP is built, we disturb the nominal values of all parameters associated with the RCP by 20%, while leaving those of the original circuit unperturbed. This models the effect of systematic variations, where the RCP parameters differ from those of the original circuit. We show the final results of the scatter plots for circuit s38417, with and without disturbance, in Figures 4.7(a) and 4.7(b), respectively. It is shown that the plots are almost identical, and the average error is 2.26% with disturbance as compared to 2.28% for the normal case.

The intuition for this can be understood as follows. The correlation between the original circuit and the RCP depends on the coefficients of the PCs in the canonical expression. The coefficients depend on the sensitivities of the delay to variations, and not on their nominal values. Although the delay is perturbed by 20%, the corresponding change in the delay sensitivity is much lower, and this leads to the small change in the accuracy of the results.

4.5 Conclusion

In this chapter, we have presented two novel techniques to automatically generate a critical path for the circuit to capture all of the parameter variations. A third approach is a simple hybrid of the two approaches that provides noticeable improvements over either individual approach. The key idea used in this paper is to take advantage of spatial correlations to build a test structure, the RCP, that captures variations in multiple critical paths in the circuit, exploiting the spatial correlation structure. Experimental results have shown that our methods produce good results.

Our current framework only handles process variations; environmental variations are addressed by adding adequate delay margins, since these are often worst-cased in practice. Addressing these through prediction remains an open area for future work. A straightforward extension of our method for very large circuits is to build not one, but a small number, of RCPs, one for each region of the circuit. The fundamental approach for building each path would be identical to the method proposed here.

Chapter 5

Statistical Analysis of Memory Systems

SRAM arrays occupy a substantial portion of the chip area in modern microprocessors, and face a number of read and write stability challenges as feature sizes shrink. Embedded DRAMs (eDRAMs) present a promising and compact memory alternative to SRAMs. However, their performance is susceptible to process variations, and accurate statistical analysis of the eDRAM cell voltage is critical to ensure correct performance. This work first develops analytical models for all leakage components that impact eDRAM performance. Next, these models are fed into two-step statistical analysis procedure. In the first step, a fast evaluation technique computes the PDF of the distribution up to the beginning of the tail, evaluated at the 99% point. Since memory designers are usually more interested in the far tails of the distribution, in the second step, extreme value theory is employed to find the distribution deeper into the tail. This method is demonstrated to predict the 99.995% distribution point with average error of 2.67% and more than 180× speedup compared to state-of-the-art methods.

5.1 Introduction

Though embedded DRAMs present a promising alternative to replace conventional six-transistor (6T) SRAM cells for on-die caches, the presilicon analysis of these cells is a critical problem, since the cell voltage must be refreshed after a certain time to avoid read

errors. The analysis problem is compounded under process variations, due to which the cell voltage for each eDRAM cell at a particular time is a random variable, instead of a specific number, across the memory system. Therefore, accurate and efficient statistical analysis of the cell voltage is very essential to ensure proper operation of an eDRAM.

In this chapter, we develop a new framework for the statistical analysis of 3T eDRAM cells, which is applicable to a wider range of DRAM designs, such as 2T eDRAM [41] and 1T1C traditional DRAM [72] cells. Our contribution is in formulating the problem and developing appropriate modeling approaches. We analyze the effects of process variations in eDRAMs and address the problem of statistical memory analysis by applying algorithmic techniques from the field of statistical static timing analysis of logic circuits as well as theory of extreme statistics. We focus our attention on the analysis of a single eDRAM cell: while this cell is relatively small, due to the high repetitive nature of memory systems, designers are extremely interested in the tail of the distribution of the statistics for its performance. Obtaining this tail involves a substantially large number of Monte Carlo simulations, which can be computationally intensive. This chapter tackles the problems by employing analytical models to significantly reduce the run time, while maintaining the quality of the result.

The rest of the chapter is organized as follows. Section 5.2 outlines the basics of gain cell eDRAM operation, followed by a problem formulation in Section 5.3. Next, Section 5.4 describes our models for the leakage currents and the cell voltage at a given time, as functions of the process parameters. Our first approach, using a moment-matching-based method to evaluate the PDF/CDF of the cell voltage with our model, is then illustrated in Section 5.5, Section 5.6 illustrates how we specifically evaluate the tail of the PDF/CDF. The experimental results are presented in Section 5.7, and Section 5.8 concludes this chapter.

5.2 3T Embedded DRAM Operation

Figure 5.1 illustrates an implementation of an eDRAM gain cell using three PMOS[1] transistors. In the gain cell, PS is the cell storage transistor, PR is the read access

[1] Currently, PMOS devices are chosen over NMOS because they have significantly smaller gate leakage current. This preference may not hold in the future, as high-k dielectric gates become more prevalent, but the fundamental techniques in this work are applicable even to cells with NMOS devices.

transistor and PW is the write access transistor. Except for the selected cell to be written, all write word lines (WWLs) are normally biased with negative V_{GS} to prevent large subthreshold currents from flowing from the write bit line (WBL) to the storage node, corrupting the stored data.

The read and write operations proceed as follows.

- During *read mode*, the read bit line (RBL) is predischarged to logic-0 and the read word line (RWL) is activated by setting it to logic-0. A data-0 at the storage node would charge RBL, while data-1 would leave the RBL voltage unchanged.

- During *write mode*, WWL is set to a small negative voltage[2] to activate PW. This negative voltage is required to write data-0 through PW to the cell storage node, to ensure that the node is fully discharged (a zero voltage would result in a V_{th} drop across the transistor)

- Similarly, when the line WWL is set high, its voltage level is set to $V_{DD} + V_{bst}$, which ensures that if WBL is high due to a write to a different cell on the same WBL, the subthreshold leakage for the unselected cell remains negligible.

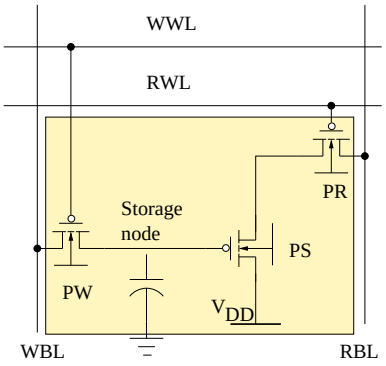


Figure 5.1: A 3T PMOS eDRAM gain cell.

In this work, we are interested in studying the data hold mode of the eDRAM operation to ensure that the data stored is not corrupted. During this mode, the storage node is left floating: due to this, the cell is susceptible to leakage currents that degrade

[2] This is analogous to the idea of applying a small boost over V_{DD} to the access transistor in a conventional 1T1C DRAM, where the boost is required to ensure that data-1 is correctly written.

the cell voltage over time. Therefore, the eDRAM requires periodic refreshing to ensure correct operation. Furthermore, on-die cache memory is accessed more frequently than off-die main memory, which places the constraint that the refresh time must be shorter. Given this context, the concept of signal retention time is a critical issue in eDRAM design.

For the implementation of Figure 5.1, we consider two cases: when the data stored in the cell is a logic 0 or a logic 1. When a data-0 is being held, the worst case corresponds to the scenario when cell storage nodes are surrounded by high voltage nodes, i.e., WWL, WBL, RWL, and RBL are all at logic-1. This creates leakage paths from the storage node to all of these nodes.

For the data-1 case , the worst case setting, with all of these lines at logic-0, is not realizable: the WWL signal must be at logic-1 for the retention case (else the cell is being written into), and the RWL and RBL lines are set to zero only while reading the cell, and every read is followed by a writeback step. In addition, if WBL is at logic 0, then the gate voltage V_G is significantly greater than the source voltage V_S for the PW transistor, and this limits the subthreshold voltage. Moreover, when the cell is in hold mode with data-1, the subthreshold leakage current flows from the storage node to WBL, but there is also incoming current from the WWL line and the V_{DD} line on the storage transistor. In equilibrium, these junction band-to-band leakage and gate overlap leakage components are equal to the negative-biased subthreshold leakage after some signal loss: this value is typically sufficiently close to the logic-1 level. When the cell stores data-0 in hold mode, the leakage sources include subthreshold voltage, junction, gate, and gate overlap leakages, all of which contribute to degrading the cell signal. The net result is that data-1 retention is generally not an issue in these cells, and the data retention time of data-0 much shorter than that of data-1.

The definition of the retention time, t_{RET}, is illustrated in Figure 5.2. To operate correctly, the eDRAM must satisfy two conditions:

1. The voltage difference between the data-1 cell voltage, V_{D1}, and the data-0 cell voltage, V_{D0}, must be greater than a specification, Δ, i.e.,

$$V_{D1} - V_{D0} > \Delta \tag{5.1}$$

   This ensures that the difference between the voltage levels is large enough so that

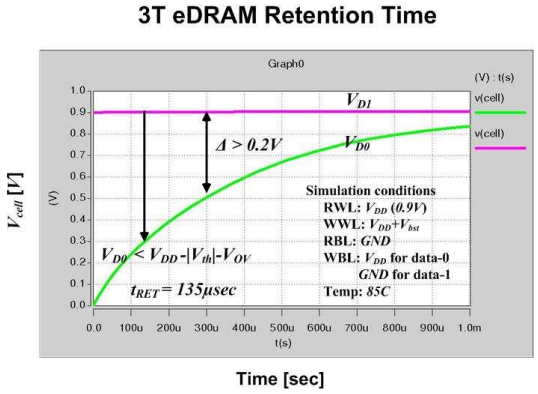


Figure 5.2: Retention Time for a 3T eDRAM.

the sense amplifier can reliably distinguish between the 1 and 0 data.

2. Voltage V_{D0} should be low enough to turn on the storage transistor PS and meet the read speed target, i.e.,

$$V_{D0} < V_{DD} - |V_{th}| - V_{OV}, \tag{5.2}$$

where V_{OV} is a safety margin and V_{th} is the threshold voltage.

The retention time is defined as the time for which these conditions are certain to hold. In the example shown in Figure 5.2, where $V_{D1} = V_{DD} = 0.9V$, $V_{th} = V_{OV} = 0.3V$, the second constraint kicks in before the first, and the retention time is found to be $135\mu s$.

These two requirements could be applied to various types of eDRAM cells when determining their retention time. The voltage margins for $V_{D1} - V_{D0}$ and V_{D0} mentioned above are specific to the eDRAM circuit, as the memory size, sense amplifier type, and read reference schemes play a significant role in overall chip performance.

As supported by silicon measurements in [73], for the 3T PMOS eDRAM cell to work reliably at $V_{DD} = 0.9$V, the corresponding requirements are $\Delta = 0.2$V and $V_{OV} = 0.3$V. In general, a CAD tool can reasonably expect to obtain constraints such as these from a designer. For the specific 3T PMOS cell used in this work, as observed earlier, V_{D1} is close to V_{DD} and is relatively constant over time, while V_{D0} experiences significant change. As a result, for this design, the two conditions discussed above become $V_{D0} < V_{D1} - 0.2$V and $V_{D0} < V_{DD} - |V_{th}| - 0.3$V. We see that in this case the second condition implies the first. Therefore the cell voltage level of V_{D0} is important.

5.3 Problem Formulation

Process variations further complicate the issue of leakage and data retention. Under these variations, the cell voltage of each eDRAM cell at a given time is not a specific number, but is at a certain value only with a certain probability. In other words, the cells within an eDRAM array experience different levels of variations, and consequently, some experience more serious signal loss than others.

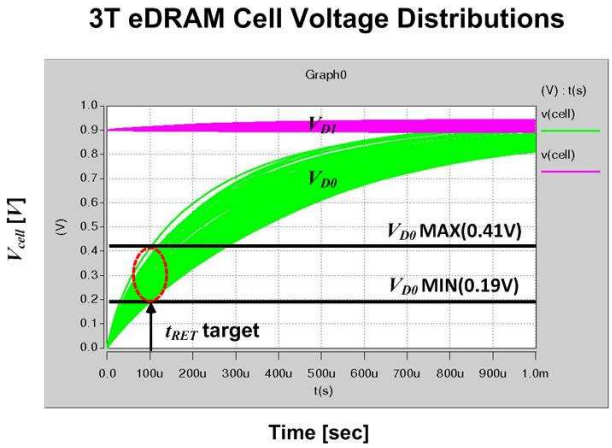


Figure 5.3: Variation of cell voltage for a 3T eDRAM.

Figure 5.3 shows an example of cell voltage variations, obtained by Monte Carlo simulations using SPICE. It is observed that the cell voltage at $t_{RET} = 100\mu$s can range from around $0.19V$ to $0.41V$, which is a large range. It also illustrates that the data-0 hold mode is much more serious than the data-1 hold mode. Since a designer must ensure, to a very high likelihood, that all cells are working correctly, it is essential to accurately estimate the cell voltage in the worst case. As an aid, it is useful to provide the full probability density function (PDF) or cumulative density function (CDF) of the worst-case cell voltage across all cells of the memory array at any given time. However, because this process is intended as an inner loop for the design procedure, Monte Carlo simulation is usually too slow, and it is important to build efficient analysis techniques.

The goal of this work is to develop statistical analysis techniques to solve this problem. Statistical techniques have been widely employed to analyze timing and power of

logic circuits in recent years [8, 11, 21]. However, these methods cannot be directly extended to memories, since the metrics for memories are entirely different from those for logic circuits. For memory systems, with the above motivation, it is useful to perform an analysis at the level of a single cell, to capture the statistical variations of its parameters. Since these cells are relatively small (3T in this case), it is possible to develop accurate SPICE-based models to facilitate accurate analysis. Then we can apply the result to a large memory array with repeated cells.

In spite of its importance, the general area of statistical analysis of memory cells has not attracted commensurate research effort. Work in this area is largely restricted to a few papers that study SRAMs. In [44], the failure probabilities of read and write operations are analyzed. Voltage scaling characteristics of different near-threshold SRAM cells are simulated in [45], based on yield estimation using importance sampling. In [46], the authors use fast conditional importance sampling to select repair elements at beginning-of-life (BOL) to improve the end-of-life (EOL) functionality of SRAM under negative bias temperature instability (NBTI) effects. A lower-bound for variability-related yield in SRAM is developed in [47] using the concept of maxima in extreme value theory, as discussed in Section 5.6. In [48], a modified Monte Carlo based approach called statistical blockade was proposed to analyze the statistical behavior of SRAM. To the best of our knowledge, techniques introduced in this chapter is the first work to develop such a statistical analysis tool for eDRAMs, where the metrics and cell voltage degradation mechanisms are very different from SRAMs. Our approach is also the first one to use analytical methods to evaluate the extreme tail of a distribution and significantly reduces runtime as compared to Monte Carlo based approaches.

Our work begins by examining the root causes of cell voltage degradation, and develop accurate models for all leakage components. From this analysis, we develop a closed-form formula for the cell voltage, at any given time point, as a function of the process parameters. Next, we employ a moment-based PDF/CDF evaluation technique based on APEX [19] to obtain the final PDF/CDF. We demonstrate that the accuracy of our method is comparable to Monte Carlo simulations using SPICE yet we save a substantial amount of run time. We consider this to be a preprocessing step.

For a large memory system, the tail of the distribution is of particular importance for designers to ensure acceptable yield. To achieve better accuracy, we borrow ideas from

the extreme value theory (EVT) to specifically deal with the tail of the distribution in order to obtain both accurate and efficient predictions in this portion of the distribution. The tail of a distribution obtained by a moment-based method such as APEX is known to have limited accuracy; however, we find that it is a reasonable starting point for identifying the beginning of the tail, which is then handled using extreme value theory.

5.4 Modeling eDRAM Behavior

5.4.1 Accurate Leakage Modeling

As stated in Section 5.2, for the eDRAM cell of Figure 5.1, the worst case scenario corresponds to a data-0 held at the storage node, surrounded by all other nodes at logic-1. The sources of leakage in this mode include the gate leakage I_{gate} of the storage transistor PS, the junction band-to-band leakage I_{junc} from the body of the write access transistor PW to the storage node, and the gate overlap leakage I_{gov} of PW. All of these leakage sources are functions of process parameters. For example, it is well known that the gate leakage and the gate overlap leakage each has a strong exponential dependency on oxide thickness, T_{ox}; junction band-to-band leakage is related to the transistor doping concentration N_A; and subthreshold leakage is a function of the threshold voltage V_{th}.

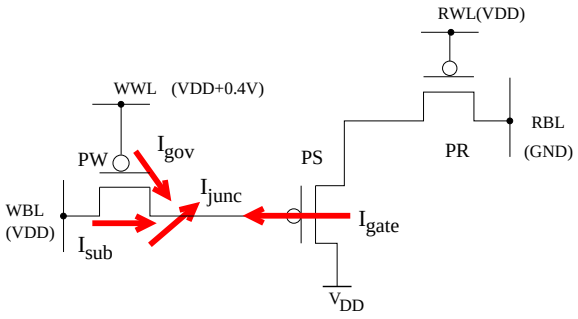


Figure 5.4: Leakage components for an eDRAM cell.

These leakage currents collectively determine the cell voltage, V_{cell}, but on the other hand, there is a cyclic dependency as V_{cell} also determines the amount of leakage due to each of these sources. Therefore, we can write an implicit equation for V_{cell}, based

on models for each of these leakage components, expressed as functions of process parameters.

The subthreshold leakage current I_{sub} of a transistor under weak inversion can be written as [74, 75]:

$$I_{sub} = K_1 V_t^2 \frac{W}{L} e^{\frac{(|V_{GS}|-V_{th})}{\eta V_t}} \left(1 - e^{-\frac{|V_{DS}|}{V_t}}\right), \tag{5.3}$$

where V_t is the thermal voltage The value of K_1 can be viewed as a constant for a given technology.

In general, the gate leakage I_{gate} is a complex nonlinear function of the process parameters and voltages. We use a simplified equation from [76] to approximate I_{gate} as a function of W, T_{ox} and V_{GS}:

$$I_{gate} = K_2 W \left(\frac{V_{GS}}{T_{ox}}\right)^2 e^{-\frac{\alpha T_{ox}}{|V_{GS}|}} \tag{5.4}$$

where K_2 and α are experimentally fitted constants based on SPICE simulations. We can further simplify this equation into a quadratic function of V_{cell}. Using a Taylor series expansion, we obtain:

$$I_{gate} \approx K_2 W e^{-\beta T_{ox}} \left(\frac{V_{GS}^2}{T_{ox}^2} - \frac{\alpha}{T_{ox}}|V_{GS}| + \frac{\alpha^2}{2} + m_0\right) \tag{5.5}$$

where β is independent of V_{GS}, and m_0 is a constant correction term.

We use a similar process to approximate the gate overlap leakage of the write access transistor PW.

The junction leakage can be approximated by the equation below:

$$I_{junc} = W|y_2 - y_1| \frac{AEV_{app}}{\Sigma_g^{1/2}} \exp\left(-\frac{B\Sigma_g^{3/2}}{E}\right) C \tag{5.6}$$

where $C = (1 - \delta_g V_G)\,(1 - \lambda\,(V_{app}))$ is a correction term, and $\lambda(V_{app})$ is a fitting function. The value of $W|y_2 - y_1|$ is the effective area of the junction, and A, B, δ are all experimentally derived constants. A detailed explanation and the meanings of the other terms are provided in [77].

5.4.2 A New Model for V_{cell}

The voltages included in the equations in Section 5.4.1 are all related to the cell voltage, V_{cell}. Computing V_{cell} based on the above approaches is unnecessary when we address the problem of modeling variations about a nominal design point. Using a first order Taylor expansion, they can be simplified into

$$\begin{aligned} I_{sub} &= a_1 V_{cell} + a_2 \\ I_{gate} &= b_1 V_{cell}^2 + b_2 V_{cell} + b_3 \\ I_{gov} &= c_1 V_{cell}^2 + c_2 V_{cell} + c_3 \\ I_{junc} &= d_1 V_{cell}^2 + d_2 V_{cell} + d_3. \end{aligned} \tag{5.7}$$

The total leakage current is equal to the sum of I_{sub}, I_{gate}, I_{gov} and I_{junc}. We denote the storage node capacitance by C_S, which consists mainly of the gate capacitance of the storage transistor. By adding the three leakage components together, we obtain:

$$I_{sub} + I_{gate} + I_{gov} + I_{junc} = -C_S \frac{dV_{cell}}{dt}. \tag{5.8}$$

If we define:

$$\begin{aligned} P_1 &= b_1 + c_1 + d_1 \\ P_2 &= a_1 + b_2 + c_2 + d_2 \\ P_3 &= a_2 + b_3 + c_3 + d_3, \end{aligned} \tag{5.9}$$

then we have

$$P_1 V_{cell}^2 + P_2 V_{cell} + P_3 = -C_S \frac{dV_{cell}}{dt} \tag{5.10}$$

where P_1, P_2 and P_3 are all *functions of the process parameters*, which are random variables. To find the value of V_{cell} at a given time t_0, we have:

$$-\int_0^{V_{cell}} \frac{C_S}{P_1 V_{cell}^2 + P_2 V_{cell} + P_3} dV_{cell} = \int_0^{t_0} dt. \tag{5.11}$$

Solving this for V_{cell}, we obtain:

$$V_{cell} = \frac{2P_3 \tan\left(\frac{t_0}{2C_S}\sqrt{4P_1P_3 - P_2^2}\right)}{\sqrt{4P_1P_3 - P_2^2} - P_2 \tan\left(\frac{t_0}{2C_S}\sqrt{4P_1P_3 - P_2^2}\right)}. \tag{5.12}$$

Equation (5.12) can be easily modified to be in the style of

$$2\arctan\left(\frac{V_{cell}\sqrt{4P_1P_3-P_2^2}}{2P_3+P_2V_{cell}}\right)C_S=\sqrt{4P_1P_3-P_2^2}t_0 \tag{5.13}$$

from which we can fit C_S using a small set of Monte Carlo simulations. Under process variations, the value of V_{cell} calculated by Equation (5.12) is a function of the underlying process parameters, which are random variables with certain distributions. The PDF of V_{cell} cannot be easily computed in closed-form using this equation. Therefore, we derive the second order Taylor expansion of Equation (5.12) with respect to the process variations at their nominal values.

5.5 Statistical Analysis of V_{cell} using Moment-Based Methods

We assume that all process parameters are uncorrelated Gaussian-distributed variables. Correlated Gaussian variables, on the other hand, can also be expressed in terms of uncorrelated variables by a linear transformation step called principal component analysis (PCA), as is shown in [8]. In such a case, it is trivial to show [19] that the linearly transformed variables can be substituted in the quadratic leakage power expressions to obtain another quadratic in the independent variables. Therefore, this assumption is general enough to take into account all Gaussian-distributed parameter variations.

All of the variables representing process parameters are placed in a vector $\mathbf{Y}=[Y_1,Y_2,\cdots,Y_N]^T$, and the vector containing their corresponding nominal values is $\mu=[\mu_1,\mu_2,\cdots,\mu_N]^T$. Defining $\Delta Y_i=(Y_i-\mu_i)/\sigma_i$, where σ_i is the standard deviation of the i^{th} parameter, and $\Delta Y=[\Delta Y_1,\Delta Y_2,\cdots,\Delta Y_N]^T$, it is easily seen that $\Delta Y\sim N(\mathbf{0},\mathbf{I})$. Using a Taylor expansion, we obtain:

$$V_{cell}=V_{cell,0}+\mathbf{U}^T\mathbf{\Delta Y}+\mathbf{\Delta Y}^T\mathbf{V}\mathbf{\Delta Y} \tag{5.14}$$

where $V_{cell,0}$ is the value of V_{cell} calculated at nominal parameter values, $\mathbf{U}$ is a vector containing the first-order Taylor series coefficients, scaled by their standard deviations, and $\mathbf{V}$ is a matrix containing the scaled second-order Taylor series coefficients.

The PDF of Equation (5.14) can be approximated using the moment-matching method, APEX, as [19] introduced in Section 2.4.3. As is discussed in Chapter 2,

the APEX technique approximates the PDF of Equation (5.14) by an impulse response of order M:

$$h(t) = \begin{cases} \Sigma_{i=1}^{M} r_i e^{p_i t} & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{5.15}$$

The unknowns in Equation (5.15), called *residues* r_i and *poles* p_i, can then be computed by matching the *time moments* of Equation (5.15) and the time moments of Equation (5.14). Then the approximate PDF of V_{cell}, written as $h(t)$, as well as the corresponding CDF $s(t)$, can be easily obtained.

The overall statistical analysis procedure for finding the body of the distribution of the cell voltage of eDRAM cells at a given time, t_0, can be summarized in Algorithm 5 below. Typically, we would apply this algorithm using $t_0 = t_{RET}$, the target retention time of the cell.

Algorithm 5 Statistical cell voltage analysis for eDRAM.

1: Characterize constants that are specific to the provided technology and that are independent of the process parameters considered as variation sources, such as K_2, α, β for the gate leakage, and K_1 for the subthreshold leakage.
2: Characterize the capacitance C_S using a small number of Monte Carlo simulations. Substitute the nominal values of process parameters, and calculate all Taylor coefficients required for Equation (5.14).
3: Use APEX to evaluate the PDF/CDF of V_{cell}.
4: If the design is changed, go to Step 2. Otherwise Stop.

5.6 Statistical Analysis of V_{cell} using Extreme Value Theory

For large memory systems, to ensure acceptable yield, designers are more concerned with extreme cases. Therefore, an accurate estimation of the tail of the distribution for the cell voltage is important. Besides, it is known that AWE-based methods do not perform well for the far tail of the distribution. An area in the statistics domain called extreme value theory (EVT) [78] suggests that the tail of any kind of distribution will converge to a certain kind of distribution beyond a large enough threshold. In the VLSI

design context, these approaches have been used for maximum power estimation in [49] and for SRAM analysis in [47]. However, most of these approaches are either Monte Carlo-based or numerically-based and require long times to simulate for points in the tail of the distribution.

5.6.1 The Basics of EVT

Extreme value theory is based on order statistics. Consider a random variable X with cumulative distribution function (CDF) $F(x)$. If we draw n samples from its population, namely, $\mathbf{X} = [X_1, \cdots, X_n]$, and rearrange them in nondecreasing order as $X_{1,n} \leq X_{2,n} \leq \cdots \leq X_{n,n}$, where $\{X_{1,n}, \cdots, X_{n,n}\}$ is a permutation of $\{X_1, \cdots, X_n\}$, then the kth term $X_{k,n}$ of this ordered sequence is called the kth *order statistic*.

We now introduce two concepts corresponding to different extreme behaviors, the *maxima* and the *exceedance*. If we obtain p sets of n samples, $\mathbf{X}^1, \cdots, \mathbf{X}^p$, and pick the maximum unit in each sample (i.e., the nth order statistic in each of the p samples), we obtain a new sample set

$$\mathbf{X}_m = \{X^i_{n,n}, 1 \leq i \leq p\} \tag{5.16}$$

This sample of the p maxima is one example of an extreme order distribution.

A second and separate notion is that of the exceedance. Given a sample of size n, $\mathbf{X} = [X_1, \cdots, X_n]$, this corresponds to the r largest elements of the sample, i.e.,

$$\mathbf{X}_e = \{X_{n-r+1,n}, \cdots, X_{n,n}\} \tag{5.17}$$

The concept of exceedance provides a natural mechanism to model the tail of a distribution. The CDF of the exceedance $F_t(x)$ is a conditional distribution because we are considering the distribution only over a predetermined threshold t. This conditional CDF can be written as

$$\begin{aligned} F_{condi}(x) &= P\left(X \leq x | X > t\right) = \frac{P\left(X \leq x, X > t\right)}{P\left(X > t\right)} \\ &= \frac{F(x) - F(t)}{1 - F(t)} \end{aligned} \tag{5.18}$$

where $F(x)$ is the parent CDF of the random variable X. Subtracting the threshold t from X, we obtain

$$F_t(z) = P\left(X - t \leq z | X > t\right) = \frac{F(z+t) - F(t)}{1 - F(t)}. \tag{5.19}$$

From this equation it is easy to derive that

$$F(z+t) = (1 - F(t))\, F_t(z) + F(t) \tag{5.20}$$

If we can obtain the conditional tail distribution $F_t(z)$ accurately, we can calculate the CDF values of the tail portion of the original distribution by Equation (5.19). It is known that for most distributions, the CDF of the exceedance would converge to a generalized Pareto distribution (GPD), as specified below.

$$G_{\xi,\beta}(z) = \begin{cases} 1 - \left(1 - \xi\frac{z}{\beta}\right)^{1/\xi}, & \xi \neq 0; \quad z \in (\xi, \beta) \\ 1 - e^{-z/\beta}, & \xi = 0; \quad z \leq 0 \end{cases} \tag{5.21}$$

where

$$D\,(\xi, \beta) = \begin{cases} [0, \infty) & \xi \leq 0 \\ [0, \beta/\xi] & \xi > 0. \end{cases} \tag{5.22}$$

It is noted that when $\xi = 0$, the CDF of the exceedance is the exponential distribution.

$$F_t(z) = 1 - e^{-z/\beta} \tag{5.23}$$

The case of $\xi > 0$ corresponds to PDFs with bounded tails. When ξ is significantly larger than zero, the corresponding PDFs it represent can have abrupt truncations. On the other hand, the case of $\xi < 0$ corresponds to PDFs with unbounded tails, and can have distributions with very large tails at their extreme values. In most practical cases, however, the distribution is smooth, and the tail portion is small, and therefore we use the case $\xi = 0$ to model the tail of the cell voltage for eDRAM. Our results in Section 5.7 demonstrate the this approach works well.

The exponential distribution has the useful property of being *memoryless*. If we replace $F(x)$ with the exponential distribution function in Equation (5.19), we obtain

$$\begin{aligned} F_t(z) = \frac{F(z+t) - F(t)}{1 - F(t)} &= \frac{e^{-t/\beta} - e^{-(z+t)/\beta}}{e^{-t/\beta}} \\ &= 1 - e^{-z/\beta} \end{aligned} \tag{5.24}$$

It is observed that the final result is independent of t. This tells us that if we are given an exponential distribution, then no matter what threshold we select, it will generate the same conditional distribution. Therefore beyond a certain point if the tail of the

distribution can be approximated by the exponential distribution, the threshold chosen has no effect over the final conditional distribution result. Mathematically speaking, if we have two thresholds t_1, t_2, $t_1 < t_2$, then if $F_{t_1}(z) = 1 - e^{-z/\beta}$, we have

$$F_{t_2}(z) = \frac{F(z + t_2) - F(t_2)}{1 - F(t_2)} \tag{5.25}$$

From Equation (5.20)We also know that

$$\begin{aligned} F(t_2) &= F(t_2 - t_1 + t_1) \\ &= (1 - F(t_1))\, F_{t_1}(t_2 - t_1) + F(t_1) \\ F(z + t_2) &= F(z + t_2 - t_1 + t_1) \\ &= (1 - F(t_1))\, F_{t_1}(z + t_2 - t_1) + F(t_1) \end{aligned}$$

Putting these results into Equation (5.25),we obtain

$$\begin{aligned} F_{t_2}(z) &= \frac{(1 - F(t_1))\,(F_{t_1}(z + t_2 - t_1) - F_{t_1}(t_2 - t_1))}{(1 - F(t_1))\,(1 - F_{t_1}(t_2 - t_1))} \\ &= \frac{F_{t_1}(z + t_2 - t_1) - F_{t_1}(t_2 - t_1)}{1 - F_{t_1}(t_2 - t_1)} \\ &= \frac{e^{-(t_2 - t_1)/\beta} - e^{-(z + t_2 - t_1)/\beta}}{e^{-(t_2 - t_1)/\beta}} \\ &= 1 - e^{-z/\beta} \end{aligned} \tag{5.26}$$

which proves that the threshold we choose has no effect over the final conditional distribution result as long as the approximation using exponential distribution holds at these threshold points. This allows flexibility in choosing the threshold during implementation.

Because of the memoryless property of the exponential distribution, under the limitations of the approximation in Equation (5.23), we have the flexibility to choose any threshold that is large enough. Therefore we can characterize the parameters in $\mathbf{Y}_{tail}$ as either their 99% points or 99.9% points. After we obtain the final exponential distribution for $V_{cell,tail}$, we can use it to determine the probability at the point of interest.

5.6.2 Finding the Tail of the Distribution

Due to the fact that we are now specifically looking at the tail of the distribution, the previous Taylor series expansion about the mean in Equation (5.14) is inaccurate in this

faraway region. To construct a new model, we use the first order Taylor series expansion around the tail instead.

$$V_{cell,tail} = V_{cell,tail0} + \mathbf{U}_{tail0}^{T} \mathbf{\Delta Y}_{tail0} \tag{5.27}$$

where $\mathbf{\Delta Y}_{tail0}, i = Y_i - Y_{i,tail}$, where $Y_{i,tail}$ is an appropriate threshold, t_{Taylor}, from which the tail can be computed. In our experiments, we set this to be the 99% point of the underlying process parameter distribution.

Physically, the idea of performing Taylor series expansions maps well to the design of memory cells. If we look at, for example, the effect of T_{ox} on the retention time, it is typically the low-T_{ox} samples that correspond to high leakage, and therefore, higher V_{cell} at the prescribed retention time. Therefore, we perform a Taylor series expansion about the tail at the left (lower end) of the distribution, which translates to the tail at the right (upper end) of the V_{cell} distribution.

Since analytical models at the tail portion are difficult to derive, we can use a small number of simulations to fit the coefficients of the Taylor series around the tail of the distribution. The number of simulations required is linear in the number of random variables considered, which is minimal due to the small number of transistors for an eDRAM cell.

As seen from Equation (5.27), the computation of the distribution of $V_{cell,tail}$ requires the calculation of the distribution of a weighted sum of random variables, $\mathbf{\Delta Y}_{tail0}$. If each of these is a Gaussian, their weighted sum is also a Gaussian, and our goal is to find the tail of this sum.

We approximate the tail of the weighted sum of these distributions by the weighted sum of the tails. For this approximation, we only use the tail points for $\mathbf{Y}$ when calculating $\mathbf{\Delta Y}_{tail0}$. Therefore, $\mathbf{\Delta Y}_{tail0}$ can be viewed as exponentially distributed with the appropriate sign changes (e.g., corresponding to considering the left tail of T_{ox} to compute the right tail of V_{cell}). To aid in this computation, we use the following result:

Observation [79]: The sum of k independent exponentially distributed random variables with different means follows a hypoexponential [79] distribution. If the k random variables are $X_1, X_2, \cdots, X_k$ with rate λ_i for X_i, then the CDF of $\mathbf{X} = \Sigma_{i=1}^{k} X_i$ can be

written as

$$H(x) = 1 - \alpha e^{x\Theta}\mathbf{1}, \tag{5.28}$$

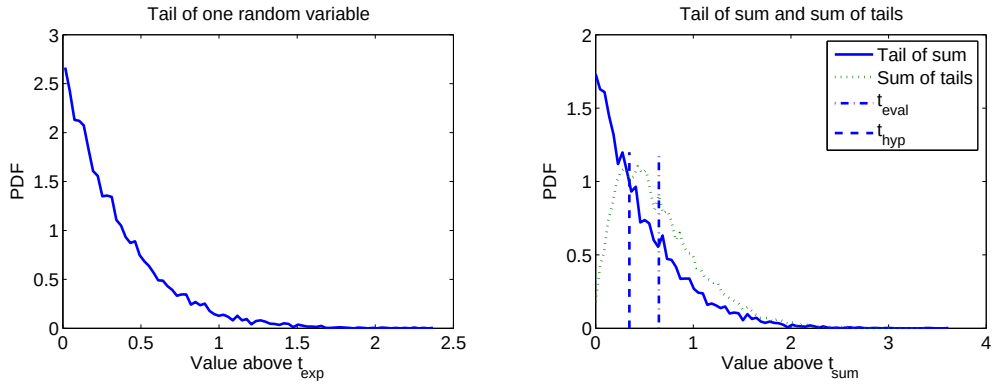
and the PDF is

$$h(x) = -\alpha e^{x\mathbf{\Theta}}\mathbf{\Theta}\mathbf{1}, \tag{5.29}$$

where $\mathbf{\Theta}$ *is a matrix constructed by the rates of the exponentially distributed random variables, namely*

$$\Theta = \begin{bmatrix} -\lambda_1 & \lambda_1 & 0 & \cdots 0 & 0 & \\ 0 & -\lambda_2 & \lambda_2 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & -\lambda_{k-2} & \lambda_{k-2} & 0 \\ 0 & 0 & \cdots & 0 & -\lambda_{k-1} & \lambda_{k-1} \\ 0 & 0 & \cdots & 0 & 0 & -\lambda_k \end{bmatrix}, \tag{5.30}$$

and $\alpha = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$. *The vector* $\mathbf{1}$ *contains all ones. The factor* $e^{x\mathbf{\Theta}}$ *is the matrix exponential [80] of* $x\mathbf{\Theta}$.



(a) (b)

Figure 5.5: A comparison between the sum of tails and the tail of the sum: (a) Monte Carlo points showing the tail of a single Gaussian PDF, (b) the sum of two such tails, shown by the dotted line, and the tail of the sum of two distributions, shown by the solid line.

Therefore, the sum of tails is a hypoexponential distribution described by Equation (5.29). Figure 5.5(a) shows a set of Monte Carlo samples that represent the tail of a single Gaussian random variable, truncated at the 99 percentile threshold t_{exp}[3] and the sum of the truncation points gives us the threshold t_{sum}: it is visually shown that the PDF for each single random variable can be approximated by an exponential distribution. The sum of two of these exponentials results in a hypoexponential, illustrated by the dotted line in Figure 5.5(b). Note that near the threshold, this differs from the tail of the sum, which is an exponential. This is not surprising: if the variables are independent, then

$$f(x_1 + x_2 = t_{sum} + c) = \int_{-\infty}^{\infty} f_{x_1}(x_1 = t) f_{x_2}(x_2 = t_{exp} + c - t) dt \tag{5.31}$$

where $f(.)$ is a PDF. When we work with the sum of the tails, the values of f_{x_i} differ from the original values since the PDF is zero at all values prior to the tail threshold, t_{exp}; if we work with the tail of the sum, these values are nonzero. The effect is diluted as c moves further away from the threshold t_{sum} as what we consider is the conditional distribution of the final tail. Ideally, what we want is that the conditional distribution of the sum of tails beyond a threshold larger than t_{sum}, equals to the conditional distribution of the tail of sum beyond that threshold. As we move away from t_{sum}, the contribution from parameter values less than t_{exp} becomes increasingly smaller toward the final conditional distribution. It is therefore reasonable to assume that the distribution is more accurate as we move away from the threshold, t_{sum}. Using this idea, we can truncate (and renormalize) the hypoexponential distribution at some point t_{hyp}, as shown by a dashed line in Figure 5.5(b). Due to the memoryless property of the exponential distribution, the same distribution may be extrapolated back to t_{sum}, as also shown in a dashed line in Figure 5.5(b) and eventually back to the tail threshold we want.

To convert the hypoexponential distribution to an exponential, based on its values beyond a threshold, t_{hyp}, shown as the dashed line in Figure 5.5(b), we match this conditional distribution at some point $t_{eval} > t_{hyp}$, shown as the dot-dash line in Figure

[3] Realistically, the location of this threshold may be different for various elements to be summed.

5.5(b), with the exponential form of Equation (5.23). Mathematically:

$$\frac{H(t_{eval}) - H(t_{hyp})}{1 - H(t_{hyp})} = 1 - e^{-(t_{eval} - t_{hyp})/\beta} \tag{5.32}$$

$$\text{i.e., } \beta = -\frac{t_{eval} - t_{hyp}}{\log\left(1 - \frac{H(t_{eval}) - H(t_{hyp})}{1 - H(t_{hyp})}\right)} \tag{5.33}$$

From this equation, β can be easily computed, and we have the tail distribution in the exponential form. It can be seen that β is related to both the mean and the variance of the hypoexponential distribution, and the relationship is implied during the calculation of $H(t_{eval})$ and $H(t_{hyp})$.

There are four specific parameters that we must select:

- t_{exp} corresponds to the point in the parameter space beyond which the tail of a function is modeled as a truncated exponential. The value of t_{exp} could be different for various process parameters.

- t_{Taylor} is the point in the parameter space about which the Taylor series expansion in (5.27) is performed in the parameter space. This value could be different for various process parameters.

- t_{sum} is a point in the performance space at which the parameters are equal to their respective t_{exp} points.

- t_{eval} is a point in the performance space at which the hypoexponential approximation of the performance function (in our case, V_{cell} at t_{RET}), based on the sum of the tails, is matched to an exponential distribution.

- t_{hyp} is the point in the performance space beyond which the sum of the tails provides a reasonable approximation to the tail of the sums, according to the argument made above.

The above parameters must be chosen carefully. As stated earlier, further into the tail, the approximation using the sum of tails for the tail of the sum would be more accurate, because points smaller than t_{exp} contribute very little to the integral. However, because the first order Taylor serious expansion is performed around t_{Taylor}, in order to maintain

the accuracy of this expansion, the points t_{hyp}, t_{Taylor}, and t_{eval} should not be too far away.

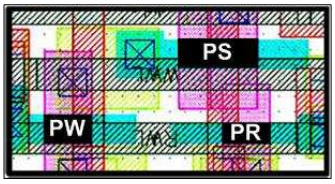
In our framework, we choose their relative positions as a preprocessing step by examining the curve of $V_{cell,tail0}$ with respect to $\mathbf{\Delta Y}_{tail0}$ and choose t_{Taylor} such that the linear relationship holds for all elements in $\mathbf{\Delta Y}_{tail0}$, while making t_{Taylor} as large as possible. The parameter t_{hyp} must be smaller than t_{eval} to make Equation (5.32) valid, but it should be close enough to t_{hyp} and t_{Taylor} for the first-order Taylor expansion to be accurate in this region. These relative positions are tuned once and can be used for other experimental setups.

5.7 Experimental Results

In this work, for the eDRAM gain cell in Figure 5.1, the sizes of each transistor and the layout are shown in Table 5.1. Unless otherwise stated, the activating voltage for WWL is set to be $V_{DD} + 0.4V$, and V_{DD} is 0.9V.

Table 5.1: Size of each transistor and the layout of the 3T eDRAM gain cell (0.9V, 65nm LP PMOS).

	PR	PS	PW
W (nm)	150	225	150
L (nm)	60	90	90



The parameter variations considered in our experiments include the threshold voltage, V_{th}, and the oxide thickness, T_{ox}, which are assumed to be Gaussian-distributed and uncorrelated. However, as mentioned in Section 5.4, our analysis technique can incorporate all Gaussian parameter variations, including correlated variations.

5.7.1 Leakage characterization

To evaluate the body of the distribution using techniques described in Section 5.4, a characterization process must be performed only once for a given process, to determine the equations that characterize leakage, as described in Section 5.4.1. These equations must accurately fit the behavior of V_{cell} over the expected range of parameter variations.

In order to verify that our model accurately captures all leakage components for nominal process parameter values, we plot each leakage component respectively as a function of the cell voltage in Figure 5.6. The circled points are values generated by SPICE, and the solid lines are approximations using our model. It is observed that the two overlap almost exactly, and our model tracks the leakage components very well with changing V_{cell}.

Taking all leakage components into consideration, we can evaluate how well Equation (5.12) tracks the transient simulation results of V_{cell} using SPICE. The result is shown in Figure 5.7(a) and the match is observed to be excellent. The circled points are the SPICE results, while the solid line corresponds to Equation (5.14).

To illustrate that our model captures the parameter variations well, we show as an example the plot of the gate leakage for transistor PS when V_{cell} is zero with varying T_{ox}, in Figure 5.7(b).

5.7.2 V_{cell} Distributions Using Moment Matching

We assume that V_{th} and T_{ox} are Gaussian-distributed, centered at their nominal values, and with 3σ values being 21% and 5% of their mean, respectively. Although our model is generally applicable to a large number of process parameter variations, for simplicity of implementation, we first examine the impact of V_{th} variations and T_{ox} variations on V_{cell}, respectively.

The following analysis demonstrates that most of the variations of V_{cell} are due to the T_{ox} variations because of the negative V_{GS} biasing for the write transistor PW introduced in Section 5.2, which means using $V_{DD} + 0.4V$ instead of V_{DD} for the gate voltage of PW. Figure 5.8 shows the results of SPICE Monte Carlo simulations over the range of V_{th} and T_{ox}. The solid line incorporates variations in V_{th}, while the dotted line ignores them: it is clearly seen that the PDF of V_{cell} in either plot is almost the same.

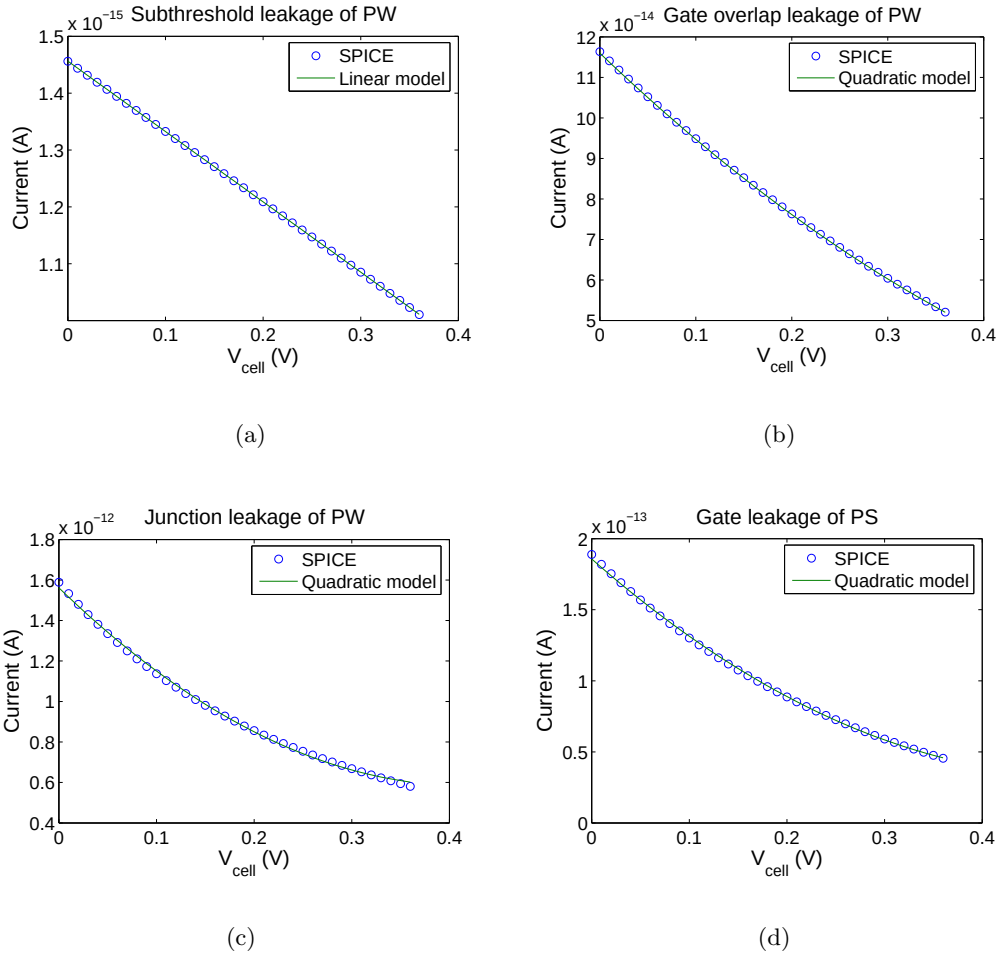
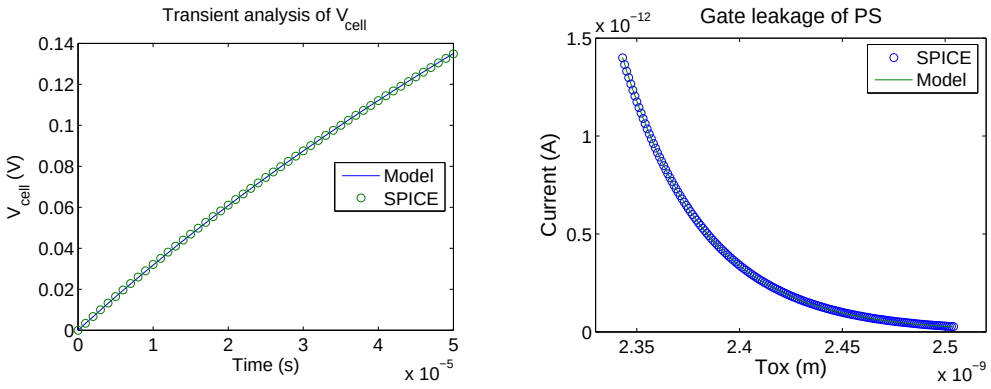


Figure 5.6: Leakage models for (a) subthreshold leakage of transistor PW, (b) gate overlap leakage of transistor PW, (c) junction leakage of transistor PW and (d) gate leakage of transistor PS.



(a) (b)

Figure 5.7: (a) Transient simulation of V_{cell}. (b) Gate leakage with zero cell voltage and varying T_{ox} for transistor PS.

Therefore in our implementation of the tail distribution, we only consider T_{ox} to be the source of variations.

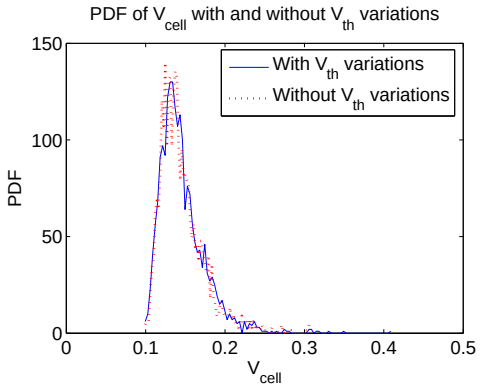


Figure 5.8: PDF of the V_{cell} distribution with and without V_{th} variations

Algorithm 5 is then applied with an impulse function of order 4 to approximate the PDF of V_{cell} up to the 99% point for an eDRAM at $t_0 = 50\mu s$. Because of the nature of moment-matching-based methods, the PDF provided by APEX may have small oscillations for the smallest values of V_{cell} and may go below zero. This issue, however, is not relevant because in almost all cases we are interested in the PDF at

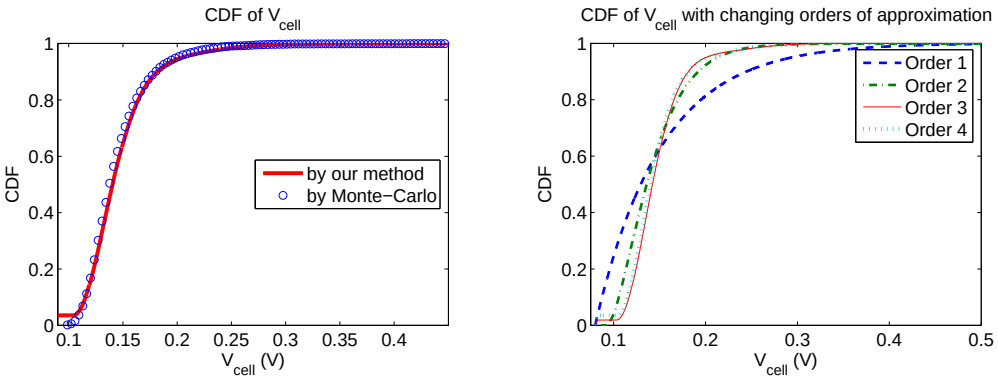
its other end, as the CDF goes towards 1. However, we include a simple heuristic that keeps the PDF consistently nonnegative, without impacting accuracy:

$$PDF(t) = \begin{cases} \omega h(t) & h(t) \geq 0 \\ 0 & h(t) < 0 \end{cases} \tag{5.34}$$

where ω is a constant such that $PDF(t)$ integrates to 1. The cumulative distribution function, $CDF(t)$, can be obtained analogously by scaling $s(t)$ of Equation (2.6) of Chapter 2.

The results of the CDF are shown in Figure 5.9(a). This CDF can be regarded as an estimate of the cell voltage distribution across all cells of a large eDRAM array. For comparison, we also plot the CDF across all cells of a 10kb eDRAM array, using the circled points, obtained by Monte Carlo simulation using SPICE. The solid line indicates the function $CDF(t)$ obtained from our analysis, and this is seen to be very close to the Monte Carlo results. There are two kinds of error metric. One is *yield prediction error*, and the other is *point prediction error*. We take the value of V_{cell} at the 99% point of the distribution obtained using the model, and determine the yield for the Monte Carlo simulation results at this point, Y. The *yield prediction error* at the 99% point is defined to be $\frac{|Y-99\%|}{99\%}$. This error is found to be 0.3% for the eDRAM cell with sizes defined by Table 5.1. If we take the 99% point of the distribution obtained using our model and using Monte Carlo simulations respectively, and then calculate the error between the two, then we obtain the *point prediction error*. Due to the condition of the CDF function at the tail, the point prediction error is usually much larger than the yield prediction error. We observe that the point prediction error at the 99% point is 2.18% for the eDRAM cell defined in Table 5.1. Excluding the characterization step which is carried out only once for a process, the run time of our method is 0.94 seconds, compared to 6.3 minutes cost by Monte Carlo simulation using SPICE. The speedup is expected to increase with the size of the memory array.

The run time savings is largely attributable to the fact that a low order of the APEX algorithm is sufficiently accurate. In Figure 5.9(b), we show the CDF results using orders 1, 2, 3 and 4, respectively. It is observed that order 1 and order 2 approximations are less accurate, while the CDF of order 3 and order 4 almost overlap entirely with each other. Further increasing the approximation order would not help with the accuracy and it is known that APEX has stability issues for high order of approximations. Therefore



(a) (b)

Figure 5.9: (a) CDF of V_{cell}. (b) CDF with changing orders of approximation.

for our problem, we conclude that a reduced order model of order 3 or 4 is sufficient. We observe that in general, the APEX method used to evaluate the PDF/CDF scales well with increasing number of parameters [19], so that a larger number of parameters can easily be handled.

A common procedure for design improvement is to change the sizes of transistors in the eDRAM cell and the WWL voltage. These design parameters are inputs to our algorithm and our approach can robustly produce accurate results with changing inputs. We show the 99% point prediction errors for a list of different size configurations of transistors PW and PS in Table 5.2, because it is the metric that generates larger errors.

To illustrate the impact on the CDF result by changing sizes of transistors, we show in Figure 5.10 the comparison between the CDF results of original sizes listed in Table 5.1, and with transistor width of the storage transistor PS sized up to be $255nm$. The CDF is observed to be shifted toward left and our model captures the shift accurately.

As an example to show that our analysis method is robust to different WWL voltages, we tested the result for the original size configuration using a WWL voltage of $V_{DD} + 0.2V$: for this scenario, the prediction error for the 99% point is 8.27%.

Table 5.2: Yield prediction errors at the 99% point for different size configurations.

size configuration				prediction error at 99% point
PW		PS		
W (nm)	L (nm)	W (nm)	L (nm)	
150	90	225	90	2.18%
150	90	240	90	6.88%
150	90	255	90	7.77%
180	90	225	90	0.29%
150	120	150	120	2.38%

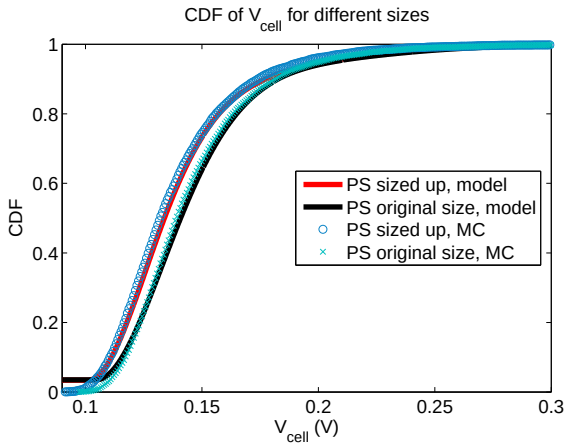


Figure 5.10: The CDF distribution for various transistor sizes.

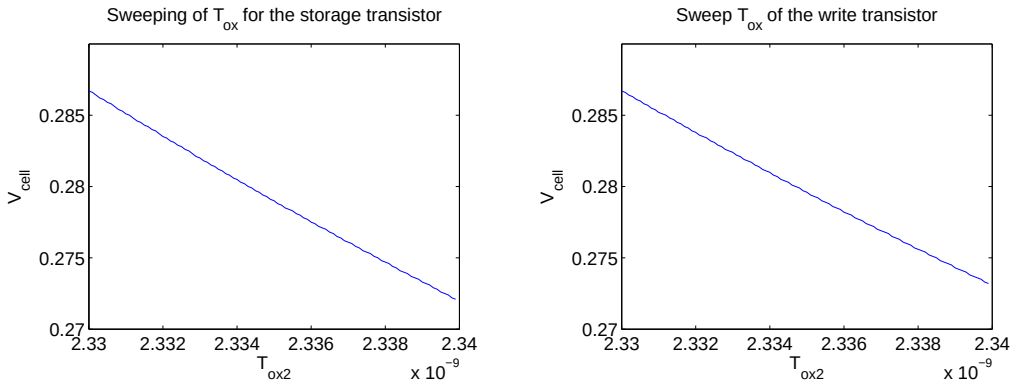
5.7.3 V_{cell} Distributions Using Extreme Value Theory

Next, we compare our model of the tail distribution with Monte Carlo simulations. As is illustrated in Section 5.6, if the tail follows an exponential distribution, because of the memoryless property of the exponential distribution, ideally the distribution is the same no matter what tail threshold we choose as long as the threshold is large enough so that the points beyond that can be characterized as the tail. In our experiments, we choose the 99% CDF point to be tail threshold. The value of this threshold can either be obtained by a reasonable number of Monte Carlo simulations or by the analytical method introduced in Section 5.4. To accurately capture the error generated by the tail distribution evaluation technique illustrated in Section 5.6, in this work we use Monte

Carlo simulations to determine this threshold value. However, it is obvious that the two analytical methods can be combined to form a fast simulation framework.

By sweeping the parameters in the region of interest, it is easily seen that V_{cell} decreases as the T_{ox} of either the read transistor or the storage transistor increases: this is shown in Figure 5.7.3. These figures also show that the assumption of a linear relationship between V_{cell} and T_{ox} in the region of interest is a good approximation. Therefore when characterizing the T_{ox} distribution, we use the 1% points for the T_{ox} distribution, for both the storage and the write transistors, as the threshold of their left tails. We perform the Taylor expansion around the threshold points of the T_{ox} values, and model the symmetric portion of these two tail distributions as exponential distributions. Then, as illustrated in Section 5.6, we use a hypoexponential distribution to model the conditional distribution of V_{cell} and then convert it back to an exponential distribution.

The results of applying this approach are shown in Figure 5.12. The circled points are the Monte Carlo results of the tail, and the solid curve is the exponential distribution fit for the conditional distribution, we see that the two results mostly coincide with each other. The hypoexponential distribution is also shown as an intermediate step, using the dashed line. V_{tail} in the figure is the tail threshold, corresponding to t_{exp} in our earlier notation.



(a) (b)

Figure 5.11: The V_{cell} vs. T_{ox} for (a) the storage transistor and (b) the write transistor

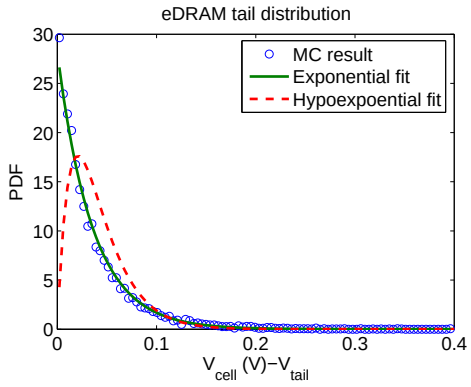


Figure 5.12: PDF of the conditional tail distribution

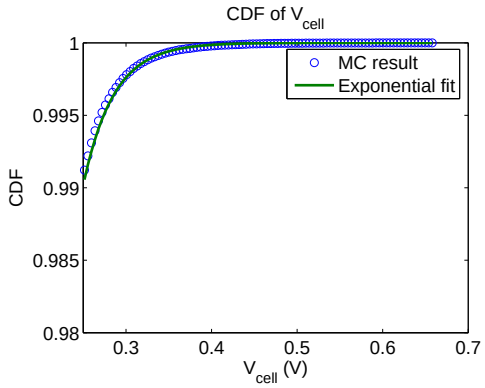


Figure 5.13: CDF of the tail portion of the original distribution

Once we compute this conditional distribution, it is easy to covert it back to the tail of the original distribution using Equation (5.20). The resulting CDF, and a comparison of Monte Carlo results, are shown in Figure 5.13.

The runtime of the tail distribution evaluation technique is the sum of three components:

- the time required to compute the threshold value, $t_{exp} = V_{tail}$,
- the simulations required to fit the coefficients with respect to the random variables, which are observed to take 1.1 seconds, and
- the time required to calculate the exponential distribution using the sum of the tails, which is around 1.2 seconds.

For the first component, if we use 10^4 Monte Carlo simulations to calculate the tail threshold, this involves a runtime of 378 seconds, and the total runtime of our tail evaluation technique is 6.3 minutes. In contrast, a set of 10^6 Monte Carlo simulations for the 99.995% point takes 10.5 hours. If we use the analytical method in Section 5.4, to calculate the tail threshold, the run time savings would be even more significant. All runtimes are reported on a Linux PC with 3.2GHz frequency and 2GB memory.

We also compare our method with the *statistical blockade* approach in [48]. This approach uses a number of Monte Carlo simulations to get the threshold of the 99% point, and borrows ideas from the data mining community and uses a *support vector machine* (SVM) as a classifier to identify simulation points that are more likely to generate tail points. In order not to be conservative in predicting tail points, a smaller threshold (97% point as in [48]) is used in the SVM. Only these points, identified by the SVM, are simulated using SPICE to generate the tail distribution.

The classifier is first trained using 1000 points, and appropriate scaling of the SVM is considered as the number of tail points is significantly smaller than the number of non-tail points. Applying this approach to the eDRAM problem, we get runtime savings as compared to 10^6 Monte Carlo simulations due to the filtering of the SVM classifier. The number of Monte Carlo points filtered using the SVM, the number of real tail points that have V_{cell} values actually larger than the 99% threshold among the filtered points, and the number of real tail points in the 10^6 original Monte Carlo simulations

are listed in Table 5.3 in the last three columns, respectively. It is noted that apart from the simulations to find the threshold, and the simulations for the training points, the statistical approach generally can obtain 10× runtime savings with a small sacrifice in the accuracy. Even so, approximately 10^5 Monte Carlo points must be simulated in order to obtain most of the tail points in the original 10^6 Monte Carlo simulations. Our approach, on the other hand, only requires as few as two Monte Carlo simulations to get the first order Taylor series expansion around the tail threshold, hence greatly reduces the runtime compared to the statistical blockade approach.

A comparison of the two approaches are listed in Table 5.4. In this table, we compare the 99.995% point for each distribution under different transistor sizes using our method and the statistical blockade (Statblock) method, respectively. To find this point, we set $F_t(z) = 99.5\%$, and $F(t) = 99\%$, and from Equation (5.20),

$$F_t(z+t) = (1 - 99\%) \times 99.5\% + 99\% = 99.995\% \tag{5.35}$$

Therefore the 99.5% point of the tail corresponds to the 99.995% point of the original distribution. It is noted that we can always choose a high threshold for the tail, and the extreme value theory suggest that the high the threshold is, the more likely that the distribution would converge to the generalized Pareto distribution, of which the exponential distribution is a special case. Therefore our method will have even more accurate results. Moreover, inspecting further into the tail for the purely Monte Carlo methods would require more experiments, thus making the run time savings of our method more attractive. For this work, we believe that the 99.995% point is sufficient to illustrate the idea. The runtime listed does not include the runtime to calculate the tail threshold for either case. It is noted that if we use Monte Carlo methods to obtain this threshold, the runtime of this part will be the same. If we use the analytical method in Section 5.4 to calculate the threshold, the runtime will be further reduced from Monte Carlo methods.

From Table 5.4, it is noted that our method provides predictions with error less than 3% for most cases, and runs much faster than the statistical blockade approach. The statistical blockade approach can be made faster by either increasing the threshold for the SVM, or reducing the number of the original Monte Carlo points generated to feed into the SVM. Increasing the threshold significantly quickly renders the method

Table 5.3: Monte Carlo simulation savings for the statistical blockade approach

Size configuration				MC points filtered	True tail points among the filtered points	True tail points among the 10^6 MC points
PW		PS				
W (nm)	L (nm)	W (nm)	L (nm)			
150	90	225	90	112562	9538	10513
150	90	240	90	108561	9338	10385
150	90	255	90	108561	8847	9920
180	90	225	90	112524	9708	10505
150	120	150	120	113028	9821	10571

Table 5.4: Error and runtime comparison of our method and statistical blockade

Size configuration				Prediction error at 99.995% point		Runtime	
PW		PS					
W (nm)	L (nm)	W (nm)	L (nm)	Our method	Statblock	Our method	Statblock
150	90	225	90	2.05%	0.45%	2.3sec	1.2hr
150	90	240	90	1.28%	0.32%	2.3sec	1.1hr
150	90	255	90	2.98%	1.46%	2.3sec	1.1hr
180	90	225	90	1.13%	0.59%	2.3sec	1.2hr
150	120	150	120	5.30%	0.48%	2.2sec	1.2hr

inaccurate because most points filtered out are larger than the new threshold, thus skewing the statistics. Therefore, for fair comparison, we reduce the number of Monte Carlo points to be 500000 as input to the SVM. The results are shown in Table 5.5, and it is noted that the accuracy is slightly worse than our approach for most cases, while the runtime is still larger. It is noted that the statistical blockade approach cannot be made as efficient as our method because less than 10 Monte Carlo simulations would not produce any meaningful distribution.

Finally we test our method using APEX to get the tail threshold and our tail analysis technique to get further into the tail. The result is a much faster framework with run time totalling 3.2 seconds, as compared to statistical blockade method with 50000 initial MC points totalling 9.8 minutes. A comprehensive comparison of accuracy and run time

Table 5.5: Statistical blockade results with 500000 Monte Carlo points

Size configuration				Prediction error at 99.995% point	MC points simulated	True tail points	Runtime
PW		PS					
W (nm)	L (nm)	W (nm)	L (nm)				
150	90	225	90	3.59%	5666	455	3.6min
150	90	240	90	5.02%	5387	458	3.4min
150	90	255	90	4.24%	5387	435	3.4min
180	90	225	90	4.36%	5575	472	3.5min
150	120	150	120	3.25%	5606	487	3.5min

is shown in Table 5.6. The runtimes shown are based on the average runtime on all five sizes, as the runtime for each size is practically very similar.

Table 5.6: Error comparison of our method and statistical blockade

Size configuration				Prediction error at 99.995% point			
PW (nm)		PS (nm)					
W	L	W	L	MC	Statblock	MC+EVT	APEX+EVT
150	90	225	90	-	3.59%	2.05%	1.56%
150	90	240	90	-	5.02%	1.28%	1.33%
150	90	255	90	-	4.24%	2.98%	3.76%
150	120	150	120	-	5.30%	3.25%	5.30%
180	90	225	90	-	4.36%	1.13%	1.40%
Runtime				10.5 hrs	9.8 min	6.3 min	3.2 sec

5.8 Conclusion

In this work, we have developed a novel statistical cell voltage analysis tool to aid in the design of embedded DRAMs. Experimental results show that our approach is both accurate and efficient.

Chapter 6

Conclusion

As feature sizes continue to shrink and these variations affect both logic and memory circuits, it is widely acknowledged that process parameter variations cannot be neglected in modern VLSI designs. For logic circuits, while presilicon analysis and optimization have been researched for several years, post-silicon tuning is still an emerging area with challenging issues to address. Accurate post-silicon delay prediction is essential to facilitate post-silicon tuning in order to reliably improve yields. This thesis has aided these tuning techniques by providing more robust and reliable post-silicon delay prediction approaches than critical path replica while still maintaining the efficiency of testing time. For memory circuits, eDRAMs are becoming increasingly popular as candidates for the on-die cache and variation-aware analysis tools play vital roles to ensure that these designs meet specifications. This thesis has provided an accurate and efficient analysis technique for the cell voltage of eDRAM, including tail analysis.

For the post-silicon statistical delay analysis work, the work in this thesis has assumed that the parameter variations are Gaussian-distributed and that first-order approximations for the delay models are adequate. For non-Gaussian parameter variations and nonlinear delay model approximations, the SSTA procedure used here can be replaced by existing non-Gaussian, nonlinear SSTA techniques, and the notion of the conditional PDF proposed here still holds. However, if the delay of the original circuit and the delays of the test structures are no longer Gaussian-distributed, the conditional distribution may not strictly follow a closed form. Approximation and numerical techniques will have to be explored to solve the problem, as is done in most non-Gaussian,

nonlinear SSTA techniques. Furthermore, the spatial correlation model used in this work is based on academic models and in practice, this model should be characterized accurately using real silicon data because it directly impacts the accuracy of the delay prediction.

Our work on statisical memory analysis of the eDRAM work can be extended to statistically analyze the data retention time as well as the read speed. The tail analysis technique is also potentially applicable to any large-scale circuit with highly repeated cells. To use the statistical analysis result to perform memory optimization is also an interesting topic that needs further exploration.

References

[1] R. Rao, A. Devgan, D. Blaauw, and D. Sylvester. Parametric Yield Estimation Considering Leakage Variability. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 442–447, June 2004.

[2] S. S. Sapatnekar. *Timing*. Kluwer Academic Publishers, Boston, MA, 2004.

[3] S. R. Nassif. Design for Variability in DSM Technologies. In *Proceedings of the IEEE International Symposium on Quality Electronic Design*, pages 451–454, March 2000.

[4] J. J. Liou, A. Krstic, L-C. Wang, and K. T. Cheng. False-Path-Aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 566–569, June 2002.

[5] J. J. Liou, K. T. Cheng, S. Kundu, and A. Krstic. Fast Statistical Timing Analysis by Probabilistic Event Propagation. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 661–664, June 2001.

[6] A. Agarwal, D. Blaauw, V. Zolotov, S. Sundareswaran, M. Zhao, K. Gala, and R. Panda. Path-Based Statistical Timing Analysis Considering Inter- and Intra-die Correlations. In *Workshop Notes, ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pages 16–21, December 2002.

[7] A. Devgan and C. Kashyap. Block-based Static Timing Analysis with Uncertainty. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 607–614, November 2003.

[8] H. Chang and S. S. Sapatnekar. Statistical Timing Analysis Considering Spatial Correlations using a Single PERT-Like Traversal. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 621–625, November 2003.

[9] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order Incremental Block-Based Statistical Timing Analysis. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 331–336, June 2004.

[10] J. Singh and S. S. Sapatnekar. Statistical Timing Analysis with Correlated Non-Gaussian Parameters using Independent Component Analysis. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 155–160, July 2006.

[11] Y. Zhan, A. J. Strojwas, X. Li, L. Pileggi, D. Newmark, and M. Sharma. Correlation-Aware Statistical Timing Analysis with Non-Gaussian Delay Distributions. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 77–82, June 2005.

[12] H. Chang, V. Zolotov, S. Narayan, and C. Visweswariah. Parameterized Block-Based Statistical Timing Analysis with Non-Gaussian Parameters, Nonlinear Delay Functions. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 71–76, June 2005.

[13] V. Khandelwal and A. Srivastava. A General Framewok for Accurate Statitical Timing Analysis Considering Correlations. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 89–94, June 2005.

[14] Y. Liu, L. T. Pileggi, and A. J. Strojwas. Model Order-Reduction of RC(L) Interconnect including Variational Analysis. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 201–206, June 1999.

[15] J. D. Ma and R. A. Rutenbar. Interval-Valued Reduced Order Statistical Interconnect Modeling. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 460–467, June 2004.

[16] Y. Liu, S. R. Nassif, L. T. Pileggi, and A. J. Strojwas. Impact of Interconnect Variations on the Clock Skew of a Gigahertz Microprocessor. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 168–171, June 2000.

[17] S. X. Shi, A. Ramalingam, D. Wang, and D. Z. Pan. Latch Modeling for Statistical Timing Analysis. In *Proceedings of the Design, Automation & Test in Europe*, pages 1136–1141, March 2008.

[18] M. C-T. Chao, L-C. Wang, K-T. Cheng, and S. Kundu. Static Statistical Timing Analysis for Latch-based Pipeline Designs. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 468–472, November 2004.

[19] X. Li, J. Le, P. Gopalakrishnan, and L. T. Pileggi. Asymptotic Probability Extraction for Non-Normal Distributions of Circuit Performance. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 2–9, November 2004.

[20] E. T. A. F. Jacobs and M. R. C. M. Berkelaar. Gate Sizing Using a Statistical Delay Model. In *Proceedings of the Design, Automation & Test in Europe*, pages 283–291, March 2000.

[21] H. Chang and S. S. Sapatnekar. Full-Chip Analysis of Leakage Power Under Process Variations, Including Spatial Correlations. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 523–528, June 2005.

[22] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester. Statistical Estimation of Leakage Current Considering Inter- and Intra-Die Process Variation. In *Proceedings of the ACM International Symposium on Low Power Electronics and Design*, pages 84–89, August 2003.

[23] A. Srivastava, R. Bai, D. Blaauw, and D. Sylvester. Modeling and Analysis of Leakage Power Considering Within-Die Variations. In *Proceedings of the ACM International Symposium on Low Power Electronics and Design*, pages 64–67, August 2002.

[24] S.Mukhopadhyay and K. Roy. Modeling and Estimation of Total Leakage Current in Nano-scaled CMOS Devices Considering the Effect of Parameter Variation. In *Proceedings of the ACM International Symposium on Low Power Electronics and Design*, pages 172–175, August 2003.

[25] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. 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*, 37:1396–1402, November 2002.

[26] J. W. Tschanz, S. Narendra, R. Nair, and V. De. Effectiveness of Adaptive Supply Voltage and Body Bias for Reducing the Impact of Parameter Variations in Low Power and High Performance Microprocessors. *IEEE Journal of Solid-State Circuits*, 38:826–829, May 2003.

[27] J. W. Tschanz, S. Narendra, A. Keshavarzi, and V. De. Adaptive Circuit Techniques to Minimize Variation Impacts on Microprocessor Performance and Power. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 23–26, May 2005.

[28] M. Nakai, S. Akui, K. Seno, T. Meguro, T. Seki, T. Kondo, A. Hashiguchi, H. Kawahara, K. Kumano, and M. Shimura. Dynamic Voltage and Frequency Management for a Low-Power Embedded Microprocessor. *IEEE Journal of Solid-State Circuits*, 40:28–35, November 2005.

[29] M. Elgebaly and M. Sachdev. Variation-Aware Adaptvie Voltage Scaling System. *Proceedings of the IEEE International Symposium on VLSI Technology, Systems and Applications*, 15:560–571, November 2007.

[30] B. Lee, L. Wang, and M. S. Abadir. Refined Statistical Static Timing Analysis Through Learning Spatial Delay Correlations. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 149–154, July 2006.

[31] L. Wang, P. Bastani, and M. S. Abadir. Design-Silicon Timing Correlation–A Data Mining Perspective. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 385–389, June 2007.

[32] A. Davoodi and A. Srivastava. Variability Driven Gate Sizing for Binning Yield Optimization. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 956–964, July 2006.

[33] V. Khandelwal and A. Srivastava. Variability-Driven Formulation for Simultaneous Gate Sizing and Postsilicon Tunability Allocation. In *Proceedings of the International Symposium on Physical Design*, pages 11–18, March 2007.

[34] M. Abranmovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller. A Reconfigurable Design-for-Debug Infrastructure for SoCs. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 7–12, July 2006.

[35] M. Mani, A. Singh, and M. Orshansky. Joint-Design-Time and Post-Silicon Minimization of Parametric Yield Loss using Adjustable Robust Optimization. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 19–26, November 2006.

[36] A. Drake, R. Senger, H. Deogun, G. Carpenter, S. Ghiasi, T. Nguyen, N. James, M. Floyd, and V. Pokala. A Distributed Critical-Path Timing Monitor for a 65nm High-Performance Microprocessor. In *Digest of Technical Papers, IEEE International Solid-State Circuits Conference*, pages 398–399, February 2007.

[37] N. Callegari, P. Bastani, L. Wang, S. Chakravarty, and A. Tetelbaum. Path Selection for Monitoring Unexpected Systematic Timing Effects. In *Proceedings of the Asia-South Pacific Design Automation Conference*, pages 781–786, January 2009.

[38] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers. Matching Properties of MOS Transistors. *IEEE Journal of Solid-State Circuits*, 24(5):1433–1440, May 1989.

[39] M. Ichihashi, H. Toda, Y. Itoh, and K. Ishibashi. 0.5V Asymmetric Three-Tr. Cell (ATC) DRAM Using 90nm Generic CMOS Logic Process. In *Proceedings of the IEEE International Symposium on VLSI Circuits*, pages 366–369, June 2005.

[40] W. Luk, J. Cai, R. Dennard, M. Immediato, and S. Kosonocky. A 3-Transistor DRAM Cell with Gated Diode for Enhanced Speed and Retention Time. In *Proceedings of the IEEE International Symposium on VLSI Circuits*, pages 184–185, June 2006.

[41] D. Somasekhar, Y. Ye, P. Aseron, S. Lu, M. Khellah, J. Howard, G. Ruhl, T. Karnik, S. Y. Borkar, V. De, and A. Keshavarzi. 2GHz 2Mbit 2T Gain-Cell Memory Macro with 128GBytes/sec Bandwith on 65nm Logic Process. In *Digest of Technical Papers, IEEE International Solid-State Circuits Conference*, pages 274–275, February 2008.

[42] J. Barth, W. Reohr, P. Parries, G. Fredeman, J. Golz, S. Schuster, R. Matick, H. Hunter, C. Tanner, J. Harig, H. Kim, B. Khan, J. Griesemer, R. Havreluk, K. Yanagisawa, T. Kirihata, and S. Iyer. A 500MHz Random Cycle 1.5ns-Latency, SOI Embedded DRAM Macro Featuring a 3T Micro Sense Amplifier. In *Digest of Technical Papers, IEEE International Solid-State Circuits Conference*, pages 486–487, 2007.

[43] IBM to Ditch SRAM for Embedded DRAM on Power CPUs, 2007. (Available at `http://www.itjungle.com/breaking/bn021407-story01.html`).

[44] K. Agarwal and S. Nassif. Statistical Analysis of SRAM Cell Stability. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 57–62, July 2006.

[45] G. C. Chen, D. Blaauw, T. Mudge, D. Sylvester, and N. S. Kim. Yield-Driven Near-Threshold SRAM Design. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 660–666, November 2007.

[46] R. Kanj, R. Joshi, C. Adams, J. Warnock, and S. Nassif. An Elegant Hardware-Corroborated Statistical Repair and Test Methodology for Conquering Aging Effects. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 497–504, November 2009.

[47] R. Aitken and S. Idgunji. Worst-Case Design and Margin for Embedded SRAM. In *Proceedings of the Design, Automation & Test in Europe*, pages 1289–1294, March 2007.

[48] A. Singhee and R. Rutenbar. Statistical Blockade: Very Fast Statistical Simulation and Modeling of Rare Circuit Events and Its Application to Memory Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28:1176–1187, August 2009.

[49] N. E. Evmorfopoulos, G. I. Stamoulis, and J. N. Avaritsiotis. A Monte Carlo Approach for Maximum Power Estimation Based on Extreme Value Theory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(4):415–432, April 2002.

[50] J. Le, X. Li, and L. Pileggi. STAC: Statistical Timing Analysis with Correlation. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 343–348, June 2004.

[51] A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula. Statistical Timing Analysis Using Bounds and Selective Enumeration. In *Workshop Notes, ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pages 29–36, December 2002.

[52] A. Devgan and C. Kashyap. Block-based Static Timing Analysis with Uncertainty. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 607–614, November 2003.

[53] A. Agarwal, D. Blaauw, V. Zolotov, S. Sundareswaran, M. Zhao, K. Gala, and R. Panda. Statistical Delay Computation Considering Spatial Correlations. In *Proceedings of the Asia-South Pacific Design Automation Conference*, pages 271–276, January 2003.

[54] M. Berkelaar. Statistical Delay Calculation, a Linear Time Method. In *Workshop Notes, ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pages 15–24, December 1997.

[55] M. Orshansky and K. Keutzer. A General Probabilistic Framework for Worst Case Timing Analysis. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 556–561, June 2002.

[56] S. Tsukiyama, M. Tanaka, and M. Fukui. A Statistical Static Timing Analysis Considering Correlations Between Delays. In *Proceedings of the Asia-South Pacific Design Automation Conference*, pages 353–358, January 2001.

[57] A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula. Computation and Refinement of Statistical Bounds on Circuit Delay. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 348–353, June 2003.

[58] S. Naidu. Timing Yield Calculations Using an Impulse-Train Approach. In *Proceedings of the IEEE International Conference on VLSI Design*, pages 219–224, January 2002.

[59] A. Agarwal, D. Blaauw, V. Zolotov, and S.B.K. Vrudhula. Statistical Timing Analysis Using Bounds. In *Proceedings of the Design, Automation & Test in Europe*, pages 10062–10067, March 2003.

[60] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical Timing Analysis Using Bounds and Selective Enumeration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22:1243–1260, September 2003.

[61] H. Chang and S. S. Sapatnekar. Statistical Timing Analysis Under Spatial Correlations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24:1467–1482, September 2005.

[62] C. E. Clark. The Greatest of A Finite Set of Random Variables. *Operations Research*, 9:145–162, March 1961.

[63] C. Ebeling and B. Lockyear. On the Performance of Level-Clocked Circuit. In *Proceedings of the Advanced Reserch in VLSI*, pages 342–356, 1995.

[64] A. P. Hurst and R. K. Brayton. Advantages of Latch-Based Design Under Process Variations. In *International Workshop on Logic & Synthesis*, June 2006.

[65] R. Chen and H. Zhou. Statistical Timing Verification for Transparently Latched Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25:1847–1885, September 2006.

[66] L. T. Pillage and R. A. Rohrer. Asympotic Waveform Evaluation for Timing Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9:352–366, April 1990.

[67] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar. Mathematically Assisted Adaptive Body Bias (ABB) for Temperature Compensation in Gigascale LSI Systems. In *Proceedings of the Asia-South Pacific Design Automation Conference*, pages 559–564, January 2006.

[68] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis (3rd ed.)*. Prentice Hall, Upper Saddle River, NJ, 1992.

[69] Q. Liu and S. S. Sapatnekar. Confidence Scalable Post-Silicon Statistical Delay Prediction under Process Variations. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 492–502, June 2007.

[70] J. P. Fishburn and A. E. Dunlop. TILOS: A Posynomial Programming Approach to Transistor Sizing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 326–328, November 1985.

[71] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F. Lu, and I. L. Markov. Capo: Robust and Scalable Open-Source Min-Cut Floorplacer. In *Proceedings of the International Symposium on Physical Design*, pages 224–226, April 2005.

[72] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits (2nd Edition)*. Prentice Hall, Upper Saddle River, NJ, 2003.

[73] K. C. Chun, P. Jain, and C. H. Kim. A 0.9V, 65nm Logic-compatible Embedded DRAM with 1ms Data Retention Time and 53% Less Static Power than a Power-Gated SRAM. In *Proceedings of the ACM International Symposium on Low Power Electronics and Design*, page 119, August 2009.

[74] Y. Taur and T. H. Ning. *Fundametals of Modern VLSI Devices*. Cambridge University Press, Cambridge, UK, 1998.

[75] D. Foty. *MOSFET Modeling with SPICE*. Prentice Hall PTR, Upper Saddle River, NJ, 1997.

[76] N. S. Kim, T. Austin, D. Baawu, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan. Leakage Current: Moore's Law Meets Static Power. *IEEE Computer*, 36:68–75, December 2003.

[77] S. Mukhopadhyay, A. Raychowdhury, and K. Roy. Accurate Estimation of Total Leakage Current in Scaled CMOS Logic Circuits Based on Compact Current Modeling. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 169–174, June 2003.

[78] E. Castillo. *Extreme Value Theory in Engineering*. Academic Press, San Diego, CA, 1988.

[79] M. F. Neuts. *Matrix Geometric Solutions in Stochastic Models*. Dover Publications Inc, Mineola, NY, 1981.

[80] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1994.