FAST AND ACCURATE STATISTICAL TIMING ANALYSIS OF DIGITAL CIRCUITS FOR TIMING YIELD ESTIMATION BASED ON TRANSISTOR LEVEL SIMULATIONS

by

Alp Arslan Bayrakçi

A Dissertation Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Computer Engineering

Koç University

October, 2010

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

Alp Arslan Bayrakçi

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Co-Chairs of Supervisory Committee:

____

Assist. Prof. Serdar Taşıran

____

Assoc. Prof. Alper Demir

Reading Committee:

____

Prof. Süleyman Özekici

____

Assoc. Prof. Alper T. Erdoğan

____

Assist. Prof. Alkan Kabakçıoğlu

Date: ___________________________

ABSTRACT

As the Integrated Circuit (IC) technology scales down to deep sub-micron regime in terms of sizes of transistors used inside circuits, the IC manufacturing process suffers from circuit parameter variations, which cause uncertainties in the speed of the chips. The statistical variations of manufacturing process have increased to a non-negligible level, which necessitates statistical timing analysis considering the variations. As a result of the parameter variations, each manufactured chip of the same circuit has different parameter values and thus a different speed performance. The manufactured chips, which pass the speed tests are packaged for marketing and others that fail the tests are discarded. One of the main aims of statistical timing analysis is to estimate timing yield, which is simply the fraction of chips that pass the speed tests. Almost all proposed statistical timing analysis methods for digital circuits are block (gate) level methods and they are called statistical static timing analysis (SSTA) methods, as they are direct generalizations of deterministic static timing analysis (DSTA) to the statistical case. However, block level statistical timing analysis lacks accuracy as it contains many approximations. In this thesis, we try to fill the gap for accurate statistical timing analysis based on transistor level circuit simulations. For this purpose, we first propose a new comprehensive statistical timing analysis tool that combines different techniques in the literature for modeling variations and extracting the statistically critical paths in a circuit. But our main novel contribution is timing yield estimation using importance sampling in a novel manner in order to speed up transistor level Monte Carlo (TL-MC) statistical timing analysis for obtaining both an accurate and efficient timing yield estimation method. We test our method on ISCAS'85 circuits and the results show that our IS based yield estimation method improves the speed performance two orders of magnitude on the average.

ÖZETÇE

Tümleşik devre (yonga) teknolojisi, devrelerde kullanılan transistörlerin boyutu bakımından mikron altı rejime indikçe tümleşik devre üretim işlemi, yongaların hız performanslarında belirsizliğe sebep olan devre parametreleri değişkenliklerinden muzdarip hale gelmektedir. Üretim işlemindeki istatistiksel değişkenliklerin göz ardı edilemez seviyelere ulaşması bu değişkenlikleri hesaba katan istatistiksel zamanlama analizini zorunlu kılmıştır. Parametre değişkenliklerinin bir sonucu olarak aynı devreye ait olan üretilmiş her yonga farklı parametre değerlerine ve dolayısıyla farklı bir hız performansına sahiptir. Hız testinde başarılı olan yongalar satış için paketlenirken başarısız olanlar atılır. İstatiksel zamanlama analizinin ana amaçlarından birisi hız testlerini geçecek yongaların oranı olan zamanlama verimini tahmin etmektir.Sayısal devreler için önerilmiş olan istatiksel zamanlama analizlerinin neredeyse hepsi blok (mantık geçidi) düzeyinde çalışan metotlardır ve bunlara istatiksel statik zamanlama analizi ismi verilir, çünkü bu metotlar istatiksel olmayan statik zamanlama analizinin istatiksel duruma doğrudan genellemeleridir. Ancak blok düzeyi istatiksel zamanlama analizi birçok yaklaşım ve tahmin içermesi sebebiyle doğruluktan yoksundur. Bu tezde, transistör düzeyinde devre simülasyonlarına dayalı, doğru istatistiksel zamanlama analizi boşluğunu doldurmaya çalışıyoruz.Bu amaçla, ilk olarak, bir devre içindeki değişkenlikleri modellemek ve istatistiksel olarak kritik olan yolları belirlemek için literatürdeki farklı teknikleri birleştiren yeni ve kapsamlı bir istatistiksel zamanlama analizi aracı öneriyoruz.Ama bizim esas orijinal katkımız, zamanlama verimini doğru ve hızlı bir şekilde tahmin eden bir metot elde etmek için önem örneklemesi yöntemini farklı bir şekilde transistör düzeyi Monte Carlo istatiksel zamanlama analizinin hızını arttırmak için kullanmaktır. Metodumuzu ISCAS85 değerlendirme devrelerinde test ettik ve sonuçlar bizim önem örneklemesi tabanlı zamanlama verimi tahmin metodumuzun, hızı ortalama 150 kat arttırdığını gösterdi.

ACKNOWLEDGMENTS

I would like to express my deepest thanks to my supervisors Assist. Prof. Serdar Taşıran and Assoc. Prof. Alper Demir. I am grateful to them as they had accepted me as their student in the middle of my PhD process. They have helped me continuously and in patience whenever I needed. They have always answered my questions and problems with valuable solutions, which incredibly helped to proceed in my research. I have learned too much from them. I have been glad to be their student and astonished by their enthusiasm, smartness and dedication to research.

I am grateful to The Scientific and Technical Research Council of Turkey (TÜBİTAK) for providing financial support under the National Doctoral Funding Programme.

I would like to thank my previous supervisor Ahmet Akkaş for his trust on me more than my trust on myself. He had always been gracious, encouraging and kind to me.

Thanks to my thesis committee members, Prof. Süleyman Özekici, Assoc. Prof. Alper Tunga Erdoğan and Assist. Prof. Alkan Kabakçıoğlu for the precious time they have spent in reviewing my thesis and the insightful questions they have raised. I also thank Prof. Irşadi Aksun and Prof. Süleyman Özekici as they showed me how a director can also be kind and caring for the students. I would like to thank my colleagues for their support and especially Önder Şuvak for helping me in editing my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

BL	Block (gate) Level
BL-MC	Block Level Monte Carlo
CNF	Conjunctive Normal Form
DAG	Directed Acyclic Graph
DSTA	Deterministic Static Timing Analysis
DTA	Deterministic Timing Analysis
EDA	Electronic Design Automation
IC	Integrated Circuit (Chip)
IS	Importance Sampling
MC	Monte Carlo
NRMSE	Normalized Root Mean Square Error
PDF	Probability Density Function
PDM	Polynomial Delay Model
RMSE	Root Mean Square Error
RV	Random Variable
SAT	Satisfiability
SLE	Stochastic Logical Effort
SSTA	Statistical Static Timing Analysis
STA	Statistical Timing Analysis
STD-MC	Standard Monte Carlo
TA	Timing Analysis
TL	Transistor Level
TL-MC	Transistor Level Monte Carlo

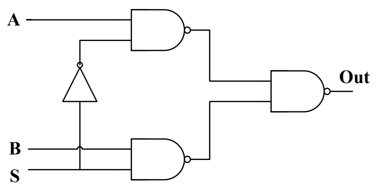
Chapter 1

INTRODUCTION

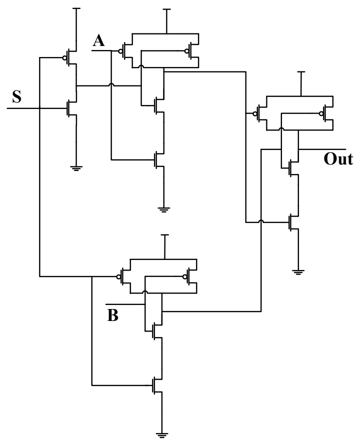
1.1 Digital Circuits

Digital circuits are based on binary logic (logic-0 and logic-1). They constitute the mainstream of the electronic market. They are manufactured as *integrated circuits* (IC) or *chips* on the surface of a thin substrate of a semiconductor material (silicon). Figure 2.9 shows an example manufactured and packaged IC that belongs to Nvidia company. Today, digital circuits are almost everywhere. Processors, micro controllers, most of the computer hardware are examples of digital ICs. They are used inside cars, all household appliances, cell phones, camcorders, automated production line systems and almost in any electrical device.

Block (gate) level (BL) and Transistor Level (TL)



(a) Gate level representation



(b) Transistor level (TL) representation

Figure 1.1: Multiplexor circuit

Digital circuits are composed of logic gates[1], which are made up of transistors. Using the logic gates; decoders, encoders, multiplexors, adders, arithmetic logic units (ALU), registers and control units are designed as basic building blocks of digital ICs. Digital ICs can be analyzed either at block (gate) level (BL) or, for a more detailed analysis, at transistor level (TL). Figure 1.1 shows a multiplexor circuit representation both at gate level and at transistor level.

Importance of Speed and The Trends

Speeding up digital circuits is a major optimization task that facilitates the marketing and enables very complex functions to be implemented by digital circuitry. Timing Analysis (TA) aims to estimate speed performance of circuits before manufacturing even at very early design stages in order to optimize the circuits accordingly. Therefore, TA of ICs is essential for IC designers. There is an extreme competition to manufacture faster circuits and the designers and manufacturers work very hard to accelerate the designs.

As a result of those efforts for the last 50 years, the number of transistors that can be placed inexpensively on an integrated circuit has doubled approximately every two years, a trend well predicted by and named after Intel co-founder Gordon E. Moore as *Moore's law*. The trend has been practiced for more than half a century until now. The transistors are getting smaller and smaller and as a result the speed performance of the ICs and the functionality that can be embedded on an IC with the same size are increasing with a similar trend. This fast improvement trend has caused the transistor sizes scale down to 45nm ranges in 2010 from 10,000nm in 1971 (Intel 4004) and more than half a billion transistors (Intel Core2 Quad, 2006) can be placed in one single chip whereas this number was only 2300 in 1971 (Intel 4004). Accordingly, the speed of the processors has increased from 108 KHz in 1971 (Intel 4004) to 3 GHz in 2003 (Intel Pentium 4).

1.2 Statistical Nature of Timing Analysis

Decreasing sizes of transistors result in manufacturing of digital ICs to become much more difficult and prone to variations of parameters like transistor gate length, oxide thickness, doping concentration, supply voltage, temperature and etc. Performance (speed) variability due to the statistical parameter variations and environmental fluctuations has become more significant. Because of these

[1]Logic gates have different types like AND gate, OR gate, NAND gate, NOR gate, NOT gate (inverter) and etc. They are also called cells and the types of logic gates that will be used in the digital IC depends on the utilized cell library.

variations, each manufactured chip (IC) comes with different properties. Manufactured chips have to be discarded if they cannot pass the tests including those concerning the speed. A speed test failure occurs when a manufactured circuit has a speed less than the minimum allowable speed, i.e. speed constraint. The fraction of manufactured chips that pass the timing (speed) tests is called *timing yield* whereas the fraction of manufactured chips that fail is called *timing loss* as they are thrown out. Digital IC designers give their designs to a manufacturing company and they have to pay for each manufactured chip to the manufacturing company. Therefore, they need to estimate the yield and optimize their design accordingly until it reaches the desired yield before manufacturing. Overestimating the timing yield results in an unexpected loss after manufacturing, whereas an underestimation of timing yield results in unnecessary design efforts and loss of time that may even cause marketing failures.

Today, in nanometer regime, the statistical variations have increased to a non-negligible level with a trend inversely proportional to the gate sizes. For instance, according to International Technology Roadmap for Semiconductors (ITRS) 2009 report [1], which estimates the trends in semiconductor manufacturing technology, threshold voltage $3\sigma/\mu$ ratio, where σ is the standard deviation and μ is the mean, is expected to increase up to 50% and for now there are no manufacturing solutions to resolve this.

As a result, there is an increasing need for comprehensive statistical timing analysis methods that consider the statistical parameter variations between different chips called inter-die variations and in the same chip called intra-die variations, topological and spatial correlations, random parameters with non-normal probability density functions (PDF) and non-linearity of the parameter-speed relationships. Another issue that must be covered by statistical timing analysis methods is the extraction of statistically critical paths, which determine the speed performance of the circuit and should be optimized for better speeds. There are many companies like Intel, IBM, Cadence, Synopsis and many academic institutions like Berkeley, Massachusetts Institute of Technology (MIT), University of Minnesota, Carnegie Mellon, which try to develop a statistical timing analysis tool that can estimate timing yield accurately and efficiently. There is a huge amount of research on this topic and there are hundreds of published papers in journals and top tier conferences under the topics like *statistical timing analysis, yield estimation, design for yield*.

1.3 Contributions

In traditional VLSI design methodologies, designers choose to perform fast and approximate timing analysis called deterministic static timing analysis (DSTA) to estimate the speed of their designs and optimize accordingly. However, they also prefer detailed transistor level (TL) circuit simulations as a final verification before timing sign-off because of the accuracy of TL simulations. One would ideally like to perform a similar transistor-level, but statistical timing analysis for timing yield estimation. Taiwan Semiconductor Manufacturing Company (TSMC), a leading chip manufacturer, has already announced the insertion of transistor-level statistical timing analysis into its new reference design flow in order to enhance timing accuracy [62].

We have developed, and describe in Chapter 3, a comprehensive statistical timing analysis methodology. While some elements of this methodology are borrowed from previous work (such as the quad-tree model [2] for capturing spatial correlations in modeling intra-die variations) and not necessarily the most comprehensive implementations, our work presents a statistical timing analysis tool that employs new gate delay models and addresses some other important open problems (such as statistically critical path identification) and offers reasonable and practical solutions.

The main, novel contribution of the work described in Chapter 4 is in devising a unique importance sampling scheme for accelerating timing yield computations based on transistor level Monte Carlo (MC) simulations, which fills a gap in statistical design methodologies and enables final transistor-level verification before timing sign-off. The technique we propose aims to improve the accuracy of the yield estimates obtained from a given number of TL simulations. Alternatively, our improved MC estimator achieves the same accuracy as the standard transistor level Monte Carlo (TL-MC) estimator but at a cost of much fewer number of TL simulations. This is made possible by using importance sampling technique that we combine in a novel manner with a cheap-to-evaluate but approximate gate delay model, which will be explained in Section 3.2. We use the cheap gate delay model to guide the generation and selection of sample points in the parameter probability space in a TL simulation based MC method for timing yield estimation.

The approach proposed in this thesis is based on the premise that, given the magnitude of process parameter variations and the non-linear dependence of gate and circuit speed (or delay) on these variations, the only sufficiently reliable and accurate method for final timing yield verification before sign-off is TL timing simulation. However, we realize that transistor level Monte Carlo estimation of timing yield will never become efficient enough for use in a loop for timing optimizations. As such, the Monte Carlo timing yield estimation technique based on TL simulations we propose in this thesis is meant not as a replacement for fast block level statistical timing analysis methods, but rather, as a complement to them. In fact, in the timing analysis methodology we describe in this thesis, the statistically critical paths on which we perform transistor-level Monte Carlo analysis are identified using a fast but approximate block-based statistical timing analysis technique.

1.4 Outline

Chapter 2 provides an overview of timing analysis with an emphasis on statistical variations and the Monte Carlo technique. The fundamentals of timing analysis, basic definitions and notation, theory of Monte Carlo methods given in this chapter are used throughout the thesis. Chapter 3 proposes a statistical timing analysis methodology, which consists of modeling both inter-die and intra-die variations with spatial correlations using a quad-tree model [2], performing Monte Carlo based block level statistical timing analysis using a polynomial gate delay model (PDM) proposed by us, extracting the statistically critical path candidates and then, in order to determine the true statistically critical paths, eliminating the false paths using a satisfiability method similar to [3]. Chapter 4 presents the main novel contribution of this thesis, which is a transistor level yield estimation method based on well-known variance reduction technique called importance sampling (IS). IS based yield estimation increases the efficiency without decreasing the accuracy so that it can be used as a final accurate yield estimation tool before manufacturing. Convergence analysis for the IS estimator is performed and theoretical expression for the error of the estimator is derived in this chapter. Chapter 5 applies the IS based yield estimation for ISCAS'85 benchmark circuits to test its performance in terms of both accuracy and efficiency. Chapter 6 provides the conclusion and a list of future tasks.

Chapter 2

OVERVIEW OF TIMING ANALYSIS AND MONTE CARLO METHOD

This chapter reviews both deterministic and statistical timing analysis after providing the fundamentals. At the end of this chapter, we provide the preliminaries and a review of Monte Carlo techniques used for timing yield estimation. Section 2.1 not only presents the fundamentals and the definitions used by timing analysis but also compares the timing analysis of combinational and sequential circuits. Section 2.2 provides the previous work for the deterministic timing analysis where the parameter variations are ignored. In that section, the transistor level timing simulation and the static timing analysis methods, which are also at the core of the statistical timing analysis methods, are explained. Also the false path detection problem and the gate delay models utilized by timing analysis tools are reviewed. Section 2.3 clarifies the need for statistical timing analysis by explaining the manufacturing variations and then gives the literature review for statistical static timing analysis (SSTA), which constitutes the majority of research in statistical timing analysis topic. Section 2.4 first provides the preliminaries that consists of the basic definitions and notation used throughout this thesis, the general Monte Carlo method and the general importance sampling method and then it explains the standard Monte Carlo estimation of loss due to the parameter variations. Section 2.4 is especially important as the definitions and notations introduced in it are used by Chapter 3 and 4.

2.1 Fundamentals of Timing Analysis (TA)

When the input of a logic gate makes a transition, the output requires a nonzero interval called *propagation delay* to change accordingly. This is unavoidable because the propagation delay is required by a logic gate in order to charge or discharge the capacitance at its output. Due to the overlapping conductive regions inside the gates, there are capacitances at the input(s) and output of every logic gate, which are called *input capacitance* and *load capacitance*, respectively. Both input and load capacitances are modeled by capacitors connected to the ground. As the gates are connected to each other, the input capacitance of a gate becomes the load capacitance of another gate, whose output is connected to its input. There is another concept related with load capacitance called *fanout*, which is equal to the normalized load capacitance according to the input capacitance of the corresponding gate. Load capacitance or fanout is the major component that determines the propagation delay of a gate and therefore utilized even by the most elementary gate delay models.

Figure 2.1 shows the input and the output voltage waveforms of an inverter. As the gate is an inverter, when the input signal makes a transition from low voltage (logic-0) to high voltage (logic-1), the output signal makes a transition from high to low and vice versa. The time difference between the instances when the input and the output signals reach half of the high voltage is called *propagation delay* of the corresponding gate or *gate delay*. According to the transition of the output signal, the propagation delay is called low to high propagation delay (tp_{L2H}) or high to low (tp_{H2L}) propagation delay. Also the *input slope* is demonstrated in Figure 2.1. Input (output) slope is the time required for the input (output) of a gate to make a transition between 10% and 90% of high voltage. Similar to the propagation delay, according to the direction of the transition there are low to high (InS_{L2H}), high to low (InS_{H2L}) input slopes and low to high ($OutS_{L2H}$), high to low ($OutS_{H2L}$) output slopes. These basics are used in the timing analysis of digital circuits and will be referred throughout the thesis.

There are two types of digital circuits: *combinational* and *sequential*. Below, these two types of circuits are explained and discussed from the perspective of timing analysis.

Combinational Circuits

In combinational circuits or combinational logic, the outputs of the circuit depend only on the current inputs of the circuit. There is no loop and no memory element like register or flip flop in combinational circuits. For this reason they can be represented as directed acyclic graphs (DAG) as will be referred later in Section 2.2.2. The multiplexor in Figure 1.1 is a simple example for combinational circuits.

TA in Combinational Circuits

A combinational circuit generally consists of many logic gates[1] and the propagation delays of the gates that are connected to each other accumulate until an output of the circuit is reached. A *path* in a combinational circuit is simply an unbroken route starting from an input of the circuit and ending

[1]For instance, there are more than a thousand gates on the average for the ISCAS'85 benchmark circuits used in Chapter 5.

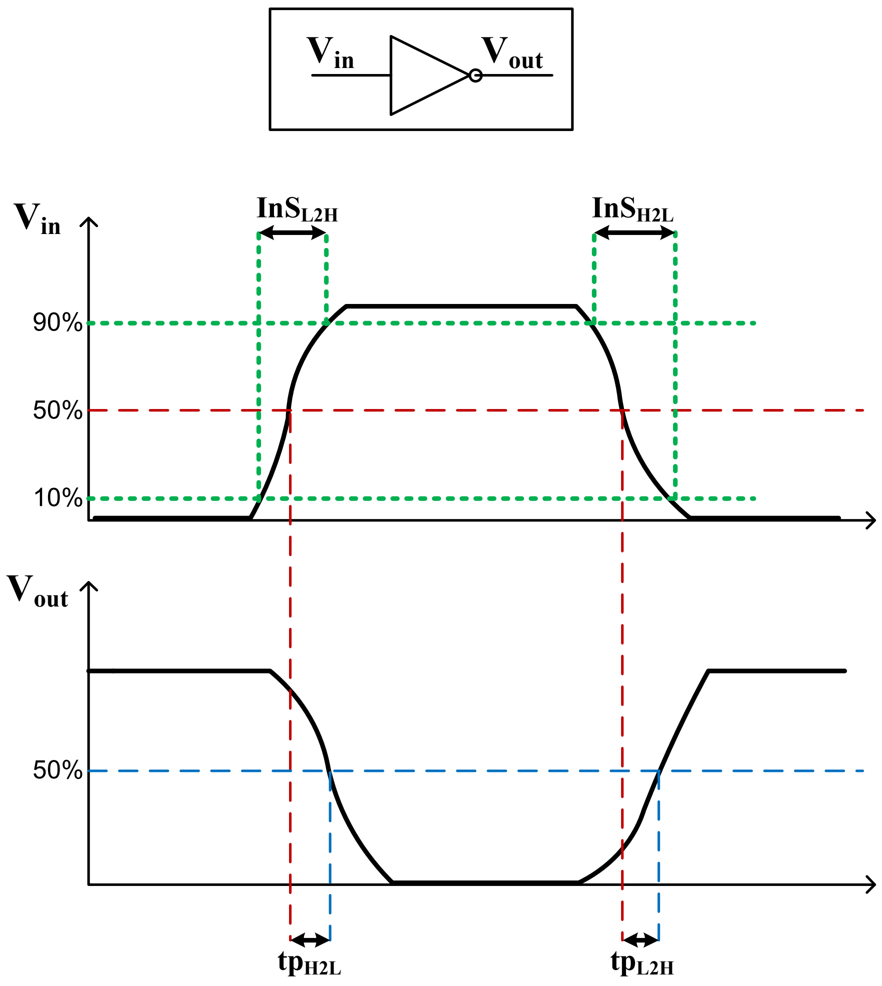


Figure 2.1: Propagation delay and input slope for an inverter

at an output of the circuit. For instance, in Figure 1.1(a), the route from **A** to **Out** is a path of the multiplexor circuit. It is clear that as there are no loops in combinational circuits there is no path with infinite number of gates. Similar to tp_{L2H} and tp_{H2L} defined above, each path has a high to low and low to high propagation delay between its input and output. The maximum of these two delays is called *path delay*. The delay for a path that consists of logic gates can be computed by TL timing simulation described in Section 2.2.1 or by simply adding the individual propagation delay of each gate on the path as explained in Section 2.2.2. The delay of a combinational circuit (*circuit delay*) is equal to the maximum of its path delays as, practically, there are many paths in a combinational circuit. The path having the maximum delay is called *critical path*[2] and it determines the delay of the circuit. The circuit delay limits the speed of the circuit and therefore, designers try to optimize the critical path in order to speed up the circuit. Timing analysis (TA) of combinational circuits aims to estimate the circuit delay and extract the critical paths in a circuit in order to optimize the circuit to obtain better speeds. TA is applied at many different levels of design by automated software tools developed by Computer Aided Design (CAD) engineers.

Sequential Circuits

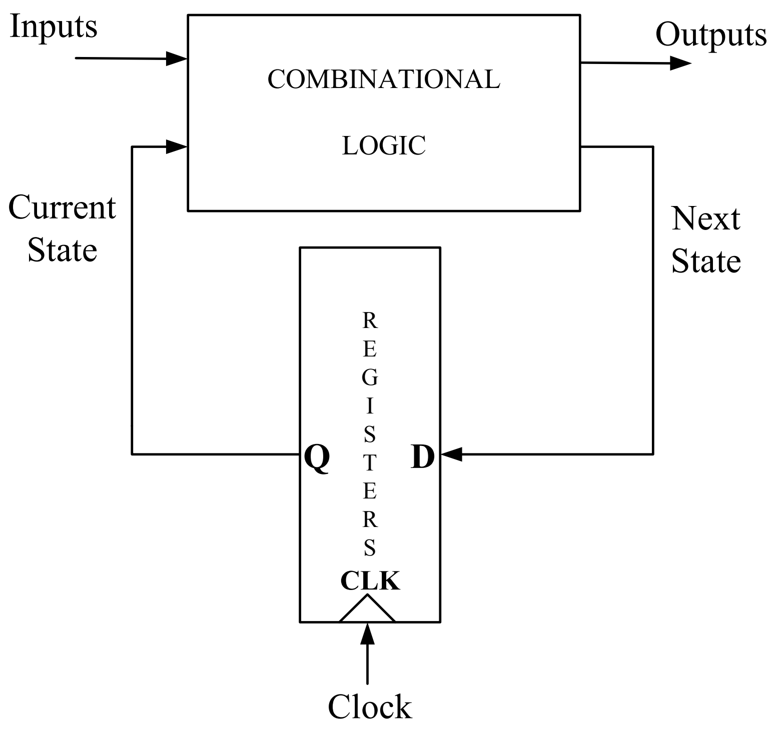


Figure 2.2: Block diagram of a sequential circuit

In sequential circuits, the outputs of the circuit depend not only on the current values of the inputs but also on the previous values of the inputs. A sequential circuit has different states, and the current state determines the current response of the circuit to the inputs. The current state of the circuit is determined according to the past inputs and past states of the circuit. Memory elements

[2]A more formal definition of critical path will be made in Section 2.2.3.

like registers are used in sequential circuits to store the state of the system. Registers are connected to an input signal called clock, which determines when the content of the registers will be switched. Figure 2.2 shows a general structure for a sequential circuit. The registers store the next state, which is computed by the combinational circuit according to the current state and the current inputs of the circuit. When a clock edge[3] is detected, the value inside the registers is loaded to the output of the registers as the current state and the registers store the next state value. Then, a new next state is computed according to the current state and the inputs. The time difference between two consecutive clock edges that the registers switch their content is called clock period. The clock frequency, which is equal to the reciprocal of the clock period, represents the speed of sequential circuits. For instance, a 3 GHz processor has a clock frequency of 3 GHz, which means that the registers switch their content 3 billion times in one second.

TA in Sequential Circuits

Analyzing the timing of a sequential circuit boils down to analyzing the timing of its combinational part. Without loss of generality, we can assume that registers switch content at positive clock edges. In order to operate accurately, registers need the data at their inputs (D in Figure 2.2) to be stable for a time period called *setup time*, i.e. t_{setup}, before the positive edge arrives. Also, there is a delay, t_{CLKtoQ}, between the positive clock edge arrival and the switching of the data at the output of the register (Q in Figure 2.2). In other words, after the positive clock edge arrives, t_{CLKtoQ} time is required to have the new data at Q output of the register. Then, t_{comb} amount of time, which is equal to the combinational circuit delay, is required for the combinational logic to compute the new outputs and the next state. This computation should finish before t_{setup} time from the next positive edge. This results in the constraint shown in (2.1)

$$T_{CLK} > t_{CLKtoQ} + t_{comb} + t_{setup} \tag{2.1}$$

where T_{CLK} is the period of the clock signal, i.e. the time difference between two consecutive positive edges as shown in Figure 2.3. The constraint in (2.1) shows that the main component that determines the clock period is the combinational circuit delay as the other two delays, i.e. t_{CLKtoQ} and t_{setup}, are well-known and already optimized constant delays, which are much smaller than

[3]Clock edge means a high to low (negative) or low to high (positive) transition of the clock signal. Registers are triggered either by positive or by negative clock edge.

t_{comb}. T_{CLK} solely determines the speed of the circuit, which is simply equal to the maximum clock frequency, i.e. $1/T_{CLK}$[4]. As a result, TA of a sequential circuit, reduces to TA of the combinational part in it. For this reason, TA of combinational circuits is focused on throughout this thesis. The particular problems of sequential circuits like clock slew and skew are outside the scope of this thesis.

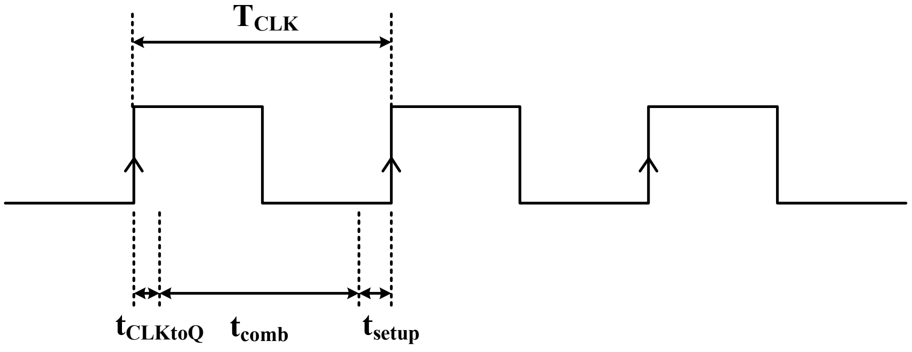


Figure 2.3: Clock signal and delays in a sequential circuit

2.2 Deterministic Timing Analysis (DTA)

Timing analysis aims to estimate the circuit speed (performance) by computing the worst case delays as explained briefly in the previous section. Until the 1990s, the variations in digital circuit parameters were relatively small and therefore, they were ignored. All circuit components were assumed to have deterministic parameters, which result in deterministic gate delays and consequently deterministic circuit delays. As a result, deterministic timing analysis (DTA) was preferred by the designers for estimating the speed of the circuits when the variations were ignored.

DTA can be classified into two: transistor level (TL) timing simulation and block level (BL) deterministic static timing analysis (DSTA). Timing simulation at transistor-level provides precise estimates but it is inefficient to be used incrementally to improve the designs. On the other hand, DSTA, which is performed at a higher level, i.e. gate (block) level, is a widely adopted method with linear run time complexity. However, it cannot totally replace the timing simulations due to the lack

[4]For instance, a processor having 2 GHz clock has a clock period of 2 ns.

of accuracy when compared with TL timing simulation. As a result, the electronic design automation (EDA) engineers prefer such a scheme for deterministic timing analysis: They perform DSTA multiple times at different levels of design in order to optimize the circuit performance accordingly and to extract the path(s), which are responsible for the slowness of the circuit, i.e. the critical path(s). Then, before handing over their designs to manufacturing, they apply TL timing simulation especially for the extracted critical path(s) in order to obtain the most accurate estimation for the speed performance of the circuit.

The description and details of TL timing simulation are given in Section 2.2.1 and the DSTA is explained in Section 2.2.2. Section 2.2.3 gives the definitions for critical path and false path and then focuses on false path problem. Section 2.2.4 reviews gate delay models in deterministic case and explains the logical effort gate delay model in more detail.

2.2.1 Transistor-Level (TL) Timing Simulation

Nodal Analysis

A digital circuit consists of transistors, resistors, capacitors and inductors at the most elementary level. Transistor level (TL) circuit simulation directly analyzes such a digital circuit by combining three types of well-known equations enumerated below into a linear matrix algebra:

1. Kirchhoff Current Law (KCL): The sum of currents leaving a node is zero. If a current is sinking to the node it has negative value.

2. Kirchhoff Voltage Law (KVL): The voltage around a cycle in the circuit is always equal to zero.

3. Device current-voltage relationships: For instance, resistors have $V = I \times R$ where I is the current passing through that resistor, V is the voltage over that resistor and R is its resistance.

The most famous TL circuit simulator is called SPICE (Simulation Program with Integrated Circuits Emphasis), which was written by Larry Nagel and released in 1972 [4]. Today, SPICE has become an electronic design automation (EDA) industry standard and large companies still continue to develop their own proprietary simulators [5]. SPICE uses *nodal analysis* which solves the three types of equations given above using linear matrix algebra. For instance, Figure 2.4 shows a simple circuit that consists of only linear elements and two nodes (node 1 and 2) and the corresponding linear matrix equation for that circuit is shown in (2.2).

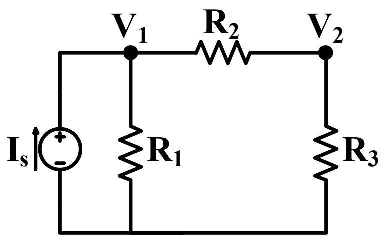


Figure 2.4: A very simple circuit with only linear elements

$$\begin{bmatrix} \frac{1}{R_1}+\frac{1}{R_2} & \frac{-1}{R_2} \\ \frac{-1}{R_2} & \frac{1}{R_2}+\frac{1}{R_3} \end{bmatrix} \times \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} I_s \\ 0 \end{bmatrix} \tag{2.2}$$

Transient Analysis Flow

If the aim of the circuit simulation is the analysis of the timing (transient) behavior of the circuit, then it is called *transient analysis*. Transient analysis is used to detect the voltage waveforms of any node in the circuit for the desired period of time. From this information the propagation delay of the circuit can be computed very easily and accurately as described in Section 2.1. For this purpose, transient analysis of SPICE first discretizes the time domain and then for each discrete time instance, it solves the nodal matrix equation in (2.2) in order to compute voltages at each node of the circuit.

The overall transient analysis methodology that considers both non-linearities and differential relationships is shown in Figure 2.5. To perform transient analysis, SPICE discretizes the time period of interest into time instances according to the fluctuation speed of the voltage signals in the circuit. If the fluctuation speed is high, it uses smaller time steps in discretization. For each time instance, SPICE performs the same operation described as follows: It uses numerical integration methods like Backward Euler or Trapezoid in order to convert differential equations into non-differential linear/non-linear equations. Then it linearizes the non-linear devices at their corresponding points of operation for being able to construct a linear matrix equation of the form in (2.2). It uses Newton-Raphson method to linearize non-linear equations. The resultant system of linear equations are solved iteratively until Newton-Raphson iterations converge. At each iteration, the system of linear equations is solved by using LU factorization to compute voltages at each node and currents at each branch of the circuit. This iterative procedure is shown as the smaller loop in Figure 2.5. As the bigger loop in this figure shows, this procedure is performed for each time instance until the end of the interested time period is reached [6].

After this transient analysis flow terminates, the propagation delay of the circuit can easily be computed as the voltage value at each node and for any time instance in the interested time period is computed. The unknown voltage values between any two time instances can easily be computed by linear interpolation. There is no significant loss in this interpolation because time steps are adjusted by SPICE such that they are small if the voltage signal fluctuates fast. In our work, a simple perl script taking the SPICE program's output as an input argument is used to perform this interpolation and determine the propagation delay of the circuit.

Computational Complexity

The number of iterations of a TL timing simulation flow depends on two loops; the bigger loop and smaller loop in Figure 2.5. The number of iterations due to the bigger loop depends on the number of time instances, which depends on the signal fluctuation speed and the length of the interested time period. The number of iterations due to the smaller loop depends on the convergence of Newton-Raphson iterations. Even in some situations, the smaller loop can fail to converge unless the convergence conditions are relaxed by the user. The most complex operation in the heart of these two loops is LU factorization used for the solution of the system of linear equations similar to shown in (2.2). LU factorization has a worst case cost of $O(n^3)$, where n is the total number of nodes and branches in the circuit. However, empirically in typical circuits, the cost can be at the order of $O(n^{1.7})$ due to the sparsity of the related matrix. As a result, the computational complexity of a transient analysis flow can be written as shown in (2.3).

$$O(N_t \times N_{NR} \times n^3) \tag{2.3}$$

where N_t is the number of time instances as a result of the discretization performed by SPICE, N_{NR} is the average number of Newton-Raphson iterations that is computed by averaging the number of NR-iterations used for each time instance and n is the total number of nodes and branches in the circuit. In typical circuit, N_{NR} is usually a couple of iterations and N_t, which is usually in the order of hundreds, actually depends on the length of the interested time period and the oscillations of the

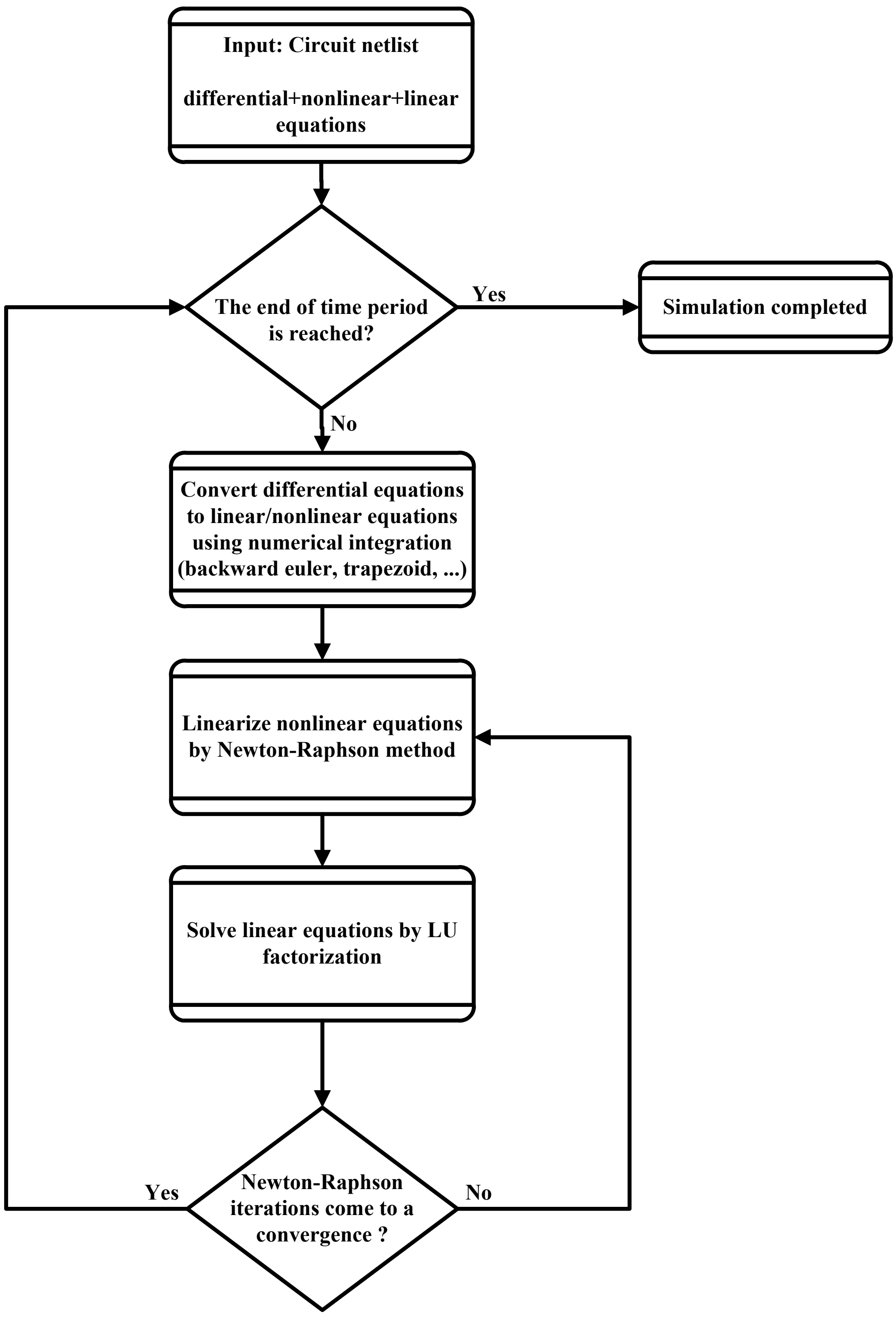

Input: Circuit netlist
differential+nonlinear+linear equations
The end of time period is reached?
Yes
Simulation completed
No
Convert differential equations to linear/nonlinear equations using numerical integration (backward euler, trapezoid, ...)
Linearize nonlinear equations by Newton-Raphson method
Solve linear equations by LU factorization
Yes
Newton-Raphson iterations come to a convergence ?
No

Figure 2.5: SPICE transient analysis flow

voltage waveforms.

Conclusion

The TL timing simulation is the most precise way for solving general circuits. For practical purposes, the results of TL timing simulation are considered to be the exact solution to any circuit provided that the sophisticated precise models for the circuit devices are utilized [6]. Despite its accuracy, this method is the slowest way of estimating the transient behavior of a digital circuit [6, 5]. Another disadvantage of circuit simulation is the necessity of exactly determining the input signals to the circuit. The input signal combination that result in a worst case delay should be known in order to detect the worst case delay of the circuit. Otherwise, all input combinations should be tested, which makes $2^2 n$ simulations where n is the number of inputs.

2.2.2 Deterministic Static Timing Analysis (DSTA) at Gate (Block) Level

Importance of DSTA

DSTA is a powerful method, which has been widely adopted in EDA community for more than two decades. It estimates the propagation delay of a combinational circuit by utilizing an algorithm called critical path method (CPM) that is used in project management. Although it is not as accurate as TL timing simulation, it is a very fast method with linear run time in terms of number of logic gates and connections in the circuit. The term ”static” is used because DSTA does not need any input vectors, which are needed in TL timing simulation. In time, DSTA has become a mature tool, which is able to take into account many aspects of timing analysis. As a result, DSTA is in the heart of almost all timing analysis and optimization tools.

Combinational Circuit $\rightarrow$ Directed Acyclic Graph (DAG)

DSTA runs at block (gate) level by first converting the circuit into a directed acyclic graph (DAG) structure. At gate level, a combinational circuit, for it does not have any loops, can be represented by a DAG as referred in Section 2.1. Figure 2.6 shows an example circuit schematic and the corresponding DAG representation. In this graph, $G(V,E)$, each vertex corresponds to a gate and each edge between the vertices corresponds to a hardwired connection between the gates, whereas the leftmost edges correspond to the primary inputs and the rightmost edge corresponds to the primary output(s). For instance, for the circuit in Figure 2.6, the primary inputs are from i_0 to i_7 and the primary output is labeled as *out*. Any route from a primary input to a primary output is called a *path*. DSTA is applied on this DAG structure, which is very similar to the activity networks in the project management.

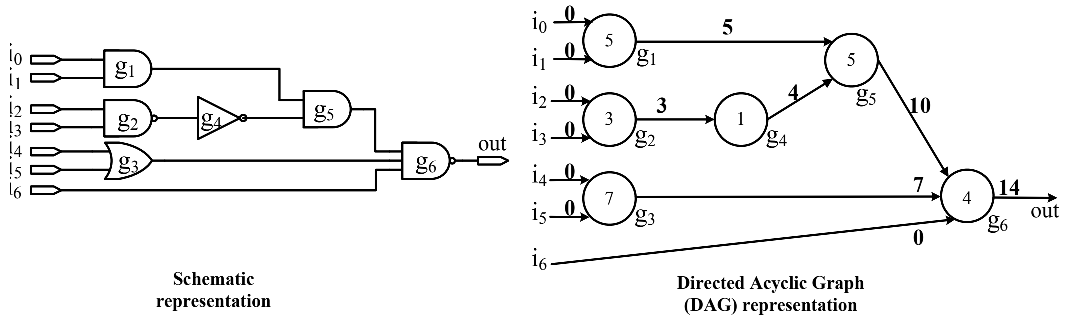


Figure 2.6: DAG representation of a combinational circuit

Circuit Delay and Arrival Time from DSTA Perspective

Transitions at the primary inputs of a circuit cause signal transitions to propagate through the circuit until the propagation reaches the output of the circuit. It was explained in Section 2.1 that each path has a propagation delay between its input and output. Different than TL timing simulation of Section 2.2.1, DSTA runs at gate level and computes the path delay by simply adding the individual gate delays on the path. As described in Section 2.1, the circuit delay is the maximum of the delays of all paths in the circuit. In other words, circuit delay is the maximum propagation delay that a circuit can practice.

Another very important concept utilized by DSTA is *arrival time*. Assuming that the primary inputs are altered at time is equal to zero, then *arrival time* at an edge represents the time when the voltage signal becomes ready at that edge by reaching the required voltage value, which is the fifty percent of the high voltage as explained in Section 2.1.

Operation of DSTA

DSTA computes not only the circuit delay but also the arrival times for all edges in the timing DAG. Actually it computes the circuit delay by computing the arrival times from left to right in a topologically sorted manner, where topological sorting is a method of arranging the vertices in a DAG, as a sequence, such that no vertex appear in the sequence before its predecessor. As a result, DSTA not only visits a vertex (gate) only once but also it never visits a vertex before visiting its predecessor.

DSTA starts from the leftmost vertices connected to the primary inputs and it performs the following operation for each vertex, until it processes all vertex: it first compares the input arrival times to the vertex, in order to find the MAXimum of them, then SUMs the gate's own propagation delay with this maximum input arrival time to compute the output arrival time for that vertex[5]. DSTA can compute all arrival times in the DAG by performing these MAX and SUM operations for each vertex in a topologically sorted manner from left to right. After DSTA is performed, the path having the biggest arrival times is the *topologically longest path* in the circuit. DSTA sets the delay of the topologically longest path as the circuit delay, which is simply the maximum or worst case propagation delay of the circuit.

In Figure 2.6, the numbers inside the vertices of DAG represent the propagation delays of the corresponding gates. Section 2.2.4 gives the details of gate delay models used to find the propagation delay of a gate. Bold numbers on the edges represent the arrival times, where all the primary inputs to the circuit are assumed to make a transition at time is equal to zero. For example, the DSTA operation on gate $g5$ can be summarized as follows: DSTA first examines the input arrival times of $g5$, takes the MAXimum of them, which is 5 then SUMs 5 with $g5$'s propagation delay, which is also 5. The resultant 10 is the output arrival time of $g5$. When DSTA performs the same operation in a topologically sorted manner for all vertices, all arrival times in the circuit are computed. The arrival time for the output of the circuit, which is 14 in this example, is the circuit delay computed by DSTA. The path starting from the primary input i_0 (or i_1) and goes through the gates g_1, g_5 and g_6 respectively is the topologically longest path in this circuit, which determines the propagation delay of the circuit.

If the desired propagation delay for the circuit is known beforehand, then assigning this desired circuit delay value to each primary output in the circuit and applying the same algorithm but this time by reversing the edges of the graph and performing subtraction instead of summation, the *required*

[5]This is true for the most elementary DSTA analysis when the slopes of the input signals are not considered. When input slopes are considered, first, the SUMmation of each input arrival time with the gate's own propagation delay is computed, where the gate's delay is computed according to the slope of the corresponding input signal. Then, the MAXimum of the resultant summations should be set as the output arrival time.

times at each edge of the graph can be computed easily. The difference between the required times and the arrival times is called *slack*. Having positive slacks in the whole graph means that there is no problem in meeting the timing requirements whereas having paths with negative slacks shows that the requirements are not met for the corresponding paths in the graph. Then, designers focus on these paths in order to satisfy the timing requirements.

Conclusion

DSTA is an approximate algorithm, which gives rough circuit delay estimates. It cannot detect false paths, which will be discussed in the next section. As a result, while DSTA is a successful and efficient algorithm, which is applied after several different levels of digital design flow like synthesis, logic optimization or placement, the state-of-the-art still does not allow DSTA to replace TL timing simulation completely due to its weaknesses [7].

2.2.3 Critical Path and False Path Detection

True (sensitizable) Path, False (unsensitizable) Path and Critical Path

Each path's input is a primary input of the circuit and its output is a primary output of the circuit. A path is called a *true (sensitizable) path* or equivalently, it is said to be *sensitized* if a transition, either from high to low or from low to high, at its input results in a transition at its output because of the signal that propagates through that path. The paths, which cannot be sensitized (activated) for any input assignment, are called *false (unsensitizable) paths*. There is no signal propagation along these paths whatever the circuit input assignment is and therefore, they are not responsible for the delay of the circuit in any case. True and false path concepts can better be realized by the examples explained below. Detection of the false paths in a circuit is called *false path problem*, which is an NP-complete problem [8].

Circuit delay is defined as the delay of the longest true path, i.e. the true path having the maximum delay in the circuit. This longest true path is called the *critical path* of the circuit as it defines the actual worst case delay of the circuit. As a result, the topologically longest path of a circuit found by DSTA, is a critical path only if it is a true (sensitizable) path. This critical path definition is different from the critical path definition in Section 2.1 as the sensitization of a path was ignored in the previous definition.

True Path Example

For instance, in Figure 2.6, the topologically longest path is $i_0 - g_1 - g_5 - g_6 - out$. Suppose that all inputs other than i_0 has a logic value of 1 and i_0 is at logic 0. In this case, if i_0 makes a transition from 0 to 1, the output of both g_1 and g_5 make a transition from 0 to 1 and the output of g_6 makes a transition from 1 to 0. Therefore, the transition of i_0 propagates down through the path $g_1 - g_5 - g_6$ and trigger the output to make a transition, which means that the path $i_0 - g_1 - g_5 - g_6 - out$ is a true (sensitizable) path. As this true path is the topologically longest path, it is also the critical path and its delay is equal to the maximum propagation delay of the circuit. In this case, the result of DSTA is correct. However, when the topologically longest path of a circuit is a false path, the result of the DSTA becomes wrong. Figure 2.7 demonstrates such a circuit.

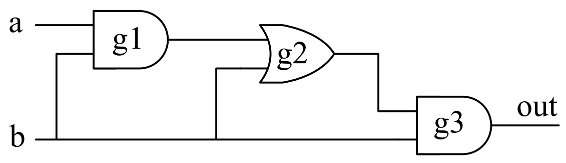


Figure 2.7: A simple example for a false path ($a - g1 - g2 - g3 - out$)

False Path Example

Figure 2.7 shows a very simple example to demonstrate false paths. Assume a delay of 1 ns for each gate in this circuit. If DSTA was performed for this circuit, it would find the paths $a - g1 - g2 - g3 - out$ and equivalently $b - g1 - g2 - g3 - out$ as the topologically longest paths and therefore, set the delay of the circuit as 3 ns. However, this would be wrong. Assume that input b has become logic 0 at time zero ($t = 0$). Then, the output of gate $g3$ (out) would become 0 immediately at $t = 1$ whatever a is. On the other hand, if input b has become 1 at $t = 0$, then the output of $g2$ becomes 1 at $t = 1$ again whatever a is. Because b is 1 at $t = 0$ and the output of $g2$ is 1 at $t = 1$, the output of $g3$ becomes 1 at $t = 2$. Actually, the transitions of a do not affect the output out. Therefore, the scenario will be either of the two situations referred above for all possible transitions of (a,b) pair, which is equal to $4 \times 4 = 16$ as a,b pair can take 4 different values[6]. As a result, the actual worst case delay for this circuit is 2 ns instead of 3 ns, which was estimated by DSTA. Also the

[6]These values are 00, 01, 10 and 11.

paths $a - g1 - g2 - g3 - out$ and $b - g1 - g2 - g3 - out$ are the false paths, as these paths are not responsible for the circuit delay. The critical path for this circuit is $b - g_2 - g_3 - out$, which is the longest true path.

Notations Used by False Path Detection Methods

As a result, false paths in a circuit should be detected in order not to waste efforts to optimize a false path and not to overestimate the circuit delay. The methods in false path detection literature generally label an input of a gate as having either a *controlling* or a *non-controlling* logic value and being a *side-input* or an *on-input*. An input to a gate has a controlling value if it can solely determine the output of the gate whatever the values of the other inputs to the same gate. Otherwise, the input has a non-controlling value as the value of the output depends also on the values of the other inputs. The controlling and non-controlling values for a gate depends on the type of the gate. For instance, a logic-1 value immediately triggers the output of an OR gate to 1 whatever values the other inputs to the OR gate have. On the other hand, a logic-0 value cannot trigger the output of an OR gate to 0 unless all other inputs are at logic-0. Consequently, logic-1 is a controlling value and logic-0 is a non-controlling value for an OR gate. Similarly, logic-1 is a controlling value for NOR gate and a non-controlling value for AND and NAND gates whereas logic-0 is a non-controlling value for NOR gate and a controlling value for AND and NAND gates. On the other hand, one input gates like inverter have no controlling and non-controlling values as the output of these gates always make a transition if their input makes a transition. Similarly, XOR and XNOR gates have no controlling or non-controlling values, because any transition (high-to-low or low-to-high) at any input of an XOR or XNOR gate result in a transition at its output. The second important concept pair used by false path detection methods is side input and on-input. For each gate on a path; if an input of the gate is on the path, it is called *on-input* and if it is not on the path, then it is called *side input*.

False Path Detection Using Static Sensitization

There has been an extensive research on false path detection problem since 1980s, and several sensitization conditions had been proposed for detecting whether a path is false, i.e. unsensitizable [9, 10, 11, 12, 13]. These methods propose different conditions for a path to be a true (sensitizable) path, which are called sensitization conditions. If a sensitization condition is satisfied by a path, then this path is called a sensitizable (true) path and otherwise, it is called a false path according to that sensitization condition.

One of the oldest and fastest sensitization condition used for false path detection is *static sensitization* [9]. It is called ”static” as it is a delay independent method ignoring the arrival times of the signals. For a path to be *statically sensitizable*, an input vector must exist such that all side inputs for that path take non-controlling values. There are no conditions for the inverters or XOR/XNOR gates on the path, because inverter does not have side input and XOR/XNOR gate makes an output transition whatever the values of its side input(s) are.

For instance, the path $i2-g2-g4-g5-g6-out$ in Figure 2.6 is a statically sensitizable path, because all side inputs take non-controlling values when a logic 1 is applied for all inputs other than i_2 to the circuit[7]. For such an input assignment, if i_2 is toggled from 0 to 1, this transition propagates through $g2-g4-g5-g6$ and the output toggles from 1 to 0. A *statically unsensitizable* path is $a-g1-g2-g3-out$ in Figure 2.7. Side inputs of $g1$ (AND gate), $g2$ (OR gate) and $g3$ (AND gate) are all connected to the same signal, called b. The non-controlling value for an AND gate is 1 whereas for an OR gate it is 0 as referred above. Therefore, all side inputs for this path cannot have non-controlling values at the same time. That violates the static sensitization condition and this path is detected as a false path under static sensitization condition.

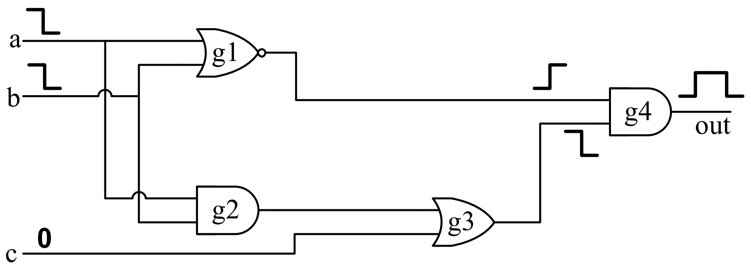


Figure 2.8: A statically unsensitizable true path ($a-g2-g3-g4-out$)

Static sensitization condition is a sufficient condition, but it is not a necessary condition [14]. In other words, if a path is statically sensitizable, then it is a true path but, in some situations, a true path may not be statically sensitizable. For instance, the path $a-g2-g3-g4-out$ in Figure 2.8 is a true path although it is statically unsensitizable. The side-input for gate $g2$ is b and the side-input for gate $g4$ is $\neg a.\neg b$, therefore the side-inputs for both $g2$ and $g4$ cannot be at non-controlling values

[7]Changing one of the i_4 and i_5 to 0 does not change anything in terms of static sensitization.

at the same time, which shows that this path is a statically unsensitizable path. However, assuming a 1 ns delay for each gate and applying high to low input transitions simultaneously at $t = 0$ results in a glitch at the output which stabilizes at $t = 3$, as shown in Figure 2.8. Therefore, the path $a - g2 - g3 - g4 - out$ is a true path as it determines the delay of the circuit as 3 ns although it is a false path under static sensitization condition. Therefore, static sensitization may underestimate the delay. Nevertheless, it does not perform bad in most of the practical cases. The results in [8,15] indicate that static sensitization does not underestimate the delay for any of the circuits in $ISCAS'85$ benchmark [16].

2.2.4 Gate Delay Modeling and Logical Effort

Deterministic Gate Delay Models

Performing DSTA, described in Section 2.2.2 requires an approximate gate delay model to determine gate delays, which were represented as numbers inside vertices in Figure 2.6. For the deterministic case, where the gate parameters are fixed, the delay of a gate mainly depends on the input slope (InS) and output load capacitance (C_L) of the gate. There are different strategies preferred by different companies:

- An obvious way of representing gate delays would be a look-up table, which has a delay value entry for each gate type and under different capacitive loads and input slopes. If a delay value corresponding to a capacitive load or input slope, which is not represented in the table is required, linear interpolation can be used. Therefore, the accuracy of utilizing such a delay look-up table depends on the number of entries.

- Alternatively, the delay of a gate can be represented by equations of the form [6]:

$$a_0 + a_1 C_L \tag{2.4}$$

$$a_0 + a_1 C_L + a_2 InS \tag{2.5}$$

$$a_0 + a_1 C_L + a_2 InS + a_3 C_L . InS \tag{2.6}$$

$$(a_0 + a_1 C_L + a_2 {C_L}^2 + \ldots + a_m {C_L}^m)(b_0 + b_1 InS + b_2 InS^2 + \ldots + b_n InS^n) \tag{2.7}$$

(2.4) is a form of logical effort delay formula, (2.5) is a basic linear equation, which also includes input slope, (2.6) and (2.7) are respectively *non-linear delay model* (NLDM) and *scalable polynomial delay model* (SPDM) used by Synopsys [6].

Logical Effort Formalism

The logical effort formalism is a fast and efficient way of determining the delay of a path in a digital circuit. The path delay is simply the sum of the delays of the gates on the path, and the delay of a logic gate r is approximated as

$$d_r^{LE} = \tau\, d \tag{2.8}$$

where d_r^{LE} is the absolute delay of a gate measured in seconds, τ is the delay of a parasitic-capacitance-free *reference inverter* driving another identical inverter, and d is the delay of the logic gate expressed in units of τ. The d factor in (2.8) models the gate delay and is given by

$$d = (p + g\,h) \tag{2.9}$$

where p represents the intrinsic (parasitic) delay, g is the logical effort, and h is the electrical effort or electrical fan-out. Logical effort g for a logic gate is defined as the (unitless) ratio of its (per) input capacitance to that of an inverter that delivers the same output current. Thus, logical effort g, is a measure of the complexity of a gate. It depends only on the gate's topology and is independent of the size and the loading of the gate. Parasitic delay p expresses the intrinsic delay of the gate due to its own internal parasitic capacitance, and it is largely independent of the sizes of the transistors in the gate. Parasitic delay p, is also a unitless quantity, it is expressed in units of τ. The electrical effort h is the ratio of the load capacitance of the logic gate to the capacitance of a particular input [17].

Logical effort model in (2.8) is a very fast model and the linear relationship between the electrical fanout and the gate delay is a reasonable assumption. However, it does not consider the effect of input slope, which may result in far off delay estimates especially for the cases when the input slope is too slow or too fast.

2.3 Statistical Timing Analysis

The impact of manufacturing process variations has increased with current fabrication technologies in nanometer regime that has scaled down to 45nm scales. These process variations result in physical parameter variations which cause electrical parameter variations and finally gate and circuit delay variations. As a result, not all of the manufactured chips can satisfy the timing requirements where the failing ones are totally discarded. This necessitates timing analysis considering statistical variations in order to modify the design accordingly until it meets the timing requirements. There are many methods proposed to fill the gap at statistical timing analysis especially during the last 15 years. Most of these methods concentrate on different statistical versions of DSTA explained in Section 2.2.2.

Section 2.3.1 explains the parameter variations in detail and investigates the need for statistical timing analysis. Section 2.3.2 gives a literature review for the statistical static timing analysis (SSTA) methods, which constitute almost all research on statistical timing analysis topic.

2.3.1 The Need for Statistical Timing Analysis

Die and Wafer

The integrated circuit (IC) manufacturing process groups a number of identical circuits, each of which is called *die* or *chip*, onto a single *wafer*. After manufacturing process terminates, each die on the wafer is packaged separately if it can pass all tests and it is discarded otherwise. The tests include timing verification to decide whether the die satisfies the speed requirements. Figure 2.9 shows the packaged Nvidia Geforce 8800 die and the corresponding wafer used for its manufacturing.

Manufacturing Process

The transistors are continuously getting smaller because smaller transistors are faster, consume less power and require less chip area. Even a manufacturing technology is named after the smallest transistor gate length it is able to produce. For instance, during the last decade, the manufacturing community has switched from 180nm technology to 45nm technology. On the other hand, manufacturing is a complex gradual process and prone to variations especially when sub-micron technologies are utilized. It comprises different techniques like lithography, etching, doping or ion implantation, chemical mechanical polishing (CMP). It is difficult to cope with this fast size re-

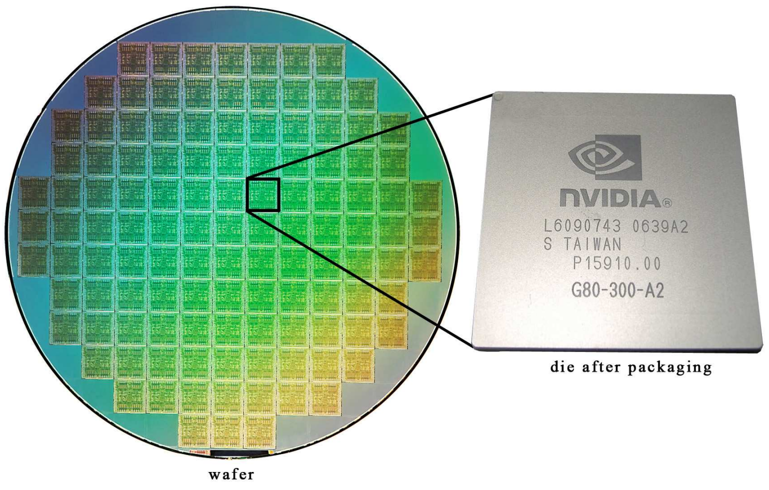


Figure 2.9: Nvidia Geforce 8800 die and its corresponding wafer

duction and improve manufacturing process sufficient for this technology improvement. Although exotic techniques such as optical proximity correction and other resolution enhancement techniques have been developed [18], the impact of manufacturing process variations have become much more significant in the nanometer regime used today.

Process Variations

The variations during the procedures like lithography, ion implantation and CMP result in physical variations like critical dimension (CD), oxide thickness, channel doping, interconnect wire width and thickness [19, 20]. Among these, CD and doping variations, which respectively result in gate length (L) and threshold voltage (V_t) variations, are the dominant factors [20]. As a result, the timing behavior of transistors and interconnects are affected and gate delays, interconnect delays and so the total circuit delay become random variables. According to International Technology Roadmap for Semiconductors (ITRS) 2009 report [1], from 2009 until 2011, CD or effective gate length variability ($3\sigma/\mu$) is reported to be 12% whereas V_t variability for typical size logic devices is reported to be 20%. The report assumes a 50% V_t variability in 2018 and there are no known manufacturing solution for such a high V_t variability. According to the report the resultant circuit timing performance variability is about 50% for 2009, which is assumed to reach up to 66% in 2018.

Inter-die and Intra-die Variations

The parameter variations are classified into two according to their spatial properties:

Inter-die Variations: Parameters may vary between two different dies on the same wafer or when the processed wafer is replaced with the new one or when a new lot of wafers start to be processed. This variation is a result of unavoidable changes in the calibration of the equipment, alignment of the wafers and mask patterns [20]. The variation of process parameters between different dies or wafers or wafer lots are called *inter-die* or *die to die* variations. All devices on the same die are affected exactly in the same manner from the inter-die variations. For instance, assume that gate length (L) is a random parameter which has inter-die variation. If there exist only inter-die variations, all transistors in the same die have the same L value. The inter-die variation only causes the transistors in different dies to have different L values.

Intra-die Variations with Spatial Correlation: The parameter variations can even exist for the devices in the same die. These variations are called *intra-die* or *within die* variations. For instance, due to intra-die variations, a transistor in a die may have a different gate length (L) than another transistor in the same die although both of them are designed to have same gate lengths. A decade ago, the intra-die variations had been a negligible portion of total variations. However, shrinking the technology below 100nm levels, the percentage contribution of intra-die variation to the total variation has increased above 50%. For instance the percentage of L intra-die variations has increased to 65% from 40% while the technology switches from 250nm to 70nm [19].

The intra-die variations exhibit spatial correlation, i.e. the correlation of a device (transistor or interconnect) parameter depends on the location of that device, because many of the manufacturing processes that cause intra-die variations change gradually from one location to another [20]. Due to the spatial correlation, the devices closer to each other are affected similarly whereas the devices far from each other have more uncorrelated parameters. In other words, the correlation of the corresponding parameters of two devices on the same die increases while the distance between them decreases. There are different models in the literature to capture the spatial correlations [2, 21]. The model used in our work is based on [2], which is explained in detail in Section 3.1.

Timing Yield Estimation Problem

As explained in Section 1.2, the variation of circuit delay as a result of all manufacturing process variations, described above, causes some of the manufactured dies fail to satisfy the timing constraints and so they are discarded. The fraction of dies that satisfies the timing constraints is called *timing yield*. Equivalently, timing yield is the probability that a manufactured die satisfies the timing constraints. Fundamental timing constraint for combinational circuits is basically an upper bound for circuit delay, i.e. T_c, such that the delay of the circuit should be below T_c to satisfy timing constraint. Figure 2.10 shows an example probability density function (PDF) for the circuit delay and the area of the shaded region in this figure corresponds to timing yield, where T_c is the timing constraint. The main aim of statistical timing analysis is to estimate timing yield before handing over the designs to manufacturing stage. According to the statistical timing analysis results, the designers make a choice either to improve their designs further or to give up optimizations and initiate manufacturing.

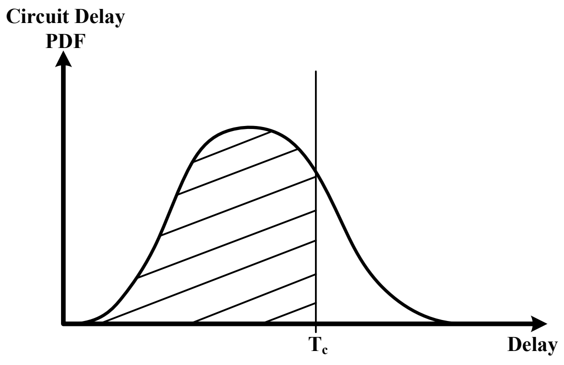


Figure 2.10: Circuit delay PDF

Corner-based Timing Analysis

Traditionally the designers try to cope with statistical parameter variations by using *corner based timing analysis*. A corner corresponds to a point in the parameter space, where each parameter gets its maximum or minimum possible value. The worst case timing behavior of the circuit can be computed by first setting the parameters at their corner values[8] and then by utilizing a deterministic timing analysis scheme as the random parameters are fixed at deterministic corner values. However, as the sources of variation increase in number, the required number of corners increases exponentially to an untenable number for a complete corner based analysis [20, 22]. Another weak-

[8]Generally, the corner values are taken as $\mu - 3\sigma$ or $\mu + 3\sigma$ values, where μ is the nominal value for the corresponding parameter and σ is the standard deviation.

ness of corner based analysis is that it is impossible to effectively take into account the intra die variations [23] although, the intra die portion of variations has even exceeded the portion of inter die variations as referred above in this section. Finally, selecting the worst case parameter values for all devices in a path represents an almost impossible and unrealistic case [24]. Therefore, the corner based analysis gives over-pessimistic yield estimates which result in wasted design efforts and time loss for unnecessarily optimizing the circuit [23, 25, 24, 22, 20]. As a result of the non-negligible manufacturing process variations and the inability of the traditional corner based methods in meeting the requirements, the statistical timing analysis has become a hot and popular research topic.

2.3.2 Statistical Static Timing Analysis (SSTA) Methods

Description of SSTA

After the turn of the millennium, we have witnessed an extensive amount of effort being expended in statistical timing analysis research. Most of this effort has been aimed at the development of statistical static timing analysis (SSTA) techniques, as a direct generalization of the deterministic static timing analysis (DSTA) algorithm, explained in Section 2.2.2, for the statistical case. A comprehensive review of the recent developments in this field that puts all relevant work into perspective is given in [20]. At this point, the SSTA problem and its key challenges are very well understood.

The fundamental functions in SSTA are SUM and MAX and they are performed for each gate in the circuit very similar to DSTA. But, due to parameter variations, instead of SUM and MAX of the deterministic values, SUM and MAX of the random variables are required in SSTA as all gate delays and arrival times are random variables due to the statistical variations. In DSTA, the deterministic circuit delay value is available when all gates are traversed by SUM and MAX computations. The same is true for SSTA but this time, the circuit delay is a random variable (RV) with a corresponding PDF instead of a deterministic value. After computing PDF of the circuit delay, timing yield can be computed by calculating the shaded area shown in Figure 2.10.

A small portion of SSTA literature, particularly the initial research includes path-based methods [26,27,28,29,30,31]. Differently from block based SSTA that will be described below, the path based methods do not follow the DSTA algorithm. Instead, these methods first compute the path delay PDF for each path by adding the individual gate delay PDFs on the path. The paths are drawn from a path set which include all paths that may affect the circuit delay for some assignments to the random parameters in the circuit. Then, they take the maximum of these path delay PDFs to obtain the circuit delay PDF. Instead of performing the MAX operation for each gate as in the block based methods, path-based methods are advantageous in the sense that they apply MAX operation only once and thus they avoid the accumulated errors due to the unavoidable approximations in MAX operations.

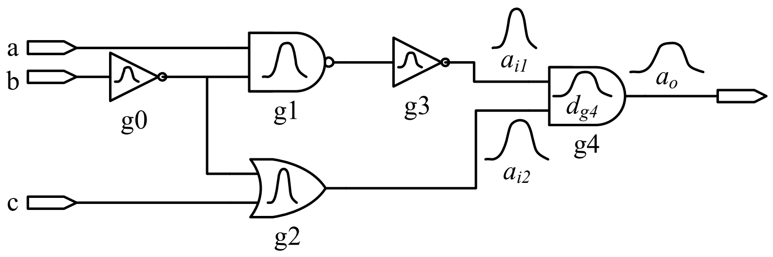


Figure 2.11: A demonstration of SSTA SUM and MAX operations on a sample gate (node)

$$a_o = max(a_{i1} + d_{g4}, a_{i2} + d_{g4}) \tag{2.10}$$

Most of the approaches to SSTA are based on what is referred to as the block-based scheme. The block based SSTA methods run at block (gate) level by following the DSTA algorithm quite closely. In order to realize the operation of block based SSTA, Figure 2.11 demonstrates the basic operation performed for gate $g4$ in the figure. The PDFs inside the gates correspond to the gate delay RVs. a_{i1} and a_{i2} are the input arrival time RVs for gate $g4$ and d_{g4} is the gate delay RV for $g4$. The output arrival time for $g4$, i.e. a_o, is computed by SUM and MAX operations as shown in (2.10). Block based SSTA performs this operation for all gates in the circuit in a topologically sorted manner similar to DSTA. The block based SSTA methods basically differ by the techniques they prefer for computing these SUM and MAX operations. These techniques differ according to the assumptions and approximations they employ in order to perform the SUM and MAX operations of two RVs. Although the difference between DSTA and block based SSTA may seem slight, SSTA has many challenges which are very difficult and sometimes impossible to solve without having exponential run time complexities. Therefore, many approximations and assumptions are involved in SSTA methods. SSTA literature will be reviewed after the challenges of SSTA are explained below.

Challenges in SSTA

One of the major challenges arises from the *topological* and *spatial* correlations. This is because the correlation information between all RVs inside the circuit must be recorded and preserved during the SUM and MAX operations. Computation of these operations on two correlated RVs are more complicated than the operations on two independent RVs. Topological correlations occur due to the *reconvergent paths* in the circuit. Two paths reconverge if both cross from a common signal and later rejoin as two distinct inputs of a gate. For instance, in Figure 2.11, paths $b - g0 - g1 - g3 - g4$ and $b - g0 - g2 - g4$ starts with the same signal (output of $g0$) and then reconverge at the input of $g4$. As a result, the input arrival times a_{i1} and a_{i2} are not independent or fully uncorrelated. This correlation between a_{i1} and a_{i2} is called topological correlation and complicates the MAX operation used to find a_o. Second type of correlations is spatial correlation explained in Section 2.3.1. Spatial correlation can affect both SUM and MAX operations. For instance, in Figure 2.11, assume that gates $g2$, $g3$ and $g4$ are spatially in a close proximity in the final layout. Then, the gate delay RVs of these gates are spatially correlated. Therefore, a_{i1}, a_{i2} and d_{g4} are correlated, which complicates SUM and MAX operations shown in (2.10). As a result, for accurate statistical timing analysis, both topological and spatial correlations must be taken into account and the operations must be modified in order to handle the correlations [20].

Another challenge in SSTA is non-linear parameter delay relationships and non-normal (non-Gaussian) parameter and delay distributions. It was explained in Section 2.3.1 that due to the variations in the physical process parameters like CD and ion implantation, the device parameters like gate length and threshold voltage and as a result gate delays become random variables. The relationships of physical process parameters with device parameters and device parameters with gate delays are non-linear. Therefore, even if the physical parameter variations are approximated by normal (Gaussian) distributions, it is impossible to model the device parameters and the gate delays with normal distributions. Despite the assumption of linearity and normality extremely simplifies the SUM and MAX procedures, these assumptions may result in far off timing yield estimates.

The MAX operation introduces another challenge. Even all gate delays and arrival times are assumed to have normal PDFs, the maximum of two normal RVs is not a normal RV [20]. However, the majority of the methods in the literature approximate the MAX result of two normal distributions with another normal distribution, which results in a loss of accuracy due to accumulated errors.

In the deterministic case, where all gate delays and arrival times are deterministic, the path having the worst arrival times is the critical path if it is a true path. However, in the statistical case, where gate delays are random variables, the critical path of the circuit depends on the values of the RVs inside the circuit. Each different assignment to RVs may result in a different critical path. The paths that become critical at least for one assignment to RVs are called *statistically critical paths*. In the case of statistical variations, the discovery of the statistically critical paths is another challenge that must be addressed at least for optimization purposes.

SSTA Literature Review

The block-based SSTA methods in the literature mainly differ according to the SUM and MAX strategy they propose and the way they handle the above challenges. The literature may be classified in many aspects: parametric or non-parametric methods, methods that consider correlations or methods that ignore them, methods according to the utilization of continuous PDFs or discrete PDFs, normal PDFs or non-normal PDFs, linear or non-linear parameter-delay relationships.

One important classification for the block-based SSTA methods is based on the types of RVs used in the circuit. In non-parametric methods, gate delays are directly represented as RVs with some statistical properties like mean and variance, whereas in parametric methods gate delays are represented in terms of other RVs, which are random device parameters like gate length and threshold voltage.

Initial efforts were more focussed on non-parametric methods. In these methods, the device parameters are not represented as RVs, only gate delays and arrival times are the fundamental RVs in the circuit. Until the circuit delay PDF characteristics are obtained, either the delay PDF properties like mean and variance are propagated or the delay PDFs are first discretized and then propagated throughout the circuit. In [32], a linear run time non-parametric SSTA algorithm propagating only the mean and variance of the arrival times is proposed. This method ignores correlations and assumes independent normal random variables for all gate delays and arrival times. In [33, 34, 35], differently from [32], the topological correlations are taken into account. Normal distribution assumption is relaxed and more efficiency is acquired by the use of discrete PDFs instead of continuous PDFs in [36, 37, 38] . Convolution is used for the SUMmation of two discrete PDFs. The papers in [39,40] add topological correlation support to this discrete framework and compute the upper and lower bounds for the CDF of the circuit delay. [41] takes into account the topological correlations and represents arrival times as CDFs and gate delays as PDFs.

Parametric SSTA methods represent gate delays and arrival times as a function of random device parameters. They are suitable to model intra-die parameter variations with spatial correlations. A linear canonical delay model widely used in parametric SSTA methods [42, 43, 20] is shown in (2.11).

$$d = d_{nom} + \sum_{i=1}^{N} a_i r_i + d_{res} \tag{2.11}$$

where d_{nom} is the nominal (mean) delay, r_i's are the independent random variables which are used to model spatial correlations and d_{res} is the residual independent variation. a_i's are called the sensitivities corresponding to r_i's.

The model is called canonical as it does not change throughout the circuit, i.e. all arrival times, gate and path delays are represented by the same canonical model. The SUM and MAX operations are applied to the canonical model such that the results of the operations are also represented by the same model. The SUM operation on such a canonical form is straightforward whereas the MAX operation is more complicated for which [42,43] use a method explained in [44].

There are different approaches to model the intra-die variations with spatial correlations as a sum of independent random variables [43,2]. [43] employs a grid model by dividing the die surface into equal sized rectangles. For each random device parameter like gate length, there is an associated random variable with each rectangle. The devices in the same rectangle are assumed to be perfectly correlated, i.e. they all have the same value for the corresponding random parameter. On the other hand, the devices in different rectangles have partially correlated parameters such that their correlation decreases as the distance between the rectangles increases. These correlated RVs are converted into uncorrelated RVs called principal components using the principal component analysis (PCA) method, which transforms space in order to get the set of uncorrelated RVs from a set of correlated RVs [6]. Then, the initial correlated RVs associated with each grid (rectangle) are expressed as the sum of these uncorrelated RVs. This avails the use of independent RVs to model spatial correlations as shown in (2.11). [2] proposes a multilevel quad-tree structure, shown in Figure 3.1, to model spatial correlations. At each level q, it divides the die surface into $2^q \times 2^q$ rectangles, each of which has its own independent random variable for each random circuit or transistor parameter. The details of this model and how it models the spatial correlations are explained in Section 3.1.

The parametric models explained above assume normal random variables and employ linear canonical models similar to (2.11). There are other papers in the literature, which employ non-linear canonical models and handle non-normal distributions to increase the accuracy, yet the relaxation of Gaussian and linearity assumptions result in significant computational overhead [20]. The approaches in [45, 46, 47] use a quadratic canonical form instead of the simple linear form in (2.11). [48] uses a linear model but considers non-normal device parameters. [49] considers both non-linearity and non-normal parameter distributions.

The collection of statistically critical paths when there are parameter variations is another research topic that is well studied in the literature [50, 51, 52, 53, 54, 55]. The probability of a path being critical, i.e. criticality, is a measure used by these papers. Other than the referred above, there are many efforts to handle interconnect wire variations, to propose solutions for statistical optimization, to perform SSTA in sequential circuits and to find statistical models for gate delays [20].

Block-based SSTA methods, reviewed above, have been preferred due to their runtime advantage when compared with other approaches to SSTA. Moreover, block-based SSTA can be performed in an incremental manner enabling its use in timing yield optimizations and for diagnostic purposes [20, 56]. On the other hand, spatial and topological correlations, non-normal process parameters and non-linear dependence of gate delay on these parameters, approximation of the maximum of two random variables (to compute latest arrival time) at every gate (node of the timing graph) are issues that must be addressed in block-based SSTA methods. In most basic form, SSTA algorithms ignore correlations, assume that all statistical process parameters and gate delays have a normal distribution and approximate the maximum of two normal random variables as another normal random variable. All of these assumptions and simplifications make it possible to obtain very efficient SSTA algorithms [56]. However, ignoring correlations and the Gaussian assumption have detrimental, and in some cases, unacceptable effects on the accuracy and meaningfulness of the results obtained by SSTA [57]. As a result, several extensions of SSTA that take correlations into account, that use non-linear gate delay models and employ non-normal distributions for the maximum of two random variables have been proposed [20]. These extensions indeed improve the accuracy of SSTA, but at the same time increase its computational complexity and may render it unusable in timing optimizations which require very efficient in-the-loop evaluations [57]. Nevertheless, block-based SSTA on an abstract timing graph is widely accepted as a useful tool and is becoming indispensable in current state-of-the-art statistical design methodologies.

2.4 Preliminaries and Monte Carlo (MC) Timing Yield/Loss Estimation

In this section, first of all, the preliminaries to understand Monte Carlo (MC) method is given. The preliminaries part below is particularly important as the basic definitions in that part are referred many times throughout this thesis and the notation introduced here is used by the methods proposed in this thesis. Upon these basics, theory of general MC estimators and a well-known variance reduction technique used to improve estimator accuracy, called importance sampling, are explained. After the preliminaries section, loss estimation using standard MC (STD-MC) method is explained by separating STD-MC into two: transistor level MC (TL-MC) and block (gate) level MC (BL-MC). TL-MC utilizes the transistor level timing simulation, explained in Section 2.2.1. It is the most precise and exact way of estimating the timing yield. On the other hand, BL-MC estimators run at block level and although they are much more accurate than SSTA methods reviewed in Section 2.3.2, they are not as accurate as TL-MC estimators.

2.4.1 Preliminaries

Basic Definitions and Notation

The random variables that represent the statistical variations in the circuit are collected into an n-dimensional vector X, with a joint probability density function (PDF) denoted by $f(X)$ which is *not* necessarily assumed to be Gaussian (normal). We note here that the number of random variables, n, is dictated by the particular inter and intra-die variations model used and is in general much larger than the number of statistical process and transistor parameters considered. For instance, in this thesis we consider only two random transistor parameters (gate length and threshold voltage), but we employ hundreds of random variables for modeling the statistical variations of the circuit. The reason for that and the details of the variation model used in our work is explained in Section 3.1.

We use $d_\pi^M(X)$ to denote the *path delay* for a *path* π computed by *method M*. The path delay naturally depends on the random variables in X, and hence, it is also a random quantity. We then define the *circuit delay* $d_C^M(X)$ computed by method M as the *maximum* path delay with

$$d_C^M(X) = max_{\pi \in \Pi_{crit}}\; d_\pi^M(X) \tag{2.12}$$

where the maximum is computed over the set of *statistically critical* paths Π_{crit}. There are many methods based on SSTA to find the statistically critical paths as mentioned above in Section 2.3.2. It will be explained in Section 3.3 what we propose to find the set of statistically critical paths.

We define an indicator random variable $I^M(T_c, X)$ as follows

$$I^M(T_c, X) = \begin{cases} 1 & \text{if} \quad d_C^M(X) > T_c \\ 0 & \text{if} \quad d_C^M(X) \leq T_c \end{cases} \tag{2.13}$$

where M is the method used for circuit delay computation and T_c is the maximum acceptable delay or timing constraint. This indicator variable "indicates" whether the delay of the circuit meets the timing constraint for a given realization of the random variables in X. We then define *Loss* computed with method M using

$$Loss^M = \int_\Omega I^M(T_c, X) f(X) dX \tag{2.14}$$

as the fraction of the circuits that fail to satisfy the timing constraint. The integral in (2.14), i.e. the expectation of the indicator variable $I^M(T_c, X)$, is computed over the domain Ω of the PDF $f(X)$ of X. Then, $Yield$, the fraction of the circuits that fulfill the timing constraint is simply given by

$$Yield = 1 - Loss \tag{2.15}$$

Therefore, yield estimation and loss estimation are equivalent in the sense that one can be derived from the other by only subtracting from 1.

One very effective method for computing expectation integrals of the form in (2.14) is the Monte Carlo technique, which we describe below.

General Monte Carlo Method

Monte Carlo (MC) techniques can be used to compute expectation integrals of the form

$$G = \int_\Omega g(X) f(X) dX \tag{2.16}$$

where Ω is the domain of the PDF $f(X)$, with $f(X) \geq 0$ for all X and $\int_\Omega f(X)dX = 1$. MC estimation of G in (2.16) is accomplished by drawing a set of independent random samples $X_1, X_2, ..., X_N$ from $f(X)$ and by using

$$G_N = (1/N) \sum_{i=1}^{N} g(X_i) \tag{2.17}$$

The estimator G_N above is itself a random variable. The theorems below show the mean (expectation) and the error for the general MC estimate G_N.

Theorem 2.4.1. *The general MC estimator in (2.17) is an unbiased estimator as its mean is equal to G, i.e.* $E\{G_N\} = G$.

Proof. As X_i's are independent identically distributed (i.i.d.) random variables

$$E\{G_N\} = \tfrac{1}{N}E\{\textstyle\sum_{i=1}^{N} g(X_i)\} = \tfrac{1}{N}.N.E\{g(X)\} = \int_\Omega g(X)f(X)dX = G \tag{2.18}$$

□

Theorem 2.4.2. *With* 95% *confidence, the error of the general MC estimator in (2.17) is as shown in (2.19):*

$$|Error| \approx \frac{2\sigma}{\sqrt{N}} = 2\sqrt{VAR\{G_N\}} \tag{2.19}$$

Proof. According to Theorem 2.4.1, the mean of MC estimator is equal to the integral G that it is trying to estimate, i.e., $E\{G_N\} = G$. As X_i's are i.i.d. RVs, the variance of G_N is $VAR\{G_N\} = \sigma^2/N$, where σ^2 is the variance of the random variable $g(X)$ given by

$$\sigma^2 = \int_\Omega g^2(X)f(X)dX - G^2 \tag{2.20}$$

The standard deviation of G_N can be used to assess its accuracy in estimating G. If N is sufficiently large, due to the Central Limit Theorem, $\frac{G_N - G}{\sigma/\sqrt{N}}$ has an approximate standard normal ($N(0,1)$) distribution. Hence,

$$P(G - 1.96\tfrac{\sigma}{\sqrt{N}} \leq G_N \leq G + 1.96\tfrac{\sigma}{\sqrt{N}}) = 0.95 \tag{2.21}$$

where P is the probability measure. The equation above means that G_N will be in the interval $[G - 1.96\frac{\sigma}{\sqrt{N}}, G + 1.96\frac{\sigma}{\sqrt{N}}]$ with 95% confidence. Thus, one can use the error measure in (2.19) in order to assess the accuracy of the estimator.

□

Several techniques, called *variance reduction techniques*, exist for improving the accuracy of MC evaluation of expectation integrals. In these techniques, one tries to construct an estimator with a reduced variance for a given, fixed number of samples, or equivalently, the improved estimator provides the same accuracy as the standard MC estimator but with considerably fewer number of samples. This is desirable because computing the value of $g(X_i)$ is typically computationally or otherwise costly. Next, a well-known variance reduction technique called importance sampling will be described.

General Importance Sampling (IS) Technique

One MC variance reduction technique is importance sampling (IS) [58, 59]. IS improves upon the standard MC approach described above by drawing samples for X from another distribution $\tilde{f}(X)$. G in (2.16) is first rewritten as below

$$G = \int_{\Omega} \left(\frac{g(X) f(X)}{\tilde{f}(X)} \right) \tilde{f}(X) dX \tag{2.22}$$

If $X_1, X_2, ..., X_N$ are drawn from $\tilde{f}$ instead of f, the improved estimator $\tilde{G}_N$ takes the form

$$\tilde{G}_N = \frac{1}{N} \sum_{i=1}^{N} g(X_i) \frac{f(X_i)}{\tilde{f}(X_i)} \tag{2.23}$$

where the factor $f(X_i)/\tilde{f}(X_i)$ has been used in order to compensate for the use of samples drawn from the biasing distribution $\tilde{f}$. As can be seen by observing (2.22), in order for the improved estimator above to be well-defined and unbiased, two requirements must hold:

1. $\tilde{f}(X)$ must be nonzero for every X for which $f(X)g(X)$ is nonzero. We refer to this as the *safety requirement*.

2. $\tilde{f}(X)$ must be a regular normalized PDF such that its integral over the whole space must be equal to 1, i.e. $\int \tilde{f}(X)dX = 1$. This is referred as *regularity requirement*.

The ideal choice for the biasing distribution $\tilde{f}$ is

$$\tilde{f}_{ideal}(X) = \frac{g(X)\, f(X)}{G} \tag{2.24}$$

which results in an exact estimator with zero variance with a single sample! However, $\tilde{f}_{ideal}$ obviously cannot be used in practice since the value of G is not known a priori. Instead, a practically realizable $\tilde{f}$ that resembles $\tilde{f}_{ideal}$ is used. The key (and also the challenge) in using IS in practical problems is the determination of an effective biasing distribution that results in significant variance reduction.

2.4.2 Standard MC (STD-MC) Loss Estimation

After the description of general MC estimation above, standard MC (STD-MC) estimator for loss with integral in (2.14), is straightforward. Equation 2.14 is obtained by replacing $g(X)$ in (2.16) with $I^M(T_c,X)$. Thus, for estimating $Loss^M$ shown in (2.14), by using MC technique, $g(X_i)$ in (2.17) must be replaced with $I^M(T_c,X_i)$. The resultant STD-MC estimator for $Loss^M$ is written as in (2.25).

$$Loss_N^M = (1/N)\sum_{i=1}^{N} I^M(T_c,X_i) \tag{2.25}$$

where M is the method used to compute $d_C^M(X)$ in (2.12), X_i's are the drawn samples according to $f(X)$ in (2.14) and T_c is the timing constraint. Provided that the method M utilized by STD-MC estimator is a very accurate method, if the number of samples N is a *big enough number*, then the estimator in (2.25) gives very accurate results. In order for standard MC analysis to be affordable, the number of samples in probability space one has to work with needs to be limited. This, however, adversely affects the accuracy of the STD-MC estimator, which has a large error for a small number of samples. The computational cost is the weakness of the STD-MC method and has prevented it from finding widespread use for practical yield estimation, even though it is widely used as a golden reference in assessing the accuracy of other timing yield estimation techniques.

STD-MC loss estimation can be divided into two: transistor level Monte Carlo (TL-MC) and block (gate) level Monte Carlo (BL-MC). Both use the STD-MC loss estimator equation in (2.25) to estimate loss but they differ in the method M they prefer to compute the indicator variable $I^M(T_c,X_i)$ in (2.25). As it is shown in (2.13), the circuit delay $d_C^M(X)$ must be computed in order to compute this indicator variable. Thus, TL-MC and BL-MC basically differ in the method M, they prefer to compute circuit delay, i.e. $d_C^M(X)$.

For each drawn sample X_i, TL-MC loss estimation employs TL simulation method, explained in Section 2.2.1, in order to compute the path delays and circuit delay in (2.12). The indicator variables for all drawn sample points are collected using (2.13). At the end, loss estimate is computed using (2.25). Because of the high computational cost of TL simulations for each sample, it is generally believed that TL-MC analysis cannot be used in practice for estimating timing yield, even though there are some arguments to the contrary [60].

BL-MC loss estimation, differently from TL-MC, uses DSTA method, explained in Section 2.2.2, to compute the circuit delay for each drawn sample point (X_i). BL-MC is computationally much cheaper than TL-MC as DSTA is much cheaper than TL simulation, which was explained in Section 2.2.1 and 2.2.2. This cheaper block-based, gate-level Monte Carlo timing analysis scheme is in fact called *golden method* as it is used to verify the accuracy of various block-based SSTA methods which employ many assumptions and approximations. Recently, variance reduction techniques such as Latin hypercube sampling [61] were used to improve the efficiency of BL-MC statistical timing analysis techniques.

We believe that sufficient accuracy and reliability in final timing yield estimation cannot be obtained even by applying Monte Carlo simulations at a high-level using a block-based scheme. We believe that accurate final verification of timing yield must have TL circuit simulation as its basis, in line with the common practice in traditional VLSI design where critical paths are simulated at transistor-level in order to verify that the circuit indeed satisfies the timing constraints. We demonstrate in this thesis that Monte Carlo transistor-level simulation in conjunction with a novel variance reduction technique can serve as an accurate yet computationally viable timing yield estimation method, to be used for final verification before timing sign-off. Next section explains further the contributions of this thesis.

Related Work

There are several works in the literature that uses importance sampling (IS) or other variance reduction techniques in order to increase the efficiency, or improve the accuracy of Monte Carlo analysis of statistical phenomena in electronic circuits. In fact, IS based Monte Carlo analysis has been used in order to estimate the yield of analog circuits [63], perform failure analysis for SRAM circuits [64,65,66], for statistical interconnect analysis [67], and even for the statistical timing analysis of digital circuits [68]. The use of simple, cheap-to-evaluate gate delay models (linear, quadratic or more sophisticated response surface models) in statistical analysis is also prevalent in the literature [69, 70, 71, 72]. Moreover, the idea of using path-based transistor-level analysis for statistical performance verification has also been explored. However, the challenge and key in using IS to achieve significant variance reduction is the non-costly determination of a useful biasing distribution. The technique we propose in Chapter 4 is novel in the sense that a cheap-to-evaluate gate delay model and approximate path-based statistical timing analysis are used in a unique way to (in effect) construct an effective biasing distribution for IS that indeed results in significant variance reduction/speed-up. Furthermore, an adaptive/automated algorithm we propose makes it possible to apply this IS technique in practice with negligible overhead. In [63], the outline and a simple analysis for an IS-like technique (called sectional weighting) that resembles the technique we propose in this thesis was given. In [63], the authors are not very encouraging regarding the use of this technique due to the insignificant speed-ups (over standard Monte Carlo) predicted by their simple analysis and due to the potentially high computational cost of forming the biasing distribution. The computational complexity of the construction of the biasing distribution we propose in this paper is not dependent directly on the dimension of the random parameter space, resulting in negligible overhead. Moreover, we achieve significant (two-orders of magnitude) speed-ups over standard Monte Carlo.

Chapter 3

STATISTICAL TIMING ANALYSIS METHODOLOGY

The statistical timing analysis methodology we propose in this chapter features the following:

- **Section 3.1:** Modeling of inter and intra-die statistical variations based on a quad-tree model that captures spatial correlations,
- **Section 3.2:** An approximate, polynomial gate delay model (PDM) that captures delay dependence on random transistor parameters, gate load and input slope,
- **Section 3.3 and 3.4:** Identification of a set of statistically critical paths for a circuit, based on a block level Monte Carlo (BL-MC) statistical timing analysis that uses PDM and a path sensitization test to identify false paths,

3.1 Quad-tree Based Parameter Variation Model

In this section, we present the statistical model we use for inter and intra-die variations in process and transistor parameters, which are explained in Section 2.3.1. The inter-die variations are perfectly spatially correlated throughout the circuit. In order to model intra-die variations and the resulting (partial) spatial correlations in the circuit, we use the quad-tree model that was proposed by Agarwal et. al. [2].

General Parameter Modeling

A statistical process or transistor parameter can be modeled as shown in (3.1), where P_{inter} represents inter-die variation and P_{intra} represents intra-die variation component. Therefore, P_{inter} is the same throughout a single die but varies between different dies, whereas P_{intra} varies even between two transistors in the same die. Due to spatial correlation, the correlation between P_{intra} components of two transistors decreases while the distance between them increases. P_{inter} has the nominal value of the corresponding statistical parameter as its mean whereas P_{intra} has zero mean. As P_{inter} and

P_{intra} are independent from each other, the total variance of a statistical transistor parameter can be written as in (3.2).

$$P = P_{inter} + P_{intra} \tag{3.1}$$

$$\sigma^2 = \sigma^2_{inter} + \sigma^2_{intra} \tag{3.2}$$

where σ^2_{inter} is the variance of P_{inter} and σ^2_{intra} is the variance of P_{intra}.

Quad-tree Model for Both Inter and Intra-die Variations

The quad-tree model can be used to model both inter-die variation and intra-die variation with spatial correlation. In the quad-tree model, the area of a die is partitioned into rectangles forming a grid structure. For each level, q, of the quad-tree model, the die area is divided into $2^q \times 2^q$ rectangles. Figure 3.1 shows the quad-tree model that is used in our work.

The top level ($q = 0$) is associated with the random variable, P_{inter}, in (3.1) and its mean is equal to the nominal value of the parameter P. Other than the top level, each level, q, in a quad-tree model is associated with a probability density function with mean μ_q and standard deviation σ_q. Each grid rectangle at each level of a quad-tree is associated with an independent random variable, $R_{intra}(x,y)_q$, with the PDF of the corresponding level, where $(x,y)_q$ shows the coordinate of the corresponding rectangle at level q. Figure 3.1 shows the $(x,y)_q$ pairs up to level 2. Therefore, each level is associated with a PDF so that all random variables at the same level has the same PDF, which is associated with that level. The mean of the PDF of every level other than the top level is zero, i.e. $\mu_q = 0$ for $q = 1,2,...,Q-1$ where Q is the number of levels. The sum of the variance of the PDF of each level other than the top level ($q = 0$) is equal to the intra-die variance as shown in (3.3).

$$\sigma^2_{intra} = \sum_{q=1}^{Q-1} \sigma^2_q \tag{3.3}$$

where Q is the number of levels and σ^2_q is the variance of the PDF corresponding to level q in the quad-tree model.

In this model, a statistical process or transistor parameter P such as channel length is expressed

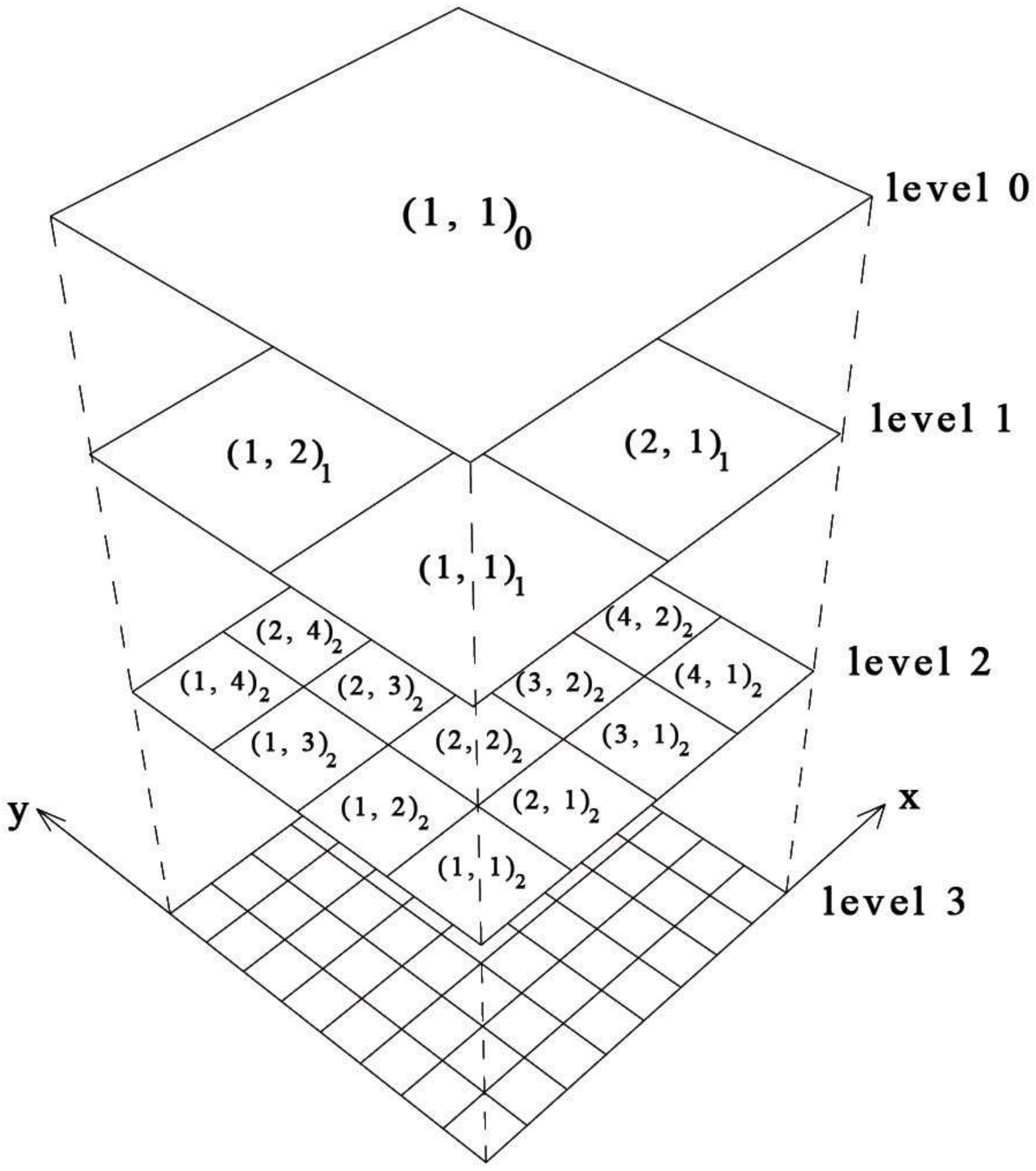


Figure 3.1: 4-level quad-tree model

as follows

$$P = P_{inter} + \sum_{q=1}^{Q-1} R_{intra}(x,y)_q \tag{3.4}$$

where the random variable P_{inter} models the perfectly correlated inter-die variations, $R_{intra}(x,y)_q$ are layout position (x,y) dependent random variables that are assigned to level q of the quad-tree model, and Q is the total number of levels in the model for both intra and inter-die variations. In most of the previous work, P_{inter} and $R_{intra}(x,y)_q$ are assumed to be independent random variables with a Gaussian distribution. In our approach, the basic statistical process and transistor parameters and the random variables in (3.4) can have arbitrary (joint) PDFs.

An example can be given for clarification. Suppose we have a gate that resides at $(5,3)_3$ where the model in Figure 3.1 is utilized. A statistical parameter corresponding to that gate may be written as

$$P = P_{inter} + R_{intra}(2,1)_1 + R_{intra}(3,2)_2 + R_{intra}(5,3)_3 \tag{3.5}$$

As a result, in a Q-level quad-tree model, $\sum_{q=1}^{Q} 2^{2(q-1)} = \frac{4^Q-1}{3}$ random variables are needed for every basic process or transistor parameter. After drawing and assigning a number to each random variable (grid) according to its corresponding PDF, a random transistor parameter is computed according to the location of the transistor as shown in the example above by (3.5). Drawing a random sample and determining the transistor parameters accordingly will be clarified more at the end of this section.

Can Quad-tree Model Capture Spatial Correlation?

As explained in Section 2.3.1, spatial correlation in statistical timing literature means that the gates closer to each other have more correlated random parameters than the gates far away from each other. Below the ability of the quad-tree model to capture this spatial correlation is investigated.

Proposition 3.1.1. *The quad-tree model captures the spatial correlation such that the gates or transistors in closer grids have more correlated parameters than the gates in grids far away from each other.*

Proof. The aim is to show that the correlation between two instances of the same statistical parameter corresponding to two different gates in the circuit increases if the distance between them decreases. Assume that P_i and P_j are the two parameter instances of the same parameter corresponding to gate i and gate j respectively. If a quad-tree model with Q levels is used, P_i, P_j and their correlation can be written as

$$P_i = P_{inter} + \sum_{q=1}^{Q-1} R_{intra}(x_i, y_i)_q \tag{3.6}$$

$$P_j = P_{inter} + \sum_{q=1}^{Q-1} R_{intra}(x_j, y_j)_q \tag{3.7}$$

$$Corr(P_i, P_j) = \frac{Cov(P_i, P_j)}{\sqrt{VAR\{P_i\}.VAR\{P_j\}}} \tag{3.8}$$

where $P_{intra} = \sum_{q=1}^{Q-1} R_{intra}(x,y)_q$, $Corr(P_i, P_j)$ is the correlation and $Cov(P_i, P_j)$ is the covariance of P_i and P_j. Therefore,

$$VAR\{P_i\} = VAR\{P_j\} = \sigma_{inter}^2 + \sigma_{intra}^2 = \sigma^2 \tag{3.9}$$

As all $R_{intra}(x,y)_q$'s and P_{inter}'s are independent, $Cov(P_i, P_j)$ becomes,

$$Cov(P_i, P_j) = VAR\{P_{inter}\} + \sum_{q=1}^{Q-1} E_q.VAR\{R_{intra}(x_i, y_i)_q\} \tag{3.10}$$

where E_q is 1 if $(x_i, y_i)_q = (x_j, y_j)_q$ and otherwise E_q is 0. Then $Corr(P_i, P_j)$ becomes,

$$Corr(P_i, P_j) = \frac{\sigma_{inter}^2 + \sum_{q=1}^{Q-1} E_q.VAR\{R_{intra}(x_i, y_i)_q\}}{\sigma^2} \tag{3.11}$$

In the quad-tree model, the gates in grids closer to each other, have more common $R_{intra}(x,y)_q$'s, therefore E_q is equal to one for more levels. Considering the fact that in the quad-tree model, $VAR\{R_{intra}(x_i, y_i)_q\}$ depends only on the level q and is equal to σ_q^2, the numerator of (3.11) will always be a bigger value for the gates residing in grids closer to each other. For instance, if the two gates reside in the same grid even at the bottom level, then E_q is one for all levels. As a result of this, $Corr(P_i, P_j)$ reaches its maximum value, 1, which means that the random parameters of two gates are fully correlated. The correlation decreases while the number of common grids of two gates decreases, i.e. the distance between two gates increases.

□

Random Sample Generation Using Quad-tree Model

Before starting timing analysis, a variation model must be constructed in order to determine the values of the random parameters like gate length and threshold voltage of each gate in the circuit. For this purpose, we use the quad-tree model, explained above. The random parameter values of each gate depends on its grid location and each random parameter is computed using (3.4). A drawn random sample point X is basically a realization for the random values of each grid location in the quad-tree model. Random sample generation using quad-tree model is best explained with an example:

For instance assume that there is only one random parameter gate length (L) and a 4-level quad-tree model shown in Figure 3.1 is used. As explained above, for level 0, the mean of the corresponding PDF is the nominal gate length and its variance is σ^2_{inter} as level 0 represents inter die variation. For other levels, the mean is equal to 0 and the variance is equal to σ^2_q. Given the total variance of gate length, i.e. σ^2_{total}, the variance of each level σ^2_q is determined in consistence with (3.2) and (3.3). Drawing a sample means that for each grid in the 4-level quad-tree model, a random value is generated according to the PDF of the corresponding layer in the quad-tree model. As we assumed a 4-level quad-tree model, a total of $\frac{4^4-1}{3} = 85$ grids and so 85 random variables should be generated for one sample drawing operation. As a result of this drawing, a 85 dimensional sample vector X is obtained. Using this X vector and the location of the gates, the gate lengths of all gates can be computed as shown in (3.4). For each gate, only four random variables are added as shown in (3.5), because there are only 4 levels. However, we still need the X vector, because different gates residing in different grids require different random variables selected from the 85 random variables stored in X. If another random parameter had existed, X would have been a $2 \times 85 = 170$ dimensional vector and for each gate two random parameters using (3.4) would have to be computed.

The random sample generation explained here is used not only for the extraction of the statistically critical paths explained in Section 3.3 but also for IS based Monte Carlo explained in Chapter 4. In our work, we consider two basic statistical parameters: the gate channel length (L) and the threshold voltage (V_t). We use four levels in the quad-tree model including a top level covering the whole area of the circuit with one grid rectangle. As a result, a total of 170 random variables exist in our circuit, as explained in the previous paragraph.

3.2 Gate Delay Models

In the timing yield estimation methodology proposed in this work, an approximate but cheap (in terms of evaluation cost) gate delay model is used as the key tool in devising an effective biasing distribution for importance sampling in a unique manner to accelerate Monte Carlo yield analysis. The gate delay model is also used for the detection of statistically critical paths, which is explained in Section 3.3. Up to now, two approximate but fast gate delay models have been proposed by us. In previous work [73], we have employed a stochastic version of the logical effort gate delay model introduced in Section 2.2.4, for this purpose. Section 3.2.1 explains this model. Because this model does not consider the effect of input slope and thus not accurate enough, we have later proposed a more advanced polynomial gate delay model (PDM), which is explained in Section 3.2.2. Concepts like input slope, load capacitance used in this section are introduced in Section 2.1.

3.2.1 Stochastic Logical Effort (SLE)

From Logical Effort (LE) to Stochastic Logical Effort (SLE)

Equation (2.9) in Section 2.2.4 provides a way of decomposing the effects of statistical parameter variations on gate delays. In a different context, Sutherland et. al [17] analyzed different semiconductor processes with varying supply voltages, and observed that almost all of the effect of process parameters and supply voltage on gate delay is captured by the reference inverter delay (τ in (2.8)), even when the parameters vary over a large range spanning different fabrication processes. The logical effort g and the unitless parasitic delay p of a gate exhibit relatively little variation with process parameters. Exploiting this observation in the context of timing yield analysis, in [74] a stochastic logical effort (SLE) model was proposed where the delay of a gate was modeled as

$$d_r^{LE}(X) = \tau(X)\,(p + g\,h) \tag{3.12}$$

where X is a vector of random variables as explained in Section 3.1 and $\tau(X)$ is the reference inverter delay when the parameters are given by X. p is the parasitic component, g is the logical effort and h is the fanout, which are described in detail in Section 2.2.4. As is apparent in this equation, in the stochastic logical effort approximation, all process and environmental variations are captured by the statistical variable τ while g, p and therefore $d = (p + g\,h)$ are assumed to be independent of process parameters. If only inter-die variations are modeled as it was assumed in [74], or equivalently, if the quad-tree model with only its top level[1] is used, then statistical parameters on the chip at all locations are perfectly correlated and X has a dimension equal to the number of random device parameters. For instance, if only gate length and threshold voltage are assumed to be random parameters, then X is a two dimensional vector. In this case, using the stochastic characterization of τ for the same reference inverter for all of the logic gates on the die captures this perfect statistical correlation among gates. We refer to the approximation given in (3.12) as *first-degree stochastic logical effort* (abbreviated as *SLE.d1*).

A further refinement of this approximation is described by the following equation

$$d_r^{LE}(X) = \tau(X)\,(p(X) + g(X)\,h) \tag{3.13}$$

where the dependency of p and g on X is also modeled. We call this model *second-degree stochastic logical effort* (*SLE.d2*). *SLE.d2* is more accurate as it considers the variations of p and g but computationally it is more expensive.

In both versions of SLE, in order to compute the delay of a path π in a circuit, we simply add the delays of the gates on π:

$$d_\pi^{LE}(X) = \sum_{r=1}^{k} d_r^{LE}(X) \tag{3.14}$$

Here $d_r^{LE}(X)$ is the delay of the r-th gate on the path π. $d_r^{LE}(X)$ is computed by evaluating (3.12) for *SLE.d1* and (3.13) for *SLE.d2*. For this evaluation, a full transistor-level simulation of the whole circuit containing the logic path is not necessary. However, the values of $\tau(X)$ (for both *SLE.d1* and for *SLE.d2*), and $p(X)$ and $g(X)$ (for *SLE.d2*) at a given X are needed. These derivations are explained below.

Computation of SLE Model Parameters

They can be computed at a given X by running transistor-level circuit simulations on small test circuits which contain only the reference inverter (for $\tau(X)$) or the gate under consideration (for $p(X)$ and $g(X)$) together with a proper driver and load circuitry [17]. Figure 3.3 shows the test circuit constructed with only reference inverters to compute $\tau(X)$. For computing $\tau(X)$, first the random parameters of all inverters in the figure are set according to X and then the number of inverters connected to the output of a previous inverter is iterated from 1 to 3. This means that h

[1]It is not rational to call it quad-tree model as the quad-tree model is proposed to model intra-die variations, which require more than one level.

is iterated from 1 to 3 as h is equal to the output load capacitance over input capacitance. At each iteration, by performing TL transient analysis explained in Section 2.2.1, the delay of the inverter between nodes 3 and 4 is recorded. As a result, a plot similar to Figure 3.2 is obtained. The x-axis

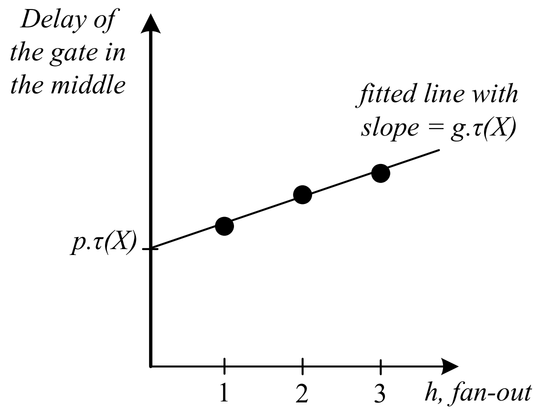


Figure 3.2: The fanout (h) vs. delay plot for the gate between nodes 3 and 4 of the test circuit

in the plot is fanout, h, and the recorded delays are marked by big black dots. A line is fitted to the marked dots as shown in the figure. This line has the slope $\tau(X)g$. As g is 1 for the reference inverter by definition, the slope is equal to $\tau(X)$. Also the point where the line intersects the y-axis is equal to $\tau(X)p$ if *SLE.d1* is used. If *SLE.d2* is used, a similar extra test circuit with a similar plot should be used for each gate type[2] to compute $p(X)$ and $g(X)$. But this time, instead of inverters, the test circuit in Figure 3.3 is constructed with the gate type, whose $p(X)$ and $g(X)$ values will be computed. The slope of the fitted line is used to compute $g(X)$, whereas the point of intersection with the y-axis is used to compute $p(X)$ of the corresponding gate type. In [73], we implemented both SLE versions above and demonstrated the results on two very simple single path circuits while considering only the inter-die variations.

Disadvantages of SLE

Both SLE versions have some disadvantages in terms of accuracy and efficiency when complex circuits with intra-die variations are considered. Starting with the accuracy problems, SLE does not

[2]Gate types refer to different gates with different number of inputs and different functionality like AND, OR, NOR, NAND, etc.

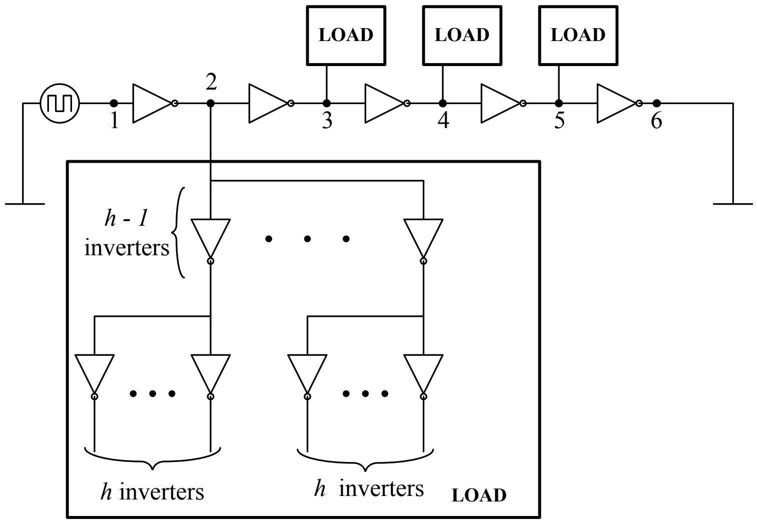


Figure 3.3: Reference circuit for $\tau(X)$ computation

model input signal's slope, although the input slope directly affects the delay of the gate especially when unbalanced loads occur in the circuit. In order to observe the lack of efficiency, assume that gate length (L) and threshold voltage (V_t) are the only random process parameters in the circuit. Then, if the intra-die variations are ignored, each sample X has two components, one for gate length and one for threshold voltage. If the intra-die variations are not ignored, each gate on a path may have a different (L, V_t) pair. In this case, even the gate types of the gates are same, for each gate, a new costly TL simulation is required to compute $\tau(X)$ (for both *SLE.d1* and for *SLE.d2*), $p(X)$ and $g(X)$ (for *SLE.d2*), which makes SLE inefficient especially when intra-die variations are not ignored.

Refinements on SLE

Alternatively, the parameters $\tau(X)$, $p(X)$ and $g(X)$ can be modeled by polynomials in terms of random process parameters like gate length. In this case, a few number of TL simulations can be enough to construct the polynomial model. As a result, for the assumption of only two random parameters as above, the delay of a gate transforms into a polynomial of L, V_t and fan-out (h). Using polynomials increases the efficiency as the only significant cost is the construction of the polynomial coefficients and this is done only once for a standard cell library[3]. After computing the coefficients for each gate type, i.e. characterizing the cell library, the delay of any gate and corresponding to any X can be computed by a simple polynomial evaluation. Here, it should be noted that similar pre-characterizations of standard cell libraries are typically performed for designers to be able to estimate the performance of their designs.

As a result, to overcome the accuracy and efficiency problems of SLE, we advance the SLE model by using polynomials and taking the effect of input slope into account. In this advanced model, gate delays are represented as a polynomial of input slope, fanout and random process parameters. This model, which we use as an approximate delay computation method throughout this thesis, is called *polynomial delay model* (PDM) and will be explained next.

3.2.2 Polynomial Delay Model (PDM)

The polynomial gate delay model uses third-degree polynomials to express the delay and the output slope as a function of the random process and transistor parameters, input slope and load (fanout) of the gate. This polynomial gate delay model (PDM) requires more computational resources to construct (but still very cheap to evaluate), but it is more accurate than the logical effort delay model and results in a much more effective biasing distribution for importance sampling, which will be clarified in Section 4.2.

If the channel length L and threshold voltage V_t are considered as the random transistor parameters, then the delay and the output slope of a gate r can be represented with

$$d_r^{PDM}(L_r, V_{tr}, h_r, InS_r) \tag{3.15}$$

and

$$OutS_r^{PDM}(L_r, V_{tr}, h_r, InS_r) \tag{3.16}$$

where L_r and V_{tr} are the random parameters for the transistors in gate r, h_r is the fanout, and InS_r is the input slope. $OutS_r^{PDM}$ is the output slope and d_r^{PDM} is the delay of gate r computed by PDM. Actually, we have considered high to low, low to high delays and high to low, low to high output slopes separately, which makes a total of four different polynomial models for each gate type instead of the two models shown in (3.15) and (3.16). But, for the sake of simplicity, we will explain our methods and implementations as if we only use the two PDM models in the above equations.

[3]A standard cell library is a collection of particular logic gates, which is used by designers to construct their digital circuit designs.

Path Delay Evaluation with PDM

Using this model, the delay of a path π with k gates in a circuit can be easily computed as follows. First, given the input slope of the first gate in the path (dictated by a primary input), the input slopes of all the other gates are computed using (3.16) and

$$InS_{i+1} = OutS_i^{PDM}(L_i, V_{t\,i}, h_i, InS_i), \quad i = 1, \ldots, k-1 \tag{3.17}$$

Then, the delay of the path is computed with

$$d_\pi^{PDM}(X) = \sum_{i=1}^{k} d_i^{PDM}(L_i, V_{t\,i}, h_i, InS_i) \tag{3.18}$$

where X is the vector that collects all of the random variable realizations used in the quad-tree model. The transistor parameters L_i and $V_{t\,i}$ are computed using X and (3.4) as explained in Section 3.1.

Construction of PDM

The polynomial delay models need to be constructed for the standard cell library that is being used. Delay look-up models for gates are routinely constructed in standard cell characterizations. These delay models have traditionally been used for static timing analysis. The delay model extraction needs to be done only once for a standard cell library for a given fabrication process. In order to construct the gate delay and output slope models for the gates in our library, we run SPICE simulations at suitably chosen sample points and fit third-order polynomials to the simulation data using a least-squares technique. For the results presented in this thesis, delay models were constructed with SPICE simulations run per gate at 1700 sample points in the parameter space. These 1700 sample points were generated as follows. For the two random parameters considered (L and V_t in this work), 425 sample points were placed non-uniformly in the rectangle in the L-V_t plane bounded by $\mu - 3.\sigma$ and $\mu + 3.\sigma$ for each parameter, where μ is the mean and σ is the standard deviation of the parameter. The sampling frequency was three times higher in the center $\mu - \sigma$ to $\mu + \sigma$ interval as shown in Figure 3.4. Only two samples (values) for both input slope and load were used due to almost linear dependence of delay on these parameters. As a result, we end up with $425 \times 2 \times 2 = 1700$ points at which SPICE simulations are run. For fanout and input slope only two values are used because the gate delay versus fanout or input slope relationship is almost linear. This can be seen from Figures 3.5(a) and 3.5(b). Figure 3.5(a) is created by iterating the fanout of a 4-input NOR gate and plotting the corresponding gate delays by blue asterisks whereas Figure 3.5(b) is created by iterating the input slope of a 5-input AND gate and plotting the corresponding gate delays by blue asterisks[4]. In each of these figures, the red dashed line represents a line, passing through the first and the last asterisks, for reference and the blue lines simply connect two consecutive asterisks. It can be seen from these figures that both fanout and input slope have almost linear relationships with gate delay. The linear gate delay versus fanout relationship was also assumed by the SLE model explained above.

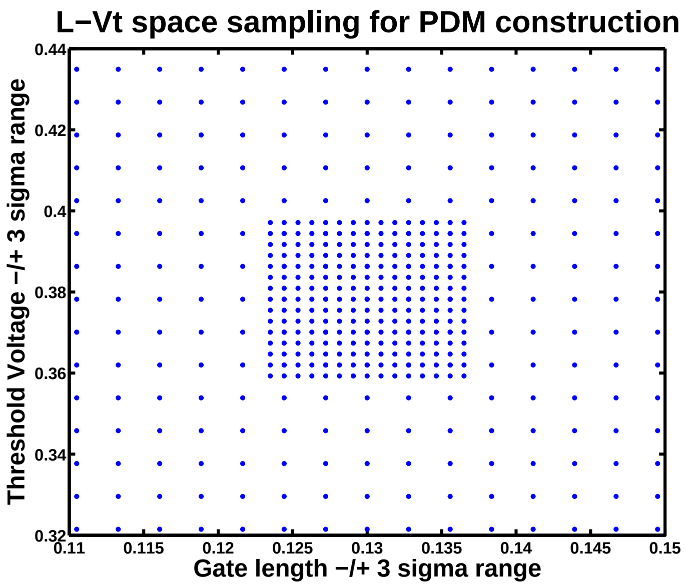
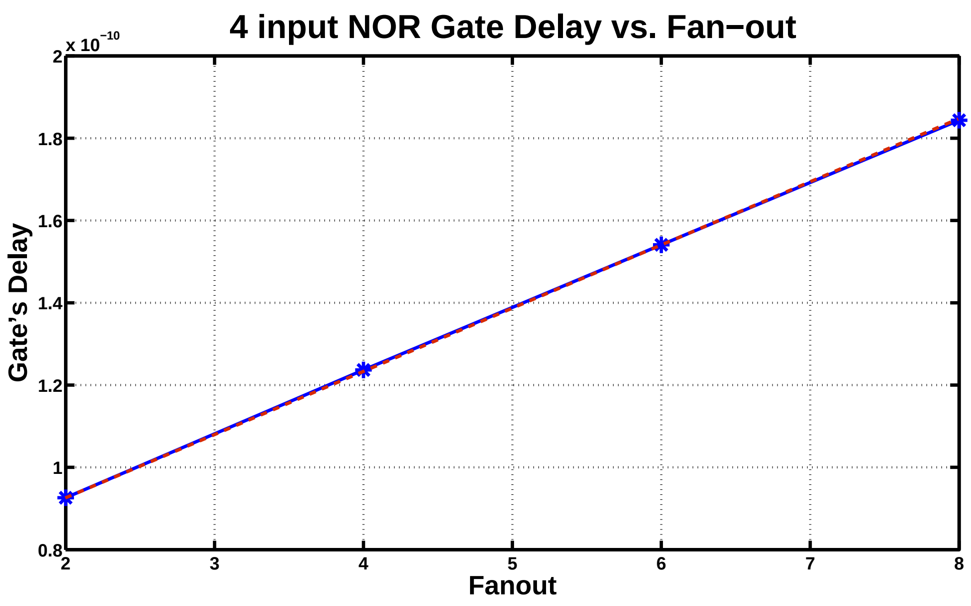


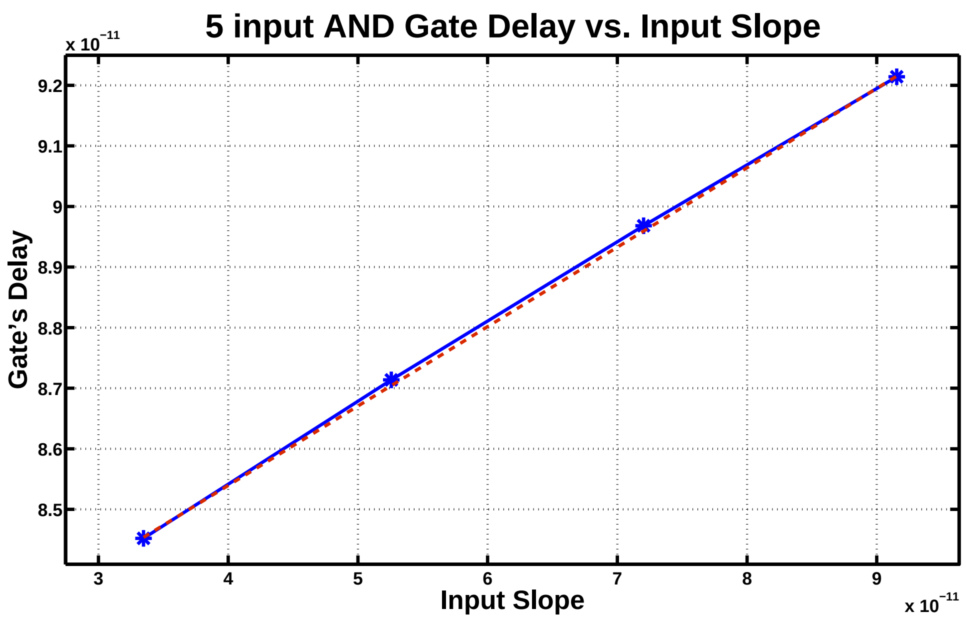
Figure 3.4: Sampling of L-V_t plane for polynomial delay model generation

We should point out that the parameter space sampling scheme described here for fitting and building the gate delay model is only rudimentary and was considered adequate for the results we present in this thesis. If a larger number of random transistor parameters are included in the gate delay model, a more efficient sampling scheme that does not have exponential complexity, such as Latin hypercube sampling [75], needs to be employed. Efficient and effective design of experiments [75] (selection of sample points in the parameter space) in statistical model fitting is a well studied problem in statistics and beyond the scope of this thesis.

[4]These two gate types are selected randomly as all gate types have similar plots.



(a) Gate delay versus fanout



(b) Gate delay versus input slope

Figure 3.5: The linearity of delay versus fanout and input slope

Accuracy of PDM

We use PDM to approximate path delays. Therefore, its accuracy should be tested by comparing path delays computed by PDM with the actual path delays computed by TL simulation. In Figure 3.6, scatter plots that show the accuracy of the polynomial delay model against SPICE TL simulations is presented. In order to generate the graph in Figure 3.6, the delay of a complete path in each circuit in the ISCAS'85 benchmark suite was determined both by TL circuit simulations ($d_\pi^{TL}(X)$) and by evaluating the polynomial gate delay model ($d_\pi^{PDM}(X)$) at a number of sample points in the parameter space. This plot has 20,000 points that are generated as explained at the end of Section 3.1 considering the intra-die variations with spatial correlation and by using the 4-level quad-tree model shown in Figure 3.1. Two random transistor parameters are considered, gate length (L) and threshold voltage (V_t). The red line in the plots is the $x = y$ line used to visualize the shifts and errors of polynomial delay model. The polynomial delay model captures the trends and relative variations in delay as a function of the transistor parameters quite accurately. However, the delay model is not accurate enough to replace transistor-level simulation in predicting timing yield with sufficient accuracy as will be seen in Chapter 5. We use this model in order to detect statistically critical paths as explained in Section 3.3 and to construct an effective biasing distribution to be used in importance sampling as explained in Section 4.2, but this model is not meant to be a replacement for transistor-level simulation in accurately determining the delay of a circuit.

Other than the visual demonstration of the path delay accuracy of polynomial delay model shown by Figure 3.6, the accuracy can be computed by the well-known *root mean square error* (RMSE) computation. (3.19) shows the RMSE computation for the path delays computed by polynomial delay model.

$$\mathrm{PDM}_{\mathrm{RMSE}} = \sqrt{\frac{\sum_{i=1}^{N} (d_\pi^{TL}(X_i) - d_\pi^{PDM}(X_i))^2}{N}} \tag{3.19}$$

where $d_\pi^{TL}(X_i)$ is the actual path delay computed by SPICE TL simulations for the given X_i sample point and $d_\pi^{PDM}(X_i)$ is the approximate path delay computed by polynomial delay model for the same X_i. X_i's are generated as explained at the end of Section 3.1 based on 4-level quad-tree model and N is the number of drawn sample points. The normalized root mean square error (NRMSE) can be computed through dividing the RMSE by the sample mean of actual path delay as shown

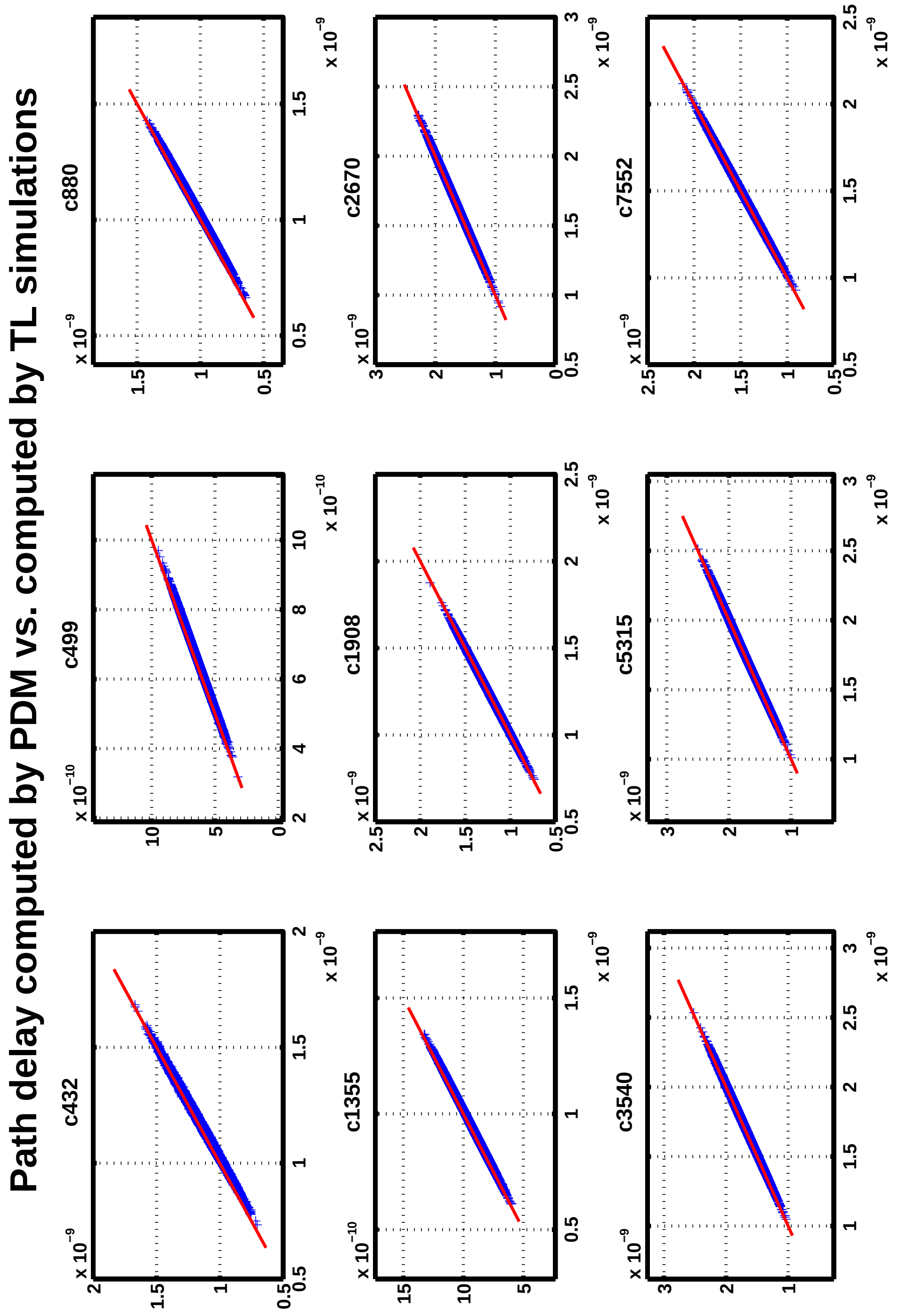


Figure 3.6: Accuracy of polynomial delay model (PDM)

in (3.20).

$$\mathrm{PDM}_{\mathrm{NRMSE}} = \frac{\mathrm{PDM}_{\mathrm{RMSE}}}{\frac{\sum_{i=1}^{N} d_{\pi}^{TL}(X_i)}{N}} \tag{3.20}$$

For the same 20,000 sample points used in Figure 3.6, the computed percentage $\mathrm{PDM}_{\mathrm{NRMSE}}$ value for a path taken from each benchmark circuit is shown in Table 3.1.

Table 3.1: Percentage NRMSE of PDM for a path in each circuit of ISCAS'85 benchmark

Circuit Name	**c432**	**c499**	**c880**	**c1355**	**c1908**	**c2670**	**c3540**	**c5315**	**c7552**
$\mathrm{PDM}_{\mathrm{NRMSE}}$ %	**1.69**	**2.52**	**1.87**	**1.66**	**0.42**	**0.77**	**1.00**	**0.44**	**0.46**

Discussion

Gate delay models are utilized in almost all statistical timing analysis methodologies. The nature of the algorithms used in statistical analysis may impose restrictions on the complexity and form of these models. For instance, in block-based statistical timing analysis (SSTA) schemes based on PDF algebra/propagation, linear or at most quadratic models are used in order to make the PDF computations tractable and practical. In our methodology, the only requirement on the delay model is that it be cheap to evaluate. Otherwise, there is no restriction on the complexity (can use higher-order polynomials) or form (not restricted to polynomial models) of the delay model. A more complex delay model may result in a larger construction cost, but again, this is done only once for a gate library for a given process. The ability to use more accurate and complex gate delay models is one of the key benefits of our methodology.

3.3 Collection of Statistically Critical Path Candidates

Reminder for Critical and Statistically Critical Paths

Critical path in a circuit is the longest true path which is responsible for the delay of the circuit as explained in Section 2.2.3. When the statistical process variations are considered, the circuit may have different critical paths for different assignments to the random variables in the circuit. A path which is critical for one assignment may not be critical for another assignment. As a result, the set of *statistically critical* paths consists of the paths that are critical for at least one assignment to random parameters. The collection of statistically critical paths is a hot research topic as explained in Section 2.3.1.

The Purpose of This Section

The purpose of this section is to explain the method we use for the extraction of the statistically critical path *candidates* in a circuit with inter and intra-die variations. They are only candidates because some of them may be false paths and so not responsible from the circuit's delay as explained in Section 2.2.3. Next section explains how we detect these false paths among these candidates and eliminate them to form a statistically critical paths set. Below, the overview and the details of the method used for the detection of statistically critical path candidates will be given.

Overview of The Method

For the extraction of the statistically critical path candidates in a circuit, a block level Monte Carlo (BL-MC) method based on DSTA as explained in Section 2.4.2 is preferred for our work. A quad-tree structure explained in Section 3.1 is used to model the inter die variations and intra-die variations with spatial correlation. The BL-MC is applied as follows:

1. According to the quad-tree model, N random sample points X_i's are drawn.

2. For each X_i, the random parameter values, i.e. gate length (L) and threshold voltage (V_t) in our case, for all gates in the circuit are computed from X_i as described at the end of Section 3.1. Then, having L and V_t corresponding to each gate, input slopes and gate delays are computed using PDM as explained in Section 3.2.

3. Knowing the gate delays (e.g. numbers inside the nodes of timing DAG in Figure 2.6), a DSTA analysis of Section 2.2.2 is performed to compute the circuit delay and the topologically longest path corresponding to the sample point X_i.

4. At the end of DSTA the discovered topologically longest path for sample X_i is recorded to the statistically critical path candidates set. At the end of N iterations, this set has the paths each of which is the topologically longest path for at least one of the N sample points.

The value of N can be very high as DSTA with PDM polynomial evaluations is a fast operation that can be repeated for many times. Below one iteration of this method is explained in detail starting from the layout representation of a circuit.

Details of the Method

We start with a design exchange file (def) representation of a circuit [76]. Def file is the output of layout (place and route) tools and has cell info, layer information, complete net list connectivity, floor plan specs, physical location of every design instance (gate), routing geometry data and etc.

The def files are converted into a timing DAG $C(V,E)$, as demonstrated in Figure 2.6. In this DAG structure, each logic gate has an ID number and important features extracted from the def file. These features are *type*, *ancestors*, *neighbors* and *fanout* of the logic gate. For instance, a gate with ID 654 from an example circuit has the following features:

type: 'and2'
ID: 654
neighbors: [747 742 737 730 725 721 719 714 712]
ancestors: [630 645]
fanout: 2.42
L: 0.1299
V_t: 0.2032

These features show that this example gate is a 2 input AND gate with *ID* 654 and *fanout* 2.42. The gate's output is connected to 9 gates, whose IDs are shown by *neighbors* attribute. It has two inputs connected to the outputs of the gates with IDs 630 and 645 as shown by *ancestors* feature. The last two features are the random parameters for the gate. As we assume two random parameters, gate length (L) and threshold voltage (V_t), for each gate, there are two features corresponding to L and V_t.

A random sample X_i is drawn according to the 4-level quad-tree model as explained in Section 3.1 before performing a DSTA analysis as the overview of the method suggests. As we assume two random parameters (L and V_t), this results in 170 dimensional sample point vector X_i as computed in Section 3.1. After the random sample X_i is drawn, the L and V_t values of each gate are computed using Eqn. 3.4 given the random sample X_i and the position of the corresponding gate extracted from the def file in the quad-tree model as described in Section 3.1. After drawing the random sample X_i and determining the random parameters (L and V_t) of all gates according to the X_i, DSTA algorithm can be applied as explained below.

Our DSTA implementation on a circuit timing DAG ($C(V,E)$) with features explained above is shown in Algorithm 1. The algorithm takes the circuit DAG C including the gate features introduced above and the PDM coefficients as input arguments. $PrimaryInSlope$ in the algorithm corresponds to the default input slope for the primary inputs of the circuit. $V(IDnum)$ expression in the algorithm returns a pointer to the gate (vertex) with ID $IDnum$. First, the algorithm topologically sorts the timing DAG such that the ID of every vertex comes after the IDs of all its predecessors and these topologically sorted IDs of the gates are put in $SortedIDs$ array. Then, by the help of this topologically sorted array, the algorithm processes the gates one by one from leftmost gates connected to the primary inputs to the rightmost gates connected to the primary outputs. For each gate, it computes the possible maximum output arrival time and records it as $arrivaltime$ feature of the gate. The algorithm also records the corresponding ancestor gate, which causes this maximum output arrival time as $delayfather$. As a result, Perform-DSTA algorithm inserts new features like $arrivaltime$, $gatedelay$, $outputslope$, $delayfather$ for each gate (vertex) of C:

- $arrivaltime$: The computed maximum output arrival time for the corresponding gate.

- $gatedelay$: Gate's own delay computed by PDM method and used in the computation of $arrivaltime$.

- $outputslope$: The output slope of the gate, which is required for PDM delay and output slope computation of the gates connected to the output of this gate.

- $delayfather$: The ID of the ancestor gate which results in maximum output arrival time for the corresponding gate.

After this DSTA analysis, it is very easy to find the topologically longest path. This path is found by first finding the gate having the maximum output arrival time. Such a gate is certainly a rightmost gate, whose output is a primary output. This gate's ancestor, which is responsible from its output arrival time to be such high, can be found by looking at the gate's $delayfather$ feature. Then all gates in the topologically longest path are traced by iteratively following the IDs in $delayfather$ features of the succeeding gates until a leftmost gate, whose inputs are primary inputs is reached.

The operation explained above, which consists of drawing a random sample according to quad-tree model, determining the parameters of the gates inside the circuit according to this random sample, then performing the algorithm shown by Alg. 1 and determining the topologically longest path in the circuit is repeated for N different sample points X_i and all collected topologically longest paths constitute the statistically critical path candidates set.

An Alternative Method

Alternatively, DSTA could be performed only once but instead of collecting only the topologically longest path, all paths having delays closer to the topologically longest path could be collected as statistically critical path candidates. Because, one can argue that only the paths in a delay proximity to the topologically longest path may become the topologically longest path for a different assignment to the random values in the circuit. We have implemented this alternative method and its details are in Section A. Other than our implementation, there are well known algorithms [77] for extracting the K-most critical paths in the circuit.

Conclusion

For the solution of this problem, instead of BL-MC explained above, effective block-based SSTA statistically critical path extraction methods (e.g. in [78, 79]) could have been used as well. In any case, the extraction of these critical path candidates is very important for both timing analysis and optimization of digital circuits.

After collecting the statistically critical path candidates, these candidates should be tested by a sensitization criteria as explained in Section 2.2.3, to eliminate the false paths, because false paths are not responsible for the delay of the circuit in any case. The next section explains the method we use for detecting and eliminating the false paths in the statistically critical path candidates set.

Algorithm 1 Perform-DSTA($C(V,E)$, PDM model for delay and output slope)

```
1.  $SortedIDs =$ Topological-Sort(C)
2.  **for** $current = 1$ **to** $length(SortedIDs)$ **do**
3.    $r = V(SortedIDs(current))$
4.    **if** $r.ancestors == \emptyset$ **then**
5.      $r.outputslope = OutS_r^{PDM}(r.L, r.V_t, r.fanout, PrimaryInSlope)$
6.      $r.gatedelay = d_r^{PDM}(r.L, r.V_t, r.fanout, PrimaryInSlope)$
7.      $r.arrivaltime = r.gatedelay$
8.      $r.delayfather = \{\ \}$
9.    **else**
10.     $r.arrivaltime = 0$
11.     **for** $AncestorInd = 1$ **to** $length(r.ancestors)$ **do**
12.       $ancestor = V(r.ancestors(AncestorInd))$
13.       $CandidateOutputslope = OutS_r^{PDM}(r.L, r.V_t, r.fanout, ancestor.outputslope)$
14.       $CandidateGatedelay = d_r^{PDM}(r.L, r.V_t, r.fanout, ancestor.outputslope)$
15.       $CandidateArrivaltime = CandidateGatedelay + ancestor.arrivaltime$
16.       **if** $CandidateArrivaltime > r.arrivaltime$ **then**
17.         $r.arrivaltime = CandidateArrivaltime$
18.         $r.gatedelay = CandidateGatedelay$
19.         $r.outputslope = CandidateOutputslope$
20.         $r.delayfather = ancestor.ID$
21.       **end if**
22.     **end for**
23.   **end if**
24. **end for**
```

3.4 False Path Detection Using Satisfiability

Overview

In Section 3.3, the method we use to collect the statistically critical path candidates is explained in detail. These candidates should be tested in order to detect the true paths and label them as the *statistically critical paths*. The method proposed in this section discriminates the statistically critical path candidates using the *static sensitization* condition. The false path issue and the static sensitization are explained in Section 2.2.3. As a reminder, for a path to be statically sensitizable, all side inputs on this path should be able to take non-controlling values at the same time for at least one input assignment. If a path is statically sensitizable it is definitely a true (sensitizable) path, thus a signal can propagate through this path and this path can be responsible from circuit delay. Our method, similar to [3], converts the problem into a *boolean satisfiability* (SAT) problem and uses popular SAT solvers in order to decide whether a path is statically sensitizable, i.e. true path.

Boolean Satisfiability Problem

A boolean function consists of logic operations AND, OR and NOT. For the boolean formulas in this thesis, $\neg$ represents a NOT, $\vee$ represents an OR and $\wedge$ represents an AND. Any boolean function can be represented by using these operators. (3.21) and (3.22) are two examples for boolean functions.

Boolean satisfiability problem is a decision problem and its aim is to detect whether a given boolean (logic) function is *satisfiable*. A boolean function is satisfiable if it can evaluate to TRUE (logic-1) for an assignment to the boolean variables (literals) in boolean function. The detection of this assignment is also included in the solution of satisfiability problem. Satisfiability is the first problem that is proven to be NP-complete. For instance, the boolean formula in (3.21) is satisfiable for the input assignment $A, B = 1, 1$ as it evaluates to logic-1 (TRUE) for this assignment, whereas the formula in (3.22) is unsatisfiable, which means that there is no input assignment that makes the function evaluate to TRUE (logic-1). An assignment that results in boolean function evaluate to TRUE is said to satisfy the boolean function. Although the problem is proven to be NP-complete, there are many heuristics called SAT solvers and they are very efficient and successful in determining whether a given boolean function is satisfiable or unsatisfiable and if it is satisfiable, they can detect which input assignment satisfies the function. There are even competitions to pick the best SAT solver of the year [80].

$$(a \vee \neg b) \wedge (\neg a \vee b) \tag{3.21}$$

$$(a \vee \neg b) \wedge (\neg a \vee b) \wedge \neg b \tag{3.22}$$

Conjunctive Normal Form (CNF)

In boolean logic, a function is in *conjunctive normal form* (CNF) if it is a conjunction (AND) of clauses, each of which is a disjunction (OR) of literals (boolean variables) where negative (NOT) literals are possible. It is also called *product of sums* form. Two boolean functions in (3.21) and (3.22) are written in CNF as the clauses of OR functions are connected by AND operator. All logic functions corresponding to the logic gates in a standard cell library can be converted into a CNF form. CNF is an important standard for satisfiability because almost all SAT solvers accept boolean functions only in CNF form as an input. For this reason, if it is desired to know whether a boolean function is satisfiable or not, then it should first be converted into a CNF form before testing its satisfiability by a SAT solver.

Representation of Combinational Circuits as a CNF Satisfiability problem

Table 3.2: Gate type equations and the corresponding CNF formulas

gate type	**equation**	**CNF formula**
not	$x = \neg a$	$(a \vee x) \wedge (\neg a \vee \neg x)$
and	$x = a \wedge b$	$(\neg a \vee \neg b \vee x) \wedge (a \vee \neg x) \wedge (b \vee \neg x)$
or	$x = a \vee b$	$(a \vee b \vee \neg x) \wedge (\neg a \vee x) \wedge (\neg b \vee x)$
nand	$x = \neg(a \wedge b)$	$(\neg a \vee \neg b \vee \neg x) \wedge (a \vee x) \wedge (b \vee x)$
nor	$x = \neg(a \vee b)$	$(a \vee b \vee x) \wedge (\neg a \vee \neg x) \wedge (\neg b \vee \neg x)$

A simple boolean equation can be represented as a CNF satisfiability problem. Table 3.2 shows the CNF equivalents of the boolean equalities corresponding to different gate types. For each gate type, the assignments to the variables that make the corresponding CNF formula evaluate to TRUE, give all allowable values that the variables can get without violating the equation of the corresponding gate type. For instance, for NOT gate, the output (x) is the complement of the input (a), i.e.

$x = \neg a$. The only allowed assignments for (a,x) pair are $(0,1)$ and $(1,0)$. These two possible assignments are the only assignments that satisfy the corresponding CNF formula. Similarly for AND gate with equation $x = a \wedge b$ the possible assignments for (a,b,x) are $(0,0,0)$, $(0,1,0)$, $(1,0,0)$, $(1,1,1)$ and these four assignments are also the only assignments that satisfy the corresponding CNF representation of the equation. Using this fact we can also convert a combinational circuit into a CNF satisfiability instance. In such a case, the variables in the circuit can only take the logic values that satisfy the corresponding CNF formula. For instance, the simple circuit in Figure 3.7 can be represented as a CNF formula showed in (3.23). An assignment to variables, which satisfies this CNF, has the logic values that the variables in the circuit are able to get.

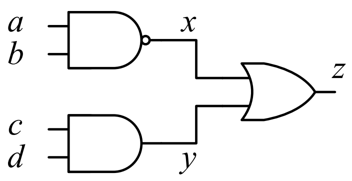


Figure 3.7: Sample combinational circuit

$$\begin{aligned} CNF = {} & (\neg a \vee \neg b \vee \neg x) \wedge (a \vee x) \wedge (b \vee x) \\ & \wedge(\neg c \vee \neg d \vee y) \wedge (c \vee \neg y) \wedge (d \vee \neg y) \\ & \wedge(x \vee y \vee \neg z) \wedge (\neg x \vee z) \wedge (\neg y \vee z) \end{aligned} \tag{3.23}$$

The Satisfiability Based Method to Detect True Paths

We can insert test conditions to the circuit CNF formulas and check whether these conditions are satisfied. For instance, the static sensitization condition was that all side inputs of the path under consideration should have non-controlling values. Assume that the static sensitization of the path $a-x-z$ in Figure 3.7 will be tested. The side inputs for this path are b and y. Simultaneously, b should be 1 and y should be 0 in order to have non-controlling values for the side inputs of the path $a-x-z$. As explained in Section 2.2.3 in detail, this is because b is a side input of a NAND gate and y is a side input of an OR gate. We can insert this condition at the end of the CNF description of the circuit in (3.23) as shown in (3.24).

$$
\begin{aligned}
& CNF_{a-x-z} = (\neg a \vee \neg b \vee \neg x) \wedge (a \vee x) \wedge (b \vee x) \\
& \wedge(\neg c \vee \neg d \vee y) \wedge (c \vee \neg y) \wedge (d \vee \neg y) \\
& \wedge(x \vee y \vee \neg z) \wedge (\neg x \vee z) \wedge (\neg y \vee z) \\
& \wedge b \wedge \neg y
\end{aligned}
\tag{3.24}
$$

For the CNF formula in (3.24) to be satisfiable, i.e. evaluate to TRUE; obviously b should be 1 and y should be 0. Alternatively, if CNF_{a-x-z} is satisfiable then the side inputs can simultaneously take non-controlling values and therefore the path $a-x-z$ is statically sensitizable, which means it is certainly a true path. By using this satisfiability based true path detection method, the static sensitization of a path in the statistically critical path candidates set is checked as follows:

1. First of all, the CNF formula corresponding to the circuit under consideration is generated.

2. Secondly, the non-controlling values for the side inputs of the path under consideration are determined and these are inserted to the circuit's CNF formula as conditions similar to what is done in (3.24).

3. The satisfiability of the resultant CNF formula is tested by a SAT solver like [81]. If the path is satisfiable then this means the path is statically sensitizable and a true path and thus it is kept. If it is unsatisfiable, then the path is discarded.

When this operation is applied to all paths in the *statistically critical path candidates* set, a new set called *statistically critical paths set* consisting of only true paths is obtained.

The next chapter explains a novel timing yield estimation method based on importance sampling and transistor level Monte Carlo simulations. The method uses the statistically critical paths set, which is generated as explained in this chapter. The aim of this method is to speed up the transistor level statistical timing analysis so that an accurate timing yield estimation can be performed as a final verification before timing sign-off.

Chapter 4

MC YIELD ESTIMATION WITH IMPORTANCE SAMPLING AND TRANSISTOR LEVEL SIMULATION

In this chapter, a novel loss estimation method is proposed: an improved loss estimator which is based on importance sampling (IS) that significantly accelerates the convergence of the transistor level Monte Carlo (TL-MC) estimator *without* forfeiting accuracy and enables its use in practice. Section 4.1 provides the details of transistor level Monte Carlo (TL-MC) loss estimator that is briefly reviewed in Section 2.4.2. Section 4.2 presents the novel importance sampling (IS) loss estimator. Section 4.3 explains the heuristic algorithm that practically avails the IS loss estimation. Section 4.4 provides theoretical analysis for the means, variances and errors of standard TL-MC and IS estimators and deduces a speed-up expression by comparing the speeds of both estimators.

4.1 Standard Transistor Level Monte Carlo (TL-MC) Loss Estimator

The actual loss can be precisely computed only by detailed transistor level analysis. Therefore, using Eqn. 2.14, the actual loss can be written as

$$Loss^{TL} = \int_{\Omega} I^{TL}(T_c, X) f(X) dX \tag{4.1}$$

Ω is the region, where PDF $f(X)$ is defined. The superscript $\cdot^{TL}$ indicates that the value of indicator random variable $I^{TL}(T_c, X)$ defined by (2.13) is computed based on transistor-level (TL) simulations, that is, the path delays and hence the circuit delay in (2.12) are computed with TL simulations. Computation of $d_{\pi}^{TL}(X)$ requires that the parameters of the gates inside the path π are set according to X as explained at the end of Section 3.1 and then a SPICE TL simulation (transient analysis) is performed as explained in Section 2.2.1.

However, the integral in (4.1) does not have an analytical solution and it requires an infinite number of TL simulations as it is an integration over a continuous Ω region. Monte Carlo (MC) method is widely used to estimate such complex integrations. As shown in (2.25), the standard

N-sample MC estimator (TL-MC) for $Loss^{TL}$ in (4.1) is given by

$$Loss_N^{TL} = \frac{1}{N}\sum_{i=1}^{N} I^{TL}(T_c, X_i) \tag{4.2}$$

where $Loss_N^{TL}$ is the loss estimate, T_c is the timing constraint that shows the maximum allowed circuit delay and X_i is a random sample point. Indicator variable is the same as in (4.1), but this time it is computed only for N times. In (4.2) above, the N samples for the random variables, X_i, $i = 1 \ldots N$, are drawn from the joint PDF $f(X)$ and for every sample X_i, the random parameters for the transistors in the circuit are computed as described at the end of Section 3.1. Then, a TL simulation with SPICE, reviewed in Section 2.2.1, is performed for each path in the set of statistically critical paths Π_{crit} (obtained as described in Section 3.3 and 3.4) to compute the path delays $d_\pi^{TL}(X)$, finally, (2.12), (2.13) and (4.2) are used to compute the loss estimate $Loss_N^{TL}$. As Theorem 2.4.1 proves, the TL-MC estimator in (4.2) converges to the actual loss, $Loss^{TL}$. Therefore, if *enough*[1] number of samples are drawn, TL-MC estimator can precisely estimate loss.

When compared with other methods like SSTA reviewed in Section 2.3.2, TL-MC loss estimator described above results in accurate yield estimation results, because it is based on TL simulations as opposed to an approximate gate delay model, and the maximum operation in (2.12) is *not* approximated in any manner. However, the standard TL-MC estimator typically requires too many samples (N) to converge. For each sample, one needs to perform TL simulations for all of the statistically critical paths, and hence, the computational cost of the TL-MC estimator could become prohibitive for practical use.

4.2 Transistor Level Importance Sampling (IS) Loss Estimator

The Choice for Biasing Distribution

Considering the general IS technique, described in Section 2.4.1, the IS based transistor level Monte Carlo estimator for loss

$$Loss_N^{IS} = \frac{1}{N}\sum_{i=1}^{N} I^{TL}(T_c, X_i)\frac{f(X_i)}{\tilde{f}(X_i)} \tag{4.3}$$

draws the samples X_i from another, biasing distribution $\tilde{f}$. We propose the following biasing distribution to be used in the IS estimator above

$$\tilde{f}(X) = \frac{I^{PDM}(T_c{}^{\varepsilon}, X)\, f(X)}{Loss^{PDM,\varepsilon}} \tag{4.4}$$

[1]The enough number will be clarified in Section 4.4.

where the loss estimate $Loss^{PDM,\varepsilon}$ and $I^{PDM}(T_c{}^{\varepsilon}, X)$ are computed based on the approximate but cheap gate delay model PDM described in Section 3.2, without performing any TL simulations. $I^{PDM}(T_c{}^{\varepsilon}, X)$ is computed as shown in (2.13) using PDM method for computing the path delays instead of M. Computation of $Loss^{PDM,\varepsilon}$ will be described later in this section.

Substituting the biasing distribution $\tilde{f}$ in (4.4) into (4.3), and performing some simplifications based on the fact that $I^{PDM}(T_c{}^{\varepsilon}, X_i)$ takes the value 1 for all samples drawn from $\tilde{f}(X)$, we arrive at a simplified form of the IS estimator

$$Loss_N^{IS} = \frac{Loss^{PDM,\varepsilon}}{N} \sum_{i=1}^{N} I^{TL}(T_c, X_i) \tag{4.5}$$

where the samples X_i are drawn from $\tilde{f}(X)$ in (4.4).

Requirements on Biasing Distribution

For the IS estimator in (4.3) and (4.5) to be well-defined and unbiased, two requirements introduced in Section 2.4.1 must be satisfied:

1. Assume that Θ represents the region where $\tilde{f}(X)$ is defined and nonzero. Then, the *safety requirement* necessitates that Θ region covers the space where $I^{TL}(T_c, X) f(X)$ is non-zero. In other words, for every sample X_i, $\tilde{f}(X_i)$ must be non-zero if $I^{TL}(T_c, X_i) f(X_i)$ is non-zero.

2. The *regularity requirement* necessitates that

$$\int_{\Theta} \tilde{f}(X) dX = \int_{\Theta} \frac{I^{PDM}(T_c{}^{\varepsilon}, X) f(X)}{Loss^{PDM,\varepsilon}} dX = 1 \tag{4.6}$$

As $Loss^{PDM,\varepsilon}$ is a constant value, this simplifies into

$$Loss^{PDM,\varepsilon} = \int_{\Theta} I^{PDM}(T_c{}^{\varepsilon}, X) f(X) dX. \tag{4.7}$$

Safety Requirement

In (4.4) above, the target delay is set to $T_c{}^{\varepsilon} = T_c - \varepsilon$ where ε is a margin parameter. This margin parameter is introduced in order to guarantee that $\tilde{f}(X_i)$ is nonzero everywhere $I^{TL}(T_c, X_i) f(X_i)$ is nonzero, i.e., $I^{PDM}(T_c{}^{\varepsilon}, X_i)$ must take the value 1 everywhere $I^{TL}(T_c, X_i)$ is 1. The margin parameter ε must be large enough so that the indicator variables never assume the values $I^{PDM}(T_c{}^{\varepsilon}, X_i) = 0$

(the timing constraint ${T_c}^{\varepsilon}$ is satisfied according to the PDM model) and $I^{TL}(T_c, X_i) = 1$ (the actual circuit fails to satisfy the timing constraint according to TL simulations) for any of the sample points, X_i. This condition is called the *margin condition*. We note here that the *safety requirement* above dictates that the margin condition is satisfied. In the next section, we present an algorithm for computing $Loss_N^{IS}$. As this algorithm explores a set of sample points, it also gathers the data required for computing a value of ε that satisfies the margin condition. For ease of exposition, we continue the mathematical presentation of our method as if ε is determined first, before computing $Loss_N^{IS}$. In reality, the algorithm carries out the $Loss_N^{IS}$ computation and ε determination concurrently.

Regularity Requirement

As the *regularity requirement* above dictates $Loss^{PDM,\varepsilon}$ in (4.5), should be computed as in (4.7). However, similar to (4.1), $Loss^{PDM,\varepsilon}$ integration shown in (4.7) cannot be computed analytically. To overcome this problem, $Loss^{PDM,\varepsilon}$ is estimated using a standard MC (STD-MC) estimator explained in Section 2.4.2 based on the approximate but cheap gate delay model (PDM) described in Section 3.2, without performing any TL simulations as follows

$$Loss_K^{PDM,\varepsilon} = \frac{1}{K}\sum_{i=1}^{K} I^{PDM}({T_c}^{\varepsilon}, X_i) \tag{4.8}$$

for which one can afford to use a very large number of samples K, since the evaluation of $I^{PDM}({T_c}^{\varepsilon}, X_i)$ for every sample is very cheap based on the approximate delay model. As MC estimators are unbiased, using $Loss_K^{PDM,\varepsilon}$ with a very large K instead of $Loss^{PDM,\varepsilon}$ in (4.5) can satisfy the regularity requirement. The experiments, presented in Chapter 5, show that satisfying the safety requirement is more difficult and critical for the efficiency and the accuracy of the IS estimator.

IS Estimator Evaluation

In evaluating the IS estimator, in order to draw a sample from $\tilde{f}(X)$ in (4.4), we first draw a sample X_i from $f(X)$ as described at the end of Section 3.1. We keep the sample if $I^{PDM}({T_c}^{\varepsilon}, X_i)$ evaluates to 1 at the sample point and discard it otherwise. Again, the evaluation of $I^{PDM}({T_c}^{\varepsilon}, X_i)$ is performed cheaply based on the delay model, described in Section 3.2. Each kept sample constitutes one of the X_i in (4.5). $Loss_N^{IS}$ is then computed by determining whether $I^{TL}(T_c, X_i) = 1$ for each such kept sample, i.e., by performing TL SPICE simulations.

4.3 ε-Margin Detection

This section presents CompLossMC-IS (Alg. 2), the algorithm for determining $Loss_N^{IS}$ as described by Eqn. 4.5. To accomplish this, CompLossMC-IS first generates a set of NS sample points $\mathcal{X} = \{X_1, X_2, ..., X_{NS}\}$ from the distribution f. Each of these sample points is drawn using the 4-level quad-tree model as shown in Figure 3.1 and described in Section 3.1. The choice of NS will be discussed later in this section. Let Y_i be these sample points in decreasing order of d_C^{PDM}, i.e., $\mathcal{X} = \{Y_1, Y_2, ..., Y_{NS}\}$ such that $d_C^{PDM}(Y_i) \geq d_C^{PDM}(Y_j)$ if $i < j$. Using the sample set $\mathcal{X}$, CompLossMC-IS must compute

- the subset $\mathcal{W} = \{Y_1, Y_2, ..., Y_N\} \subseteq \mathcal{X}$ consisting of all sample points for which $I^{PDM}(T_c^{\varepsilon}, Y_i)$ evaluates to 1 (using the gate delay model),
- the subset $\mathcal{Q} \subseteq \mathcal{W}$ of sample points for which $I^{TL}(T_c, Y_i) = 0$ (by performing TL simulations),
- the set $SafeMargin = \{Y_{N+1}, Y_{N+2}, ..., Y_{N+SM}\}$ (to be defined below) and the corresponding value of ε, and
- using ε above, the value of $Loss^{PDM,\varepsilon}$ as in (4.8).

Then, the loss estimate $Loss_N^{IS}$ will be computed as

$$Loss_N^{IS} = Loss^{PDM,\varepsilon} . \frac{|\mathcal{W} - \mathcal{Q}|}{|\mathcal{W}|} \tag{4.9}$$

where $|\mathcal{W} - \mathcal{Q}|$ shows the size of the set difference of $\mathcal{W}$ from $\mathcal{Q}$ and $|\mathcal{W}|$ shows the size of set $\mathcal{W}$. The first factor on the right hand side of (4.9) is the IS biasing factor, and the second factor is the fraction of points in $\mathcal{W}$ which result in a loss value. It can be seen that the numerator of the second factor ($|\mathcal{W} - \mathcal{Q}|$) represents the summation in (4.5) and the denominator ($|\mathcal{W}|$) represents N in (4.5).

The only non-straightforward task that the algorithm must carry out is the determination of the margin parameter ε. ε uniquely determines $\mathcal{W}$, $\mathcal{Q}$, $Loss^{PDM,\varepsilon}$, and thus $Loss_N^{IS}$. The requirements on ε are discussed next.

Algorithm 2 CompLossMC-IS (NS, SM, T_c)

```
1.  Generate $NS$ sample points $\{X_1, X_2, ..., X_{NS}\}$ from $f(X)$.
2.  For each $X_i$, compute $d_C^{PDM}(X_i)$.
3.  Let $\mathcal{X} = \{Y_1, Y_2, Y_3, ..., Y_{NS}\}$ be the $NS$ samples
    in decreasing order of $d_C^{PDM}(Y_i)$.
4.  $i = 1$, $\mathcal{Z} = \emptyset$, $SafeMargin = \emptyset$
5.  while ($|SafeMargin| < SM$ and $i \leq NS$) do
6.    $d_C = d_C^{TL}(Y_i)$
7.    if ($d_C < T_c$) then
8.      $\mathcal{Z} = \mathcal{Z} \cup \{Y_i\}$
9.      if $SafeMargin == \emptyset$ then
10.       $\varepsilon = T_c - 0.5(d_C^{PDM}(Y_i) + d_C^{PDM}(Y_{i-1}))$
11.     end if
12.     $SafeMargin = SafeMargin \cup \{Y_i\}$
13.   else
14.     $SafeMargin = \emptyset$
15.   end if
16.   $i = i + 1$
17. end while
18. Let $N = i - SM - 1$ and $SafeMargin = \{Y_{N+1}, ..., Y_{N+SM}\}$ .
19. Let $\mathcal{W} = \{Y_1, ..., Y_N\}$
20. $\mathcal{Q} = \mathcal{W} \cap \mathcal{Z}$
21. Using a huge set of K samples compute $Loss^{PDM,\varepsilon} = \frac{1}{K}\sum_{i=1}^{K} I^{PDM}({T_c}^{\varepsilon}, X_i)$
22. $Loss_N^{IS} = Loss^{PDM,\varepsilon} \, . \, |\mathcal{W} - \mathcal{Q}| \, / \, |\mathcal{W}|$
```

Constraints on ε

For importance sampling to provide an unbiased estimator in our approach, ε must be large enough to satisfy the safety requirement that for every value of X that $f(X)I^{TL}(T_c,X)$ is non-zero, $\tilde{f}(X)$ is also non-zero. This translates to the requirement that $I^{TL}(T_c,X_i) = 1 \Rightarrow I^{PDM}(T_c{}^{\varepsilon},X_i) = 1$. Let us define ε_{abs} to be the smallest value of ε that theoretically guarantees the margin condition. ε_{abs} as a function of the timing constraint T_c is given by

$$\varepsilon_{abs}(T_c) = max_{\text{over all } X \text{ such that } d_C^{TL}(X) \geq T_c} \ (T_c - d_C^{PDM}(X))$$

However, the value of ε_{abs} is not known in practice because it requires knowledge of d_C^{TL} throughout the entire sample space. Therefore, the algorithm must try to heuristically provide a value of ε close enough to ε_{abs} in order to minimize the bias in the estimator.

On the other hand, as seen in (4.32), the closer $Loss^{PDM,\varepsilon}$ is to $Loss^{TL}$, the more speedup the IS estimator achieves over standard TL-MC estimator. Making $Loss^{PDM,\varepsilon}$ close to $Loss^{TL}$ requires that ε be kept close to a particular value ε^* that satisfies $Loss^{PDM,\varepsilon^*} = Loss^{TL}$. Thus, to make IS efficient while preserving correctness, we must choose ε as close to ε^* as possible. Similarly to the case in the paragraph above, the value of ε^* is not known in practice, since it requires the entire sample space to be covered by TL simulations.

To summarize, the algorithm must pick a value of ε as close to ε^* as possible without going below ε_{abs}. However, since neither of these quantities are known a priori, we use the heuristic algorithm in this section to compute an ε that is a good compromise. In the experiments in Chapter 5, we demonstrate that our heuristic strikes a good compromise between accuracy and efficiency in all benchmarks.

Heuristic Criterion for ε

CompLossMC-IS explores the samples Y_i in increasing order of i, i.e., in decreasing order of their d_C^{PDM} values. For a given value of *SM* (short for ”Safety Margin”), CompLossMC-IS's goal is to select ε satisfying the following property:

- There is a sequence of *SM* sample points $\{Y_{N+1},...,Y_{N+SM}\}$ that constitute the safety margin (called as *SafeMargin* in the algorithm). For each point Y in the margin,

$$d_C^{TL}(Y) < T_c$$

And ε satisfies

$$\varepsilon = T_c - 0.5(d_C^{PDM}(Y_{N+1}) + d_C^{PDM}(Y_N))$$

The safety margin (heuristically) provides confidence that the safety condition is satisfied for the remaining points for which a TL simulation has not been carried out. This is because all of these remaining samples have a value of d_C^{PDM} less than $T_c - \varepsilon$.

In Chapter 5, we show that, using a reasonably small *SM*, the heuristic criterion provides an estimator with negligible bias. CompLossMC-IS runs only *SM* additional TL simulations beyond those needed for $Loss_N^{IS}$ [2]. The computational cost of $Loss^{PDM,\varepsilon}$ determination is unavoidable with the IS estimator and is not due to the adaptive determination of ε.

Determining NS:

Roughly speaking, the user provides the algorithm with a number *NS*, and he expects to carry out approximately *NS*.*Loss* TL simulations. Since the intended use of our proposed approach is accurate, late-stage yield determination, a rough estimate for *Loss* should be available. If not, $Loss^{PDM}$ can be used as a rough guide. $Loss^{PDM}$ can be computed by (4.8) where $T_c{}^{\varepsilon}$ should be replaced by T_c. It is important to note that the choice of *NS* is guided by how small one would like the variance of the $Loss^{IS}$ estimator to be. The purpose of *NS* is not to sample the parameter space in order to determine a safe value of ε. ε is determined heuristically and this heuristic is empirically justified separately. *NS* is chosen so that roughly *NS*.*Loss* TL simulations are affordable, and the variance of the IS estimator for *NS*.*Loss* samples is as small as desired.

Alternatively, the user can start with a very small *NS* value and advance this value until he performs the maximum affordable number of TL simulations. Assuming that the upper bound for the loss is $Loss^{UB}$ and the maximum affordable number of TL simulations per path is N_{MAX}, then the two step process for determining *NS* is as follows:

1. Start with $NS = N_{MAX}/Loss^{UB}$. Perform IS estimation using CompLossMC-IS algorithm.

[2]The overhead due to the additional *SM* simulations is taken into account in the reported `Speedup` results in Chapter 5.

2. If number of TL simulations is smaller than N_{MAX}. Increment NS by adding a new sample point (X_i) and perform IS estimation with this new NS until the number of TL simulations becomes equal to N_{MAX}.

As the main source of computational cost is TL simulations, this two step process for NS determination does not introduce a serious additional cost. Because, in the repetitive IS estimations, the previously computed $I^{TL}(T_c,X_i)$ and $I^{PDM}(T_c{}^{\varepsilon},X_i)$ values are kept and also $Loss^{PDM,\varepsilon}$ is computed only for once in step 1. When this second suggestion for NS determination is compared with the previous one, it is seen that there is an additional cost in the second suggestion as the algorithm CompLossMC-IS is repeated more than once in step 2. However, in this second suggestion, the user performs exactly the maximum affordable number of TL simulations which is not guaranteed in the previous suggestion. For increasing the efficiency of the second suggestion, the number of algorithm repetitions can be decreased by adding more than one sample in step 2. But this time the guarantee to perform exactly the desired number of TL simulations is relaxed. If in step 2, k samples were added instead of 1 sample, then at worst case, at the end of the process the user would perform $N_{MAX}+k-1$ TL simulations, whereas the number of algorithm repetitions would decrease k times.

4.4 The Convergence Analysis

In this section, we present a precise analysis that quantifies the variance reduction and the speed-up obtained when we use the IS estimator instead of the standard TL-MC estimator. We start with the analysis of mean and variance of the TL-MC and IS estimators and then derive error expressions for both estimators.

Means of TL-MC and IS estimators

Theorem 4.4.1. *The mean of the standard TL-MC estimator in (4.2) is the actual loss given by the integral in (4.1). Therefore, the TL-MC estimator is an unbiased estimator.*

Proof. As X_i's are independent identically distributed (i.i.d.) random variables from $f(X)$, mean of the TL-MC estimator can be written as

$$\begin{aligned} E\{Loss_N^{TL}\} &= \frac{1}{N}\sum_{i=1}^{N} E\{I^{TL}(T_c, X_i)\} \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_{\Omega} I^{TL}(T_c, X) f(X) dX \end{aligned} \tag{4.10}$$

where Ω is the region where $f(X)$ is defined and nonzero. From (4.1), the integral above is equal to actual loss, i.e. $Loss^{TL}$.

$$\begin{aligned} E\{Loss_N^{TL}\} &= \frac{1}{N}.N.Loss^{TL} \\ &= Loss^{TL} \end{aligned} \tag{4.11}$$

□

Theorem 4.4.2. *The mean of the IS estimator in (4.5) is equal to the actual loss given by the integral in (4.1), provided that the safety and regularity requirements given in Section 4.2 are satisfied.*

Proof. As $\tilde{f}(X)$ is a regular probability density function (regularity requirement) and X_i's are i.i.d. random variables from $\tilde{f}(X)$, the mean of IS estimator can be written as

$$\begin{aligned} E\{Loss_N^{IS}\} &= \frac{Loss^{PDM,\varepsilon}}{N}\sum_{i=1}^{N} E\{I^{TL}(T_c, X_i)\} \\ &= \frac{Loss^{PDM,\varepsilon}}{N}\sum_{i=1}^{N}\int_{\Theta} I^{TL}(T_c, X)\tilde{f}(X) dX \end{aligned} \tag{4.12}$$

where Θ is the region where $\tilde{f}(X)$ is defined and nonzero. Substituting $\tilde{f}(X)$ in (4.4) and using the fact that in Θ region $I^{PDM}(T_c{}^{\varepsilon}, X)$ is always 1

$$\begin{aligned} E\{Loss_N^{IS}\} &= \frac{Loss^{PDM,\varepsilon}}{N}\sum_{i=1}^{N}\int_{\Theta} I^{TL}(T_c, X)\frac{I^{PDM}(T_c{}^{\varepsilon}, X) f(X)}{Loss^{PDM,\varepsilon}} dX \\ &= \frac{Loss^{PDM,\varepsilon}}{N}\sum_{i=1}^{N}\frac{1}{Loss^{PDM,\varepsilon}}\int_{\Theta} I^{TL}(T_c, X) f(X) dX \end{aligned} \tag{4.13}$$

According to the safety requirement, Θ region must cover the region where $I^{TL}(T_c, X) f(X)$ is nonzero. Therefore, if safety requirement is satisfied, the integration above is equal to the actual loss, i.e. $Loss^{TL}$ given by (4.1).

$$\begin{aligned} E\{Loss_N^{IS}\} &= \frac{Loss^{PDM,\varepsilon}}{N}\sum_{i=1}^{N}\frac{Loss^{TL}}{Loss^{PDM,\varepsilon}} \\ &= Loss^{TL} \end{aligned} \tag{4.14}$$

□

The analysis of means of both TL-MC and IS estimators shows that both estimators converge to the actual loss, $Loss^{TL}$. This is a very useful property that approximate loss (or yield) estimation methods based on many assumptions like SSTA methods do not have. As they both converge to the actual loss, the most important point is the convergence speed of the TL-MC and IS estimators, which is directly related with the variances of both estimators for a given number of drawn sample points for TL simulations.

Variances of TL-MC and IS estimators

Theorem 4.4.3. *The variance of the TL-MC estimator in (4.2) is given by*

$$VAR_{TL}(N) = VAR\{Loss_N^{TL}\} = \frac{Loss^{TL}.Yield^{TL}}{N} \tag{4.15}$$

where $Yield^{TL} = 1 - Loss^{TL}$ and N is the number of drawn samples or equivalently the number of TL simulations performed per path.

Proof. As X_i's are i.i.d. random variables from $f(X)$ and $E\{I^{TL}(T_c,X)\} = Loss^{TL}$, the variance of TL-MC estimator can be written as

$$\begin{aligned} VAR\{Loss_N^{TL}\} &= \frac{1}{N^2}\sum_{i=1}^{N} VAR\{I^{TL}(T_c,X_i)\} \\ &= \frac{1}{N^2}\sum_{i=1}^{N} E\{(I^{TL}(T_c,X_i) - Loss^{TL})^2\} \\ &= \frac{1}{N^2}\sum_{i=1}^{N} \left(E\{(I^{TL}(T_c,X_i))^2\} - (Loss^{TL})^2\right) \end{aligned} \tag{4.16}$$

Substituting the fact that $I^{TL}(T_c,X)$ is either 1 or 0 and so $(I^{TL}(T_c,X))^2 = I^{TL}(T_c,X)$

$$\begin{aligned} VAR\{Loss_N^{TL}\} &= \frac{1}{N^2}\sum_{i=1}^{N} \left(Loss^{TL} - (Loss^{TL})^2\right) \\ &= \frac{1}{N^2} N \left(Loss^{TL}(1 - Loss^{TL})\right) \\ &= \frac{Loss^{TL}.Yield^{TL}}{N} \end{aligned} \tag{4.17}$$

□

Theorem 4.4.4. *The variance of the IS estimator in (4.5) is equal to*

$$VAR_{IS}(N) = VAR\{Loss_N^{IS}\} = \frac{Loss^{TL}.(Loss^{PDM,\varepsilon} - Loss^{TL})}{N} \tag{4.18}$$

Proof. As X_i's are i.i.d. random variables from $\tilde{f}(X)$ the variance of IS estimator can be written as

$$\begin{aligned} VAR\{Loss_N^{IS}\} &= \frac{(Loss^{PDM,\varepsilon})^2}{N^2}\sum_{i=1}^{N} VAR\{I^{TL}(T_c,X_i)\} \\ &= \frac{(Loss^{PDM,\varepsilon})^2}{N^2}\sum_{i=1}^{N} E\left\{\left(I^{TL}(T_c,X_i)-E\{I^{TL}(T_c,X_i)\}\right)^2\right\} \end{aligned} \tag{4.19}$$

As X_i's are drawn from $\tilde{f}$,

$$E\{I^{TL}(T_c,X_i)\} = \int_{\Theta} I^{TL}(T_c,X)\tilde{f}(X)dX \tag{4.20}$$

substituting $\tilde{f}(X)$ in (4.4) and using the fact that in Θ region $I^{PDM}(T_c^{\varepsilon},X)$ is always 1,

$$\begin{aligned} E\{I^{TL}(T_c,X_i)\} &= \frac{1}{Loss^{PDM,\varepsilon}}\int_{\Theta} I^{TL}(T_c,X)f(X)dX \\ &= \frac{Loss^{TL}}{Loss^{PDM,\varepsilon}} \end{aligned} \tag{4.21}$$

Substituting this to (4.19), we get

$$VAR\{Loss_N^{IS}\} = \frac{(Loss^{PDM,\varepsilon})^2}{N^2}\sum_{i=1}^{N}\left(E\{(I^{TL}(T_c,X_i))^2\} - \left(\frac{Loss^{TL}}{Loss^{PDM,\varepsilon}}\right)^2\right) \tag{4.22}$$

As indicator variable is either 1 or 0, $E\{(I^{TL}(T_c,X))^2\} = E\{I^{TL}(T_c,X)\} = \frac{Loss^{TL}}{Loss^{PDM,\varepsilon}}$,

$$\begin{aligned} VAR\{Loss_N^{IS}\} &= \frac{(Loss^{PDM,\varepsilon})^2}{N^2}\sum_{i=1}^{N}\left(\frac{Loss^{TL}}{Loss^{PDM,\varepsilon}} - \frac{(Loss^{TL})^2}{(Loss^{PDM,\varepsilon})^2}\right) \\ &= \frac{(Loss^{PDM,\varepsilon})^2}{N^2}.N.\frac{\left(Loss^{TL}Loss^{PDM,\varepsilon}-(Loss^{TL})^2\right)}{(Loss^{PDM,\varepsilon})^2} \\ &= \frac{Loss^{TL}.(Loss^{PDM,\varepsilon}-Loss^{TL})}{N} \end{aligned} \tag{4.23}$$

□

The theorems 4.4.3 and 4.4.4 prove that for the same number of samples (N), the degree of the variance reduction we get by using the proposed IS estimator in (4.5) can be written as

$$\frac{VAR_{TL}(N)}{VAR_{IS}(N)} = \frac{Yield^{TL}}{Loss^{PDM,\varepsilon}-Loss^{TL}} \tag{4.24}$$

95% Confidence Errors of TL-MC and IS estimators

The error of an estimator is the deviance of the estimator's result from the actual loss for a general estimator. The central limit theorem suggests that for sufficiently large N, both TL-MC estimator in (4.2) and IS estimator in (4.5) have loss estimates with normal distribution. Using this fact, the errors of the standard TL-MC and IS estimators are derived and the results are compared below.

Theorem 4.4.5. *The error of the standard TL-MC estimator in (4.2) obtained with N samples, where N is sufficiently large, is*

$$Error_{TL}(N) = 2\sqrt{\frac{Loss^{TL}.Yield^{TL}}{N}} = 2\sqrt{VAR_{TL}(N)} \tag{4.25}$$

with more than 95% confidence.

Proof. According to the central limit theorem, when N is sufficiently large $\frac{Loss_N^{TL} - Loss^{TL}}{\sqrt{VAR_{TL}(N)}}$ has a normal distribution with $N(0,1)$. Hence

$$P\left(Loss^{TL} - 1.96\sqrt{VAR_{TL}(N)} \leq Loss_N^{TL} \leq Loss^{TL} + 1.96\sqrt{VAR_{TL}(N)}\right) = 0.95 \tag{4.26}$$

which means that $Loss_N^{TL}$ is in the interval $\left[Loss^{TL} - 1.96\sqrt{VAR_{TL}(N)},\ Loss^{TL} + 1.96\sqrt{VAR_{TL}(N)}\right]$ with probability 0.95. Therefore, the error of the TL-MC estimator with more than 95% confidence can be written as

$$\begin{aligned} Error_{TL}(N) &= 2\sqrt{VAR_{TL}(N)} \\ &= 2\sqrt{\frac{Loss^{TL}.Yield^{TL}}{N}} \end{aligned} \tag{4.27}$$

□

Theorem 4.4.6. *The error of the* IS *estimator in (4.3) or (4.5) obtained with N samples, where N is sufficiently large, is given by*

$$Error_{IS}(N) = 2\sqrt{\frac{Loss^{TL}(Loss^{PDM,\varepsilon} - Loss^{TL})}{N}} = 2\sqrt{VAR_{IS}(N)} \tag{4.28}$$

with more than 95% confidence.

Proof. According to Central Limit Theorem, when N is sufficiently large $\frac{Loss_N^{IS} - Loss^{TL}}{\sqrt{VAR_{IS}(N)}}$ has a normal distribution with $N(0,1)$. Hence

$$P\left(Loss^{TL} - 1.96\sqrt{VAR_{IS}(N)} \leq Loss_N^{IS} \leq Loss^{TL} + 1.96\sqrt{VAR_{IS}(N)}\right) = 0.95 \tag{4.29}$$

which means that $Loss_N^{IS}$ is in the interval $\left[Loss^{TL} - 1.96\sqrt{VAR_{IS}(N)},\ Loss^{TL} + 1.96\sqrt{VAR_{IS}(N)}\right]$ with probability 0.95. Therefore, the error of the IS estimator with more than 95% confidence can be written as

$$\begin{aligned} Error_{IS}(N) &= 2\sqrt{VAR_{IS}(N)} \\ &= 2\sqrt{\frac{Loss^{TL}(Loss^{PDM,\varepsilon} - Loss^{TL})}{N}} \end{aligned} \tag{4.30}$$

□

Discussion

In the derivation of the IS estimator error above, $Loss^{PDM,\varepsilon}$ was assumed to be a known deterministic quantity. However, $Loss^{PDM,\varepsilon}$ is estimated using the estimator in (4.8), and in fact, it is a random quantity with a nonzero variance that decreases proportionally to the number of samples K used in (4.8). In order for the error derivation for the IS estimator in (4.5) to be valid, the estimation of $Loss^{PDM,\varepsilon}$ must be performed by using a large enough number of samples in (4.8) so that it has negligible variance. This would validate its treatment as a deterministic quantity in the derivation of the error equation for the IS estimator. The use of a large number of samples in (4.8) is easily affordable, because no TL simulations are performed, only simple evaluations of the cheap delay models are needed. The results we present later show that the theoretical error expressions derived here are in excellent agreement with experimental data.

The error equations (4.25) and (4.28) that have been derived with Theorem 4.4.5 and Theorem 4.4.6 for the standard TL-MC and IS Monte Carlo estimators can be used to compare them. If the same number of samples N is used for both methods (meaning an equal number of TL simulations), then the ratio of the errors of the estimators is given by

$$ErrorRatio(N) = \frac{Error_{TL}(N)}{Error_{IS}(N)} = \sqrt{\frac{Yield^{TL}}{(Loss^{PDM,\varepsilon} - Loss^{TL})}} \tag{4.31}$$

Alternatively, suppose a bound on the allowable estimation error is given. The ratio of the number of samples (TL circuit simulations) required by the two approaches to achieve this same error bound is given by

$$\texttt{Speedup} = \frac{N_{TL}}{N_{IS}} = \frac{Yield^{TL}}{Loss^{PDM,\varepsilon} - Loss^{TL}} \tag{4.32}$$

which is obtained by solving $Error_{TL}(N_{TL}) = Error_{IS}(N_{IS})$ for $\frac{N_{TL}}{N_{IS}}$, which we call $\texttt{Speedup}$, since the number of samples used in the estimators determines the number of TL simulations that need to be performed on the statistically critical paths of the circuit. Based on (4.24) and (4.32) above, we note here that $\texttt{Speedup}$ can alternatively be computed with

$$\texttt{Speedup} = \frac{VAR_{TL}(N)}{VAR_{IS}(N)} \tag{4.33}$$

as the ratio of the variances for the standard TL-MC and IS estimators with the same number of samples $N = N_{TL} = N_{IS}$, i.e., the same number of TL simulations.

Finally, we address a question that may arise in the mind of an attentive reader. If $\texttt{Speedup}$ in (4.32) is large, one might conclude that TL simulations are not needed and $Loss^{PDM,\varepsilon}$ can simply be used as an accurate loss estimate. This conclusion would be based on the observation that $Loss^{PDM,\varepsilon}$ has to be very close to $Loss^{TL}$ if one can attain a large `Speedup` in (4.32). However, this conclusion is not correct. $Loss^{PDM,\varepsilon}$ is computed using (4.8), where ${T_c}^{\varepsilon} = T_c - \varepsilon$ with ε as the margin parameter. The margin parameter is determined by an adaptive algorithm which performs TL simulations in its search for the correct ε value. If $Loss^{PDM}$ is not computed based on the ε value found by the CompLossMC-IS algorithm described in Section 4.3, i.e. ε is assumed to be zero, then the resultant $Loss^{PDM}$ will not be close to $Loss^{TL}$. Therefore, attaining a large `Speedup` does not mean that the PDM model is by itself accurate enough for loss estimation. The PDM model needs to be in some sense "calibrated" or corrected with the TL simulations run at the critical samples in the parameter space selected using importance sampling.

Chapter 5

EXPERIMENTAL RESULTS: ISCAS'85 BENCHMARK

5.1 Experimental Setup

We first describe our experimental setup in order to help interpret our results better:

We present results on the ISCAS'85 benchmark suite [16] in this paper. We use the 0.13μ standard cell library provided by Graham Petley [82] for the transistor-level implementations of the gates needed in order construct the benchmark circuits. We have added some missing 5-,8- and 9-input gates to this library, as they are needed for some of the circuits in the ISCAS'85 benchmark suite. The layout information for the circuits, i.e., relative locations of the gates on the layout for a particular benchmark circuit (needed for the intra-die variation model that captures spatial correlations), are extracted from the def file provided on the VLSI CAD group web pages at Texas A&M University [83].

Two random transistor parameters, namely the transistor gate length L and the threshold voltage V_t, are considered. Both inter and intra-die variations for these parameters are taken into account and a statistical model as described in Section 3.1 is constructed. In this model, half of the variation is allocated to inter-die variations and the other half to intra-die variations [2], with a total $3\sigma/\mu$ ratio of 15% for both of the random parameters L and V_t [84]. In the quad-tree model [2] that captures spatial correlations, we use four grid levels (layers) as shown in Figure 3.1. We allocate half of the variation to the top level that covers the whole area of the circuit with one grid rectangle in order to capture perfectly correlated inter-die variations. The other three levels in the model capture the spatially correlated intra-die variations and are allocated one sixth of the total variation each. These allocations are done by appropriately choosing the variances of the grid random variables in the quad-tree model. As described in Section 3.1, we use 85 random variables in the quad-tree model per parameter. With two random transistor parameters, L and V_t, the random variable vector X described at the end of Section 3.1 has a dimension of 170 in all of our experiments.

For each of the circuits in the ISCAS'85 benchmark suite, we determine a set of statistically critical paths using the method described in Section 3.3 and Section 3.4. We experiment with two timing constraints for each circuit, $T_{c,low}$ and $T_{c,high}$ that result in roughly 10% and 5% loss, respectively. When we use our improved IS-based estimator for timing yield, the required margin parameter ε that was introduced in Section 4.2 is computed automatically using the algorithm described in Section 4.3. It is important to note that, as it computes ε, this algorithm carries out all TL circuit simulations required for computing the IS estimator.

With the results that we present in this section, we compare the accuracy and the efficiency of our improved Importance Sampling (IS) estimator in (4.5) against the standard Transistor Level Monte Carlo (TL-MC) estimator in (4.2). In doing so, we empirically compute the error (variance) achieved for both of the loss estimators. In order to measure the error of an estimator, we perform M (to be quantified precisely) independent repetitions of the same experiment (evaluation of the estimator). In each independent run, we compute the loss estimates with the IS estimator by using R (to be quantified precisely) independently drawn samples from the PDF $f(X)$ in the parameter space. In other words, R is equal to the NS shown in CompLossMC-IS of Section 4.3. These M independent runs constitute the samples of the loss estimator, and the variance of the loss estimator is computed over these M samples. For the IS estimator, most of the R samples are discarded as explained in Section 4.2 and Section 4.3 based on the evaluation of the PDM equations, and a reduced number (N_{IS} on the average) of TL simulations are performed. All of these N_{IS} simulations are performed as part of the iterative algorithm for computing ε. In other words, N_{IS} includes all of the TL simulations required to compute ε and $Loss_N^{IS}$. In evaluating the standard TL-MC estimator, we choose $N_{TL} = N_{IS}$ samples randomly among the R samples in every set. For the standard TL-MC estimator, the results of TL circuit simulations performed at every one of the N_{TL} sample points are used.

The $Loss^{PDM,\varepsilon}$ value that is needed for computing the IS estimator in (4.5) is computed using the PDM based estimator in (4.8) using all of the $K = M \times R$ sample points generated during all of the M runs.

The `Speedup` that we report for the IS estimator over the standard TL-MC estimator represents the ratio of the number of TL circuit simulations required by the TL-MC and IS estimators to achieve the same error, as given by (4.32). Alternatively, `Speedup` is equal to the amount of variance reduction, i.e. the ratio of the variances for the loss estimates obtained by the two estimators with the same number of samples (TL simulations), as given by (4.33).

5.2 IS Estimator Results

Experiment A: Three statistically critical paths

In this experiment, for each benchmark circuit we choose three most statistically critical paths which are identified using the scheme described in Section 3.3 and Section 3.4. We then perform the following:

We construct M sample sets, each with R samples drawn from the PDF $f(X)$ as explained in Section 3.1, with a total of $K = M \times R$ samples in the random parameter space. In this experiment, we have used $M = 250$ and $R = 200$ for both $T_{c,low}$ and $T_{c,high}$, for a total of $K = 50,000$ samples in each case. For each set, we evaluate the IS estimator. The IS estimator eliminates most of the samples in the set without performing TL simulations, as explained in Chapter 4. Loss estimates obtained with the IS estimator and the number of actual TL simulations run per path and for each set, i.e. N_{IS}^{i} for set i, are recorded. All of the sets have the same number of random samples in the parameter space, R. N_{IS} is equal to the average of the N_{IS}^{i}'s ($N_{IS} = mean(N_{IS}^{i})$) and it corresponds to the number of non-discarded samples at which TL simulations are run. For each experiment, up to SM extra TL simulations that are used in order to heuristically determine a safe value of ε are included in N_{IS}^{i}'s. This allows a fair comparison with standard TL-MC. The variance of the loss estimates over the M sets is computed as VAR_{IS}. The standard TL-MC estimator is evaluated for every set, using N_{IS} number of samples chosen randomly. Thus, loss with the TL-MC estimator is evaluated using the same number of TL simulations as the IS estimator, i.e., $N_{TL} = N_{IS}$. The variance of the loss values computed with the TL-MC estimator is computed as VAR_{TL}.

With the construction above, we have $N_{TL} = N_{IS} = N + SM$, where N is the average number of samples used for TL simulations excluding SM additional TL simulations used for margin detection. Hence, we substitute VAR_{IS} and VAR_{TL} in (4.33) in order to quantify the Speedup of the IS estimator over the standard TL-MC estimator. Our speed-up computation is reminded below by (5.1).

$$\texttt{Speedup} = \frac{VAR_{TL}}{VAR_{IS}} \qquad \text{with } N_{IS} = N_{TL} \tag{5.1}$$

The Speedup results, computed as described above for $T_{c,low}$ and $T_{c,high}$, for the ISCAS'85 benchmark suite are presented in Tables 5.1 and 5.2. In these tables, the mean values for $Loss^{IS}$ and $Loss^{TL}$ using the same number of TL simulations ($N_{TL} = N_{IS} = N + SM$) are also shown. Furthermore, we also report the loss values (labeled as *Loss*) computed using TL simulations at all 50,000 samples,

Table 5.1: Speedup for $T_{c,low}$, $M = 250$ sets, $R = 200$ samples each, $SM = 4$

Bench.	$N+SM$	$Loss$	mean $Loss^{TL}$	mean $Loss^{IS}$	$Error_{TL}$ (%)	$Error_{IS}$ (%)	Speedup
c432	29	0.1129	0.1153	0.1118	105.5	11.4	86
c499	31	0.1239	0.1234	0.1237	95.0	6.6	205
c880	29	0.1112	0.1135	0.1107	105.0	11.0	91
c1355	32	0.1297	0.1286	0.1296	93.3	6.9	181
c1908	30	0.1201	0.1133	0.1193	102.3	9.2	124
c2670	30	0.1195	0.1213	0.1190	96.6	7.0	188
c3540	29	0.1135	0.1105	0.1130	104.8	8.5	152
c5315	28	0.1137	0.1201	0.1135	106.2	7.7	191
c7552	26	0.1001	0.1117	0.0995	117.3	8.7	181

Table 5.2: Speedup for $T_{c,high}$, $M = 250$ sets, $R = 200$ samples each, $SM = 2$

Bench.	$N+SM$	$Loss$	mean $Loss^{TL}$	mean $Loss^{IS}$	$Error_{TL}$ (%)	$Error_{IS}$ (%)	Speedup
c432	16	0.0604	0.0580	0.0598	197.1	17.9	121
c499	12	0.0416	0.0393	0.0412	270.7	19.3	196
c880	14	0.0523	0.0560	0.0516	225.7	17.4	169
c1355	16	0.0637	0.0615	0.0636	189.7	12.5	229
c1908	17	0.0680	0.0671	0.0678	178.6	12.5	205
c2670	19	0.0784	0.0764	0.0779	156.1	8.8	314
c3540	19	0.0748	0.0718	0.0744	158.8	10.8	216
c5315	19	0.0776	0.0829	0.0771	157.9	10.2	242
c7552	15	0.0578	0.0621	0.0574	202.0	13.0	241

which can be regarded as the real loss value. The mean value of the $Loss^{IS}$ estimator was on the average within 0.57% of the loss value computed using TL simulations at all 50,000 samples. The mean value of the $Loss^{TL}$ estimator was on the average within 4.15% of the loss value computed using TL simulations at all 50,000 samples.

As discussed in Section 4.4, in Tables 5.1 and 5.2, $Error_{TL}$ and $Error_{IS}$ are computed by

$$Error_{TL} = 100 \times \frac{2\sqrt{VAR_{TL}}}{Loss} \tag{5.2}$$

$$Error_{IS} = 100 \times \frac{2\sqrt{VAR_{IS}}}{Loss} \tag{5.3}$$

and `Speedup` represents the ratio of the number of TL simulations required by the standard TL-MC estimator to the number of TL simulations needed by our proposed IS estimator in order to estimate the loss of the circuit with the same error (accuracy). Alternatively, if the same number of TL simulations are used for both of the estimators, the estimation variance for loss will be `Speedup` times less for our IS estimator. As seen in Tables 5.1 and 5.2, our accelerated yield estimator achieves on the average *two orders of magnitude* (185 on the average) `Speedup` over standard Transistor Level Monte Carlo.

Contrasting Absolute Errors of Estimators

In the method we propose, while computing the loss estimator $Loss^{IS}$, approximate delay computed using the PDM method (with an ε margin) is used by IS in order to achieve a low variance estimator. The question naturally arises as to how good an estimator $Loss^{PDM}$ is, and whether it itself could be used for yield estimation. We contrast the absolute errors in the following four estimators:

- $Loss^{TL}$: The Monte Carlo estimator.
- $Loss^{IS}$: The Monte Carlo estimator with importance sampling.
- $Loss^{PDM}$: The PDM estimator with no adjustment
- $Loss^{PDM,\varepsilon}$: The PDM estimator with the ε margin (i.e., with $T_c - \varepsilon$ as the timing constraint)

We computed the loss estimated by each approach. The results for $T_{c,low}$ and $T_{c,high}$ are presented in Tables 5.3 and 5.4 respectively. We have taken the $Loss^{TL}$ value computed from 50,000 samples

Table 5.3: Loss for $T_{c,low}$, four different estimators

Benchmark	*Loss*	$Loss^{IS}$	$Loss^{PDM,\varepsilon}$	$Loss^{PDM}$
c432	0.1129	0.1129	0.1510	0.0937
c499	0.1239	0.1239	0.1304	0.0804
c880	0.1112	0.1112	0.1280	0.0901
c1355	0.1297	0.1297	0.1402	0.0982
c1908	0.1201	0.1201	0.1327	0.1154
c2670	0.1195	0.1195	0.1272	0.1060
c3540	0.1135	0.1135	0.1232	0.0979
c5315	0.1137	0.1137	0.1232	0.1055
c7552	0.1001	0.1001	0.1112	0.0965

as the reference in each case. These reference loss values are labeled as *Loss* in Tables 5.3 and 5.4 and correspond to the columns with the same label in Tables 5.1 and 5.2. The $Loss^{IS}$, $Loss^{PDM}$ and $Loss^{PDM,\varepsilon}$ values were also obtained from a single run on the same 50,000 samples[1] . To compute $Loss^{PDM,\varepsilon}$ we used the ε value found during the $Loss^{IS}$ computation. As seen in the tables, $Loss^{IS}$ is bias-free in all cases. This is to be expected, since $Loss^{IS}$ is an unbiased estimator in theory – a fact experimentally demonstrated further in Section 5.4.

The value of the uncorrected PDM estimator $Loss^{PDM}$ is too far from $Loss^{TL}$ to be acceptable for most benchmarks, resulting in errors of 44.3% at most and 16.1% on the average. It is important to note that, if one had carried out block-level Monte Carlo (BL-MC)[2] statistical timing analysis using our polynomial gate delay models, this is the accuracy one would have obtained. While delay values as computed by the PDM model only correlate well with the actual values, in terms of the absolute value of delay, they are far off. Thus, while the PDM model may serve as a rough guide for timing and yield optimization, it is not accurate enough for the numerical prediction of yield. This is important as it justifies the use of our method as a final pass of yield estimation.

[1]For IS estimator, the algorithm CompLossMC-IS is given $NS = 50,000$ where all NS samples are not used for TL simulation as explained in Section 4.3

[2]BL-MC was reviewed in Section 2.4.2 and explained in detail in Section 3.3. The difference from TL-MC is that the value of the indicator variable is decided by block level DSTA instead of TL simulations.

Table 5.4: Loss for $T_{c,high}$, four different estimators

Benchmark	*Loss*	$Loss^{IS}$	$Loss^{PDM,\varepsilon}$	$Loss^{PDM}$
c432	0.0604	0.0604	0.0802	0.0501
c499	0.0416	0.0416	0.0444	0.0231
c880	0.0523	0.0523	0.0634	0.0409
c1355	0.0637	0.0637	0.0701	0.0454
c1908	0.0680	0.0680	0.0756	0.0659
c2670	0.0784	0.0784	0.0843	0.0690
c3540	0.0748	0.0748	0.0822	0.0632
c5315	0.0776	0.0776	0.0834	0.0710
c7552	0.0578	0.0578	0.0629	0.0558

The ε-corrected $Loss^{PDM,\varepsilon}$ is a better estimator for loss. The fact that $Loss^{PDM,\varepsilon}$ is in many cases close to $Loss^{TL}$ might appear to suggest that yield prediction using PDM with the ε correction is a sufficiently accurate and cheap method. However, in order to compute the ε correction factor, all the TL simulations required for computing $Loss^{IS}$ have to be carried out. Furthermore, $Loss^{PDM,\varepsilon}$ is actually not close enough to the actual loss value to be an accurate estimator in its own right. Thus, it makes more sense to use more accurate and provably unbiased estimator $Loss^{IS}$.

$Loss^{PDM,\varepsilon}$ *Estimation*

The accuracy of $Loss^{PDM,\varepsilon}$ used in IS estimation is important in terms of the regularity requirement explained in Section 4.2.

As explained in Section 4.2, $Loss^{PDM,\varepsilon}$ is estimated using (4.8) and a huge number of samples, i.e. a big K. In the experiment above 50,000 samples ($K = 50,000$) are used to estimate $Loss^{PDM,\varepsilon}$ and then this $Loss^{PDM,\varepsilon}_{50,000}$ is used as $Loss^{PDM,\varepsilon}$ for computing $Loss^{IS}_{N}$ as shown by algorithm CompLossMC-IS in Section 4.3. Figure 5.1 shows the estimate $Loss^{PDM,\varepsilon}_{K}$ versus K plot for a sample benchmark circuit (c1908), where K is iterated from 1 to 500,000. It can be verified from the figure that $K = 50,000$ is a reasonable estimate, which can be used instead of actual $Loss^{PDM,\varepsilon}$.

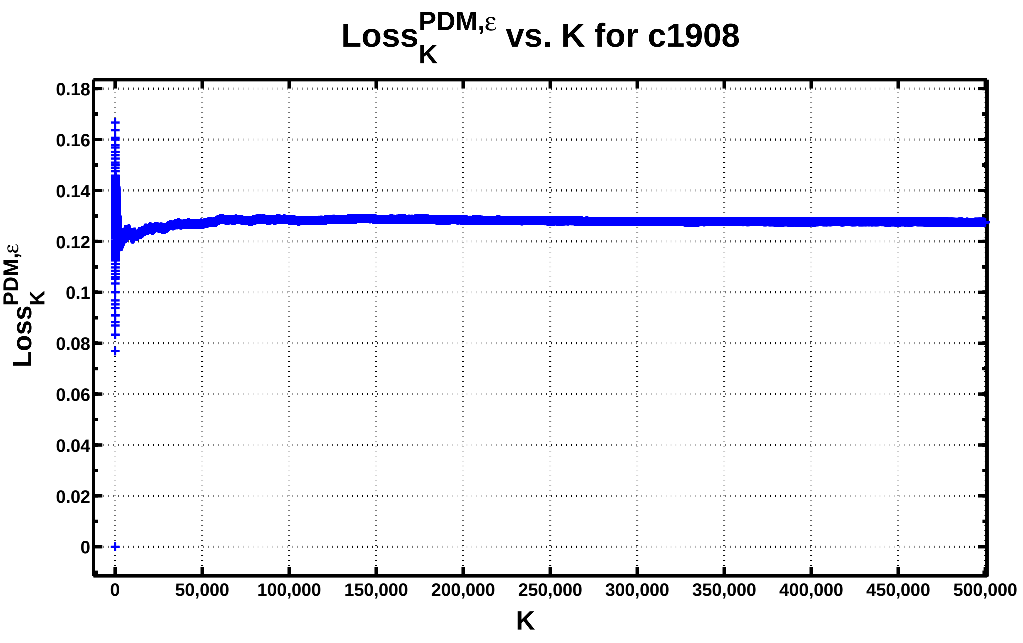


Figure 5.1: Convergence of $Loss_K^{PDM,\varepsilon}$ in (4.8) w.r.t. number of samples (K) (belongs to c1908)

VAR_{IS} Convergence

The accuracy of the variance of IS estimator used in `Speedup` computation is important as it determines the performance of our IS estimator.

In the experiments above, $M = 250$ sets are used to compute the variance of the IS estimator, i.e. VAR_{IS}. Then VAR_{IS} value is used to compute the `Speedup` acquired by IS method over standard TL-MC method. Figure 5.2 shows the reason why $M = 250$ is picked. Each plot in this figure corresponds to a different benchmark circuit and shows the computed variance of IS estimator over M sets while M is changed from 1 to 250. In other words, the x-axis shows the number of sets used for variance computation of IS estimator and y-axis shows the variance of IS (VAR_{IS}). Although for some benchmark circuits like c499 VAR_{IS} converges when 50 or more sets are used, for the general case, the variance of the IS estimator does not converge until more than 200 sets are used. Therefore, to have the accurate `Speedup` results, we have used 250 sets and corresponding 250 independent IS estimations and then the variance of the IS estimator is computed over these independent 250 IS estimations.

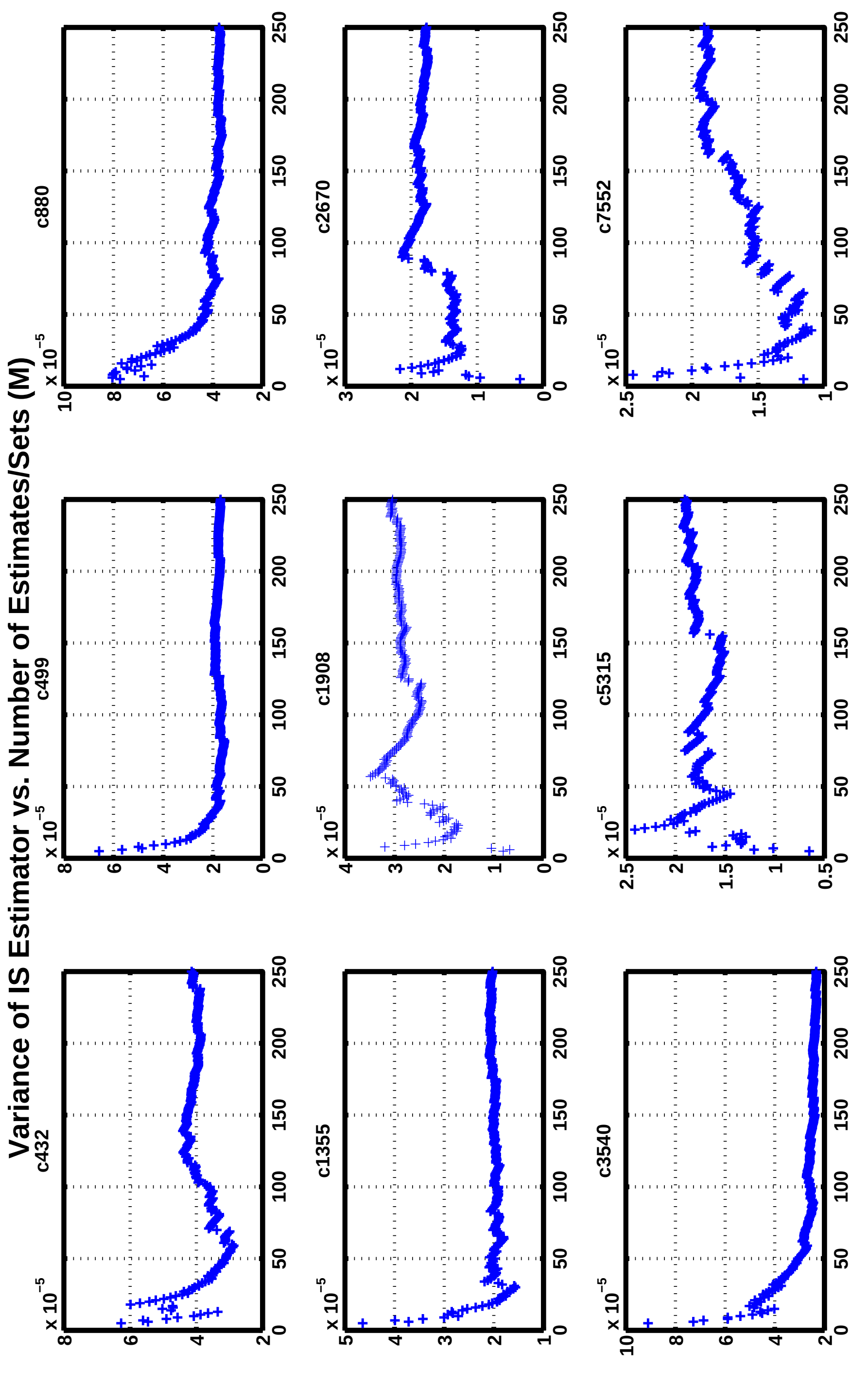


Figure 5.2: Convergence of VAR_{IS} w.r.t. number of sets used for variance computation (M)

Experiment B: Ten statistically critical paths

In this experiment, we randomly choose one of the ISCAS'85 benchmark circuits, and repeat the same experiment described above ($T_{c,low}$), but with ten most statistically critical paths [3]. In this case, we obtain a `Speedup` of 147, which is almost the same as the one obtained for the same benchmark circuit with only three critical paths. The following values were obtained in this experiment: $N = 31$, $Loss = 0.1244$, mean $Loss^{TL} = 0.1252$, mean $Loss^{IS} = 0.1238$, $Error_{TL} = 99.32$ %, $Error_{IS} = 8.2$ %. These results confirm that the `Speedup` achieved by our IS estimator is not dependent on the number of statistically critical paths considered for a circuit. The efficiency of the IS estimator does not degrade if a large number of critical paths are included in yield estimation, because the maximum of the path delays in (2.12) for the overall circuit delay is computed exactly, without employing approximations. This is a key advantage of our technique. If an approximate maximum operation is employed in computing the circuit delay from path delays, the accuracy will degrade if a large number of paths are considered.

5.3 Performance of Margin Detection Algorithm

Recall that the ε margin involved in the computation of $Loss_N^{IS}$ is arrived at using a heuristic in the algorithm CompLossMC-IS. In order to validate the computed ε values, we performed the following checks:

Confirming that $Loss_N^{IS}$ is unbiased

Results presented in Figure 5.4 in Section 5.4 below, and Tables 5.3 and 5.4 indicate that when a large number of samples NS is used, $Loss_N^{IS}$ is an unbiased estimator. Of more practical importance is the fact that when $Loss_N^{IS}$ is computed with about 30 samples, the mean of $Loss_N^{IS}$ is on the average within 0.45% of the $Loss$ value computed from 50,000 samples[4].

Exploring different values of ε

Recall that $\varepsilon_{abs}(T_c) = max_{X \text{ such that } d_C^{TL}(X) \geq T_c}(T_c - d_C^{PDM}(X))$ is the smallest value of ε that guarantees the margin condition. As the closest practical approximation to the ideal ε_{abs}, we computed ε_{abs}

[3]We were not able to run this experiment for all of the circuits in the benchmark suite due to the excessive computational resources required by the standard TL-MC technique against which we compare our proposed estimator.

[4]This is derived from Table 5.1.

Table 5.5: % IS estimator bias error versus margin parameter ε ($T_{c,low}$, $M = 100$ sets, $R = 200$ samples each)

	0	0.1 ε_{abs}	0.2 ε_{abs}	0.3 ε_{abs}	0.4 ε_{abs}	0.5 ε_{abs}	0.6 ε_{abs}	0.7 ε_{abs}	0.8 ε_{abs}	0.9 ε_{abs}	ε_{abs}	ε (Heur.)
c432	17.01	12.89	8.68	5.12	2.47	1.13	0.50	0.19	0.00	0.03	0.02	**0.94**
c499	35.06	31.50	27.77	23.90	19.72	16.16	11.80	7.94	3.56	0.99	0.01	**0.16**
c880	19.02	16.51	13.36	10.46	7.56	4.76	2.89	1.34	0.32	0.00	0.13	**0.53**
c1355	24.26	21.70	18.77	15.67	12.29	8.96	5.95	2.84	0.84	0.09	0.05	**0.06**
c1908	4.41	3.29	2.33	1.78	1.17	0.70	0.38	0.25	0.17	0.14	0.11	**0.67**
c2670	11.30	9.55	7.85	6.30	4.45	3.07	1.91	1.06	0.33	0.05	0.12	**0.40**
c3540	13.76	11.78	9.87	7.55	5.72	4.07	2.44	1.07	0.41	0.09	0.08	**0.45**
c5315	7.21	5.75	4.29	2.85	1.83	1.02	0.48	0.13	0.04	0.06	0.04	**0.19**
c7552	4.27	3.22	2.20	1.49	0.88	0.56	0.42	0.36	0.25	0.19	0.10	**0.62**

using the equation above and letting X range over the 50,000 sample points we had for each benchmark. In order to investigate the sensitivity of the yield estimator to the safety margin, we carried out the following experiment. We forced our importance sampling algorithm to use a fixed value of ε. We varied this value of ε in the range 0, 0.1 ε_{abs}, 0.2 ε_{abs}, ..., 0.9 ε_{abs}, ε_{abs}. We then computed the percentage bias error in the mean of the IS estimator for each of these values of ε, including the one our heuristic computes. The results are shown in Table 5.5. The last column presents the results for the ε our heuristic finds. The first observation is that, for each benchmark, there is a value of ε below which the bias in the loss estimator is too high. This value of ε is different for each benchmark, but is in the 0.5-0.8 ε_{abs} range. Above this value of ε, the bias is not very sensitive to the particular value of ε. The second key observation is that the heuristically found ε value for each benchmark always results in an acceptable bias (error) in the loss computed. This bias is less than 1% of the absolute loss. Given that larger approximations are probably involved in parameter variation modeling, etc., a bias error of 1% of the loss is certainly negligible.

Validating the value of SM used

For each benchmark, we explored values of SM from 1 to NS. We found that even very small values of SM result in an acceptable error in the loss computation. Larger values of SM result in values of ε closer to ε_{abs} but this results in only very small differences in the actual bias. In order to make sure that the ε value we choose provides a good compromise between accuracy and high speedup, and to keep the computational cost still low, we pick SM to be 20% of the number of points that we expect the IS approach to perform TL simulations on. When $R = 200$, this amounts to $SM = 4$ for the examples in which $T_{c,low}$ (approximate loss is 10%) is considered and to $SM = 2$ in which $T_{c,high}$ (approximate loss is 5%) is considered. With this choice, the bias error in each benchmark is within approximately 1% of the absolute loss.

When $R = 200$, it is difficult to demonstrate the effect of SM choice because even very small SM values surely result in very small bias errors. Table 5.6 demonstrates the effect of SM choice on the resultant percentage bias error of the IS estimator. For this table, $R = 500$ samples and $M = 100$ sets are preferred because a bigger R value is better for demonstrating the effect of SM choice. In this table, the SM value is iterated starting from its smallest value, 1 and the resultant percentage bias error of the IS estimator for each benchmark and each different SM value is recorded, which is computed same as the percentage bias errors in Table 5.5. We use $T_{c,low}$ as our timing constraint,

Table 5.6: The *SM* value used in CompLossMC-IS and the corresponding percentage bias error

	SM=1	**SM=4**	**SM=7**	**SM=10**	**SM=13**	**SM=16**	**SM=19**	**SM=100**
c432	2.63	0.62	0.44	0.36	0.42	0.42	0.42	0.42
c499	0.53	0.39	0.42	0.42	0.42	0.42	0.42	0.42
c880	1.60	0.11	0.24	0.21	0.25	0.25	0.25	0.25
c1355	0.30	0.16	0.18	0.18	0.18	0.18	0.18	0.18
c1908	1.26	0.77	0.83	0.85	0.85	0.85	0.85	0.85
c2670	0.59	0.25	0.29	0.29	0.29	0.29	0.29	0.29
c3540	0.75	0.36	0.39	0.39	0.39	0.39	0.39	0.39
c5315	0.55	0.38	0.43	0.43	0.43	0.43	0.43	0.43
c7552	0.80	0.59	0.59	0.59	0.59	0.59	0.59	0.59

therefore the expected loss is 10%. As explained above, we always use an *SM* value equal to the 20% of the number of sample points for which IS estimator performs TL simulations, which results in $SM = 500 \times 10\% \times 20\% = 10$. According to Table 5.6, the bias errors corresponding to $SM = 10$ is same as the bias errors when a very big *SM* value like 100 is used. The table is stopped at 100 but the bias errors do not change even when bigger *SM* values are used. This is expected because after some values of *SM*, the ε margin computed by CompLossMC-IS does not change for any of the M sets. Another important consequence deduced from Table 5.6 would be that the bias error is not intolerable even when very small *SM* values are used as argued above. For instance, instead of $SM = 10$, using $SM = 4$ in CompLossMC-IS, results in very similar bias errors for all benchmark circuits.

5.4 Experimental Convergence Analysis

The purpose of this experiment is to empirically validate the theoretical convergence analysis conducted and the error estimation equations derived in Section 4.4 for the TL-MC and IS estimators.

The results we present in Figure 5.3 experimentally confirm the theoretical error/convergence equations, (4.25) for the TL-MC and (4.28) for the IS estimators, that were derived in Section 4.4. In this figure, a plot of loss error versus the number of TL circuit simulations is shown for both estimators. The smooth curves in this plot were obtained using the theoretical error formulas, i.e.

(4.25) and (4.28). The two other curves were generated by computing loss estimate errors over 250 independent runs (M=250), each of which explore a sample set size R (number of samples drawn from the PDF $f(X)$) ranging from 1 to 200. As explained before, TL circuit simulations are performed at all of the sample points for the standard TL-MC estimator, but a reduced number of simulations are performed for the IS estimator since most of the samples are discarded based on the evaluation of the PDM equations. We observe the excellent match between the theoretical and experimental error curves in this plot, validating the $1/\sqrt{N}$ dependency of error on the number of TL simulations for both of the estimators. The significant reduction in error that the IS estimator provides is also obvious in this graph. The results in Figure 5.3 were generated with circuit **c3540** in the ISCAS'85 benchmark suite (with similar results for the other circuits). In this case, the $Loss^{PDM,\varepsilon}$ value that is needed for computing the IS estimator in (4.5) is computed using the PDM based estimator in (4.8) using all of the 50,000 sample points generated during all of the 250 runs. In empirically computing the variances of both of the estimators to generate the curves in Figure 5.3, we use the loss value computed based on the standard TL-MC estimator with TL simulations at all of the 50,000 sample points in the parameter space. Since the number of samples used here is very large, we treat this loss value as the actual loss as if it was given to us by an oracle. After the emprical variances of TL-MC (VAR_{TL}) and IS (VAR_{IS}) estimators are computed, the errors are computed as $2\sqrt{VAR_{TL}}$ and $2\sqrt{VAR_{IS}}$ respectively.

As discussed in Section 4.4, both the standard TL-MC and IS estimators are unbiased and their means converge to the actual loss if a large number of samples are used. We empirically confirm this with the plot in Figure 5.4. The curves in this plot were generated using the same experiment described above that was used to generate the error curves in Figure 5.3. In order to generate the plot in Figure 5.4, we simply compute the means of the loss estimates obtained by the two estimators over the 250 independent runs with varying number of samples, whereas variances over these 250 runs were used for Figure 5.3. We can clearly observe in Figure 5.4 that the IS estimator converges to the actual loss value much earlier, with only a few number of TL simulations.

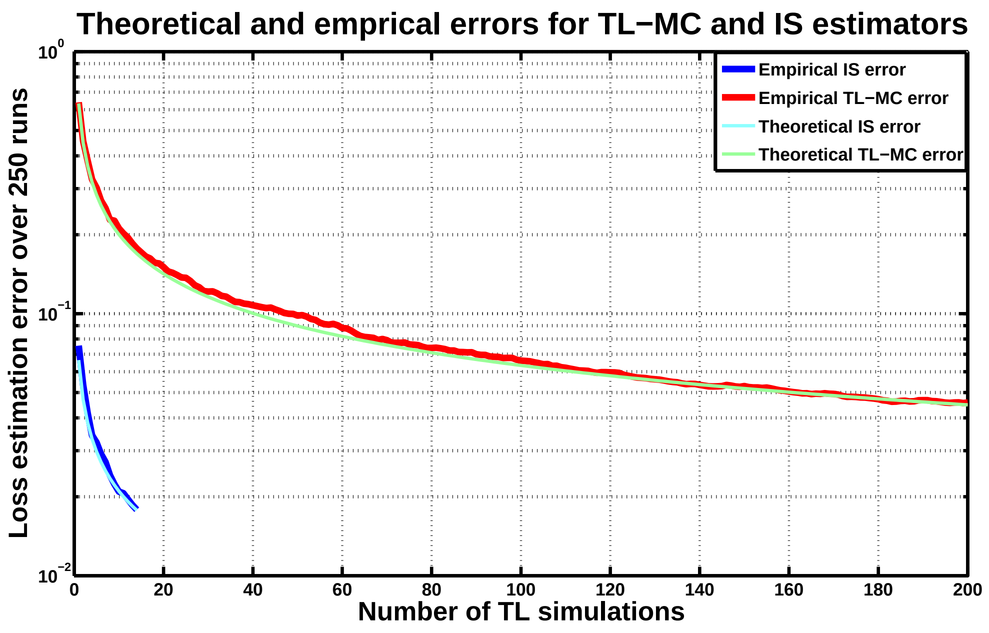


Figure 5.3: Convergence of standard TL-MC and IS estimators

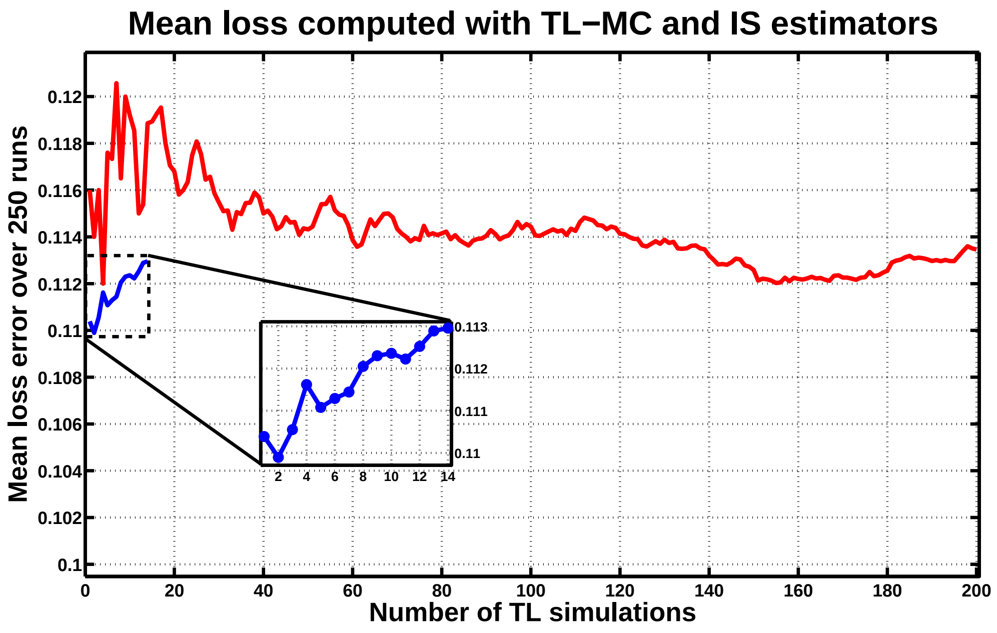


Figure 5.4: Mean loss computed with standard TL-MC and IS estimators

5.5 Discussion

In Chapter 5 the standard TL-MC method is compared with the proposed IS method for estimating timing yield. By Table 5.1 and 5.2, it is shown that IS method requires on the average 185 times less (155 times for $T_{c,low}$ and 215 times for $T_{c,high}$) TL simulations than standard TL-MC in order to compute the loss with the same accuracy with standard TL-MC method.

Up to now, in computing the `Speedup` of IS method, the additional cost for computing $Loss_K^{PDM,\varepsilon}$ by using (4.8) was ignored as no costly TL simulations are required in its computation. Although the computation of $Loss_K^{PDM,\varepsilon}$ is implemented on `MATLAB` and we use 50,000 samples to compute it ($K = 50,000$), the cost to compute $Loss_{50,000}^{PDM,\varepsilon}$ is much less than the cost required by TL simulations. Each column of Table 5.7 shows respectively from left to right the run times in seconds that is required for TL-MC, for TL simulations used for IS and for PDM evaluations used for $Loss_{50,000}^{PDM,\varepsilon}$ computation. The last column shows the new `Speedup` results without ignoring $Loss_{50,000}^{PDM,\varepsilon}$ computation cost. This table is for $T_{c,low}$, $R = 200$ and $M = 250$. For fair comparison, both TL-MC and IS estimators compared in the table result in loss estimates with the same accuracy (error). The average `Speedup` has dropped only from 155 to 120. If the polynomial evaluations used for $Loss_{50,000}^{PDM,\varepsilon}$ were performed in C program, the decrease in the `Speedup` would be negligible. This is why we had ignored additional cost of IS method for $Loss_{50,000}^{PDM,\varepsilon}$ computation.

Table 5.7: Run times for TL-MC and IS estimators considering $Loss_{50,000}^{PDM,\varepsilon}$ evaluation cost

Benchmarks	**t(**TL-MC)	**t(IS) for** TL **sim.**	**t(IS) for PDM eval.**	`Speedup`
c432	74468	870	158	72
c499	76236	372	102	161
c880	47524	522	221	64
c1355	115752	640	222	134
c1908	149048	1200	342	97
c2670	271386	1440	275	158
c3540	176002	1160	371	115
c5315	288930	1512	427	149
c7552	164413	910	369	129

An important result that can be deduced from Tables 5.3 and 5.4 is that the block level Monte Carlo (BL-MC) loss estimation is not so accurate. It is referred in Section 2.4.2 that BL-MC estimators are used as golden reference in order to test the accuracy of statistical static timing analysis (SSTA) methods. However, the results in these two tables show that the BL-MC method based on our third order polynomial gate delay model (PDM) results in errors of 44.3% at most and 16% at the average although all of the 50,000 sample points are used in loss estimation. This shows the necessity of transistor level simulation based statistical timing analysis for the final verification stage.

The `Speedup` of IS method increases while the probability of the event decreases. For instance, in our experiments the `Speedup` has increased from 155 to 215 while the approximate loss has decreased from 10% to 5%. This can also be seen from the theoretical error expression of IS estimator. Therefore, if our IS loss estimation method was applied for cases with very small loss probabilities like SRAM failures, the results would be much better. There are attempts to use different IS methods in SRAM failure analysis in the literature [65].

The most important and novel parts of our IS loss estimation methodology are the choice of a good biasing distribution (4.4), estimation of $Loss^{PDM,\varepsilon}$ (4.8) and the adaptive detection of the ε margin (CompLossMC-IS). IS method is a big step for availing TL timing analysis in statistical case and for the final verification stage before manufacturing. On the other hand, it is still slow to be used in design stages for optimizing the design.

Chapter 6

CONCLUSIONS AND FUTURE WORK

Summary

Designing fast circuits consuming less power and area is the main aim of digital IC (chip) designers. As a result, timing analysis of digital circuits is essential to estimate the circuit's speed performance and optimize the circuit accordingly until the desired performance is reached. When the stochastic nature of manufacturing and the resulting parameter variations are ignored, the strategy of Electronic Design Automation (EDA) community is as follows: They first perform deterministic static timing analysis (DSTA) to detect the critical paths in the circuit, then they perform transistor level timing simulation on these critical paths to detect the speed performance more accurately.

However, the statistical variations of manufacturing process have increased to a non-negligible level due to decreasing transistor sizes to have faster circuits covering less area. This necessitates statistical timing analysis which considers the variations and analyzes the circuits accordingly. As a result of the parameter variations, each manufactured chip of the same circuit has different parameter values and thus a different speed performance. The manufactured chips, which pass the speed tests are packaged for marketing and others that fail the tests are discarded. One of the main aims of statistical timing analysis is to estimate the timing yield, which is simply the fraction of chips that pass the speed tests.

Almost all proposed statistical timing analysis methods for digital circuits are block (gate) level methods and most of them are generalizations of DSTA to the statistical case. However block level statistical timing analysis lacks accuracy as it inherently contains many approximations like linearity and normality.

In this thesis, we try to fill the gap for accurate statistical timing analysis based on transistor level circuit simulations. For this purpose, we first propose a new comprehensive tool that combines different techniques in the literature for modeling variations and extracting the statistically critical paths of the circuit. But our main novel contribution is timing yield estimation using importance sampling in a novel manner in order to speed up transistor level Monte Carlo statistical timing analysis. We tested our method on ISCAS'85 circuits and the results show that our IS based yield estimation method improves the speed performance two orders of magnitude on the average.

Future Work

In order to improve the yield estimation methodology proposed in this thesis, other sources of variation like supply voltage, temperature, gate oxide thickness can be taken into consideration without changing any part of the methodology except for building new PDM model considering all of these parameters. In such cases, the PDM model generation cost must be decreased by the help of latin hypercube sampling, which avails the model generation by using same number of samples although the number of parameters are increased. The interconnect (wiring connection between the gates) width and height are also well known parameters that have non-negligible variations. The effect of interconnect could be inserted by modeling interconnect delays besides the PDM gate delay model.

Our IS estimator works with any approximate gate or path delay model inside. Therefore, more advanced gate delay models can increase the `Speedup` further. For instance, especially for input slope and for fanout, second degree polynomials can be fitted instead of first degree preferred for this thesis.

Different margin detection heuristics than CompLossMC-IS in Section 4.3, can be developed to improve the performance of IS estimation. But we observed that if these methods are conservative to satisfy the safety requirement given in Section 4.2, then the resultant efficiency of IS estimator decreases and otherwise if they are too loose, then the resultant IS estimator will loose accuracy, where accuracy is the main goal of IS estimation. Our heuristic gives a good compromise in between these two extremity.

There are other variance reduction techniques than IS, like control variates, stratified sampling, latin hypercube sampling and etc., which can be applied in order to increase speed without losing accuracy. Even some of them, for instance control variates and importance sampling, can be applied in a combined manner in order to have more efficiency and accuracy.

Appendix A

EXTRACTION OF TOP LONGEST PATHS

For applying the alternative method explained in Section 3.3, a new gate feature called $stepfathers$ is introduced. For a gate r, $delayfather$ feature shows the ancestor of gate r, which causes gate r to have its output arrival time, i.e. it shows the ancestor with maximum output arrival time. $stepfathers$ feature of gate r should show the ancestors which have output arrival times in a close proximity of the output arrival time of $delayfather$ of gate r. After running our default DSTA algorithm shown by Alg. 1 in order to modify timing DAG $C(V,E)$ by inserting the features $gatedelay$, $arrivaltime$, $delayfather$, the modified timing DAG $\tilde{C}(V,E)$ is the input argument of Alg. 3 below. Alg. 3 inserts the new $stepfathers$ feature to the gates in the timing DAG $\tilde{C}(V,E)$ structure. It has an input argument *ProximityCriteria* which simply determines how many of the ancestors will be set as step fathers. *ProximityCriteria* is a percentage that shows what percentage of the delay of $delayfather$ should be at a gate in order for that gate to be a $stepfathers$. It should be noted that the $delayfather$ of a gate is also a member of $stepfathers$ feature of the same gate. After Alg. 1 is run, we obtain a new feature called $stepfathers$, which consists of ancestor gates, which have output arrival times in a proximity of the output arrival time of the $delayfather$, which is the ancestor that has the maximum delay.

$stepfathers$ feature can be used to collect paths, which have delays in a proximity of maximum circuit delay. For this purpose, another algorithm (Alg. 4) is called with a vertex (gate) connected to a primary output of the timing DAG and then the algorithm records each step father in the $stepfathers$ feature of that vertex as a different path. Then, the algorithm recursively continues to perform the same operation for each step father of the first called vertex (Line 6 of Alg. 4). At the end, the algorithm collects all paths, which passes through step fathers, i.e. $PathList$. As this algorithm may collect too many paths, the paths having smaller delays than a desired lower delay bound can be thrown out after the algorithm finishes.

Algorithm 3 Insert-StepFathers($\tilde{C}(V,E)$, *ProximityCriteria*)

```
1.  for $GateInd = 1$ to number of vertices in $\tilde{C}$ do
2.    $r = V(GateInd)$
3.    $DelayFather = r.delayfather$
4.    $MaxArrivalTime = DelayFather.arrivaltime$
5.    $StepFatherCount = 0$
6.    for $AncestorInd = 1$ to $length(r.ancestors)$ do
7.      $ancestor = r.ancestors(AncestorInd)$
8.      if $(MaxArrivalTime - ancestor.arrivaltime)/MaxArrivalTime < ProximityCriteria$ then
9.        $StepFatherCount = StepFatherCount + 1$
10.       $r.stepfathers(StepFatherCount) = ancestor$
11.     end if
12.   end for
13. end for
```

Algorithm 4 Collect-NearCriticalPaths(*CurrentGate*, *CurrentPath*)

```
1. Append $CurrentGate$ to $CurrentPath$
2. if $CurrentGate$ is a primary input then
3.   Append $CurrentPath$ to $PathList$
4. else
5.   for $SFcode = 1$ to length($CurrentGate.stepfathers$) do
6.     Collect-NearCriticalPaths($CurrentGate.stepfathers(SFcode)$, $CurrentPath$)
7.   end for
8. end if
```

BIBLIOGRAPHY

[1] http://www.itrs.net/. International Technology Roadmap for Semiconductors (ITRS) Report 2009 Edition.

[2] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *ACM/IEEE International Conference on Computer Aided Design (ICCAD)*, 2003.

[3] Felipe S. Marques, Renato P. Ribas, Sachin Sapatnekar, and André I. Reis. A new approach to the use of satisfiability in false path detection. In *GLSVLSI ’05: Proceedings of the 15th ACM Great Lakes Symposium on VLSI*, pages 308–311, 2005.

[4] Laurence W. Nagel and D.O. Pederson. Spice (simulation program with integrated circuit emphasis). Technical Report UCB/ERL M382, EECS Department, University of California, Berkeley, Apr 1973.

[5] Kenneth S. Kundert. *The Designer’s Guide to Spice and Spectre*. Kluwer Academic Publishers, Norwell, MA, USA, 1995. Foreword By-Gray, Paul.

[6] Sachin Sapatnekar. *Timing*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2004.

[7] J. Bhasker and Rakesh Chadha. *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer Publishing Company, Incorporated, 2009.

[8] Luís Guerra e Silva, ao Marques-Silva, Jo L. Miguel Silveira, and Karem A. Sakallah. Satisfiability models and algorithms for circuit delay computation. *ACM Trans. Des. Autom. Electron. Syst.*, 7(1):137–158, 2002.

[9] Jacques Benkoski, E. Vanden Meersch, Luc J. M. Claesen, and Hugo De Man. Timing verification using statically sensitizable paths. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 9(10):10723–10784, 1990.

[10] D. Brand and V. S. Iyengar. Timing analysis using functional analysis. *IEEE Trans. Comput.*, 37(10):1309–1314, 1988.

[11] S. Perremans, L. Claesen, and H. De Man. Static timing analysis of dynamically sensitizable paths. In *Design Automation, 1989. 26th Conference on*, pages 568 – 573, 25-29 1989.

[12] P.C. McGeer and R.K. Brayton. Efficient algorithms for computing the longest viable path in a combinational network. In *Design Automation, 1989. 26th Conference on*, pages 561 – 567, 25-29 1989.

[13] H.-C. Chen and D.H.C. Du. Path sensitization in critical path problem. In *Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE International Conference on*, pages 208 –211, 11-14 1991.

[14] Srinivas Devadas, Abhijit Ghosh, and Kurt Keutzer. *Logic synthesis*. McGraw-Hill, Inc., New York, NY, USA, 1994.

[15] J. Marques Silva and K.A. Sakallah. Concurrent path sensitization in timing analysis. In *Design Automation Conference, 1993, with EURO-VHDL '93. Proceedings EURO-DAC '93. European*, pages 196 –199, 20-24 1993.

[16] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits. In *Proceedings IEEE International Symposium on Circuits and Systems*, page 695698, 1985.

[17] I. Sutherland, B. Sproull, and D. Harris. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann, 1999.

[18] A. Sreedhar and S. Kundu. Statistical timing analysis based on simulation of lithographic process. In *Computer Design, 2009. ICCD 2009. IEEE International Conference on*, pages 29 –34, 4-7 2009.

[19] Duane S. Boning and Sani Nassif. Models of process variations in device and interconnect. In *Design of High Performance Microprocessor Circuits*. IEEE Press, 2000.

[20] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. Statistical timing analysis: From basic principles to state of the art. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27:589–607, April 2008.

[21] Hongliang Chang and S.S. Sapatnekar. Statistical timing analysis under spatial correlations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(9):1467 – 1482, sept. 2005.

[22] C. Forzan and D. Pandini. Why we need statistical static timing analysis. In *Computer Design, 2007. ICCD 2007. 25th International Conference on*, pages 91 –96, 7-10 2007.

[23] F.N. Najm. On the need for statistical timing analysis. In *Design Automation Conference, 2005. Proceedings. 42nd*, pages 764 – 765, 13-17 2005.

[24] Hongliang Chang. *Circuit timing and leakage power analysis under process variations*. PhD thesis, University of Minnesota, Minneapolis, MN, USA, 2006. Adviser-Sapatnekar, Sachin S.

[25] C. Visweswariah. Death, taxes and failing chips. In *Design Automation Conference, 2003. Proceedings*, pages 343 – 347, 2-6 2003.

[26] A. Gattiker, S. Nassif, R. Dinakar, and C. Long. Timing yield estimation from static timing analysis. In *Quality Electronic Design, 2001 International Symposium on*, pages 437 –442, 2001.

[27] Aseem Agarwal, David Blaauw, Vladimir Zolotov, Savithri Sundareswaran, Min Zhao, Kaushik Gala, and Rajendran Panda. Statistical delay computation considering spatial correlations. In *ASP-DAC '03: Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, pages 271–276, New York, NY, USA, 2003. ACM.

[28] Rung-Bin Lin and Meng-Chiou Wu. A new statistical approach to timing analysis of vlsi circuits. In *VLSID '98: Proceedings of the Eleventh International Conference on VLSI Design: VLSI for Signal Processing*, page 507, Washington, DC, USA, 1998. IEEE Computer Society.

[29] Byungwoo Choi and D. M. H. Walker. Timing analysis of combinational circuits including capacitive coupling and statistical process variation. In *VTS '00: Proceedings of the 18th IEEE VLSI Test Symposium*, page 49, Washington, DC, USA, 2000. IEEE Computer Society.

[30] Michael Orshansky and Kurt Keutzer. A general probabilistic framework for worst case timing analysis. In *DAC '02: Proceedings of the 39th annual Design Automation Conference*, pages 556–561, New York, NY, USA, 2002. ACM.

[31] Hratch Mangassarian and Mohab Anis. On statistical timing analysis with inter- and intra-die variations. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 132–137, Washington, DC, USA, 2005. IEEE Computer Society.

[32] M. Berkelaar. Statistical delay calculation, a linear time method. In *TAU '97: Proceedings of the 3rd ACM/IEEE international workshop on Timing issues in the specification and synthesis of digital systems*, pages 15–24, 1997.

[33] Shuji Tsukiyama, Masakazu Tanaka, and Masahiro Fukui. A statistical static timing analysis considering correlations between delays. In *ASP-DAC '01: Proceedings of the 2001 Asia and South Pacific Design Automation Conference*, pages 353–358, New York, NY, USA, 2001. ACM.

[34] Jiayong Le, Xin Li, and Lawrence T. Pileggi. Stac: statistical timing analysis with correlation. In *DAC '04: Proceedings of the 41st annual Design Automation Conference*, pages 343–348, New York, NY, USA, 2004. ACM.

[35] Kunhyuk Kang, Bipul C. Paul, and Kaushik Roy. Statistical timing analysis using levelized covariance propagation. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 764–769, Washington, DC, USA, 2005. IEEE Computer Society.

[36] Jing-Jia Liou, Kwang-Ting Cheng, Sandip Kundu, and Angela Krstic. Fast statistical timing analysis by probabilistic event propagation. In *DAC '01: Proceedings of the 38th annual Design Automation Conference*, pages 661–666, New York, NY, USA, 2001. ACM.

[37] Jing-Jia Liou, Angela Krstic, Li-C. Wang, and Kwang-Ting Cheng. False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation. In *DAC ’02: Proceedings of the 39th annual Design Automation Conference*, pages 566–569, New York, NY, USA, 2002. ACM.

[38] Srinath R. Naidu. Timing yield calculation using an impulse-train approach. In *ASP-DAC ’02: Proceedings of the 2002 Asia and South Pacific Design Automation Conference*, page 219, Washington, DC, USA, 2002. IEEE Computer Society.

[39] Aseem Agarwal, David Blaauw, Vladimir Zolotov, and Sarma Vrudhula. Statistical timing analysis using bounds. In *DATE ’03: Proceedings of the conference on Design, Automation and Test in Europe*, page 10062, Washington, DC, USA, 2003. IEEE Computer Society.

[40] Aseem Agarwal, David Blaauw, Vladimir Zolotov, and Sarma Vrudhula. Computation and refinement of statistical bounds on circuit delay. In *DAC ’03: Proceedings of the 40th annual Design Automation Conference*, pages 348–353, New York, NY, USA, 2003. ACM.

[41] Anirudh Devgan and Chandramouli Kashyap. Block-based static timing analysis with uncertainty. In *ICCAD ’03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 607, Washington, DC, USA, 2003. IEEE Computer Society.

[42] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. In *DAC ’04: Proceedings of the 41st annual Design Automation Conference*, pages 331–336, New York, NY, USA, 2004. ACM.

[43] Hongliang Chang and Sachin S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single pert-like traversal. In *ICCAD ’03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 621, Washington, DC, USA, 2003. IEEE Computer Society.

[44] Charles E. Clark. The greatest of a finite set of random variables. *Operations Research*, 9(2):145 – 162, Mar. - Apr. 1961.

[45] Yaping Zhan, Andrzej J. Strojwas, Xin Li, Lawrence T. Pileggi, David Newmark, and Mahesh Sharma. Correlation-aware statistical timing analysis with non-gaussian delay distributions. In *DAC ’05: Proceedings of the 42nd annual Design Automation Conference*, pages 77–82, New York, NY, USA, 2005. ACM.

[46] Lizheng Zhang, Weijen Chen, Yuhen Hu, John A. Gubner, and Charlie Chung-Ping Chen. Correlation-preserved non-gaussian statistical timing analysis with quadratic timing model. In *DAC ’05: Proceedings of the 42nd annual Design Automation Conference*, pages 83–88, New York, NY, USA, 2005. ACM.

[47] Lizheng Zhang, Yuhen Hu, and C. Chung-Ping Chen. Statistical timing analysis with path reconvergence and spatial correlations. In *Design, Automation and Test in Europe, 2006. DATE ’06. Proceedings*, volume 1, page 5 pp., 6-10 2006.

[48] Jaskirat Singh and Sachin Sapatnekar. Statistical timing analysis with correlated non-gaussian parameters using independent component analysis. In *DAC ’06: Proceedings of the 43rd annual Design Automation Conference*, pages 155–160, New York, NY, USA, 2006. ACM.

[49] Hongliang Chang, Vladimir Zolotov, Sambasivan Narayan, and Chandu Visweswariah. Parameterized block-based statistical timing analysis with non-gaussian parameters, nonlinear delay functions. In *DAC ’05: Proceedings of the 42nd annual Design Automation Conference*, pages 71–76, New York, NY, USA, 2005. ACM.

[50] S. Tsukiyama and M. Fukui. Accuracy of the criticality probabilty of a path in statistical timing analysis. In *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, pages 707 –710, 23-27 2009.

[51] Jinjun Xiong, Vladimir Zolotov, Natesan Venkateswaran, and Chandu Visweswariah. Criticality computation in parameterized statistical timing. In *DAC ’06: Proceedings of the 43rd annual Design Automation Conference*, pages 63–68, New York, NY, USA, 2006. ACM.

[52] L.-C. Wang, Jing-Jia Liou, and Kwang-Ting Cheng. Critical path selection for delay fault testing based upon a statistical timing model. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 23(11):1550 – 1565, nov. 2004.

[53] F. Wane, Yuan Xie, and Hai Ju. A novel criticality computation method in statistical timing analysis. In *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, pages 1 –6, 16-20 2007.

[54] Yaping Zhan, A.J. Strojwas, M. Sharma, and D. Newmark. Statistical critical path analysis considering correlations. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 699 – 704, 6-10 2005.

[55] Jing-Jia Liou, A. Krstic, L.-C. Wang, and Kwang-Ting Cheng. False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation. In *Design Automation Conference, 2002. Proceedings. 39th*, pages 566 – 569, 2002.

[56] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. In *DAC '04: Proceedings of the 41st Design Automation Conference*, pages 331–336, 2004.

[57] J. Singh and S. S. Sapatnekar. A scalable statistical static timing analyzer incorporating correlated non-gaussian and gaussian parameter variations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27:160–173, January 2008.

[58] Malvin H. Kalos and Paula A. Whitlock. *Monte Carlo Methods, Volume 1, Basics*. Wiley, 1986.

[59] P. Glasserman, P. Heidelberger, and P. Shahabuddin. Importance sampling and stratification for value-at-risk. In *Proceedings of the 6th International Conference on Computational Finance*, pages 7–24. MIT Press, May 28-31 1999.

[60] L. Scheffer. The count of Monte Carlo. In *ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU)*, February 2004.

[61] Vineeth Veetil, Dennis Sylvester, and David Blaauw. Efficient monte carlo based incremental statistical timing analysis. In *DAC '08: Proceedings of the 45th annual conference on Design automation*, pages 676–681, 2008.

[62] Taiwan Semiconductor Manufacturing Company (TSMC) Reference Flow 9.0 Press Release. New TSMC reference flow 9.0 supports 40nm process technology. http://www.tsmc.com.

[63] D.E. Hocevar, M.R. Lightner, and T.N. Trick. A study of variance reduction techniques for estimating circuit yields. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2(3):180–192, July 1983.

[64] L. Dolecek, M. Qazi, D. Shah, and A. Chandrakasan. Breaking the simulation barrier: Sram evaluation through norm minimization. In *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, pages 322–329, Nov. 2008.

[65] R. Kanj, R. Joshi, and S. Nassif. Mixture importance sampling and its application to the analysis of sram designs in the presence of rare failure events. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 69–72, 0-0 2006.

[66] A. Singhee and R.A. Rutenbar. Statistical blockade: A novel method for very fast monte carlo simulation of rare circuit events, and its application. In *Design, Automation and Test in Europe Conference and Exhibition, 2007. DATE '07*, pages 1–6, April 2007.

[67] M. Zhang, M. Olbrich, H. Kinzelbach, D. Seider, and E. Barke. A fast and accurate monte carlo method for interconnect variation. In *Integrated Circuit Design and Technology, 2006. ICICDT '06. 2006 IEEE International Conference on*, pages 1–4, 0-0 2006.

[68] Guo Yu, Wei Dong, Zhuo Feng, and Peng Li. A framework for accounting for process model uncertainty in statistical static timing analysis. In *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pages 829–834, June 2007.

[69] V. Khandelwal and A. Srivastava. A general framework for accurate statistical timing analysis considering correlations. In *Design Automation Conference, 2005. Proceedings. 42nd*, pages 89–94, June 2005.

[70] Mingjing Chen and A. Orailoglu. Circuit-level mismatch modelling and yield optimization for cmos analog circuits. In *Computer Design, 2007. ICCD 2007. 25th International Conference on*, pages 526–532, Oct. 2007.

[71] S. Yaldiz, U. Arslan, Xin Li, and L. Pileggi. Efficient statistical analysis of read timing failures in sram circuits. In *Quality of Electronic Design, 2009. ISQED 2009. Quality of Electronic Design*, pages 617–621, March 2009.

[72] Bao Liu. Gate level statistical simulation based on parameterized models for process and signal variations. In *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on*, pages 257–262, March 2007.

[73] A. A. Bayrakci, A. Demir, and S. Tasiran. Fast monte carlo estimation of timing yield: Importance sampling with stochastic logical effort (ISLE). Technical Report arXiv:0805.2627, http://arxiv.org, 2007.

[74] A. Demir and S. Tasiran. Statistical logical effort: Designing for timing yield on the back of an envelope. In *ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU)*, February 2006.

[75] K-T. Fang, R. Li, and A. Sudjianto. *Design and Modeling for Computer Experiments*. CRC Press, 2006.

[76] http://dropzone.tamu.edu/ xiang/iscas.html. Copyright (c) 2003 Texas A&M Unverisity, College Station, TX.

[77] S. H. Yen, D. H. Du, and S. Ghanta. Efficient algorithms for extracting the k most critical paths in timing analysis. In *DAC '89: Proceedings of the 26th ACM/IEEE Design Automation Conference*, pages 649–654, New York, NY, USA, 1989. ACM.

[78] Y. Zhan, A. J. Strojwas, M. Sharma, and D. Newmark. Statistical critical path analysis considering correlations. In *ACM/IEEE International Conference on Computer Aided Design (ICCAD)*, November 2005.

[79] Feng Wang, Yuan Xie, and Hai Ju. A novel criticality computation method in statistical timing analysis. In *DATE '07: Proceedings of the Conference on Design, Automation and Test in Europe*, 2007.

[80] http://www.satcompetition.org. The international SAT Competitions web page.

[81] N. Eén and N. Sorensson. Minisat a sat solver with conflict-clause minimization. In *International Conference on Theory and Applications of Satisfiability Testing*, 2005.

[82] http://www.vlsitechnology.org. Graham Petley's 0.13μ Cell Library, Release 8.5.

[83] http://dropzone.tamu.edu/˜cad. VLSI Computer-Aided Design (CAD) and Test Research Group.

[84] Y. Cao, H. Qin, R. Wang, P. Friedberg, A. Vladimirescu, and J. Rabaey. Yield optimization with energy-delay constraints in low-power digital circuits. In *IEEE Conference on Electron Devices and Solid-State Circuits*, December 2003.