Probabilistic Analysis for Modeling and Simulating Digital Circuits

by

Chien-Chih Yu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering)
in The University of Michigan
2012

To Mom and My Sister

ACKNOWLEDGEMENTS

I wish to thank all the people who helped me with my research and life at the University of Michigan. I would like to thank my academic advisor Professor John Hayes not only for his advice on research but also for his firm and continuous financial support. I would also like to thank Professor Lavieri for her feedback on my research, and Professors Austin and Papaefthymiou for serving on my Thesis Committee.

I am grateful to Dr. Kenneth Zick for providing a unique opportunity to participate a satellite circuit design project at the USC Information Science Institute. I also thank Armin Alaghi with whom I collaborated. I could not have finished my last research task without his brilliant observations and ideas. I would like to thank people who helped to improve my writing skill including Professors Rod Johnson and Elizabeth Hildinger, and Jin Hu. I would like to thank Stella Hu who gave great help when I was looking for jobs.

I am also grateful to the National Science Foundation for supporting my Ph. D. studies.

I would like to acknowledge my classmates and friends including Kai-hui Chang, Smita Krishnaswamy, Sungsoon Cho, Ramashis Das, Dae Young Lee, MingChi Hsu, Andrea Pellegrini, Cheng Zhuo, Po-Chun Hsu, Rita Tsai, Wei-Ling Chiang, Cheng Zhuo, Te-Hsuan Chen, Ada Lin, Emily Yeh, and many others in America and in Taiwan.

Finally, I would thank my mom and sister. I could not have completed the entire doctoral program without their encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

Due to the rapid progress of their manufacturing technologies, integrated circuit (ICs) can now contain billions of transistors and operate at gigahertz frequencies. This great complexity has forced engineers to rely on electronic design automation (EDA) software tools to design, verify and test new ICs. Traditional EDA tools are deterministic in nature and try to explicitly address all a circuit's operating modes by examining very large input signal sets and computing their output responses. However, beyond some point, such methods must be replaced by random sampling of the inputs, an approach that is inherently probabilistic. Manufacturing process variations and soft errors caused by environmental disturbances also call for statistical approaches to gauge their impact. Hence, there is an increasing need for probabilistic characterizations of IC behavior that can be easily incorporated into EDA tools, and can be used in situations where traditional deterministic approaches are ineffective.

The goal of this dissertation is to develop ways to significantly improve the quality of the probabilistic analysis techniques required for EDA. The accuracy and scalability of these techniques is greatly affected by several factors, including the probability models employed and the handling of correlations among signals. To address such issues, we develop novel and efficient ways to sample logic circuit behavior, model the impact of soft errors, and estimate circuit reliability. First, we present a methodology for sampling input signals that improves accuracy and runtime by prioritizing the sample variables and compressing the sample space. Then, we introduce a trigonometry-based technique for

efficiently analyzing soft errors by mapping signal probabilities into angles. Finally, a reliability estimation method is described that uses probabilistic transfer matrices to calculate signal and error probability distributions in sequential circuits. Unlike previous techniques, its memory usage grows slowly even when simulating very large circuits over many clock cycles. Extensive simulation studies are presented in support of all the foregoing results.

The contributions of this dissertation identify features of probabilistic, error-inducing phenomena that can lead to significant improvements in circuit quality. They also reduce the computational overhead for probabilistic calculations which are essential for many EDA tasks.

.

CHAPTER 1
Introduction

In the semiconductor industry, the cost of designing and manufacturing integrated circuits (ICs), including financial investment, time-to-market, and human resources heavily depends heavily on the effectiveness of electronic design automation (EDA) tools. EDA refers to the use of software-implemented techniques for accelerating the design and manufacturing processes, and for improving the quality of IC-based products. EDA tools have been widely adopted for implementing routine design steps, for validating circuit functionality, for analyzing performance features such as delay and power consumption, for generating layout data, and for testing product quality.

Traditional EDA methods employ deterministic approaches, which assume that the circuit's entire structure and behavior are definite and fully predictable. However, IC complexity has reached a point where circuits contain billions of transistors and can perform billions of calculations per second. This complexity poses some new challenges for EDA. For example, random sampling is increasingly necessary to estimate important aspects of IC behavior. Furthermore, ICs are becoming sensitive to non-deterministic effects such as imperfect manufacturing processes and environmental disturbances. Hence, conventional deterministic approaches are becoming less effective for many EDA tasks.

Unlike deterministic approaches, probabilistic analysis views a circuit from a statistical perspective, and provides a natural way both to describe and simulate non-deterministic effects. It models the properties of devices and signals in terms of probabilities, and measures their characteristics using statistical techniques such as sampling and averaging. Probabilistic analysis has been applied to several areas of EDA, such as the estimation of reliability [61], testability [46], and power [47]. Recently, probabilistic analysis has also been used for measuring the impact of manufacturing variations in ICs [21][38][41], and for analyzing the vulnerability of ICs to soft errors resulting from decreasing noise margins [19][67].

In this chapter, we provide an overview of the role of probabilistic analysis in EDA. We review the basic concepts of probabilistic analysis, study several existing probability-based applications, and address the problems and limitations of existing methods. We then summarize the major contributions of this dissertation.

1.1 Background

Integrated circuit (IC) technology was invented in the late 1950s [42]. After 50 years of steady development, ICs now are widely used in personal computers, mobile phones, cars, and so on. Not only are they deeply integrated into our daily life, but they also have changed the way modern society operates. For instance, supercomputers containing vast numbers of ICs are used for weather and environmental forecasting, geological surveys in the energy industry, and structure analyses of proteins in the pharmaceutical industry [55].

Virtually all ICs are manufactured with complementary metal-oxide-semiconductor (CMOS) process technology, where the fundamental components are switching devices called transistors. The structure of a typical transistor is shown in Figure 1.1(a). There are four parts to a transistor: the gate, the substrate, and two terminals called the source and drain. The length *L* of a transistor's gate is a key parameter that is often referred to as the "feature size" of a particular CMOS manufacturing technology "node" in the industry. For instance, a 0.25-μm (0.25×10^{-6} meter) CMOS manufacturing technology indicates one that is capable of producing transistors whose gate lengths are around 0.25 μm.

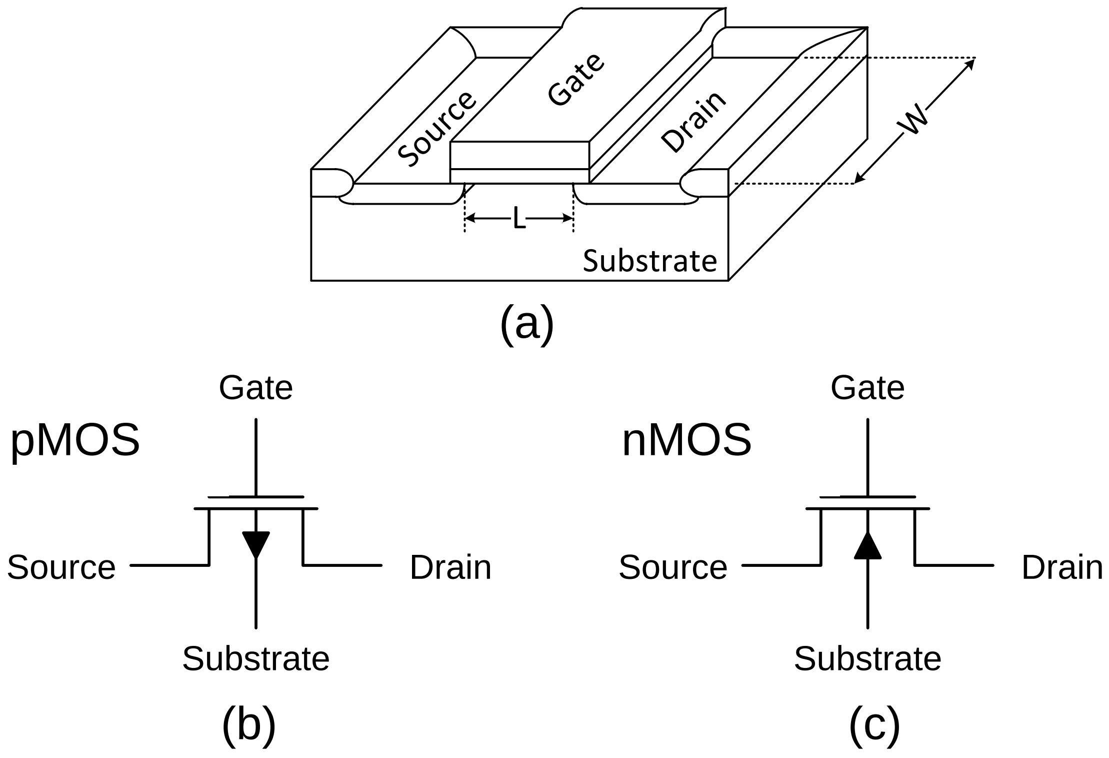


Figure 1.1. (a) Structure of a transistor; *L* and *W* denote the length and width of the gate, (b) pMOS symbol and (c) nMOS symbol.

In 1965, Gordon Moore of Intel predicted that the number of transistors in an IC would double about every 24 months, an insightful observation now referred to as Moore's Law [42]. Continuing developments in CMOS process technology suggest that Moore's Law will extend into the next decade [30][65], as shown in Table 1.1.

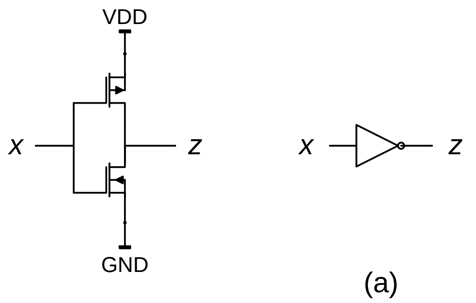
Table 1.1. Technology trends in CMOS manufacturing [65].

Year	2006	2008	2010	2012	2014
Feature size (nm)	65	45	32	22	16
Supply voltage	1.2	1.0	0.9	0.8	0.7
Gate count (10^9)	4	8	16	32	64

With modern process technologies, engineers can design ICs that are much more complex in terms of transistor count and operation speed than ever before. In 1997, for instance, Intel's Pentium II microprocessors were manufactured with a 0.25-μm CMOS technology; they contained 7.5 million transistors, and operated at 200 MHz. In 2012, Intel's Core i7 microprocessors manufactured with a 28-nm technology, contain over 1.7 billion transistors, and operate at 3 GHz. This represents an improvement of about three orders of magnitude in transistor density and computational capacity.

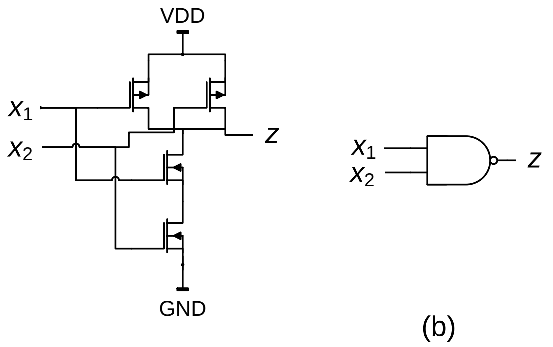
Two types of transistors are used in logic circuits, namely are n-type and p-type metal-oxide-silicon (MOS) transistors, i.e., nMOS and pMOS transistors, as shown in Figures 1.1(b) and (c). A transistor's current in a conducting channel between the source and drain is controlled by the voltage applied to the transistor's gate terminal. An important electrical parameter that characterizes a transistor's behavior is its threshold voltage Vt. For an nMOS (pMOS) transistor, if the voltage between its gate and source Vgs is higher (lower) than its threshold voltage, i.e., Vgs > Vtn (Vgs < Vtp), then the channel between the source and drain begins to conduct. We say a transistor is ON if its channel is conducting; otherwise, it is OFF. Clearly, transistors can be used as switches, and their ON and OFF states are usually referred to as 1 and 0 in logic circuits.

In addition, pMOS and nMOS transistors can be used for forming the basic components (gates) used in logic circuits. There are five types of elementary gates, namely inverter, NAND, NOR, AND and OR gates, each of which implements a specific Boolean function. Figure 1.2 shows the circuit structures, graphic symbols, and the functions of an inverter, and two-input NAND and NOR gates. AND and OR gates are typically formed by combining NAND or NOR gates with inverters.



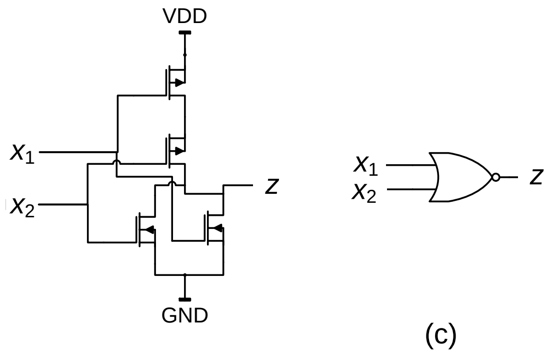
Input x_1	Output z
0	1
1	0

(a)



Input x_1x_2	Output z
00	1
01	1
10	1
11	0

(b)



Input x_1x_2	Output z
00	1
01	0
10	0
11	0

(c)

Figure 1.2. Transistor schematic, circuit symbol, and truth table of (a) an inverter, (b) a two-input NAND gate, and (c) a two-input NOR gate.

Integrated circuits can be viewed at several levels of abstraction such as the transistor, logic, register-transfer, and system levels [62], We will generally only consider the logic level, where the IC is modeled as a large set of interconnected gates that process the logic signals 0 and 1. This level has the advantage of being relatively independent of electrical and other physical details. However, it may entail special (indirect or non-deterministic) methods to analyze properties of interest such as power consumption.

Consider, for example, the task of analyzing a circuit's power consumption. A CMOS gate's maximum power consumption occurs when a signal at the gate changes from 0 to 1 or from 1 to 0. In general, a gate's dynamic power consumption is proportional to its *switching activity*, which indicates how frequently the gate's output signal changes value [22][47][58].

Figures 1.3(a) and (b) show a circuit assigned two different but similar four-bit input sequences: the number of 1s and 0s in the corresponding sequences are the same. However, the total numbers of transitions at g_1, g_2, and g_3 in Figures 1.3(a) and (b) are 6 and 4, respectively, and hence the power consumption in Figure 1.3(a) is higher by about 33% than that in Figure 1.3(b), assuming all gates have the same power consumption characteristics. Hence, the order of the applied inputs needs to be taken into consideration for accurately measuring the circuit power consumption with deterministic simulations. The total number of possible cases that exists in this small circuit is $16! = 2.09 \times 10^{13}$. As can be seen, measuring the power consumption by explicitly enumerating all possible cases is infeasible.

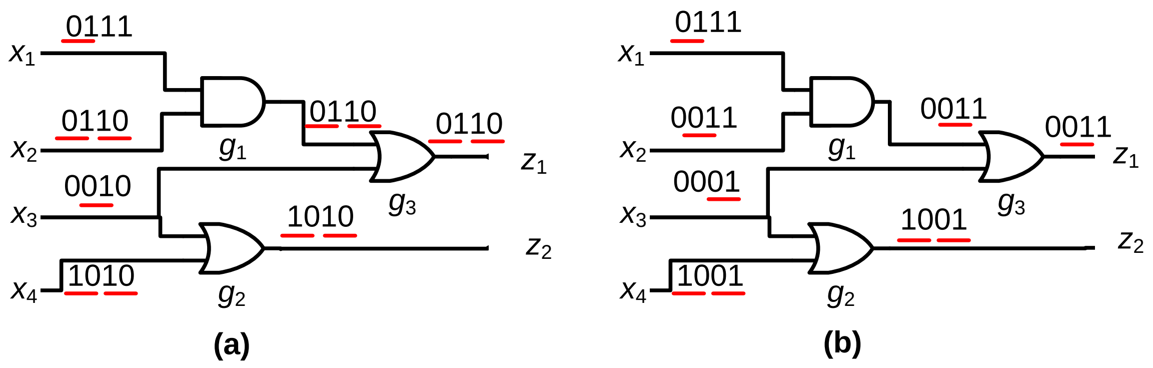


Figure 1.3. Switching activities caused by different input sequence orders; signal transitions are underlined.

Instead of exhaustively simulating circuits with all possible input sequences, Wu et al. [64] show that a circuit's power consumption can be accurately calculated by randomly sampling the switching activities of its individual gates. We will show in detail in Chapter 2 how to estimate a circuit's power consumption by probabilistic analysis of switching activities.

Deterministic approaches are inadequate for simulating the non-deterministic effects caused by process variations. These are phenomena where semiconductor fabrication processes are unable to produce identical transistors when a transistor's size shrinks below the scale of a few hundreds atoms. Performance features, such as delay and power consumption can vary from transistor to transistor, even in the same chip.

For instance, when CMOS technology is scaled to below 65 nm, precisely controlling the shape and doping concentrations of transistors during manufacture becomes difficult. The resulting process variations not only reduce manufacturing yield, but also affect circuit characteristics such as leakage current, threshold voltage, and power consumption. For instance, a major problem of 65-nm manufacturing technology is random dopant fluctuation (RDF) where the number of dopant atoms is unevenly implanted in a transistor [6]. Another source of process variation is line-edge roughness (LER), resulting from imperfect etching processes. In 65-nm CMOS technology, forming uniform shapes for individual transistors by etching is very difficult, because the etching process cannot be precisely controlled [44]. A consequence of the LER problem is an increase in the variance of channel length among transistors.

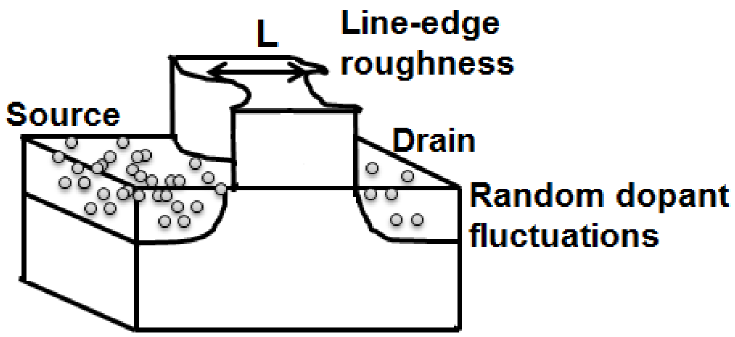


Figure 1.4 Impact of the LER and RDF phenomena on a transistor.

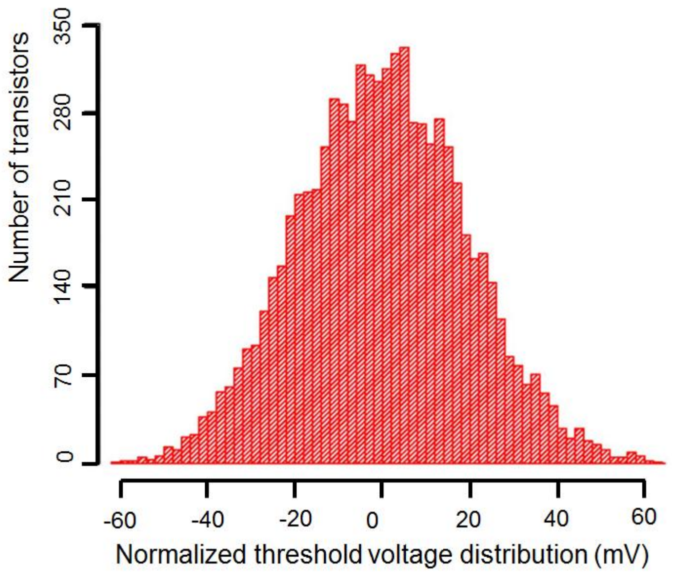


Figure 1.5. Threshold voltage variation of the IBM 65-nm silicon-on-isolator process; results are measured among 8,000 devices [4].

Figure 1.4 illustrates the impact of RDF and LER on a single transistor. Such manufacturing problems make the threshold voltages and leakage currents of individual transistors different from each other. This, in turn, affects the electrical characteristics such as threshold voltage of logic gates. Figure 1.5 shows threshold voltages of transistors in the same wafer spread across 120 mV, which is a broad range. As the result, individual chips can have unique electrical characteristics that affect their delay characteristics and power consumption. Figure 1.6 illustrates the differences in gate delay caused by process variations among three individual Intel microprocessor chips.

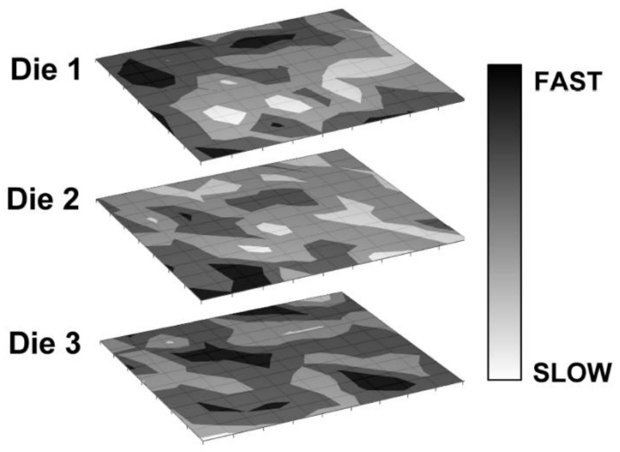
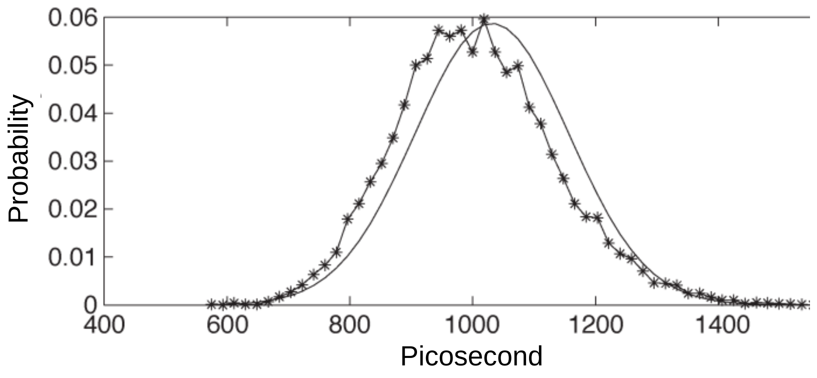


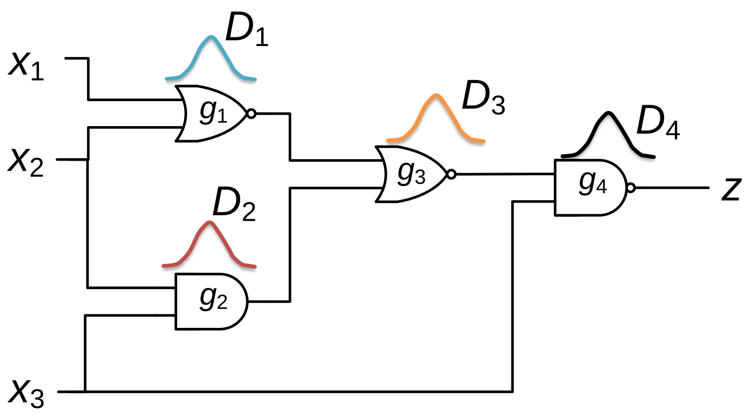
Figure 1.6. Delay profiles for three different 65-nm 80-core microprocessor chips [24].

Identifying the longest paths in an IC and evaluating their delays are important tasks for circuit design and testing [43][50]. Traditional timing analysis methods determine the longest delays from a circuit's structure, and they assume the delays of each gate type are the same. However, these deterministic methods are increasingly unable to capture the complete timing behavior of paths and gates caused by process variations.

A technique called statistical timing analysis [21][38][41] has been developed to deal with the aforementioned problems. Unlike static timing analysis methods, which model gate delays as constants, the statistical approach treats gate delays as random variables. The effects of process variations on delay are described in terms of the variances of the random variables. For a path of interest, its delay may be estimated by propagating delay variables along the path, as depicted in Figure 1.7. Probability-based timing analysis is being gradually adopted by industry [17][63].



(a)



(b)

Figure 1.7. (a) Gate delay distribution. (b) Illustration of statistical timing analysis; the delay of each gate g_i is modeled by a random variable D_i.

Recently, concern about soft errors has been increasing. Probabilistic analysis of soft errors raises issues and problems different from those mentioned so far. When transistors are manufactured at the nanometer scale, they are very sensitive to any manufacturing or electrical fluctuations, such as small process defects and unexpected voltage drops, and devices can temporarily malfunction. In other words, the behavior of a circuit is no longer fully predictable even if it has no manufacturing defects. Thus, circuit behavior is characterized by some degree of uncertainty because of the unpredictable occurrence of soft errors.

There are three major sources of soft errors: high-energy particles due to cosmic radiation, process variation, and voltage overscaling. Cosmic rays are composed of various subatomic particles such as neutrons originating from outer space [36]. The energy density of cosmic rays increases with altitude. For instance, the energy density at 30,000 feet, a typical flight altitude for commercial airplanes, is two orders of magnitude higher than the energy density at sea level [71].

When a high-energy particle hits a transistor, the state of the transistor may be temporarily changed. Such particle strikes are called *single-event upsets* (SEUs), see Figure 1.8. SEUs may lead to (soft) errors in circuits, or even to catastrophic system failures if they are successfully propagated to primary outputs or are stored by memory elements. For example, in 2008, a Qantas A330 airplane suddenly went out of control, and plunged downwards resulting in serious injuries to a flight attendant and several passengers. After examining the flight recorder, the Australian Transport Safety Bureau concluded the accident resulted from high energy neutron strikes to the aircraft’s control computers [9].

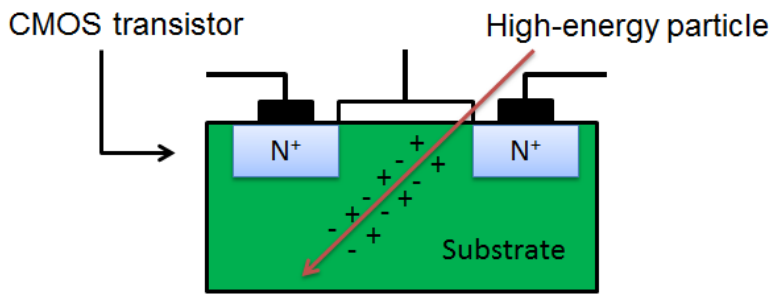


Figure 1.8 A soft error in a transistor caused by a high-energy particle.

Even though the energy density of cosmic rays at ground level is relatively low, the threshold energy for generating SEUs on a transistor has been decreasing with falling supply voltage and transistor size. Therefore, the impact SEUs on ICs can no longer be ignored [23]. Ando et al. [6] designed a series of experiments to test the susceptibility of 65-nm SPARC microprocessors to soft errors. Their experimental results show that without adequate hardware protection mechanisms, *e.g.*, error-correcting codes (ECCs), a system failure caused by soft errors will be observed every two and a half months. This failure rate is unacceptable, particularly for systems designed for critical applications, such as data centers and vehicle control.

Process variations are another source of soft errors because the threshold voltages of gates can be spread across a broad range, as shown in Figure 1.5. For instance, if a circuit is operated at a fixed supply voltage, then gates with high threshold voltages may not fully turn on, or work more slowly than expected. If such gates lie along critical paths, then incorrect results can easily be produced.

Finally, if a circuit's supply voltage is lower than a critical value, say the threshold voltage, the circuit can malfunction because some transistors no longer switch on or off correctly. Problems of this kind caused by insufficient supply voltages are called voltage overscaling. If circuits work at minimal voltage levels, noise margins cannot compensate for even slight voltage fluctuations, and voltage-overscaling problems result [12]. ICs designed for portable devices, such as mobile phones and laptops are especially prone to this issue. Their error rates (error probabilities) can grow dramatically with small decreases in supply voltage, as shown in Figure 1.9 [60].

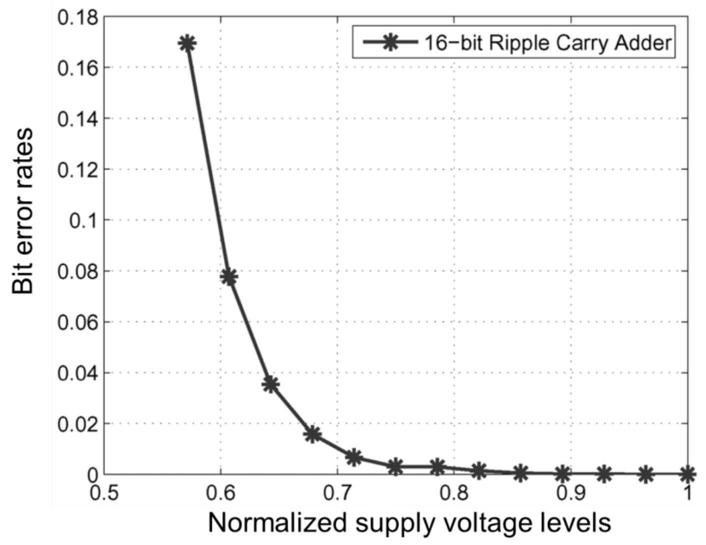


Figure 1.9. Bit error rates for different voltage overscaling levels in a 16-bit ripple-carry-adder using IBM 130 nm-process parameters; normal supply voltage is 1.35v [60].

We have illustrated several limitations of deterministic approaches for simulating typical EDA tasks and newly-arising non-deterministic effects. Unlike deterministic approaches, probabilistic analysis deals with these issues by modeling a circuit's characteristics with probabilities, and analyzing the circuit's statistical behavior.

Probabilistic analysis usually provides *average* results based on statistical estimates. Such results can be treated as a circuit's representative behavior, which is very useful for designers. Exact probability calculation methods [48] also suffer from high computational complexity, but many heuristics have been proposed for mitigating the computational overhead, while making the calculation inaccuracies manageable.

Our work aims at improving the applicability of probabilistic analysis in two areas: signal probability calculation and soft error analysis. We propose three novel and powerful methods for improving accuracy as well as efficiency for general circuit sampling, signal probability representation, and reliability estimation in sequential circuits.

1.2 Signal Probability Analysis

In this section, we define signal probability for logic circuits, discuss its basic calculation rules, and explain signal correlation, which is an important factor affecting calculation efficiency and accuracy.

The *signal probability* $p(s)$ of a logic signal s indicates the likelihood that 1 rather than 0 can be observed on s. Signal probability is determined experimentally by applying N input vectors to an n-input circuit, either physically or by simulation, and counting the number k of 1s produced on s, in which case, $p(s) = k/N$. If the applied vectors are randomly generated, such simulation procedures for obtaining signal probabilities are referred to as Monte Carlo (MC) sampling, and the total number of applied samples (input vectors) is called the sample size. The number $p(s)$ is considered to be *exact* if it equals $p^*(s)$, where $p^*(s)$ is defined as the signal probability obtained when $N = 2^n$, and the applied vectors are all different, i.e., when the simulation is exhaustive. Clearly, $p^*(s)$ is the fraction of 1s in the truth table for s. Consider the four-input circuit of Figure 1.10. If it is simulated with all 16 different input vectors as shown, then we obtain z's exact signal probability $p^*(z) = 0.5$.

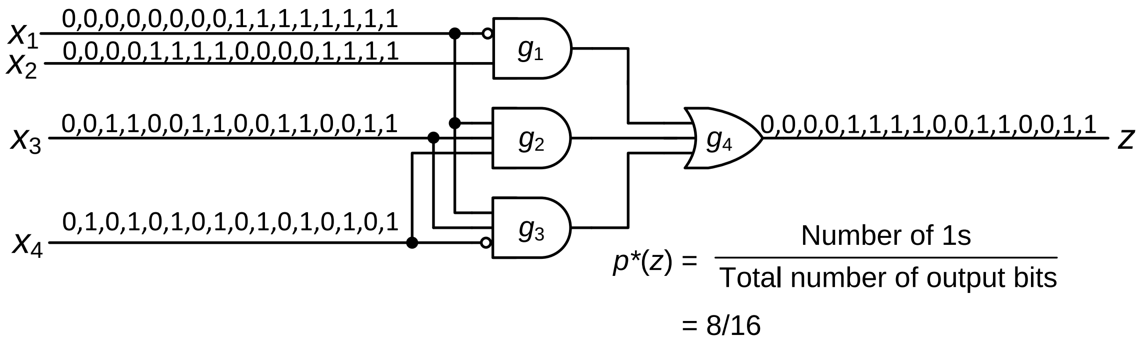
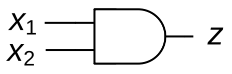


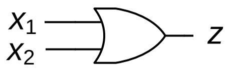
Figure 1.10 Signal probability estimation via exhaustive simulation.

Another approach for obtaining signal probabilities is probabilistic analysis. Consider the two-input AND gate g_{and} as shown in Figure 1.11(a) with inputs x_1 and x_2. Its exact output signal probability can be expressed as $p^*(g_{and}) = p^*(x_1) \times p^*(x_2) = 0.25$ because the probability of the AND gate being 1 is the probability of all its two inputs being 1. Similarly, for a two-input OR gate g_{or} in Figure 1.11(b), its probability of outputting 0 is the probability of all its inputs being 0, which can be expressed as $1 - p^*(g_{or}) = (1 - p^*(x_1)) \times (1 - p^*(x_2))$. Hence, this OR gate's exact output signal probability is given by $p^*(g_{or}) = 1 - (1 - p^*(x_1)) \times (1 - p^*(x_2)) = 0.75$.



Input x_1x_2	Output z
00	0
01	0
10	0
11	1

(a)

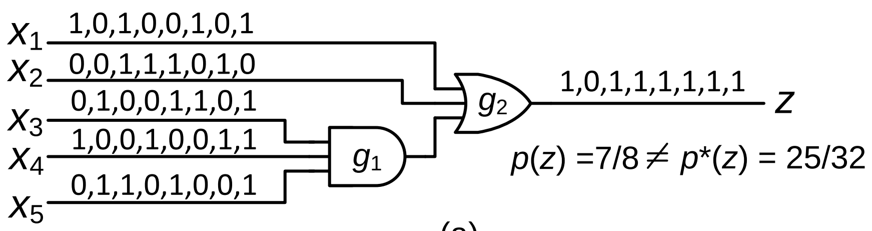


Input x_1x_2	Output z
00	0
01	1
10	1
11	1

(b)

Figure 1.11. Circuit symbol and truth table of (a) a two-input AND and (b) a two-input OR gates.

Figure 1.12 illustrates how to calculate signal probabilities via two different approaches. In Figure 1.12(a), the circuit output signal probability $p(z)$ is estimated by MC sampling. The circuit is simulated with eight input vectors selected randomly from 32 possibilities, and $p(z)$ is calculated as the fraction of 1s in z's output sequence, which is 7/8. If $p(z)$ in is evaluated by the first four samples shown, then $p(z)$ becomes 3/4, which is less accurate than the estimated result from eight samples. In the MC method, higher accuracy is achieved by applying more samples (input vectors). To be effective, however, MC must achieve a proper balance between computational effort and accuracy. Too few samples provide insufficient accuracy, while too many lead to excessive runtimes.



(a)

$$p^*(g_1) = p^*(x_3)p^*(x_4)p^*(g_5) = 1/8$$

$$p^*(z) = 1 - (1 - p^*(x_1))(1 - p^*(x_2))(1 - p^*(g_1)) = 25/32$$

(b)

Figure 1.12. Signal probability estimation for a five-input circuit via (a) MC sampling and (b) probabilistic analysis.

Figure 1.12(b) shows how $p(z)$ is evaluated using probabilistic analysis. Here the circuit is traversed from primary inputs to primary output, and the signal probabilities of the individual gates are calculated. For instance, $p^*(g_1)$ is first calculated from the signal probabilities of its input signals, $p^*(x_3)$, $p^*(x_4)$, and $p^*(x_5)$. Then, $p^*(z) = p^*(g_2)$ is obtained from $p^*(x_1)$, $p^*(x_2)$, and $p^*(g_1)$. In this case, the exact output signal probabilities can be obtained using probabilistic analysis because the structure of this circuit is relatively simple. In general, the computational overhead and accuracy of probabilistic analysis is strongly affected by circuit structure.

If a gate's inputs that are controlled by one or more common primary inputs, then these inputs should not be treated as separate independent signals. Instead, they should be considered together to avoid calculation inaccuracy. Such dependency among signals is called *correlation*. A major challenge of probabilistic analysis is how to handle correlation among signal probabilities. We next discuss what signal correlation is and how it affects probabilistic calculations.

Consider the two-input AND gate depicted in Figure 1.13(a) whose two input signals are tied together. The exact output signal probability, $p^*(g) = p^*(x_1) = 0.5 \neq p^*(x_1) \times p^*(x_1)$. This is because in this case the two input signals originate from the same source, and the probability that g has two different input values is zero. Now consider another case depicted in Figure 1.13(b). Here, $p^*(y_3)$ cannot be directly calculated from its local inputs; *i.e.* $p^*(y_3) \neq p^*(y_1) \times p^*(y_2)$ because both y_1 and y_2 are controlled by x_4. Signal correlation problems of this kind are caused by fanout-reconvergence structures, which are shown with bold lines in Figure 1.13. From a statistical viewpoint, exact signal probabilities are signal probabilities where all correlations are accurately accounted for. This requires global calculations that encompass all fanout structures. However, the complexity of analyzing these structures in a circuit is exponential in circuit size [51], which implies that determination of exact signal probabilities tends to be intractable.

A key advantage of using MC sampling for calculating signal probabilities is that all signal correlations associated with the applied samples are automatically accounted for. By increasing the number of samples, the calculated probabilities can be made as close to the exact values as desired.

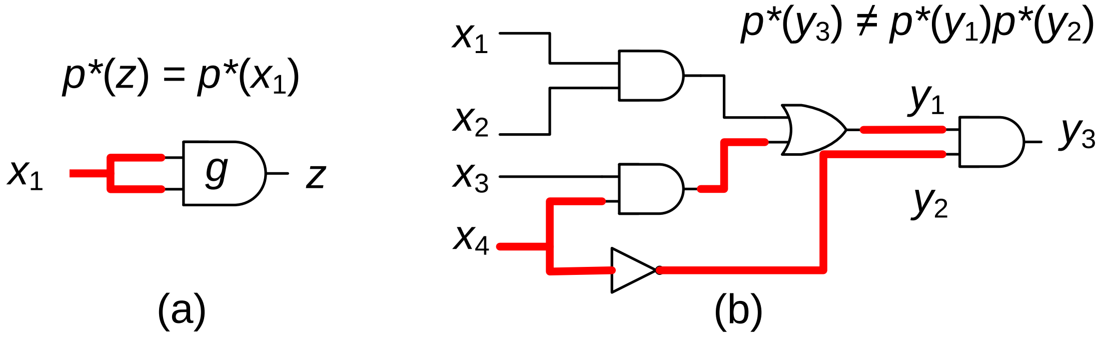
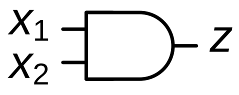


Figure 1.13 Two circuits with fanout-reconvergence structures marked with red bold lines.

1.3 Soft Error Modeling and Analysis

The effect of soft errors on a logic gate is generally described in terms of its *gate-error probability*, which indicates how likely an incorrect value will be generated by the gate due to soft errors. Such error probabilities are typically very small. Several studies indicate that the probabilities of an erroneous bit in one month of operation for 90-nm FPGAs and 65-nm DRAMs are 1.38×10^{-10} and 8.25×10^{-10}, respectively [30] [37].

Two error types, which we refer to as conditional and unconditional, have been used in probabilistic methods to model the soft-error behavior of gates or other components. These have not always been clearly distinguished in the past. A *conditional* error model allows the error probability to vary with respect to different input vectors, while an *unconditional* error model requires the error probability to be constant, and therefore independent of the input vectors. Thus, unconditional models ignore correlations among a gate's inputs.



(a)

Input	Output probabilities	
x_1x_2	$z = 0$	$z = 1$
00	$1 - p_1$	p_1
01	$1 - p_2$	p_2
10	$1 - p_3$	p_3
11	p_4	$1 - p_4$

(b)

Input	Output probabilities	
x_1x_2	$z = 0$	$z = 1$
00	$1 - p_{err}$	p_{err}
01	$1 - p_{err}$	p_{err}
10	$1 - p_{err}$	p_{err}
11	p_{err}	$1 - p_{err}$

(c)

Figure 1.14 (a) Two-input AND gate, and its behavior with (b) a conditional gate error model, and (c) an unconditional gate error model.

Figure 1.14 illustrates the two error models for a two-input AND gate. The table in Figure 1.14(b) shows an example of a conditional error. For a gate g with output z, the conditional output probability of z associated with a particular input vector v is denoted by $p(z|v)$. For example, $p(z|x_1'x_2) = p(z|01) = p_1$ in Figure 1.14(b). Figure 1.14(c) defines an unconditional error model for the two-input AND gate. In this case, the error probabilities associated with all input vectors are the same, namely p_{err}. In general, a conditional error model for an n-input gate g can employ up to 2^n distinct probability values to simulate a soft error. On the other hand, the unconditional model needs only one error probability value p_{err}. A conditional error model can simulate more complicated situations, while an unconditional one has much better scalability.

The parameter p_{err} is the probability that g outputs an incorrect value, which may be 0 or 1. The corresponding signal probability, denoted by $p(g^e)$ is the probability of g outputting 1 when it is subject to errors. In the other words, $p(g)$ and $p(g^e)$ represent g's signal probabilities in the error-free and erroneous cases, respectively. For the two-input AND gate of Figure 1.14

$$p(g^e) = 0.25(p_{err} + p_{err} + p_{err} + (1 - p_{err})) = 0.25(1 + 2p_{err})$$

So, if $p_{err} = 0.1$, then $p(g^e) = 0.3$. We will show how to calculate the erroneous signal probabilities of individual gates later in Chapters 3 and 4.

We now explain how soft-error effects are measured. The *soft-error rate* (SER) is typically used for this purpose, and is defined in terms of gate vulnerability and circuit reliability. The *vulnerability* of a particular gate g represents how easily an erroneous value can be produced by g and can be propagated to the primary outputs. The *reliability* of a circuit *C* indicates how likely an incorrect output caused by *g* will be observed at the primary outputs.

Whether or not an incorrect value (an error) can propagate to the primary outputs is determined by three masking effects: logic, electrical and temporal [45].

- *Logic masking*: all possible propagation paths of an erroneous signal to primary outputs are blocked by the circuit's connectivity, functionality or a combination of both.
- *Electrical masking*: the strength (voltage amplitude) of an erroneous signal gradually attenuates as the erroneous signal propagates, and the error eventually disappears before reaching a primary output.
- *Temporal masking*: An error signal is not stored by a memory element because the erroneous signal fails to satisfy the timing constraints, *e.g.*, the setup or hold times, of the memory element.

Figure 1.15 illustrates the three masking effects mentioned above. Suppose $x_1x_2x_3x_4$ = 0001, making y_1y_2 = 00. If the value of y_1 is flipped to 1 by a soft error, then y_1's erroneous value will be blocked by y_2, as shown in Figure 1.15(a). In addition, observe that x_3 is a redundant signal because $y_3 = x_1x_2x_4'$. Therefore, no erroneous value generated at x_3 can propagate to the primary output. In Figure 1.15(b) an error successfully propagates to the primary output because no other signal can block the erroneous signal in this case. However, the erroneous signal's strength is decreasing as it propagates from y_1 to y_3. If the signal strength falls below a threshold voltage, then an error cannot be captured and stored by the flip-flop (memory element). Finally, Figure 1.15(c) shows how an error fails to be stored by the flip-flop when it does not meet the flip-flop's timing constraints, even though its signal strength does not greatly attenuate before it reaches the flip-flop. To be captured by the flip-flop an error should arrive before the setup time, and keep its logic value stable over the duration of the setup and hold times.

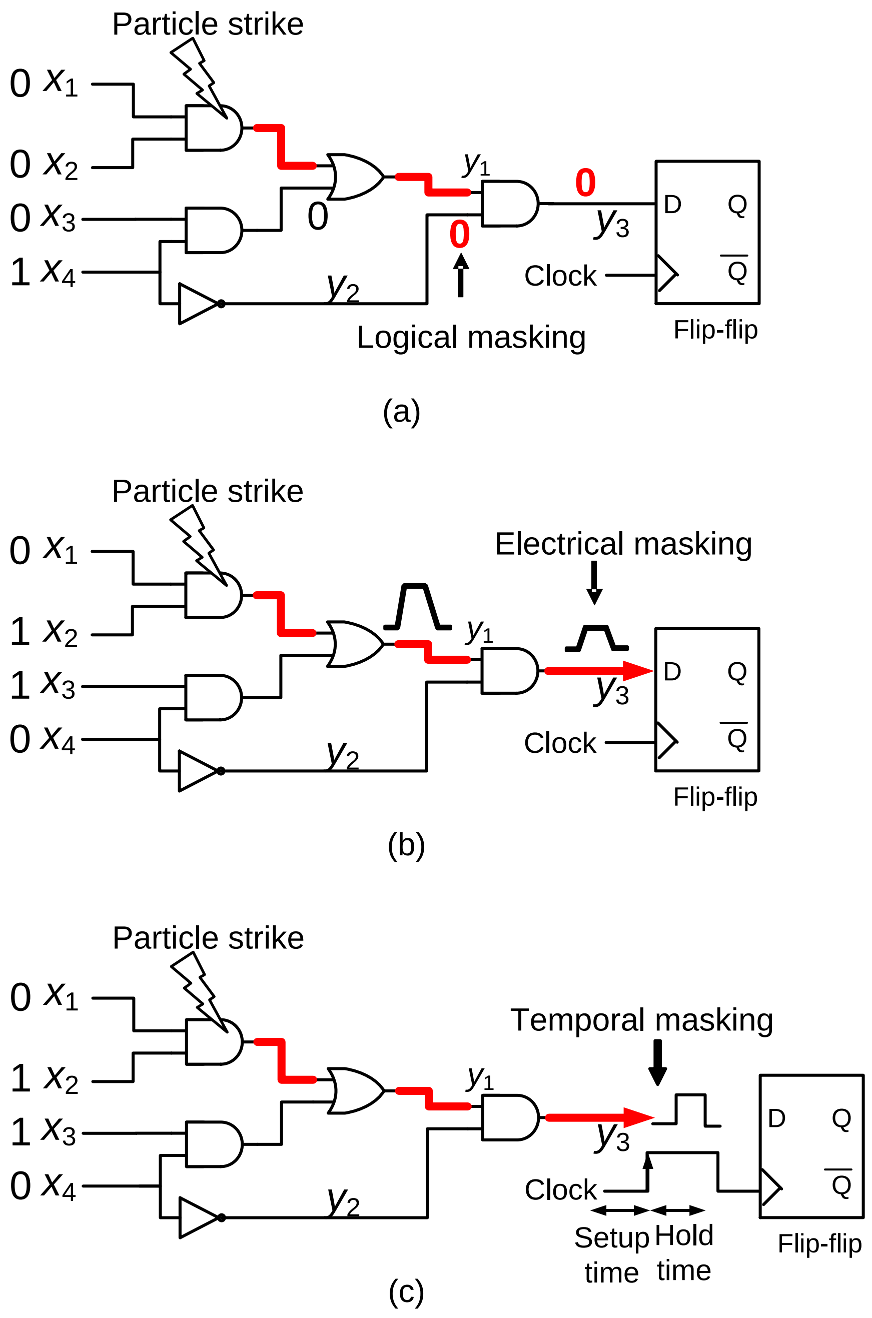


Figure 1.15 Illustration of three masking effects: (a) logical, (b) electrical, and (c) temporal.

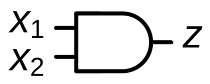
Since our research focuses on evaluating SERs at the logic level, we only consider the logical masking effect because the electrical and temporal masking effects must generally be simulated at the transistor level with considerable computational effort, and are difficult to handle at the logic level.

We next introduce probabilistic transfer matrices (PTMs), which can efficiently represent a circuit's probabilistic behavior described by either conditional or unconditional error models.

1.4 Probabilistic Transfer Matrices

Many probabilistic techniques have been developed for representing and estimating the SERs in logic circuits, such as the PTM approach [34], Bayesian networks (BNs) [48], and probabilistic decision diagrams (PDDs) [1]. We use the PTM approach because not only is it capable of simulating circuits with many different error models, but it also can provide accurate SER results due to the way it handles correlations. We will review and compare these probabilistic techniques later in Chapter 3.

Krishnaswamy et al. were the first to propose PTMs for circuit reliability analysis [34]. PTMs can explicitly include the probability of every input-output combination associated with a logic gate or a combinational circuit. Hence, they provide a general way to represent conditional errors. Figure 1.16 shows the gate PTM for a two-input AND gate whose probabilistic behavior is described in Figure 1.14(b).



(a)

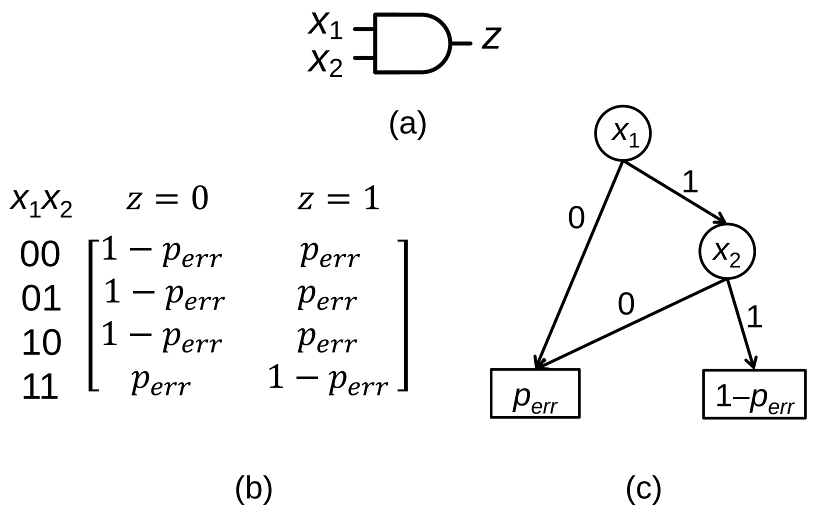
$$\begin{array}{c|cc} x_1x_2 & z=0 & z=1 \\ \hline 00 & 1-p_1 & p_1 \\ 01 & 1-p_2 & p_2 \\ 10 & 1-p_3 & p_3 \\ 11 & p_4 & 1-p_4 \end{array}$$

(b)

Figure 1.16. (a) Two-input AND gate, and (b) its PTM representation with a conditional error model.

Each PTM entry is a conditional probability associated with a corresponding input vector. For example in Figure 1.16(b), the probabilities of z being 1 associated with inputs x_1x_2 = 01 and 10 are $p(z|x_1'x2) = p_2$ and $p(z|x_1x_2') = p_3$, respectively. To reduce memory usage, matrices can be stored in the form of algebraic decision diagrams (ADDs) [10]. Like reduced ordered binary decision diagrams (ROBDDs) [16], ADDs have multiple terminal nodes to represent signal probability values, and provide a compact and canonical way to represent the probabilistic behavior associated with individual input vectors for a gate or an entire circuit.

Again, we use a two-input AND gate to illustrate the use of ADDs to reduce memory needs. Suppose the AND gate's error behavior is described by the unconditional error probability p_{err}. Figure 1.17(c) shows an ADD representation of the AND gate's PTM. A path from node x_1 to a terminal node represents a complete or partial input assignment u, and the linked terminal indicates the output probability of being 1 associated with u. For instance, the path between node x_1 and terminal p_{err} indicates $p(z|x_1') = p_{err}$, without considering x_2's value. If $x_1 = 1$, the ADD needs to use x_2's value to determine the output probability. The path between nodes x_1, x_2 and terminal $1 - p_{err}$ represents $p(z|x_1x_2) = 1 - p_{err}$.

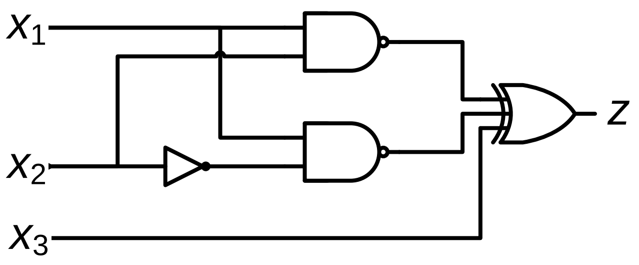


$$\begin{array}{c} x_1x_2 \\ 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cc} z=0 & z=1 \end{array} \\ \begin{bmatrix} 1-p_{err} & p_{err} \\ 1-p_{err} & p_{err} \\ 1-p_{err} & p_{err} \\ p_{err} & 1-p_{err} \end{bmatrix} \end{array}$$

Figure 1.17. (a) Two-input AND gate, (b) its gate PTM with unconditional error p_{err}, and (c) its ADD representation.

For a combinational circuit *C*, the basic PTM construction algorithm first generates a level PTM for each circuit level, and then merges level PTMs to form a circuit PTM. The signal probabilities of the primary outputs can then be obtained from the circuit PTM by matrix operations. All correlations among signals are implicitly accounted for when generating the circuit PTM. Hence, the SER of a circuit can be accurately computed from the circuit PTM by comparing the differences between the error-free and erroneous versions of the circuit's output signal probabilities.

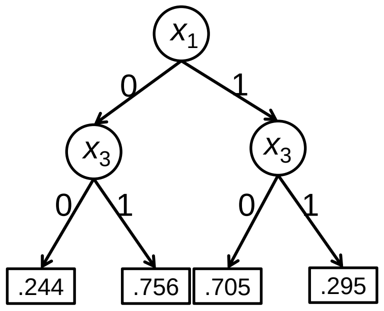
Figure 1.18 shows a three-input circuit, its PTM, and the corresponding ADD, assuming an unconditional gate-error probability $p_{err} = 0.1$. Like a gate PTM, the circuit PTM represents *z*'s signal probabilities associated with all possible input vectors. This ADD form is more compact because it has only seven nodes, whereas the PTM contains 16 entries.



(a)

$$\begin{array}{c|cc}
x_1x_2x_3 & z=0 & z=1 \\
\hline
000 & 0.756 & 0.244 \\
001 & 0.244 & 0.756 \\
010 & 0.756 & 0.244 \\
011 & 0.244 & 0.756 \\
100 & 0.295 & 0.705 \\
101 & 0.705 & 0.295 \\
110 & 0.295 & 0.705 \\
111 & 0.705 & 0.295
\end{array}$$

(b)



(c)

Figure 1.18. (a) Three-input circuit, (b) its circuit PTM for p_{err} = 0.1, and (c) the corresponding ADD.

Although the basic PTM algorithm handles the general conditional error model and accounts for all signal correlations, its practical use is limited to relatively small circuits, as the effort of accurately handling and preserving signal correlations can be huge. In addition, since circuit PTMs only store the conditional probabilities of primary outputs with respect to primary inputs, evaluating signal probabilities for individual gates of a circuit can be difficult. Like ROBDDs [16], the size of the ADDs representing PTMs highly depends on the variable ordering, and determining the best order is an NP-hard problem [34]. In Chapter 4, we develop a novel way to simplify the PTM construction process by combining circuit partitioning with a fast fault simulation technique.

1.5 Thesis Outline

We have illustrated the need for probabilistic analysis in various EDA areas, and have discussed the challenges to applying this approach. This dissertation focuses on two particular issues: (1) providing a general circuit sampling methodology that can generate more accurate signal probabilities than Monte Carlo sampling, and (2) establishing a framework for efficiently and accurately evaluating soft-error effects in logic circuits.

As discussed above, MC plays an important role in probabilistic analysis. It can estimate signal probabilities and provide a gold reference for validating the effectiveness of EDA heuristics. However, MC requires many samples to achieve high accuracy, particularly in large circuits. In Chapter 2, we propose the use of partial signal redundancy and observability to solve this scalability problem. We examine the degree of redundancy in individual signals by analyzing a circuit's structure. This helps to identify unnecessary samples and arrange sample sequences effectively.

Chapter 3 introduces a novel trigonometric model that unifies the representation of signal and unconditional error probabilities in combinational circuits. This unification reduces the computational complexity by mapping multiplications in probabilistic calculations into angular rotations (additions) within trigonometric operations.

In Chapter 4, we develop a PTM-based probabilistic calculation method for SER that is able to simulate a sequential circuit without excessive circuit duplication. We partition the circuit in a way that guarantees that the size of the resultant PTMs is manageable, and apply a fault simulation technique that accounts for signal correlation and masking effects. Since the time-frame expansion technique is not used, our method requires no additional memory when simulating sequential circuits. This property makes our approach capable of tracing the accumulated error effects in a sequential circuit over many simulation cycles.

Finally, we summarize our key contributions and discuss several possible research topics as extensions of this work.

CHAPTER 2

Circuit Sampling for Signal Probability Calculation

In this chapter, we introduce a new circuit sampling methodology for calculating signal probability in error-free combinational circuits. As stated in Chapter 1, MC (Monte Carlo) sampling is a general method of signal probability estimation; however, it has the disadvantage that the sample size grows rapidly with increasing accuracy levels. To improve the scalability of MC, we apply the concept of signal redundancy to circuit sampling. We prioritize primary inputs, and compress the sample space by analyzing the degree of redundancy and observability of individual signals. Our simulation results show that the proposed sampling method can improve the simulation efficiency by from one to three orders of magnitude, even in large benchmark circuits. A paper based on this work has been accepted for presentation at the 2012 International Conference on Computer-Aided Design [69].

2.1 Background

As noted in the preceding chapter, signal probabilities are basic to non-deterministic analysis, and MC is a natural way to calculate signal probabilities in logic circuits. In fact, MC is a widely-used technique in EDA that serves several purposes. It is used as an evaluation method when analytical approaches are infeasible, and it often serves to validate heuristic problem-solving techniques. For example, Sauer et al. present a SAT-based timing analysis technique whose results are “validated by comparison to an exact (Monte Carlo) simulation approach” [52]. Other EDA tasks commonly validated by MC include testability and power estimation [58], circuit synthesis [18], and design verification [57]. In some cases, MC is central to the main solution methodology [70].

However, conventional MC suffers from high simulation complexity particularly in large circuits due to the many samples that need to be applied. In statistics, the simulation efficiency of MC for specific functions can sometimes be enhanced by using the properties of the target functions [27]. Recently, several sampling methods for analog circuits have been proposed [31][56] for improving sample accuracy. For instance, Singhee and Rutenbar apply a quasi-Monte Carlo (QMC) technique [56] for simulating the process variation effects. However, these algorithms cannot be directly applied to digital circuits, because analog circuits are modeled by continuous functions, while digital circuits are described by Boolean functions with discrete binary values.

In this chapter, we will introduce a general sampling methodology aimed at signal probability analysis for logic circuits. Prior work on logic circuit sampling just generates random samples without considering the functionality and connectivity of the target circuit. Our proposed method can provide more accurate results by removing unnecessary samples, and by arranging the samples based on the degree of importance of the input signals.

In the next section, we briefly review several representative probability calculation methods to clarify the advantages and limitations of prior work, and the need for developing new probability calculation techniques, such as sampling in logic circuits.

2.2 Prior Work

As mentioned in Chapter 1, Parker and McCluskey proposed a method [46] that can calculate exact signal probabilities for logic circuits. Given a circuit *C*, the Parker-McCluskey method traverses *C* from primary inputs to primary outputs, and calculates the signal probabilities of *C*'s individual gates using the rules shown in Table 2.1.

Table 2.1 Signal probability calculation rules for an *n*-input elementary gate $z(x_1, x_2, \ldots, x_n)$ with independent inputs.

Gate type	**Probability calculation rule**
Primary input x_i	$p^*(x_i) = 0.5$
NOT	$p^*(z) = 1 - p^*(x)$
AND	$p^*(z) = \prod_{i=1}^{n} p^*(x_i)$
NAND	$p^*(z) = 1 - \prod_{i=1}^{n} p^*(x_i)$
OR	$p^*(z) = 1 - \prod_{i=1}^{n}(1 - p^*(x_i))$
NOR	$p^*(z) = \prod_{i=1}^{n}(1 - p^*(x_i))$

If a gate *g*'s inputs are not independently controllable from the primary inputs of *C*, then the signal probability of *g*'s output cannot be directly calculated from *g*'s inputs alone. To account for correlations, the Parker-McCluskey method calculates *g*'s signal probability in terms of the signal probabilities of *C*'s primary inputs. In other words, if *g*'s Boolean function is $f(g) = f(x_1, x_2, \ldots, x_m)$, then $p(g)$ is expressed in terms of $p(x_1)$, $p(x_2)$, …, $p(x_m)$, and correlations among gate *g* can be accounted for when $p(g)$ is accurately expressed with $p(x_i)$.

Consider the circuit in Figure 1.13(b), which is copied in Figure 2.1. The exact output signal probability $p(y_3)$ is expressed as

$$p^*(y_3) = (1 - (1 - p^*(x_1)p^*(x_2))(1 - p^*(x_3)p^*(x_4)))(1 - p^*(x_4))$$

which can be simplified to

$$p^*(y_3) = p^*(x_1)p^*(x_2)(1 - p^*(x_4)) = 0.125$$

It is worth noting that in this case, x_3 is a redundant signal because the Boolean function $y_3 = x_1x_2x_4'$ is independent of x_3. Hence $p^*(x_3)$ is irrelevant to $p^*(y_3)$.

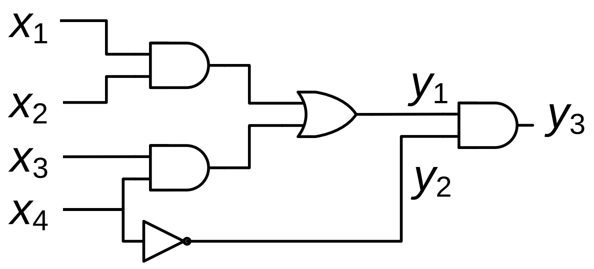


Figure 2.1. Four-input circuit (copy of Figure 1.13(b)).

Although the Parker-McCluskey method can compute exact signal probabilities, the complexity of completely accounting for all signal correlations makes it exponential in circuit size [46], and therefore impractical for large circuits.

Several heuristic approaches have been developed to reduce the complexity of handling signal correlations. We review three representative heuristic calculation approaches, namely the controllability/observability program (COP) [14], the correlation coefficient method (CCM) [25], and the Boolean approximation method (BAM) [58]. These address the signal-correlation problem by only considering correlations at individual topological levels, or by ignoring cases considered to have insignificant impact. We discuss these three heuristics because they have been adopted in several different application areas, including testability [14], switching activity [59], and reliability analysis [19][68]. Their accuracy is measured by comparing their signal probability results to the results generated by MC simulation [14][25][58].

Brglez developed a fast probability calculation method COP [14] that ignores correlation. COP therefore assumes all gate inputs in a circuit are independent, which can lead to huge calculation inaccuracy [25]. Again, taking the circuit in Figure 2.1 as an example, y_3's exact signal probability and its signal probability from the COP method are 0.125 and 0.21875, respectively, making the calculation inaccuracy 75%.

Ercolani et al. [25] developed the CCM method, which calculates signal probabilities by only considering correlations within a single topological level. Given a circuit C, CCM first levelizes C, and then accounts for signal correlations among each pair of signals at the same level. This method reduces the computational complexity of handling correlation from exponential to quadratic. In the case of Figure 2.1, the total number of correlation cases that need to be enumerated is $\binom{4}{2} + \binom{3}{2} = 9$. CCM is exact for circuits with simple structures, such as the circuit in Figure 2.1, even though it does not account for all possible correlations.

The complexity of the signal correlation problem can be further reduced by carefully analyzing circuit structure. In Figure 2.1, for example, x_4 is the only cause of this problem because its signal probability can propagate to the primary output along different paths. Uchino et al. [58] have proposed another approach called BAM that estimates a gate's correlations associated with individual primary inputs. In the Figure 2.1 case, while CCM accounts for the correlations of every pair of signals at each level, BAM recognizes x_4 is the only signal with multiple branches, and then only considers correlation when evaluating $p(y_3)$. This is because y_3 is the only signal affected by x_4 via multiple paths. In this case, like CCM, BAM yields the exact signal probabilities. It is worth noting that unlike COP and CCM, Uchino et al. [58] not only derive the complexity of BAM, but also provide an analytical upper bound of its calculation inaccuracy. Such information is crucial when we need to make a fair comparison among multiple heuristic approaches.

Although the foregoing heuristics provide fast signal probability estimation, their calculation accuracy varies from circuit to circuit. In addition, some methods [14][25] are not based on well-defined mathematical models, but are just ad-hoc solutions. Hence, measuring the quality of these methods e.g., identifying analytical bounds on their calculation inaccuracy, is difficult or even impossible.

Unlike the aforementioned heuristic approaches, MC sampling is a well-studied statistical technique whose simulation accuracy is manageable and is fully determined by the number of applied samples. Since MC can provide highly accurate results if necessary, it is suitable for validating the performance of heuristics. In the next section, we study the use of signal redundancy for simplifying the sampling complexity in logic circuits.

2.3 Signal Redundancy

We now propose ways to exploit a circuit's redundancy properties to speed up signal probability estimation. To illustrate, consider the circuit in Figure 1.10, which is copied in Figure 2.2, and realizes $z(x_1,x_2,x_3,x_4) = x_1'x_2 + x_1x_3$. Input x_4 is *redundant* in the usual sense that z is independent of x_4, a fact that can be exploited to reduce sampling effort. While exhaustive simulation of a four-input circuit requires 16 samples, the *full* redundancy of x_4 reduces the circuit to a three-input one that can be exhaustively simulated with only 8 samples. The sample input vectors shown on the figure apply all possible combinations to the non-redundant inputs x_1, x_2 and x_3, so the simulated value of $p(z)$ is exact.

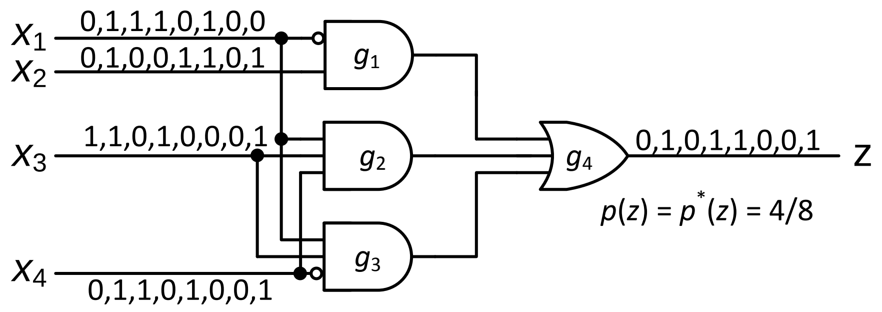


Figure 2.2. Circuit illustrating signal probability estimation via sampling; (copy of Figure 1.10).

Of course, fully redundant inputs such as x_4 in the above example rarely occur in practice. We address a new type of *partial redundancy* where independence on input variables is conditional on specific values applied to other variables. For instance, z in Figure 2.2 becomes independent of x_2 and x_3, when $x_1 = 1$ and 0, respectively. Hence, x_2 and x_3 can be considered partially redundant with respect to x_1. Most of a circuit's primary inputs are partially redundant; however, the degree of their redundancy varies widely. Some inputs become redundant much more often than others. We will show how such partial redundancies can be exploited to speed up simulation.

We propose an extension to the MC approach we call *Reduced-Ordered Monte Carlo* (ROMC), which has better performance than conventional MC. It incorporates two main techniques to improve the quality of circuit simulation. The first technique orders and prioritizes the primary inputs according to their *observability*. For instance, consider the problem of estimating the signal probability of z_1 in Figure 2.3. Some primary inputs such as x_{10} are more observable at z_1 than others, and some, such as x_2, are difficult to observe. Low-observability primary inputs tend to become partially redundant due to other primary input assignments, while high-observability primary inputs are unlikely to become redundant. The concepts of observability and redundancy discussed here are loosely related to the influence property of Boolean functions [33], and observability as used in the testing context [15]. We employ the SCOAP testability measurement heuristic [15] to quickly find the most observable primary inputs.

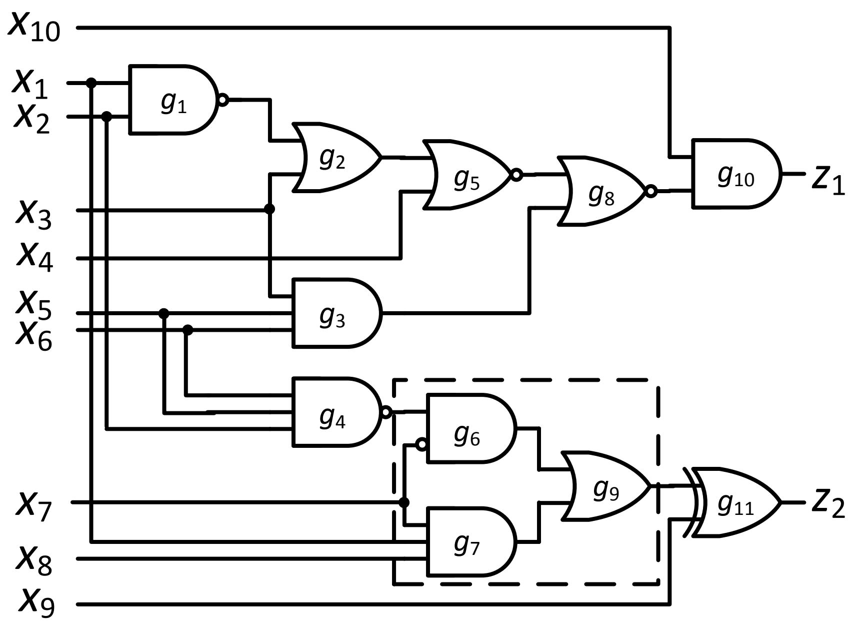


Figure 2.3. Ten-input, two-output circuit; a multiplexer-like structure formed by g_6, g_7 and g_8 is highlighted.

The second technique involves finding *compatible* primary inputs, which have the property that only one of the primary inputs can affect certain primary outputs at any time. This relation among input variables can also be interpreted as a kind of partial redundancy. For example, x_4 and x_5 in Figure 2.3 are compatible primary inputs with respect to primary output z_1. If $x_3 = 1$ (0), then z_1 becomes independent of x_4 (x_5). So both x_4 and x_5 are partially redundant primary inputs associated with x_3, and they do not affect output z_1 at the same time. Such behavior resembles that of a multiplexer, whose control (select) signals ensure that only one data input D_i is connected to the multiplexer's output at a time; the others are effectively masked. The data inputs $\{D_i\}$ are viewed here as compatible. More generally, we find compatibility relations among the inputs of multiplexer-like structures. One such structure is marked by dashed lines in Figure 2.3, where x_1 and x_8 are compatible with respect to x_2, x_5 and x_6. We will show that compatible primary inputs can share resources, such as input samples or randomness sources that drive the samples. This can lead to large efficiency gains in signal probability estimation.

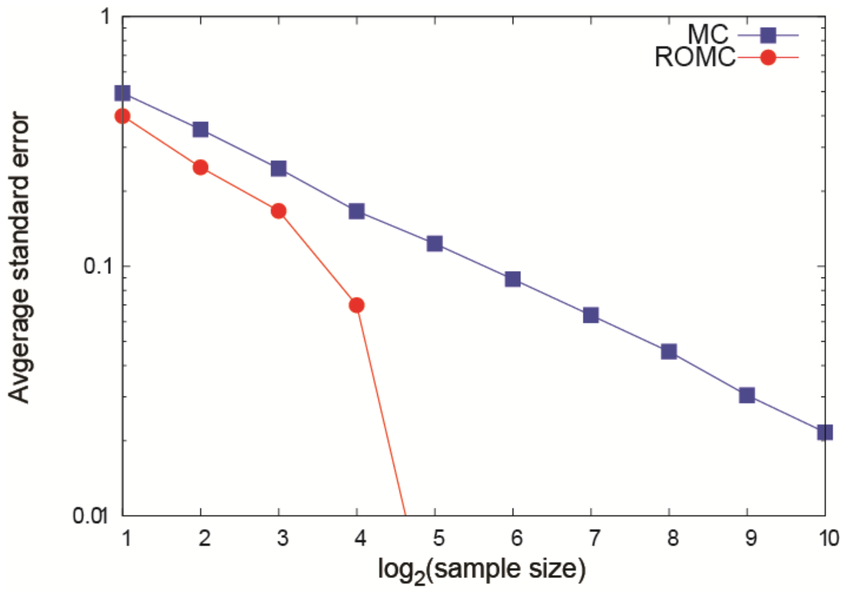


Figure 2.4. Accuracy comparison between ROMC and a typical MC calculation of the average output signal probability for the circuit in Figure 2.3.

Figure 2.4 illustrates the accuracy improvement from applying ROMC to the circuit in Figure 2.3. Here, accuracy is measured in terms of standard error [27], and refers to the difference between the exact probability $p^*(z)$ and the probability $p(z)$ estimated by the simulation for various sample sizes; the errors are averaged over the two outputs z_1 and z_2. For a given accuracy level, ROMC is significantly faster than MC because it achieves the required accuracy with fewer samples. Furthermore, in this case, ROMC can produce exact signal probabilities with only 32 samples.

2.4 Sampling Concepts

This section reviews some basic concepts that are needed later. First, we formalize the signal probability estimation problem introduced in the previous section, and then discuss some basic properties of MC simulation. Lastly, we review the Boole-Shannon expansion.

We formulate signal probability computation in general terms, so that it applies to many different EDA situations. The *signal probability estimation problem* may be stated as follows. Given a combinational circuit C (including the pseudo-combinational equivalent of a sequential circuit) with n primary input and m primary output signals, as well as a set of N sample input sequences, calculate the (average) signal probabilities of the primary outputs. Figure 2.2 illustrates this for $n = 4$, $m = 1$ and $N = 8$. We make the usual assumption [25][58] that each primary input x_i has signal probability $p(x_i) = 0.5$ and is independent of all other primary inputs. This means that their joint probability distribution is the product of their individual distributions, and captures the informal notion of "random" sampling.

Consider a n-input, single-output circuit C that implements the Boolean function $z(x_1,x_2,\ldots,x_n)$, where the primary inputs have the same signal probability $p(x_i) = 0.5$ and are all independent. The signal probability estimation problem for C is to determine $p^*(z)$.

Since it is usually not feasible to find the exact value of $p^*(z)$ if C is large, we aim to find an *estimator* probability $p(z)$ that provides a good estimate of $p^*(z)$. For a given estimator $p(z)$, a bias β and a variance $var[p(z)]$ are defined as follows [27]:

$$\beta = E[p(z) - p^*(z)] \tag{2.1}$$

$$var[p(z)] = E[(p(z) - p^*(z) - \beta)^2] \tag{2.2}$$

where $E[\boldsymbol{X}]$ denotes the expected value of $\boldsymbol{X}$. The square root of the variance of an estimator is its *standard error*. It is usually desirable for an estimator to have $\beta = 0$, in which case it is called *unbiased*, and to have a small variance [27]. An unbiased estimator with variance zero is exact, that is, $p(z) = p^*(z)$.

MC is a method of estimating $p^*(z)$ by applying N uniform and independent random samples to the primary inputs and collecting N samples at the output z. Let $z(i)$ denote the ith value of z associated with the ith sample. The estimator $p(z)$ can be defined as

$$p(z) = (1/N)\sum_i z(i)$$

It can be shown that $p(z)$ in this case is unbiased [27] and that the variance of the estimation is

$$var[p(z)] = (1/N)\, p^*(z)\, (1 - p^*(z)) \tag{2.3}$$

So the variance and the standard error of an MC simulation can be reduced by increasing the sample size N. However, reducing the standard error by a factor of k requires the sample size to be increased by factor of k^2 making it impractical if the desired error level is low. Consequently, many techniques for variance reduction without increasing N have been proposed [27][49]. The method proposed in this dissertation can be seen as another variance reduction technique.

Although the input samples for MC are usually assumed to be independent, which allows samples to be repeated, it is also possible to use MC with non-repeating samples

[49]. We call the former *MC with sample replacement* (MCW) and the latter *MC without sample replacement* (MCWO). Like MCW, MCWO is an unbiased estimation method, but its variance follows that of a hypergeometric distribution [49]. For an MCWO estimator $p(z)$

$$E[p(z)] = p^*(z)$$

$$var[p(z)] = (1/N)p^*(z)(1 - p^*(z))\,(2^n - N)/(2^n - 1)$$

Recall that n denotes the number of the primary inputs in C. For a fixed sample size N, if the number of primary inputs n is reduced by some number, then the variance will definitely become smaller. The change can be significant if N is comparable to 2^n. This is not true in the case of MCW since its variance is independent of n. This fact turns out to be important in our proposed method since we aim at reducing the sample space of the MC, and can benefit from it only if we use non-repeating samples.

Finally, we summarize the Boole-Shannon expansion and provide a compact version for later use. An n-variable Boolean function $F(X)$ can be expanded around any variable x_i according to the following formula [26]

$$F(x_1, x_2,.., x_i,.., x_n) = x_i'\cdot F_{xi'} + x_i\,.F_{xi}$$

where $F_{xi'} = F(x_1, x_2,..,0,..,x_n)$ and $F_{xi} = F(x_1, x_2,..,1,..,x_n)$ denote the negative and positive *cofactors* of F, respectively, with respect to x_i.

For example, consider the function generated by g_8 in Figure 2.3. It can be expanded around x_3 thus:

$$g_8(x_1,x_2,x_3,x_4,x_5,x_6) = x_3'((x_1x_2)' + x_4) + x_3(x_5x_6)'$$

The negative and positive cofactors are $(x_1x_2)' + x_4$ and $(x_5x_6)'$, respectively. The variable x_4 (x_5) does not appear in the positive (negative) cofactor due to partial redundancy; the value of x_3 always blocks the propagation path from x_4 (x_5) to g_8.

As this example suggests, Boole-Shannon expansions and cofactors provide a tool for describing signal compatibility. We can also express signal probabilities in terms of cofactors:

$$p^*(F) = \frac{1}{2^k}\sum_0^{2^k-1} p^*(F_i) \tag{2.4}$$

It is also useful to define the Boole-Shannon expansion with respect to a product of k variables $X_k \subseteq X$. In this case, we are dealing with cube cofactors [26] such as $F_{x1'x2'\ldots xk'}$ and $F_{x1x2\ldots xk}$. With this notation, cofactors are awkward to write, so we denote $F_{x1'x2'\ldots xk'}$ by F_0, $F_{x1'x2'\ldots xk}$ by F_1, and $F_{x1x2\ldots xk}$ by $F_{2^{k-1}}$. The cube expansion around X_k can then be expressed compactly as

$$F(X) = x_1'x_2'\ldots x_k'F_0 + x_1'x_2'\ldots x_kF_1 + \ldots + x_1x_2\ldots x_k F_{2^{k-1}} \tag{2.5}$$

2.5 Variable Ordering

As discussed in Section 2.3, fully redundant primary inputs can be exploited to reduce the sample size for simulation, simply by not wasting samples on them. Similarly, the partially redundant, or less observable, primary inputs that become redundant frequently can be ignored in favor of more observable ones. (Of course, we cannot ignore them completely like redundant primary inputs.)

Accordingly, we propose the following sampling approach. Suppose we want to use $N = 2^k$ samples to estimate the signal probability $p(F)$ of function F with an n-member input set X. Select k variables $X_k = x_1, x_2,..., x_k \subseteq X$ that are more observable than the others. Generate N samples that include every possible value of X_k exactly once, and assign random values to the remaining $n - k$ variables. The following samples illustrate this for $k = 3$ and $n = 9$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
0	0	0	0	1	0	1	1	0
0	0	1	1	0	1	1	0	1
0	1	0	0	1	0	1	1	1
0	1	1	0	1	1	1	0	0
1	0	0	1	1	1	0	1	0
1	0	1	1	0	1	1	1	0
1	1	0	1	0	0	1	0	1
1	1	1	1	1	0	1	0	0

The Boole-Shannon expansion (2.5) enables us to express the above sampling scheme in the following way. Estimate the signal probability of each of the cofactors F_0, $F_1,\ldots, F_{N\text{-}1}$ with one sample and average the estimates. In practice, estimating the cofactors with one sample is sufficient. For instance, with a good variable ordering, the cofactors will be close to 1 or 0, so a single sample provides a very good estimate.

$$p(F) = (1/N)(\, p(F_0) + p(F_1) + \cdots + p(F_{N\text{-}1})\,) \tag{2.6}$$

The average estimate $p(F)$ is unbiased. Its variance is therefore

$$var[p(F)] = (1/N^2)(var[p(F_0)] + var[p(F_1)] + \ldots + var[p(F_{N\text{-}1})]) \tag{2.7}$$

The variance of each cofactor estimate $p(F_i)$ is that of conventional MC with one sample, so

$$var[p(F_i)] = p^*(F_i)(1 - p^*(F_i))$$

Now consider two exemplary cases. First, suppose that $p^*(F_i) = 0.5$ for $i = 0, 1, 2, ..., N - 1$; then $var[p(F_i)] = 1/4$. In this case, according to (2.7) we have $var[p(F)] = 1/4N$, which means that the proposed approach is no better than conventional MC. Next, suppose that $p^*(F_i) = 0$ for $i = 0, 1, 2, ..., (N/2) - 1$ and $p^*(F_i) = 1$ for $i = N/2, (N/2) + 1, (N/2) + 2, ... , N - 1$. This time $var[p(F_i)] = 0$, so $var[p(F)] = 0$; in other words, the estimate in this case is exact. The difference between the two cases sketched above is that in the first one, $x_1, x_2, ..., x_k$ have high redundancy and low observability at the output F. This is why the corresponding cofactor has probability 0.5. In the second case, however, $x_1, x_2, \ldots, x_k$ have high observability at F, and after assigning a value to them, completely determine the value of F.

Thus, for a given function F, we need to look for k variables that minimize $var[p(F)]$. This means we have to consider the possible choices, calculate the probability of the corresponding cofactors, and calculate $var[p(F)]$. If all choices must be considered, this problem is more difficult than exhaustive simulation, and hence is not feasible in practice. We therefore use heuristic observability evaluation methods to find the most observable variables.

Consider the circuit in Figure 2.5. The output z has exact probability $p^*(z) = 0.5$. If we try to estimate this value with a four-sample simulation, the MC method yields a variance of $1/16 = 0.063$ according to (2.3). Now apply the proposed technique to this example. Since we have a total of four samples, we need to find the two most observable variables, which are obviously a and b. The variance of the estimation is given by (2.7) thus:

$$var[p(F)] = (1/16)(var[p(F_{a'b'})] + var[p(F_{a'b})] + var[p(F_{ab'})] + var[p(F_{ab})])$$

= (1/16)($var[p(cd)] + var[p(1)] + var[p(c'+d')] + var[p(0)]$) = 0.023

which implies that this technique yields a better estimate.

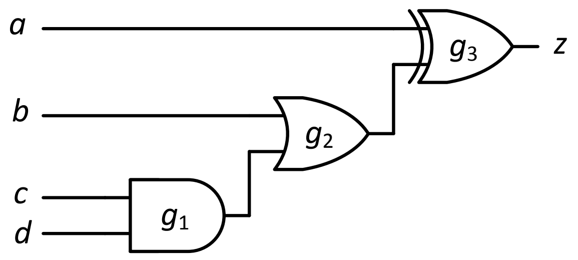


Figure 2.5. Variable ordering; *a* and *b* are more observable than *c* and *d*.

2.6 Sample Space Reduction

Next, we introduce a technique to shrink a circuit's sample space by converting it to another circuit with fewer primary inputs but the same output probabilities. As discussed in Section 2.4, this sample space reduction results in better signal probability estimates without the need to increase the sample size.

Consider again the function $z(x_1,x_2,x_3,x_4) = x_1'x_2 + x_1x_3x_4 + x_1x_3x_4'$ of Figure 2.2 The redundant primary input x_4 allows the sample space to be reduced from 16 to 8. Such redundant primary inputs can be identified by their Boolean difference. The *Boolean difference* of $z(x_1, x_2,\cdots, x_i, \cdots, x_n)$ with respect to input x_i is defined as

$$\frac{dz}{dx_i} = z(x_1, x_2, .., 0, .., x_n) \oplus z(x_1, x_2, .., 1, .., x_n)$$

where $\oplus$ denotes XOR. As stated first by Akers [5], it follows immediately that x_i is redundant if and only if

$$\frac{dz}{dx_i} = 0 \tag{2.8}$$

We want to show that partially redundant primary inputs also reduce the sample size, and can be detected in a similar way. Consider again the partially redundant inputs in Figure 2.2. Input x_2 (x_3) becomes redundant if $x_1 = 1$ (0), so at least one of them is always redundant in the sense that x_2 and x_3 do not affect the output at the same time. Knowing this, we can further reduce z's sample space by tying x_2 and x_3 together to form a single primary input for simulation purposes. This reduces the number of samples for exhaustive simulation from eight to four, namely, $x_1x_2x_3x_4$ = 0000, 0110, 1000, 1110, while z's signal probability remains unchanged. So just by obtaining the output values for these few samples, we can determine the exact signal probability $p^*(z)$.

In general, two partially redundant inputs x_i and x_j of a function z that never affect z together have the following property in terms of the Boolean difference:

$$\frac{dz}{dx_i}.\frac{dz}{dx_j} = 0 \qquad (2.9)$$

We call such inputs *compatible*. For example, inputs x_2 and x_3 of $z(x_1,x_2,x_3,x_4) = x_1'x_2 + x_1x_3x_4 + x_1x_3x_4'$ are compatible because

$$\frac{dz}{dx_2}.\frac{dz}{dx_3} = x_1'.x_1 = 0$$

Theorem 2.1: If two inputs x_i and x_j of an n-input function $f(x_1,x_2,.., x_i,..,x_j,..,x_n)$ are compatible, then the $(n - 1)$-input function $g = f(x_1,x_2,.., x_i,..,x_i,..,x_n)$ obtained by equating x_j and x_i in f has the same signal probability as f, that is, $p^*(g) = p^*(f)$

Proof: Equation (2.9) implies that for all the possible values of the variables other than x_i and x_j, we have either $\frac{df}{dx_i} = 0$ or $\frac{df}{dx_j} = 0$. Hence, $f_{x'_i} = f_{x_i}$ or $f_{x'_j} = f_{x_j}$. Applying

Boole-Shannon expansion to these equations yields $f_{x'_i x_j} = f_{x_i x_j}$ and $f_{x'_i x'_j} = f_{x_i x'_j}$, or $f_{x_i x'_j} = f_{x_i x_j}$ and $f_{x'_i x'_j} = f_{x'_i x_j}$. Therefore,

$$p^*\left(f_{x_i x_j}\right) + p^*\left(f_{x'_i x'_j}\right) = p^*\left(f_{x_i x'_j}\right) + p^*\left(f_{x'_i x_j}\right)$$

Also, according to (2.4) we have

$$p^*(f) = \frac{1}{4}\left[p^*\left(f_{x'_i x'_j}\right) + p^*\left(f_{x'_i x_j}\right) + p^*\left(f_{x_i x'_j}\right) + p^*(f_{x_i x_j})\right]$$
$$= \frac{1}{2}\left[p^*\left(f_{x'_i x'_j}\right) + p^*(f_{x_i x_j})\right]$$

Since g is the same as f with x_i and x_j connected, we can write $g_{x_i} = f_{x_i x_j}$ and $g_{x'_i} = f_{x'_i x'_j}$. Consequently,

$$p^*(g) = \frac{1}{2}\left[p^*(g_{x'_i}) + p^*(g_{x_i})\right] = \frac{1}{2}\left[p^*\left(f_{x'_i x'_j}\right) + p^*(f_{x_i x_j})\right] = p^*(f) \quad (2.10)$$

So tying together a pair of compatible primary inputs yields a circuit with fewer inputs but the same signal probability. More generally, an input set $\mathbb{X}$ of a function forms a *compatible set*, if all x_i-x_j pairs in $\mathbb{X}$ are compatible. All the members of $\mathbb{X}$ can be tied together without altering signal probabilities. This is an instance of a general compatibility relation [26], and so is reflexive, symmetric but not necessarily transitive. It can be shown that the ten primary inputs of Figure 2.3 include 18 compatible pairs and four compatible triples, but no larger ones.

Determining compatible sets with special properties is a difficult task that arises often in EDA, for example, in state minimization for finite-state machines, and resource scheduling for high-level synthesis [26][39]. These problems can be modeled by *compatibility graphs*, or their complements, *conflict graphs*. The problem of interest here is finding a minimal number of compatible sets to cover all primary inputs and minimize the relevant sample space, where the chosen sets must form a disjoint cover (partition) of the primary inputs. This problem is related to the clique partitioning problem in graph theory [39].

There is usually more than one optimum solution to our primary input partitioning problem, and they all reduce the sample space equally. However, some partitions provide better simulation accuracy due to differences in their observability properties. For the circuit of Figure 2.3, our proposed method ROMC selects the compatible sets $\{x_1, x_5\}$,$\{x_2, x_8\}$,$\{x_3, x_7\}$,$\{x_4, x_6\}$,$\{x_9, x_{10}\}$, which leads to a five-input circuit with the same output signal probabilities as the 10-input original; see Figure 2.6. Other nearly equivalent solutions exist such as: $\{x_1,x_5\}$, $\{x_2,x_6,x_8\}$, $\{x_3,x_7\}$, $\{x_4, x_9\}$, $\{x_{10}\}$.

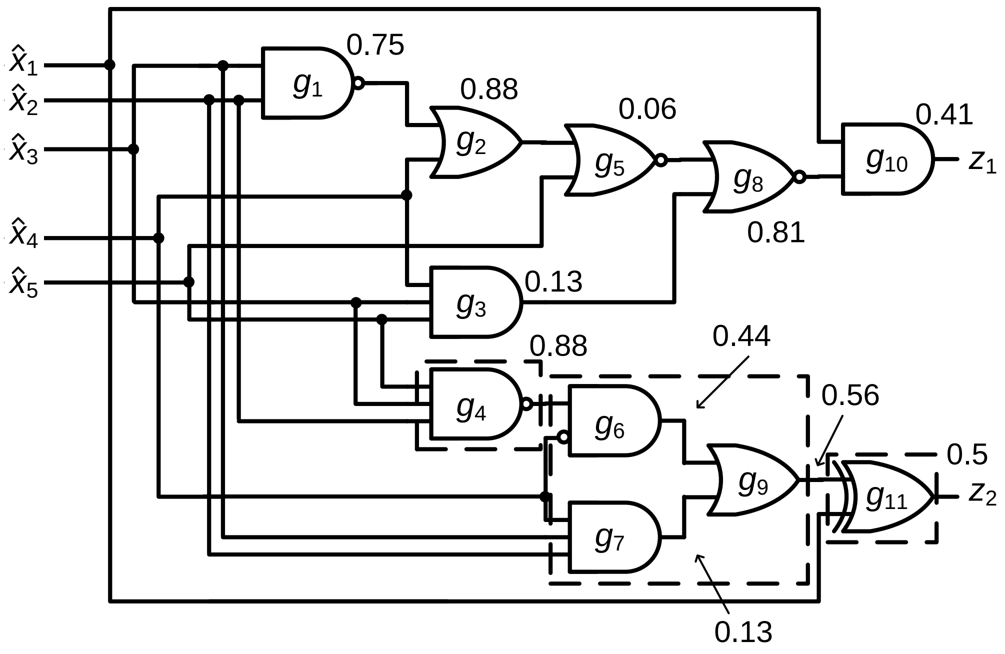


Figure 2.6. Five-input circuit obtained from Figure 2.3 by connecting compatible primary inputs; the supergates of z_2 are marked by dotted lines.

2.7 Implementation Issues

This section presents our circuit sampling algorithm Reduced-Ordered Monte Carlo (ROMC), which incorporates the variance and sample-space reduction techniques introduced in Sections 2.5 and 2.6. Figure 2.7 shows the pseudo-code for ROMC, as well as some of its main procedures.

Given an n-input m-output circuit C and a sample size $N = 2^k$, ROMC first finds an observability value for each primary input via a modified SCOAP algorithm [15]. The modifications arise from differences between observability in the testing context and ROMC's. First, a fanout stem's SCOAP observability is the minimum value among its fanout branches; this is replaced by the average value of the branches' observability. For testing purposes, observing a signal at one primary output is enough, but for ROMC, a good observable signal is one that is observable at many primary outputs. A second modification is that SCOAP calculates the worst-case observability for both the 0 and 1 values of a signal, whereas ROMC computes the average of these values.

Next, ROMC determines all compatibility relations among the primary inputs and constructs a primary input compatibility graph. The **Partition_to_Supergates** procedure searches for multiplexer-like structures in order to detect compatible primary inputs. These structures are relatively easy to recognize, and are naturally confined to the given circuit *C*'s supergates, which are sub-circuits that encapsulate maximal fanout-reconvergence structures. Of course, it is possible that the entire original circuit is a supergate, in which case no simplification results from supergate partitioning. The **Compatible_Pair_Detection** procedure partitions *C* into supergates and recursively searches them for compatible signals. For instance, there are three supergates for primary output z_2 in Figure 2.6, namely: $SG(g_{11}) = \{g_{11}\}$, $SG(g_9) = (g_9, g_7, g_6)$, and $SG(g_4) = \{g_4\}$. $SG(g_9)$ is actually a multiplexer, so some of its inputs are compatible. These compatibility relations are propagated toward the primary inputs and lead to the conclusion that x_8 is compatible with x_5 and x_6.

```
ROMC(Circuit C, primary inputs X, Sample size N = 2^k) {
        // Returns estimated signal probabilities of the primary outputs
        Observability Values X_obs. = Observability_Evaluation(C, X)
        Edge List E = Compatible_Pair_Detection(C, X)
        Graph G = G(X, E)     // G is the compatibility graph of C;
          vertices are primary inputs, edges connect compatible primary inputs
        Reduced primary input Set X̂ = Clique_Partitioning(G)
        Observability Values X̂_obs = Add X_obs. of connected primary inputs in X̂
        Ordered List L = Sort primary inputs in X̂ by their observability X̂_obs.
        Sample Set S = All combinations of first k primary inputs in L,
                     with random values assigned to remaining primary inputs
   return Monte_Carlo(C, S)   // Returns signal probabilities generated by
             applying Monte-Carlo simulation to C with sample set S
     }

Observability_Evaluation(Circuit C, primary inputs X){
        // Returns observability value of each primary input
        return Modified_SCOAP(C, X)
     }

Compatible_Pair_Detection (Circuit C, primary input X){
        // Returns list of compatible primary input pairs
        Supergate Partition SGP = Partition_to_Supergates(C)
                 // Partitions C into fanout-free network of subcircuits
   if (SGP = C) then    // Identify the compatible signals in C's primary inputs
           for each primary input pair (x_i, x_j) in X
                  if (x_i and x_j are compatible) then
                         add (x_i, x_j) to list L
              return L
       else   //Recursively identify the compatible signals in C's supergates
              for each subcircuit C′ in SGP
                  L_i′  = Compatible_Pair_Detection(C′, X ′)
                      // Recursive call to supergate subcircuits
                  for each pair (x_i, x_j) in L_i′
                    propagate the relation to the primary inputs
                        add compatible primary inputs to list L
                      // Combine recursive results
   return L
}
```

Figure 2.7. Pseudo-code for the ROMC simulation algorithm.

After constructing the compatibility graph G for the original primary input set X, ROMC performs clique partitioning on G to find a reduced set $\hat{X}$ of $\hat{n} \leq n$ primary inputs that form a primary input cover. Equivalently, one could construct the primary input's conflict graph $\bar{G}$, and solve the coloring problem for $\bar{G}$, i.e., find the minimum number of vertex colors such that all adjacent vertices have different colors [39]. There are many fast heuristic algorithms to solve this well-studied problem [13]. Figure 2.8 shows the conflict graph of the circuit in Figure 2.3 and its coloring solution. The primary inputs (vertices) with the same color label are those that are tied together in Figure 2.6.

Next, ROMC sorts the primary inputs according to their observability. The observability value of each newly formed primary input $\hat{x}_i$ is the sum of the observability values of the corresponding original primary inputs. For instance, in Figure 2.6, x_9 and x_{10} are replaced by a new primary input $\hat{x}_1$, hence $\hat{x}_1$'s observability is the sum of those of x_9 and x_{10}. The new primary inputs are then ordered by observability; in the case of Figure 2.6, the ordering is $\hat{x}_1\ \hat{x}_4\ \hat{x}_3\ \hat{x}_2\ \hat{x}_5$.

At this point, ROMC generates samples based on its compatibility and observability figures. For the example (Figure 2.6), if the sample size $N = 8$, then primary inputs $\hat{x}_1$, $\hat{x}_4$ and $\hat{x}_3$ are chosen as the most observable. ROMC applies all 8 combinations 000, 001,...,111 to these three primary inputs, and assigns random bit sequences to $\hat{x}_2$ and $\hat{x}_5$. It simulates the circuit with the resulting 8 samples.

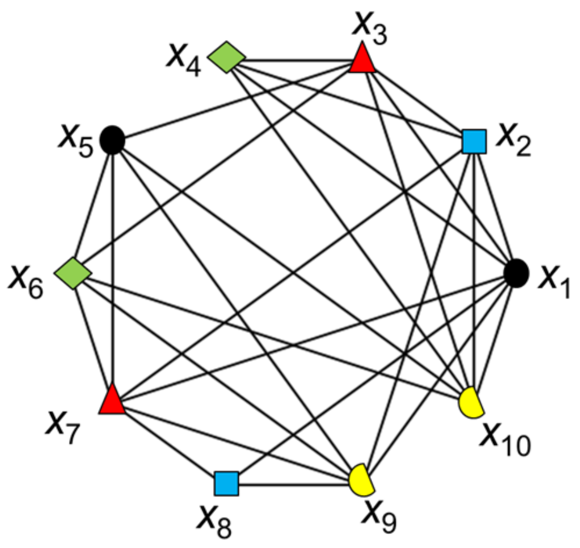


Figure 2.8. Conflict graph for the circuit in Figure 2.3; primary inputs with the same color labels are compatible and can share samples.

2.8 Experimental Results

To gauge the efficiency and accuracy of our approach, we applied ROMC to signal probability estimation for representative ISCAS-85 and LGSyn-93 benchmark circuits [3]. Accuracy was measured in the following way: For an n-input m-output circuit C, reference (gold) signal probabilities were generated for all m primary outputs using conventional MC. If $n \leq 31$, then C was exhaustively simulated; otherwise, C was simulated with 2^{31} random samples. This sample size 2^{31} produces results that are accurate enough for verifying ROMC's performance considering the relatively small size of the benchmark circuits. Then, for a fixed sample size $N = 2^k$, ROMC and MC simulate C 100 times. The accuracy for $N = 2^k$ is measured in terms of the average standard errors of the estimated results [27].

Each circuit was simulated with sample sizes ranging from 2^7 to 2^{24}. We found that ROMC can identify compatible primary inputs and determine each primary input's observability quickly, even in fairly large circuits such as misex3, which contains over 3,600 gates. The runtime overhead for compatible signal identification and observability estimation was less than 2 seconds on an Intel Quad-Core 2.35GHz, 64-bit PC with 4G RAM, which was used for all benchmarks.

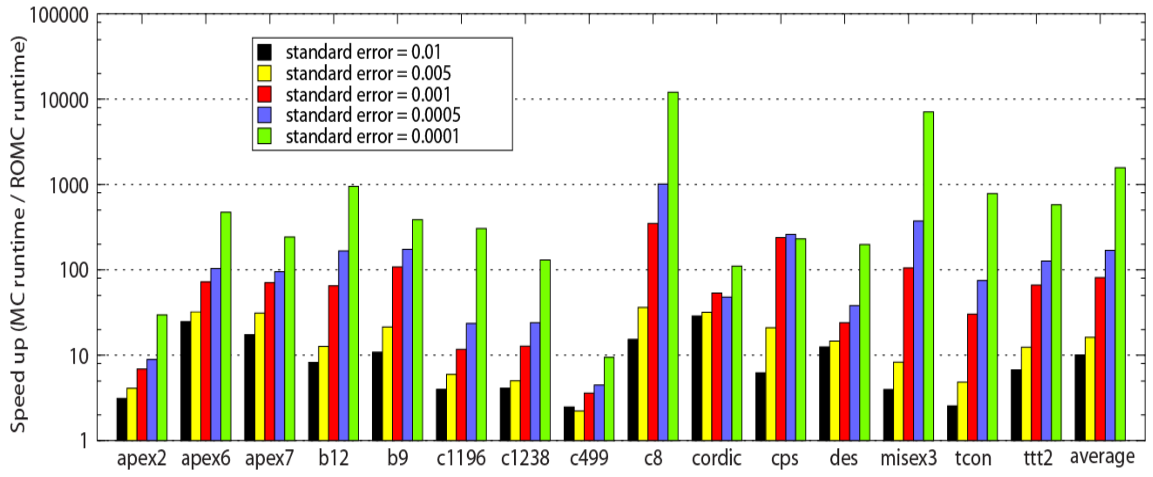


Figure 2.9. Speed-up of ROMC over MC for the benchmark circuits at five accuracy levels defined by standard error.

Figure 2.9 shows the runtime improvements for the representative benchmark circuits. The improvement at each accuracy level is measured by the ratio between the MC and ROMC sample sizes needed to achieve the required accuracy. From the figure, we see that ROMC can reduce runtimes by from one to three orders of magnitude. The average runtime improvement for an estimated error of 10^{-4} is nearly three orders of magnitude. In addition, these simulation results also show that ROMC's runtime improvement grows with increasing accuracy levels. In other words, ROMC can produce very accurate results with far fewer samples than MC. This suggests that ROMC is well-suited to applications that require highly accurate signal probabilities, such as power estimation [58].

We would like to emphasize that a variance reduction method might produce much worse results than the ones produced by MC if it is not designed carefully [27], and developing a variance-reduction technique suitable for all kinds of circuits is a challenge. ROMC, however, produces no sample variance higher (worse) than the variances generated by MC, which suggests that it is a very broadly applicable method. As the results show, it is effective for various types of circuits such as the arithmetic and error detection circuits found in the ISCAS-85 [28] and LGSyn-92 [3] benchmark sets.

Table 2.2. Performance of compatibility and observability determination for selected benchmark circuits including the largest benchmark circuits.

Circuit	No. of gates	No. of primary inputs	No. of inputs removed by compatibility analysis	Runtime (s) for compatibility and observability analysis
misex3	3624	14	0	0.923
seq	2875	41	1	0.430
apex5	2418	115	0	0.176
c5315	2307	178	20	2.902
i10	2195	257	1	5.947
des	2007	256	1	0.279
c3540	1669	50	1	4.718
apex3	1038	54	7	0.248
c1196	529	31	2	0.130
c1238	508	31	3	0.131
cm150a	96	21	14	0.006
i2	36	201	65	0.222

As Table 2.2 indicates, ROMC can identify compatible PIs and determine the observability of each primary input efficiently. The runtimes for compatible signal identification and observability determination in all cases were less than six seconds. Most of the benchmarks contain no more than two or three compatible primary inputs, because of the design effort usually devoted to logic minimization and, implicitly, to redundancy removal. In a few cases, however, such as i2 and C5315, ROMC discovers many compatible primary inputs. In C5315, for instance, no PO is associated with more than 67 different primary inputs. This means that without the proposed compatible input identification technique, a lower bound on sample size for exhaustive simulation is 2^{67}. After compatible signals are identified, this lower bound can be reduced by a factor of 2^{20} to 2^{47}. In the case of i2, the size of the sample set for exhaustive simulation is reduced by 2^{65}, a huge reduction. Although this technique does not work well for all types of circuits, it is still effective for realistic circuits such as C5315, which is a nine-bit arithmetic logic unit [28].

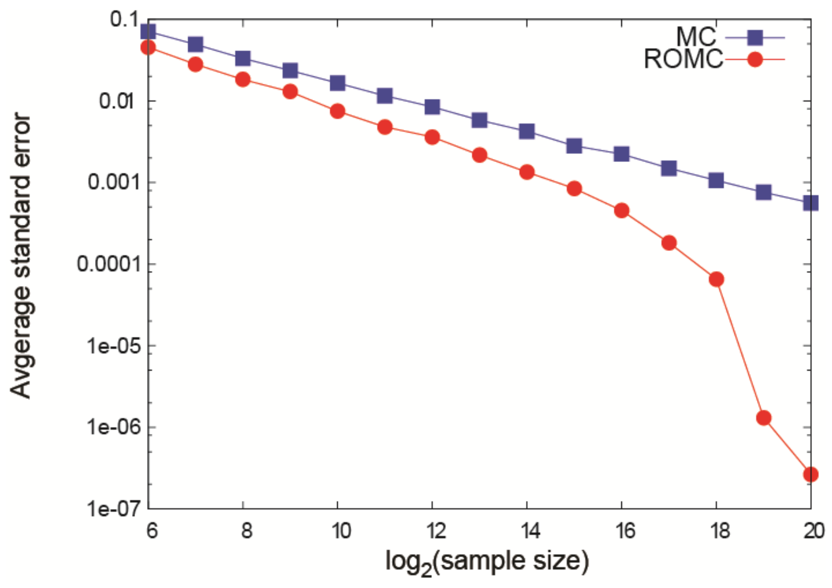


Figure 2.10. Comparison between MC and ROMC for C1196.

Figure 2.10 compares MC and ROMC for the C1196 circuit where the standard error of ROMC falls dramatically with increasing sample size. The big accuracy improvement from N = 18 to N = 20 results from a combination of the compatibility and MCWO features of ROMC. The sample space of C1196 is reduced from 2^{23} to 2^{21} due to the presence two compatible primary input pairs. MCWO can reduce the sample variance exponentially when the sample size is close to that of the entire sample space, which is 2^{21} in this case.

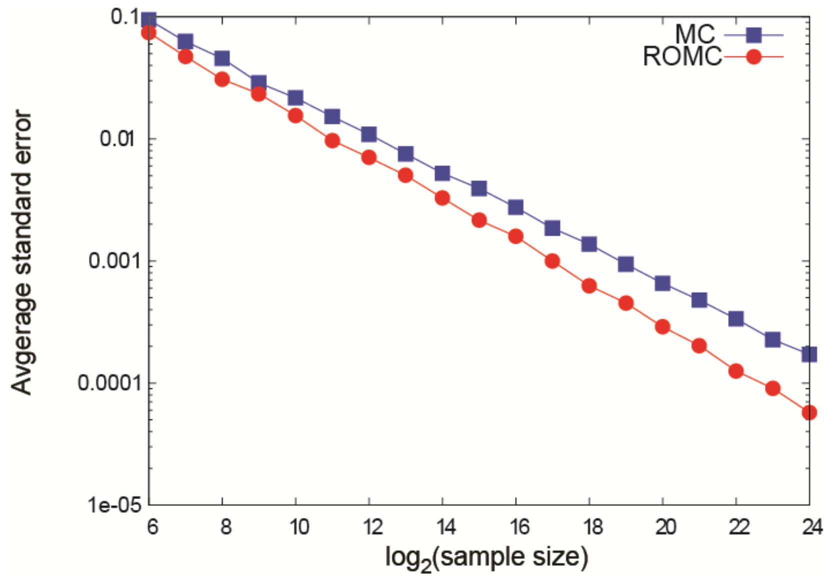


Figure 2.11. Comparison between MC and ROMC for C499.

Although ROMC does not produce negative results in any of the selected benchmarks, there are a few cases where ROMC produces insignificant runtime improvement. Figure 2.11 shows the runtime comparison for C499, which is an error-correction circuit [28]. The small improvement is due to the fact that all primary inputs of this circuit tend to be equally important (of the same low redundancy) for all primary outputs.

2.9 Case Study

Finally, we use a typical EDA task, power consumption estimation, to demonstrate how the proposed ROMC method can be applied to other EDA areas.

As mentioned in Chapter 1, a gate's power consumption is proportional to its switching activity. This can be accurately estimated from the equation

$$P_{\text{dyn}} = 1/2 C_{\text{L}} V^2 f \alpha$$

where C_{L}, V, f and α are the gate's output capacitance, applied voltage, working frequency, and switching activity, respectively [47][58][64]. Of these, the switching activity α is the most difficult to determine. Suppose, however, that all inputs are independent and G's output signal is s. We can then use the variance of $p(s)$, which is $p(s)(1 - p(s))$, to represent α, and hence calculate P_{dyn} [47][58][64].

Consider again the circuit of Figure 2.6, which is reproduced in Figure 2.12. By computing the signal probability of the intermediate lines, we can evaluate the circuit's overall switching activity, and hence its power consumption. In Figure 2.12, gate output lines are marked with the dynamic power consumption of individual gates in microwatts, assuming an IBM 130-nm process with a 1.2 volt supply and a 1 GHz clock frequency. It is worth noting that ROMC estimates the power values with less than 1% error via only 16 samples, while MC has a 12% error with the same number of samples.

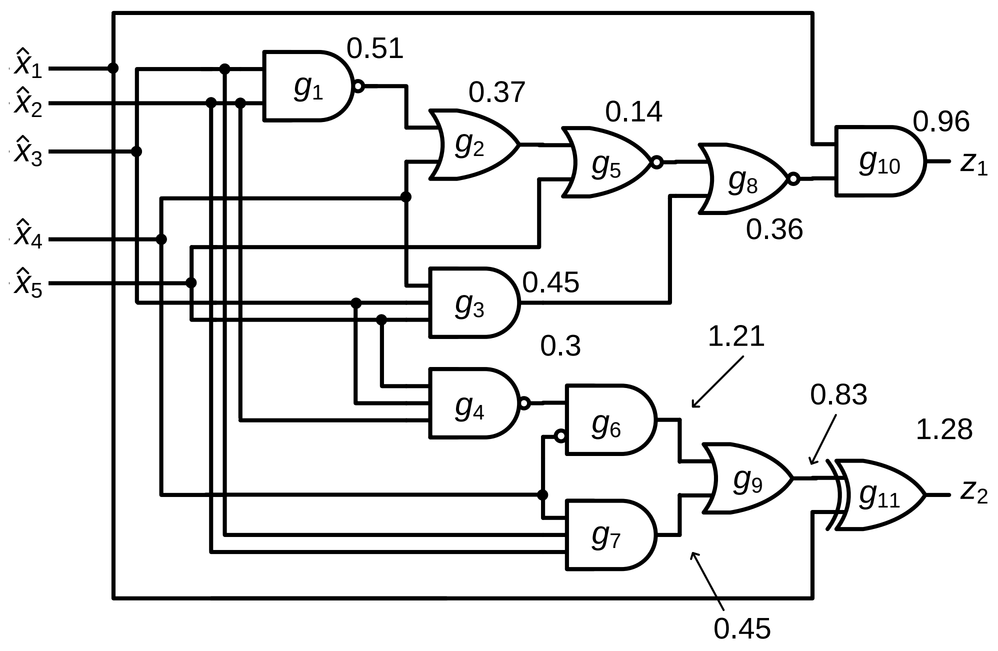


Figure 2.12. Power consumption of the five-input circuit in Figure 2.6.

2.10 Summary

We have proposed an MC-based algorithm ROMC for signal probability estimation in logic circuits. As discussed in Chapter 1, accurate signal probabilities are important to EDA applications based on probabilistic analysis. Existing probabilistic methods for signal probability calculation are confined to small circuits if they attempt to account for all signal correlations, even those having insignificant impact on calculation accuracy, or else provide highly inaccurate results due to inadequate handling of signal correlations.

MC sampling, on the contrary, can precisely control its calculation accuracy by the number of applied samples, unlike probabilistic methods such as CCM and BAM. ROMC, like MC methods, preserves this computational property, but is faster than conventional MC, because it exploits two redundancy properties of logic signals (compatibility and observability) in novel ways to remove unnecessary samples and reduce sample variance.

We have presented experimental results showing that the proposed approach reduces simulation time by as much as three orders of magnitude on a representative set of benchmark circuits. In addition, it can reduce signal probability estimation errors for a given runtime budget. These two nice properties make ROMC particularly suitable for EDA tasks such as measuring power consumption.

CHAPTER 3

Trigonometry-based Probability Modeling

In Chapter 1, we discussed how soft errors have become a major concern of circuit design because of the decreasing noise margins of ICs. This chapter presents a novel probabilistic algorithm, Trigonometry-based Probability Calculation (TPC), aimed at logic circuits employing the unconditional error model. TPC formulates signal probability in terms of trigonometric functions controlled by angles. Gate-error probability is modeled in TPC by a small angular deviation, and is treated as a parameter of signal probability. Consequently, a gate’s error-free signal probability and error probability are fully integrated. With the proposed trigonometric representation, typical (Taylor) expansion techniques can be applied to simplify probabilistic calculations. The overall efficiency and accuracy can be managed by the number of expansion terms used. Our experimental results show that TPC scales well to circuits with tens of thousands of gates, and its average accuracy, even with error probabilities as small as 10^{-8}, is over 96%. This work was presented at the DATE (Design, Automation, and Test in Europe) conference in 2011 [68].

3.1 Background

The gates of modern ICs are very sensitive to non-deterministic effects caused by high energy particles, process variations, and voltage overscaling. A key characteristic of gate-error probabilities is that they are usually very low, e.g., 10^{-8} to 10^{-10} [30][37]. In practice, analyzing such non-deterministic phenomena by physical simulation methods like radiation testing is expensive [29]. Monte Carlo (MC) sampling is also widely used for this purpose [11][70]. However, MC sampling can be very time-consuming, especially when the occurrence probabilities of the error phenomena of interest are so low. As we discussed in Chapter 2, large sample sets are necessary for accurately capturing such rare events.

Recently, several methods have been proposed for evaluating the soft-error rate (SER) [1][7][20][34][48] based on various probabilistic theories. The main differences between these techniques lie in their error models, the way they simulate masking effects, the completeness of their correlation analysis, and their ability to simulate situations where several errors occur in the same clock cycle.

In Chapter 1, we briefly introduced the probabilistic transfer matrix (PTM) approach [34]. Now we review and compare the other major probabilistic techniques for SER estimation for combinational circuits: Bayesian networks (BNs) [48], probabilistic decision diagrams (PDDs) [1], the Four-Event (FE) method [7], and the Single-Pass (SP) method [20].

The BN method uses conditional error models, while the PDD, FE, and SP methods employ unconditional ones. Like the PTM approach, the BN and PDD methods preserve complete information about signal correlations by mapping circuits into certain data structures, namely junction trees and decision diagrams, all of which can be very complex. The PTM, BN, and PDD methods are therefore limited to relatively small circuits, since the complexity of analyzing and storing correlation information using their various data structures is huge. On the other hand, the FE and SP methods do not account for all correlations among erroneous and error-free signals. Therefore, these two approaches are more scalable but much less accurate than the others.

Rejimon and Bhanja presented a method of estimating circuit reliability using Bayesian networks [48]. Like the PTM approach, the BN technique uses a conditional error model. Given an error-free circuit *C*, this method first creates an erroneous circuit model C^e from *C*. Then each primary output of *C* is connected to the corresponding output of C^e by an XOR gate. To evaluate circuit reliability, a junction tree for a new circuit comprising *C*, C^e and XOR gates is constructed. A node of the junction tree may contain several gates that are highly correlated. An important property of junction trees is that the path from one junction-tree node to another is unique [34]. After the junction tree construction, the signal probabilities can be evaluated, but the evaluation complexity is exponential in the maximum number of gates at a junction-tree node. The BN method is practical only for small circuits due to its use of the conditional error model and exhaustive correlation estimation.

Probabilistic decision diagrams (PDDs) [1] are single-terminal, weighted and directed acyclic graphs (DAGs). They form the basis of a signal probability and reliability calculation algorithm for combinational circuits. Unlike the PTM and BN methods, the PDD approach describes error behavior with the simpler unconditional error model. A circuit's PDD is constructed as follows. First, for each gate, generate its PDD based on its error probability. Second, recursively merge the gate PDDs from inputs to outputs. Like ROBDDs and ADDs, PDDs are canonical for circuits with the same functions and error probabilities [1]. To guarantee canonicity, all redundant sub-graphs and unnecessary nodes must be eliminated from a PDD. The size of the final PDD is determined not only by the correlation properties of the circuit, but also by the variable order. Like the PTM method, this method is unsuited to calculating the probabilistic behavior of individual gates in a single PDD.

To deal with the complexity of handling correlation information, Asadi and Tahoori developed a soft-error reliability estimation method, called Four-Event (FE), which also employs unconditional error models [7]. The FE method describes signal behavior by means of four different states: 1, 0, e and e', where e and e' represent an erroneous value and its negation, respectively. It calculates the probability that an erroneous value originating from a particular gate g will be observed at a primary output. For a faulty gate g, FE first assigns e to g, and then simulates the circuit. The error probability associated with g is equivalent to the probability that e or e' is observed at the primary outputs. Paths between g and the primary outputs are called *on-paths*, while the others are *off-paths*. A major advantage of this method is that all correlations among on-paths (the correlations of e and e') are accounted for. However, the off-path correlations are ignored, which can dramatically affect accuracy. For instance, in Figure 3.1 the propagation probabilities of the induced error to the primary output using the FE and exact methods are 0.375 and 0.5, respectively, so the computational error of FE can be as high as 25%.

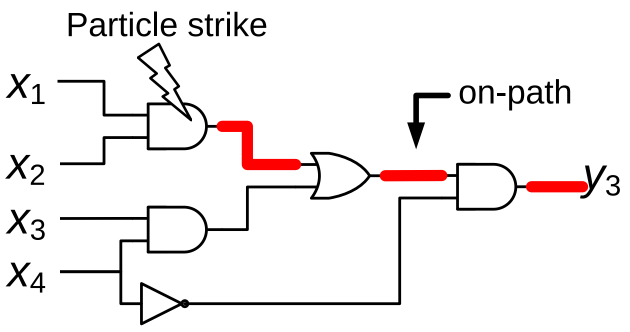


Figure 3.1. Illustration of the FE method; an on-path is denoted by red bold lines.

The SP method [19][20] only examines the correlations between consecutive levels of the target circuit. It first employs CCM to calculate signal probabilities for error-free versions of individual gates, and then evaluates reliability or error susceptibility based on the error-free probabilities. Such calculations implicitly assume that the occurrence of errors does not affect the signal probabilities. As a result, the accumulated effects of multiple soft errors are underestimated, and the overall accuracy diminishes significantly with very small error probabilities. For instance, in [19] the average computational inaccuracy for error probabilities of 0.3 and 0.05 are reported as 0.46% and 5.63%, respectively, which suggests that the inaccuracy grows rapidly with decreasing error probabilities. This makes the SP approach unsuitable for estimating soft-error effects because, in practice, soft error probabilities are very low, e.g., 10^{-9}.

We have shown that the existing methods of soft-error analysis are either confined to small circuits because of their inefficient handling of correlation, or else provide fast but inaccurate simulation results. This makes it difficult for these methods to perform accurate SER simulation when used with realistic error models. In the next section, we introduce a trigonometry-based probability model that can integrate error-free and error probabilities efficiently. We will show later that the proposed TPC method is particularly suitable for dealing with low error probabilities, which are an important feature of soft-error models.

3.2 Unconditional Error Representation

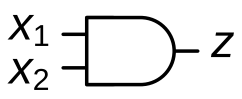
We first derive a mathematical expression for unconditional errors, and introduce some basic properties of unconditional models. We then briefly discuss an error representation technique using exclusive-or (XOR) gates, which has been adopted in many error-estimation methods. We explain why this approach can dramatically increase complexity, making it impractical for large circuits.

Consider a two-input AND gate with the unconditional error model shown in Figure 3.2. This is compared with a conditional gate-error model in Figure 1.14. Let the signal probabilities associated with four different input vectors be $p(x_1'x_2')$, $p(x_1'x_2)$, $p(x_1x_2')$, and $p(x_1x_2)$, respectively. Since the four probabilities are mutually exclusive, z's erroneous signal probability $p(z^e)$ can be expressed as

$$p(z^e) = p(x_1'x_2')p_{err} + p(x_1'x_2)p_{err} + p(x_1x_2')p_{err} + p(x_1x_2)(1 - p_{err})$$

which reduces to

$$p(z^e) = p(z)(1 - p_{err}) + (1 - p(z))p_{err} \tag{3.1}$$



(a)

Input	Output probabilities	
x_1x_2	$z = 0$	$z = 1$
00	$1 - p_{err}$	p_{err}
01	$1 - p_{err}$	p_{err}
10	$1 - p_{err}$	p_{err}
11	p_{err}	$1 - p_{err}$

(b)

Figure 3.2. (a) Two-input AND gate, and (b) its behavior with an unconditional error model.

Equation (3.1) shows that for a particular gate, the erroneous version of its signal (output) probability $p(z^e)$ can be obtained from its error-free signal probability $p(z)$ and the gate-error probability p_{err}. The maximum and the minimum values of $p(z^e) - p(z)$ are p_{err} and $-p_{err}$ when $p(z) = 0$ and 1, respectively. Thus for an arbitrary $p(z)$, $p(z^e)$ is closer to 0.5 than $p(z)$. In other words, the presence of errors increases the degree of "uncertainty". Moreover, it is worth noting that $p(z^e) - p(z) = 0$ when $p(z) = 0.5$, which means, from the statistical point of view, errors have no significant impact on signals whose signal probabilities are very close to 0.5.

We now sketch a widely-used method for error representation in simulation algorithms. Equation (3.1) can be interpreted as defining the signal probability expression for a two-input XOR gate, as illustrated in Figure 3.3(b). This XOR gate has two uncorrelated input signals z and x^e, where x^e serves as a random variable whose probability of being 1 is p_{err}. The values of x^e form a stochastic binary sequence, whose bits are temporarily uncorrelated. Hence, the overall error effect of a circuit can be modeled by connecting each gate in the circuit to a new primary input via an XOR gate, as in Figure 3.3(b). Several error estimation algorithms [1] explicitly or implicitly employ this technique in their evaluation procedures. Clearly, this error representation method will double the number of gates and primary inputs in a circuit, so the complexity of processing a XOR-extended circuit can be much higher than that of the original version.

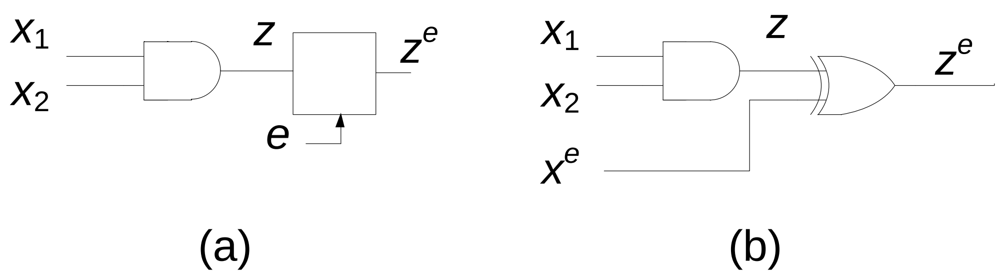


Figure 3.3. (a) Two-input AND gate with unconditional error *e*; (b) Using a two-input XOR gate to model the error.

3.3 Trigonometric Representation of Probability

We show next how signal probabilities can be efficiently described by a trigonometric approach, where signal probabilities are represented by angles. We also introduce a corresponding representation for error probabilities, the *trigonometric error model* (TEM), which models a gate's error probability as a small positive or negative addition to its signal probability angle. Consequently, TEM calculations avoid both the insertion of XOR gates and the multiplication steps required by (3.1).

3.3.1 Trigonometric Signal Probability

Let s be a signal in a logic circuit. For any input probability distribution of s, $0 \leq p(s), p(s') \leq 1$ and $p(s) + p(s') = 1$. For a given pair $p(s)$ and $p(s')$, there exists an angle θ, where $0 \leq \theta \leq \pi/2$, such that $\sin^2 \theta = p(s)$ and $\cos^2 \theta = p(s')$, a consequence of the fundamental trigonometric identity $\sin^2\theta + \cos^2\theta = 1$. This implies that a signal probability can be fully described in terms of certain trigonometric functions controlled by a single angle we call the *signal probability angle* (SPA). The SPA θ of s can be expressed as

$$\theta = \cos^{-1}(\sqrt{p(s)}) \quad \text{or} \quad \theta = \sin^{-1}(\sqrt{p(s')}) \tag{3.2}$$

For example, the SPA for $p(s) = 0.25$ is π/3.

3.3.2 Trigonometric Error Model

Unlike the XOR-based approach of (3.1), TEM formulates a gate's error probability as a linear rotation associated with its (error-free) signal probability angle by combining the trigonometric representation and the Taylor expansion technique. TEM is more scalable because it maps probabilistic calculations into rotations without inserting extra gates or signals.

Recall that for a particular signal s, if $p(s) \leq 0.5$, then $p(s^e) \geq p(s)$; otherwise, if $p(s) \geq 0.5$, then $p(s^e) \leq p(s)$. This property means if an error-free SPA is less (greater) than π/4, then its erroneous SPA is greater (less) than the error-free version. Such small rotations reflect signal errors, which we call *error probability angles* (EPAs). Given a gate's error-free signal and error probabilities, the EPA is first calculated, and then the overall signal probability is obtained by adding or subtracting the EPA from the error-free SPA.

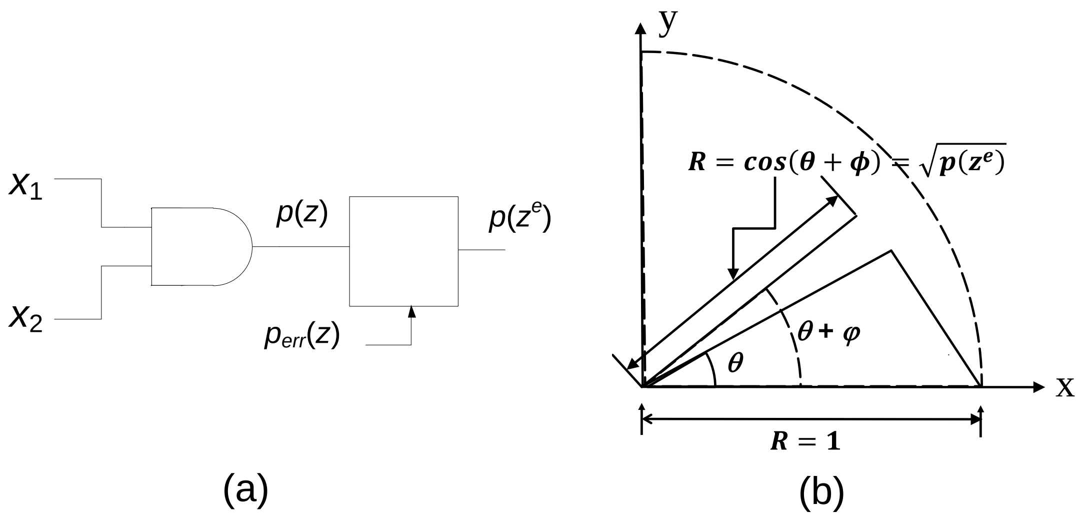


Figure 3.4. (a) Two-input AND gate with an unconditional error; (b) its trigonometric probability representation.

Figure 3.4 illustrates how an EPA can be calculated from the corresponding error-free SPA and gate-error probability. Assume that $p(z) \leq 0.5$, and $p(z) > p_{err}$. Then

$$\cos^2(\theta - \phi) = p(z)(1 - p_{err}) + \big(1 - p(z)\big)p_{err} \tag{3.3}$$

where θ and ϕ ($\phi \geq 0$) are the SPA and EPA of z, respectively. This equation can be re-written as

$$(\cos\theta\cos\phi + \sin\theta\sin\phi)^2 = p(z)(1 - p_{err}) + \big(1 - p(z)\big)p_{err}$$

If p_{err} is very small, then ϕ is also very small, so $\cos\phi$ and $\sin\phi$ can be approximated by $1 - \phi^2/2!$ and ϕ, the initial terms of their respective Taylor series expansions. By definition, $\cos^2\theta = p(z)$ and $\sin^2\theta = 1 - p(z)$, so

$$\big(1 - 2p(z)\big)\phi^2 + 2\sqrt{p(z)\big(1 - p(z)\big)}\phi - \big(1 - 2p(z)\big)p_{err} = 0$$

This has the solution

$$\phi = \frac{\sqrt{p(z)\big(1-p(z)\big)+\big(1-2p(z)\big)^2 p_{err}} - \sqrt{p(z)\big(1-p(z)\big)}}{1-2p(z)} \tag{3.4}$$

(Note that the other solution is invalid because $\phi \geq 0$.) We can replace (3.4) by its Taylor series expansion at p_{err} = 0:

$$\phi = \frac{\big(1-2p(z)\big)}{2\sqrt{p(z)\big(p(z')\big)}} p_{err} + \frac{(2p(z)-1)^3}{8\sqrt{\big(p(z)p(z')\big)^3}} p_{err}^2 + \cdots$$

Since p_{err} is very small, we can approximate ϕ by the first term of its Taylor series. Thus, ϕ can be re-stated as

$$\phi \cong \big(1-2p(z)/2\sqrt{p(z)p(z')}\big)p_{err} = -\cot(2\theta)\, p_{err} \tag{3.5}$$

By combining Equations (3.3) and (3.5), we get $p(z^e) \cong \cos^2(\theta + \cot(2\theta)p_{err})$, where $\theta = \cos^{-1}(\sqrt{p(z)})$ and $p(z) \leq p_{err} \leq 0$. If $p(z) \leq p_{err}$, then z's EPA can be calculated in a similar way, which yields $\phi \cong \big(1-2p_{err}/2\sqrt{p(z)p(z')}\big)p(z)$. Note that if $p(z) \leq p_{err}$, there is no need to evaluate signal correlation on z later, since z's signal behavior is not controlled by its input signals, but is mainly determined by "random noise". As can be seen from (3.5), given a gate's error-free SPA, the resulting EPA is a small rotation that is linear in the gate-error probability. The rotation operations in (3.5) can be easily realized by additions. The XOR-based technique discussed in Section 3.2 models an unconditional error probability with multiplications and additions, while only additions are required with TEM. With its trigonometric representation, TEM is computationally much simpler than the XOR-based technique.

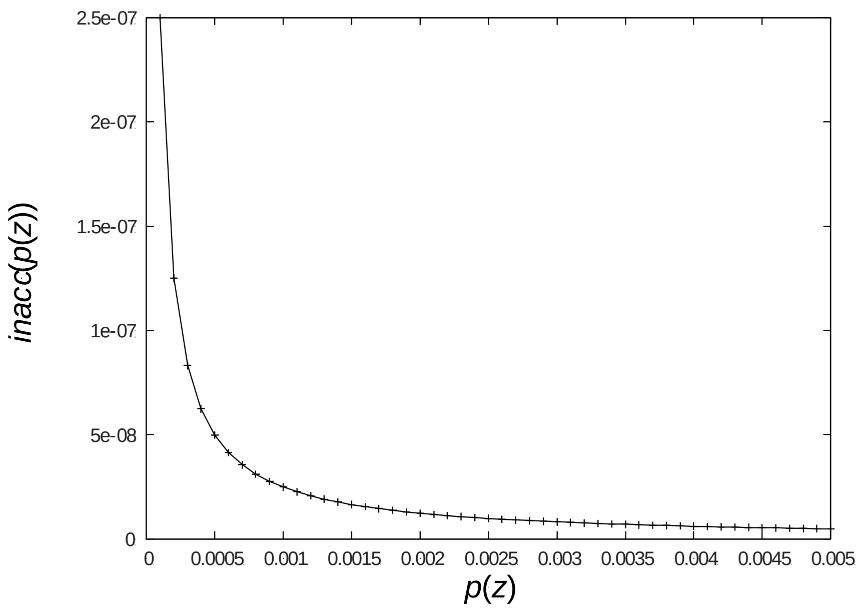


Figure 3.5. Computational inaccuracy *inacc*($p(z^e)$) for $p(z) = 10^{-4}$ to 5×10^{-3} with gate-error probability $p_{err} = 10^{-5}$.

Next, we examine the computational inaccuracy of (3.5) with respect to $p(z)$. Given a gate-error probability p_{err}, this inaccuracy can be measured by the differences in signal probabilities between the exact and approximated cases, which is given by |Eq. (3.1) – Eq. (3.3)|, where ϕ in (3.3) is replaced by (3.5), namely, $inacc\big(p(z^e)\big) = p_{err}^2(p(z') - p(z))^3/4p(z)p(z')$. The worst case happens at $p(z) = p_{err}$ and the maximum of *inacc*($p(z^e)$) is less than $0.25p_{err}$. Figure 3.5 shows a plot of *inacc*($p(z^e)$) for $p_{err} = 10^{-5}$. As can be seen, the computational inaccuracy decreases sharply from 2.5×10^{-7} to 4.8×10^{-9} when $p(z)$ just slightly increases from 10^{-4} to 5×10^{-3}. Suppose that $p(z)$ is a uniformly-distributed random variable. The average inaccuracy of (3.5) is

$$inacc_{avg}\big(p(z^e)\big) = \int_{p(z)=p_{err}}^{p(z)=0.5} (0.5 - p_{err})^{-1} \times inacc\big(p(z^e)\big)dp(z)$$

The average inaccuracies for p_{err} = 10^{-5} and 10^{-6} are 4.56×10^{-10} and 5.71×10^{-12}, respectively. Thus the average inaccuracy is quadratic in the given gate-error probability. As a result, the accuracy increases rapidly with decreasing error probabilities, making TEM particularly suitable for estimating practical non-deterministic phenomena.

3.4 Probabilistic Calculation Algorithm

We now present the TPC algorithm for calculating signal probabilities in circuits with errors. TPC combines the TEM error representation and the efficient BAM heuristic [58] for signal probability estimation. A useful feature of TPC is that its efficiency and accuracy can be controlled by carefully selecting the terms that make a significant contribution to the results, and deleting other terms.

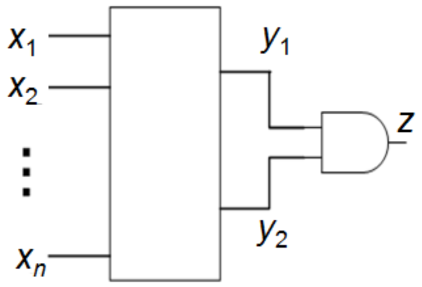


Figure 3.6. Single-output circuit with *n* inputs.

3.4.1 Correlation Handling

TPC incorporates BAM to account for correlation among signals because BAM can handle correlation more accurately and efficiently than other published methods. As discussed in Chapter 2, BAM was developed by Uchino et al. [58] to estimate switching activity in error-free combinational circuits. Without losing generality, we again use a two-input AND gate to illustrate the basic concepts of BAM. In Figure 3.6, *z* is the output of an AND gate whose inputs are y_1 and y_2, and x_1, x_2, $\cdots$,x_n are the circuit's primary inputs, assuming the circuit is error-free. The Boole-Shannon expansion of z with respect to x_i is $z = x_i z_{x_i} + x_i{}' z_{x_i'}$ where $z_{x_i}$and $z_{x_i'}$ are the cofactors with respect to x_i. The exact signal probability $p^*(z)$ of z can be written as

$$p^*(z) = p(x_i)p(z_{x_i}) + p(x_i)p(z_{x_i'})$$

Suppose that x_i is the only primary input shared between y_1 and y_2, then $p(y_1) = p(x_i)p(y_1) + p(x_i')p(y_{1_{x_i'}})$ and $p(y_2) = p(x_i)p(y_{2_{x_i}}) + p(x_i')p(y_{2_{x_i'}})$. The exact signal probability of z can then be expressed as

$$p^*(z) = p(x_i)p(x_i')\left(p\left(y_{1_{x_i'}}\right) - p\left(y_{1_{x_i}}\right)\right)\left(p\left(y_{2_{x_i'}}\right) - p\left(y_{2_{x_i}}\right)\right) + p(y_1)p(y_2) \quad (3.6)$$

The second term in (3.6) is the product of y_1 and y_2's signal probabilities without considering correlation, while the first term "compensates" for the signal correlation caused by the common input x_i shared between y_1 and y_2. The general form of (3.6) for n shared primary inputs can be approximated by

$$p(z) = \sum_{i=1}^{i=n} p(x_i)p(x_i')\left(p\left(y_{1_{x_i'}}\right) - p\left(y_{1_{x_i}}\right)\right)\left(p\left(y_{2_{x_i'}}\right) - p\left(y_{2_{x_i}}\right)\right) + p(y_1)p(y_2) \quad (3.7)$$

which, for brevity, we write as $p(y_1)@p(y_2)$. Equation (3.7) only accounts for correlation of n major cases caused by individual primary inputs, whereas exact correlation calculation would require explicitly enumerating 2^n cases. In addition, z's cofactor probabilities associated with x_i are

$$p\left(z_{x_i'}\right) \cong p\left(y_{1_{x_i'}}\right)p\left(y_{2_{x_i'}}\right) \text{ and } p\left(z_{x_i}\right) \cong p\left(y_{1_{x_i}}\right)p\left(y_{2_{x_i}}\right) \quad (3.8)$$

The complexity of BAM is $O(nN)$ where n and N are the numbers of inputs and gates in the circuit, respectively. Generally, $n \cong N^{0.5}$, so BAM's complexity is about $O(N^{1.5})$ [58].

Other heuristics besides BAM could be used for the correlation analysis, such as CCM, mentioned in Chapter 2 for this purpose [19][20]. However, unlike BAM, which only analyzes gates with common primary inputs, CCM only accounts for the correlations between any two gates at the same level, even if they have no common predecessor gates. This tends to make CCM's complexity quadratic in the size of the circuit. The simulation results in [58] show that BAM and CCM achieve similar accuracy, but BAM is generally faster by an order of magnitude.

3.4.2 Signal Probability Estimation

We use the two-input AND in Figure 3.4 to explain how the signal probability of a gate with errors is calculated with TEM and BAM. Assume that y_1 and y_2 are uncorrelated. Let θ_i be the SPA of $p(y_i)$, and let $p_{err}(y_i)$ be the gate-error probability associated with y_i. From (3.5), z's error-free signal probability is

$$p(z) = \prod_{i=1}^{2} \cos^2\left(\theta_i + \cot(2\theta_i)\, p_{err}(y_i)\right) \tag{3.9}$$

In fully expanded form, expression (3.9) contains 12 terms. Of these terms, some are higher-order terms in $p_{err}(y_i)$ e.g., $\sin^2\theta_1\sin^2\theta_2\sin^2(\cot(2\theta_1)p_{err}(y_1))\sin^2(\cot(2\theta_2)p_{err}(y_2))$, which is bounded above by $p^2_{err}(y_1)p^2_{err}(y_2)$. Such higher-order terms make an insignificant contribution to the overall result when $p_{err}(y_1)$ and $p_{err}(y_2)$ are very small. In that case, $p(z)$ estimated by removing $p_{err}(y_1)$ and $p_{err}(y_2)$*'s* higher-order terms remains very accurate. If we calculate $p(z)$ only using terms containing $p^i_{err}(y_1)\, p^j_{err}(y_2)$ where $i + j \leq 2$, we get:

$$p(z) \cong \prod_{i=1}^{2} p(y_i)\,(1 - 2\sum_{i=1}^{2} p_{err}(y_i) + \sum_{i=1}^{2} \frac{p_{err}(y_i)}{p(y_i)}$$

$$+ \sum_{i=1}^{2} \frac{\left(1-2p(y_i)\right)^2}{4p^2(y_i)} p_{err}^2(y_i) + 4\prod_{i=1}^{2} p_{err}(y_i)) \tag{3.10}$$

$$+(1 - 2\sum_{i=1}^{2} p_{err}(y_i)) \prod_{i=1}^{2} p_{err}(y_i)$$

The maximum computational inaccuracy of (3.10) is $p^2_{err}(y_1)\ p^2_{err}(y_2)$. If each $p_{err}(y_i) = 10^{-8}$, this inaccuracy is less than 10^{-32} If we need to reduce computational overhead further, we can just drop terms containing $p^i_{err}(y_1)\ p^j_{err}(y_2)$ where $i + j \geq 2$, so (3.10) is further simplified to

$$p(z) \cong \prod_{i=1}^{2} p(y_i) \left(1 - 2\sum_{i=1}^{2} p_{err}(y_i) + \sum_{i=1}^{2} \frac{p_{err}(y_i)}{p(y_i)}\right) \tag{3.11}$$

The maximum inaccuracy of (3.11) is $p_{err}(y_1)\ p_{err}(y_2)$. Again, if $p_{err}(y_i) = 10^{-8}$, this is less than 10^{-16}, which from the practical viewpoint is still good enough for most applications. Equations (3.10) and (3.11) can be easily extended to other gate types. The cofactor probabilities can be estimated in a similar way. For instance, z's positive cofactor probability associated with x_i is

$$p(z[x_j]) \cong \prod_{i=1}^{2} p(y_i[x_j]) \left(1 - 2\sum_{i=1}^{2} p_{err}(y_i) + \sum_{i=1}^{2} \frac{p_{err}(y_i)}{p(y_i[x_j])}\right)$$

Now we explain how the signal correlation is estimated. A gate's unconditional error probability is independent of its inputs and indeed, of all circuit signals. In this case, $p_{err}(y_1)$ and $p_{err}(y_2)$ are uncorrelated with y_1 and y_2 and any preceding gates of y_1 and y_2. Therefore, in (3.10) and (3.11) there is no need to consider signal correlation for any term in $p(y_i)$ and $p_{err}(y_i)$. The term $\prod_{i=1}^{2} p(y_i) = p(y_1)p(y_2)$ is the only one requiring correlation analysis because it implicitly assumes y_1 and y_2 are uncorrelated. If y_1 and y_2 can be represented as functions of n primary inputs as shown in Figure 3.6, the operation of $p(y_1)p(y_2)$ in (3.10) and (3.11) is replaced by $p(y_1)@p(y_2)$ using BAM. For instance, (3.11) becomes

$$p(z) \cong p(y_1) * p(y_2)\left(1 - 2\sum_{i=1}^{2} p_{err}(y_i) + \sum_{i=1}^{2} \frac{p_{err}(y_i)}{p(y_i)}\right)$$

```
  TPC_algorithm(combinational circuit CC, gate-error probability
configuration ge_conf, primary input signal probabilities pi_sp)
  Levelize CC

  Initialize all primary input pi_sp and gate-error probabilities gp_conf

  For each topological level l (from primary inputs )

    For each gate z at the l^th level
      Use TEM to model the error probabilities as rotations

      Calculate z's output probability using Eq.(3.10) or Eq.(3.11)
      (Correlation is handled using BAM)

      For each primary input x
        Calculate z's cofactor probabilities associated with x
```

Figure 3.7. TPC probability estimation algorithm.

The overall computational complexity of TPC is $O(nN)$, where n and N are the numbers of inputs and gates in the circuit, so TPC has near-linear runtime complexity in the general case.

3.5 Simulation Results

To evaluate the performance of the proposed TPC method, and to check its accuracy, we used it to calculate signal probabilities for various benchmark circuits with three representative gate-error probabilities p_{err} = 10^{-3}, 10^{-4} and 10^{-8} applied to all gates. All primary input signal probabilities are assumed to be 0.5. The selection of benchmarks is constrained by the published, and often incomplete, data available. Note that our TPC program is quite general and places no a priori restriction on the error probability values. The experiments were performed on an Intel Quad-Core, 2.35 GHz, 64-bit PC, with 4GB RAM, and running under Linux.

Runtime, memory usage and accuracy data for gate-error probability 10^{-8} appear in Table 3.1. To measure the accuracy for such low error probabilities, we also used MC sampling to estimate signal probabilities with 320 million random input vectors. The inaccuracy figures were determined by comparing the TPC and MC results, where the MC results are assumed to be accurate.

Table 3.1. Performance of TPC and MC for gate-error probability 10-8 with selected ISCAS-85 benchmarks (including the largest ones).

Circuit	Runtime (s)		TPC memory usage (MB)	Inaccuracy (%)
	TPC	MC		
C1196	0.054	3508	1.03	6.038
C1238	0.071	3348	1.04	6.154
C1355	0.174	3612	2.23	0.189
C1908	0.462	5619	2.75	5.235
C2670	0.459	8609	3.49	6.090
C3540	1.454	11063	5.53	7.614
C5315	2.319	15283	6.39	7.989
C6288	11.290	15186	6.81	3.559
C7552	5.911	23096	10.06	7.507

The maximum runtime, memory usage, and computational inaccuracy of our TPC program for gate-error probability $p_{err} = 10^{-8}$ are 11.29 sec. (for C6288), 10 MB (for C7552), and 7.98% (for C5315), respectively. Figure 3.8 shows the inaccuracy defined by comparing the results of TPC and MC sampling for the ISCAS-85 benchmarks and several values of p_{err} The average computational error with $p_{err} = 10^{-3}$ is about 9.27%, but it drops to 5.4% with error probability $p_{err} = 10^{-4}$. The runtime and accuracy data in Table 3.1 and Figure 3.8 indicate that the TPC method can efficiently produce accurate results in cases with gate-error probabilities of 10^{-4} or less. These simulation results also suggest that, unlike the FE method [7], TPC is capable of handling circuits where errors simultaneously occur at multiple gates, which makes TPC more general than FE. Moreover, unlike the SP method, the accuracy of the TPC approach grows with decreasing gate-error probabilities. As a result, TPC may provide less accurate results for high error probabilities like 10^{-3}; however, as discussed in Section 3.3, such high probabilities are rarely encountered in real applications. More importantly, its accuracy with very low error probabilities makes it particularly suitable for soft-error estimation.

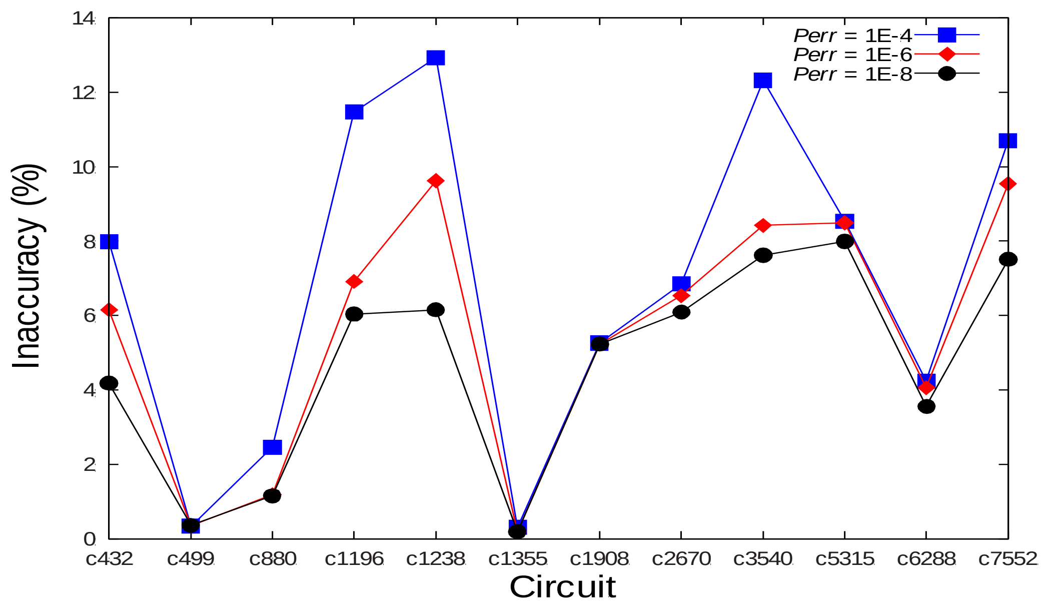


Figure 3.8. Average computational error of the TPC method for various gate-error probabilities.

Table 3.2 compares the TPC and signal probability methods for evaluating circuit reliability. The SP data is taken from [20]. These results show that TPC is faster than SP by one or two orders of magnitude, mainly due to the differences between their respective use of BAM and CCM for handling signal correlation. Generally, the TPC method is efficient and accurate for practical applications like transient-error estimation, whereas the SP method is more suitable for applications with very high error probabilities.

Table 3.2. Runtime comparison between the TPC and SP methods for reliability estimation.

		Runtime (sec.)	
Circuit	**No. of gates**	**TPC**	**SP [20]**
C499	650	0.080	15.40
C1355	653	0.417	14.70
C1908	699	0.892	30.06
C2670	756	0.901	1.11
frg2	1024	4.000	0.30
C3540	1466	2.97	234.63
i10	2643	19.473	145.35

3.6 Summary

We have presented a new trigonometric framework TPC for probabilistic calculations intended to handle very-low-probability soft errors of the type usually found in practice. Prior methods for soft-error estimation are typically limited by error modeling and signal correlation considerations to relatively small circuits or to relatively inaccurate analysis of large circuits. The TPC approach, on the other hand, is able to produce accurate results, even in the case of large circuits, because of its novel use of trigonometric operations and its improved heuristics for handling correlation. An interesting feature of TPC is that its computational inaccuracy falls as gate-error probabilities are reduced to levels that better reflect the actual frequency of soft errors in ICs.

CHAPTER 4

Soft-Error Estimation in Sequential Circuits

In the previous chapter, we introduced a method that can estimate soft-error effects in combinational circuits using unconditional error models. In this chapter, we propose a SER estimation method, SAMPLE (Scalable and Accurate Matrix-based Probabilistic Algorithm for Logic Signal Estimation) for sequential circuits based on probabilistic transfer matrices (PTMs). PTMs can accurately represent the probabilistic behavior for a gate or even for an entire circuit with conditional error models, and they can be used for calculating signal and error probabilities in sequential circuits. However, a key challenge of applying PTMs to probabilistic calculation is to reduce the computational overhead of constructing the PTMs and to control the size of resulting matrices.

SAMPLE dramatically reduces the complexity of PTM calculations and makes the PTM size manageable by circuit partitioning and by avoiding massive matrix multiplications. In addition, unlike existing methods that can only simulate small sequential circuits for just a few simulation cycles, SAMPLE is capable of simulating large sequential circuits, such as circuits containing 20,000 gates over hundreds of cycles without significantly increasing memory usage. This work was presented at the VLSI Test Symposium in 2010 [67].

4.1 Prior Work

Chapter 3 introduced several probabilistic techniques for simulating soft errors in combinational circuits. In this chapter, we propose an SER estimation method for sequential circuits. Simulating the soft-error effect in sequential circuits is more challenging than in combinational circuits, because the computational complexity is affected not only by issues such as error modeling, masking effects, and signal correlation, but also by two new issues: (1) the inefficiency of existing methods for simulating sequential circuits over many cycles, and (2) the fact that several soft errors can occur in the same simulation run.

A commonly-used technique for tracing the behavior of a sequential circuit from the first simulation cycle to an arbitrary simulation cycle is time-frame expansion [15][40]. To simulate a circuit *SC* over *w* cycles, the time-frame expansion technique first replicates the combinational parts of *SC* $w - 1$ times, and then connects the secondary outputs of replica *i* to the corresponding secondary inputs of replica $i - 1$. The "expanded" circuit becomes a pseudo-combinational circuit, and SER methods developed for handling combinational circuits can be directly applied to it. Figure 4.1 shows an example with $w = 3$. The size and the total number of primary inputs of the expanded circuit are *w* times larger than the original version. This can consume large amounts of memory space, which may make the time-frame expansion technique unsuitable for processing large circuits over many simulation cycles.

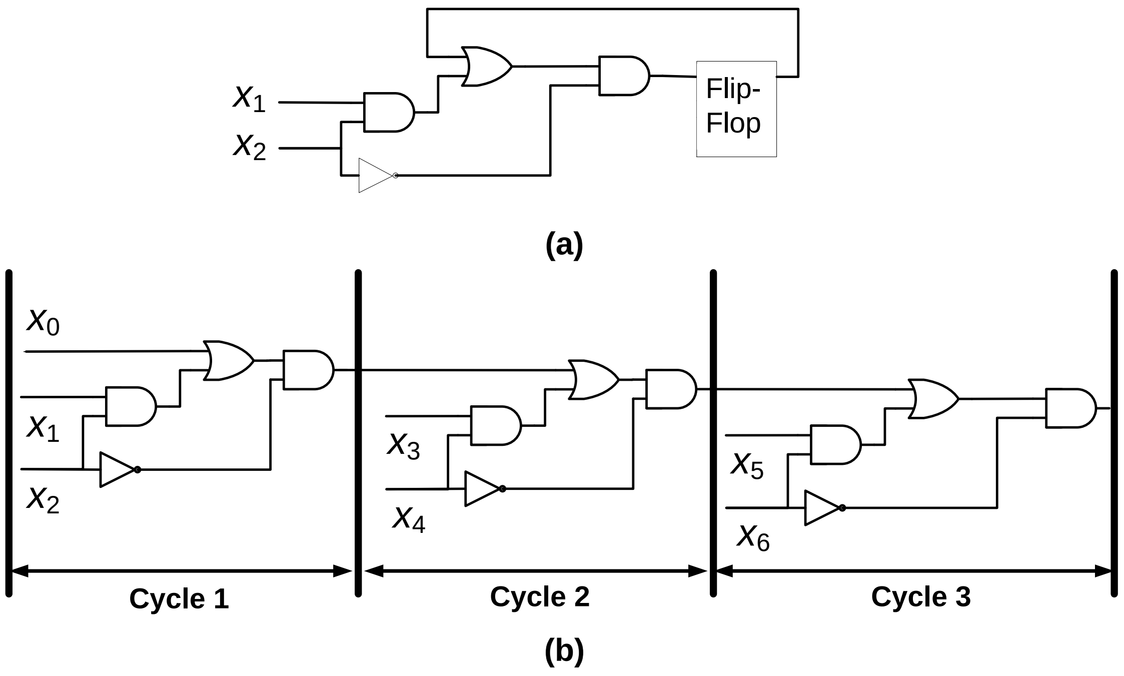


Figure 4.1. Illustration of the time-frame expansion technique: (a) original sequential circuit and (b) expanded circuit with three time frames.

Another major modeling issue is that most soft-error estimation methods for sequential circuits assume that an error occurs only in the first clock cycle of a *w*-cycle simulation with no new errors occurring in subsequent cycles. This assumption reduces computational overhead as well as memory usage [8][40]. Although the after-effects of the initial error may linger, the simulated environment is implicitly assumed to be error-free in cycles 2 through *w*. However, this assumption is unrealistic, and often underestimates the effect of soft errors. An accurate sequential soft-error analysis method should allow a soft error to occur in any clock cycle, and be capable of simulating the interactions among errors occurring in different cycles.

One representative SER estimation method for sequential circuits is MARSS [40], which is capable of simulating all three masking effects at the gate level, as shown in

Figure 1.15. MARSS defines SER in terms of the *mean error susceptibility* (MES), the density of high energy particles per unit area. For a signal *s* in a sequential circuit, the MES of *s* is the probability that an erroneous value is observed on *s* in a particular cycle. It is calculated at the electrical (transistor) level in terms of the duty cycle, signal amplitude (voltage), and gate-error probabilities.

However, MARSS has difficulty simulating circuits over many cycles because of the way it employs the time-frame technique. As discussed above, the circuit size and the number of primary inputs and outputs grow linearly with the number of simulated cycles, and memory usage becomes unmanageable after a few simulated cycles. Take S298, a small 86-gate ISCAS-89 benchmark circuit, as an example. MARSS requires 6,900 seconds to evaluate soft-error effects in S298 for a 10-cycle simulation, whereas the runtime for the proposed SAMPLE method under the same conditions is less than 0.4 seconds. Yet another source of inaccuracy in MARSS is that it assumes an error occurs only in the first clock cycle of a *w*-cycle simulation, and no new errors arrive in subsequent cycles.

4.2 SER Measurement in Sequential Circuits

We defined signal probability for combinational circuits in Chapter 1, and we now extend this definition to sequential circuits. Consider a sequential circuit *SC* that is being analyzed or simulated over *w* clock cycles. Let $v(k)$ denote the input vector applied to *SC* in cycle *k*, where $1 \leq k \leq w$, and *w* is the maximum number of simulation cycles. The input vectors are characterized by some probability distribution, which determines the probabilities of all signals appearing in *SC* at any time. The probability of a particular gate g in SC outputting a 1 in clock cycle k is its signal probability, and is denoted by $p(g(k))$.

Computing signal probabilities in a sequential circuit is more difficult than in a combinational circuit because feedback loops in the sequential circuits can change the signal probabilities of the secondary inputs in every simulation cycle. This significantly complicates the simulation process in runtime as well as in memory usage.

The occurrence of one or more errors e at gate g during the first k clock cycles of circuit operation can change $p(g(k))$ to $p(g^e(k))$. Thus, the difference in signal probabilities between the error-free and erroneous cases $|\, p(g(k)) - p(g^e(k))\, |$ indicates the overall impact of e on signal s in cycle k. Given a sequential circuit SC, we define the *circuit error probability* (CEP) in the kth cycle, denoted $CEP(k)$, as the average difference in signal probabilities between the error-free and erroneous cases over all m primary outputs Z of SC.

$$CEP(k) = (\textstyle\sum_{z \in Z} |\, p(z(k)) - p(z^e(k))\, |)/m$$

CEP represents how likely an incorrect signal is to be observed at a typical primary output z in a particular cycle. The SER of a sequential circuit in each cycle will be measured here by the corresponding CEP. We also introduce a general sequential soft-error model denoted $SE(k)$, which allows errors to occur in the first k clock cycles, where $1 \leq k \leq w$ and w is the number of simulation cycles. The $SE(w)$ error model thus allows soft errors to occur in any cycle during the entire simulation period. Most existing error models, on the other hand, are of the $SE(1)$ type.

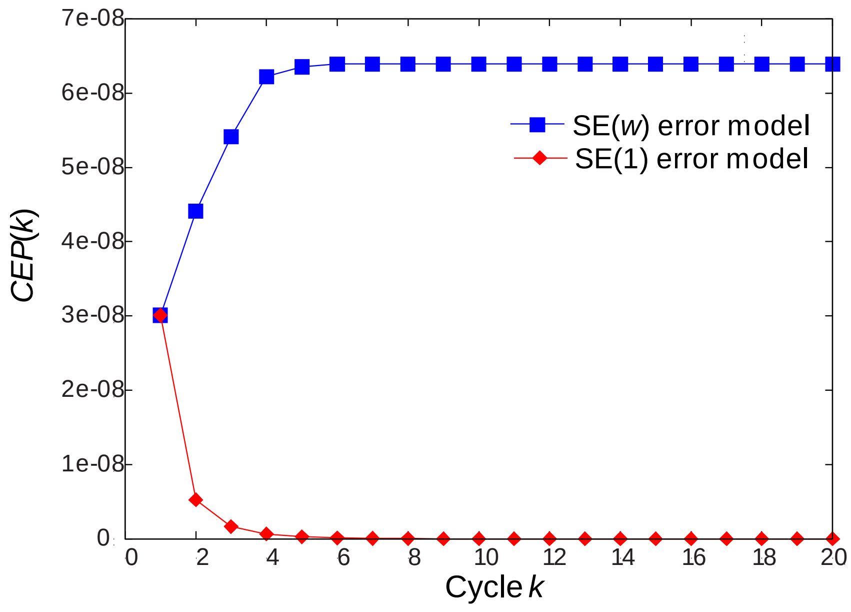
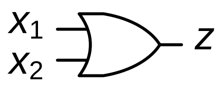


Figure 4.2. Circuit error probabilities for the S298 benchmark assuming a gate-error probability of 10^{-7}.

Figure 4.2 demonstrates that the chosen value of k in $SE(k)$ can have a dramatic impact on SER calculations, even for small sequential circuits. It shows the soft-error effect for the ISCAS-89 benchmark S298 under typical and identical simulation conditions with $k = 1$ (the standard model) [40] and $k = 20$. Observe that the two predicted behaviors are qualitatively as well as quantitatively different. In this case, the CEP for the SE(1) type decreases with increasing clock cycles because no soft errors will occur between the 2nd and wth cycles, whereas the CEP for the $SE(w)$ type increases with time reflecting the interaction among soft errors arriving in different cycles.

4.3 PTM-Based Analysis

We discussed the basic concepts of PTMs and their use for modeling a gate or a circuit's error behavior in Chapter 1 without providing the calculation details. We now introduce the basic operations of PTMs and show how PTMs are typically used in probabilistic calculations [34]. A PTM for an n-input m-output component is a $2^n \times 2^m$ matrix M whose $(i, j)^{\text{th}}$ element is the probability of output j occurring in response to input i. The PTM of a fault-free component is called its ideal transfer matrix (ITM) and the probability of every correct output value in the ITM is 1. Figure 4.3 shows a two-input OR gate, its ITM, and a general PTM. In this gate PTM, p_1, p_2, p_3 and p_4 are conditional gate-error probabilities associated with the corresponding input vectors (rows). Recall that the error probability with respect to input vector v is denoted by $p_{err}(z|v)$; for example, $p_{err}(z|x_1x_2 = 01)$ and $p_{err}(z|x_1x_2 = 10)$ are p_2 and p_3, respectively.



(a)

$$\begin{array}{c} x_1x_2 \\ \begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array}$$

(b)

$$\begin{array}{cc} z = 0 & z = 1 \\ \end{array}$$
$$\begin{bmatrix} 1 - p_1 & p_1 \\ p_2 & 1 - p_2 \\ p_3 & 1 - p_3 \\ p_4 & 1 - p_4 \end{bmatrix}$$

(c)

Figure 4.3. (a) Two-input OR gate, (b) its ITM, and (c) a PTM with various error probabilities for each input vector.

PTM algorithms involve several types of matrix operations, one of which is the tensor or Kronecker product. Given an $a \times b$ matrix A and a $c \times d$ matrix B, their *tensor product* $TP = A \otimes B$ is an $ac \times bd$ matrix whose elements are:

$$TP(i_0 \ldots i_{n+p-1}, j_0 \ldots j_{m+q-1}) = A(i_0 \ldots i_{n-1}, i_0 \ldots j_{m-1}) \times B(i_n \ldots i_{n+p-1} \ldots j_{m+q-1}) \qquad (4.1)$$

For example,

$$\begin{bmatrix}1 & 0 & 2\\0 & 1 & 0\end{bmatrix} \otimes \begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} = \begin{bmatrix}1 & 2 & 0 & 0 & 2 & 4\\3 & 4 & 0 & 0 & 6 & 8\\0 & 0 & 1 & 2 & 0 & 0\\0 & 0 & 3 & 4 & 0 & 0\end{bmatrix} \qquad (4.2)$$

Next we describe how the PTM of an l-level combinational circuit can be constructed from the PTMs of its component gates and wires. First, derive ITMs or PTMs for all components individually. Then for each topological level i containing h component PTMs $\{M_{ij}\}$, form the level PTM $Mi = M_{i1} \otimes M_{i2} \otimes \cdots \otimes M_{ih}$ by repeatedly applying the tensor product. Finally, using ordinary matrix multiplication, multiply all l level PTMs together to form the circuit PTM $M = M_1 \cdot M_2 \cdots M_l$.

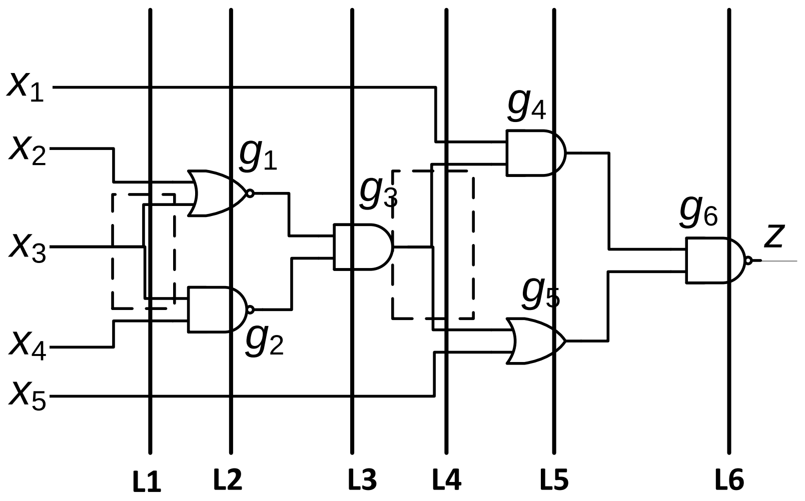
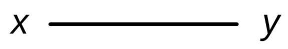


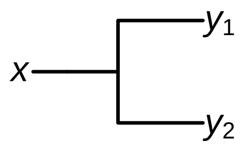
Figure 4.4. Circuit C demonstrating PTM construction; dashed lines enclose fanout gates.

The six-level circuit C in Figure 4.4 shows how a circuit PTM is constructed. First, insert explicit wiring and fanout “gates” into C as needed. It requires six different types of gate PTMs to construct C’s circuit PTM including wire gates. The PTM of a two-input OR gate is shown in Figure 4.3(c), and the others appear in Figure 4.5. Assuming all wires in *C* are error-free, the wiring and fanout gate PTMs here are identical to their ITMs; see Figures 4.5(a) and (b).



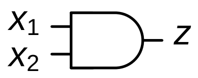
$$\begin{array}{c|cc} x \backslash y & 0 & 1 \\ \hline 0 & 1 & 0 \\ 1 & 0 & 1 \end{array}$$

(a)



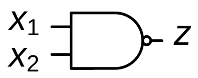
$$\begin{array}{c|cccc} x \backslash y_1y_2 & 00 & 01 & 10 & 11 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{array}$$

(b)



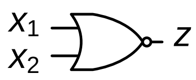
$$\begin{array}{c|cc} x_1x_2 \backslash z & 0 & 1 \\ \hline 00 & 1-p_1 & p_1 \\ 01 & 1-p_2 & p_2 \\ 10 & 1-p_3 & p_3 \\ 11 & p_4 & 1-p_4 \end{array}$$

(c)



$$\begin{array}{c|cc} x_1x_2 \backslash z & 0 & 1 \\ \hline 00 & p_1 & 1-p_1 \\ 01 & p_2 & 1-p_2 \\ 10 & p_3 & 1-p_3 \\ 11 & 1-p_4 & p_4 \end{array}$$

(d)



$$\begin{array}{c|cc} x_1x_2 \backslash z & 0 & 1 \\ \hline 00 & p_1 & 1-p_1 \\ 01 & 1-p_2 & p_2 \\ 10 & 1-p_3 & p_3 \\ 11 & 1-p_4 & p_4 \end{array}$$

(e)

Figure 4.5. Gate PTMs of (a) a single wire, (b) a fanout node, (c) a two-input AND, (d) a two-input NAND, and (e) a two-input NOR gate.

After all wiring and fanout gate PTMs are inserted, construct level PTMs M_1 to M_6 for each level of logic. The PTMs of a gate g, a single wire and a j-branch fanout gate are denoted by G, the identity matrix I_2, and F_j, respectively. The level PTMs are

$$M_1 = I_2 \otimes I_2 \otimes F_2 \otimes I_2 \otimes I_2$$

$$M_2 = I_2 \otimes G_1 \otimes G_2 \otimes I_2 \otimes I_2$$

$$M_3 = I_2 \otimes G_3 \otimes I_2 \otimes I_2$$

$$M_4 = I_2 \otimes F_2 \otimes I_2$$

$$M_5 = G_4 \otimes G_5$$

$$M_6 = G_6$$

The final circuit PTM is

$$M = M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot M_5 \cdot M_6$$

which is a 32×2 matrix.

Once the overall PTM is known, the joint output signal probabilities J can be calculated very easily by multiplying the input signal probability distribution, denoted by a (row) vector V, by the circuit PTM M thus:

$$J = V \cdot M \tag{4.3}$$

For example, if all gates in C have the same gate output error probability $p_{err} = 0.1$ associated with any input vector v, then C's joint output signal probabilities are

$$J = [0.03125\ 0.03125\ \ldots\ 0.03125] \cdot M = [0.81, 0.19]$$

since C is a single-output circuit, $J[0, 1]$ is z's erroneous signal probability $p(z^e)$.

The circuit PTM M of a single-output circuit with r primary inputs is a $2^n \times 2^1$ matrix. For such two-column PTMs, one entry associated with the input vector v (in the v^{th} row) is the probability that the circuit produces the correct result, while the other entry is the error probability with respect to v. For a circuit (or a sub-circuit) PTM M, the error probability associated with a particular input vector v is denoted by $p_{err}(M|v)$. Like the gate-error probability, the error probability of a circuit is also the conditional probability of one particular input vector. For a single-output combinational circuit C whose ITM and PTM are I and M, respectively, if the input probability distribution is V, C's joint output signal probabilities J_{ITM} and J_{PTM} can be obtained using (4.3). The CEP of C is given by

$$| J_{\text{ITM}}[0, 1] - J_{\text{PTM}}[0, 1] |$$

Although Krishnaswamy et al. [34] proposed using algebraic decision diagrams (ADDs) and developed several heuristics to reduce the computational overhead of large matrix operations as mentioned in Chapter 1, their modified PTM calculations are restricted to small combinational circuits because they still construct the circuit PTM from large level PTMs using many large matrix multiplications and tensor operations.

The PTM construction approach presented above has the advantage of correctly accounting for all a circuit's signal probabilities, even when the signals are highly correlated due to fanout and subsequent reconvergence. In other words, all logical masking effects are implicitly accounted for. However, this accuracy comes at considerable computational cost [34]. To reduce this cost, we developed a new PTM construction algorithm that combines a new partitioning method and fault simulation technique. Our algorithm can accurately generate a circuit PTM, even with very small probability values without using any explicit matrix multiplication or tensor operations.

4.4 Circuit Partitioning

To estimate signal probability efficiently in sequential circuits, we introduce two new concepts: circuit partitioning and multi-cycle probability calculation. We first partition the combinational part of a sequential circuit into its supergates, and derive the circuit PTM. The gate signal probabilities for each cycle can be obtained by directly multiplying the circuit PTMs by the corresponding input probability distributions.

A major challenge in signal probability computation is to estimate the output probability of a fanout-reconvergent structure quickly, while maintaining high accuracy. Several methods have been developed for this purpose [25][58]. We again apply the supergate partitioning method presented in Chapter 2. Recall that a supergate typically encloses one or more fanout-reconvergence structures associated with its output. As shown Figure 4.6, the circuit in Figure 4.4 can be partitioned into two supergates, namely, $SG(g_6) = \{g_4, g_5, g_6\}$ and $SG(g_3) = \{g_1, g_2, g_3\}$.

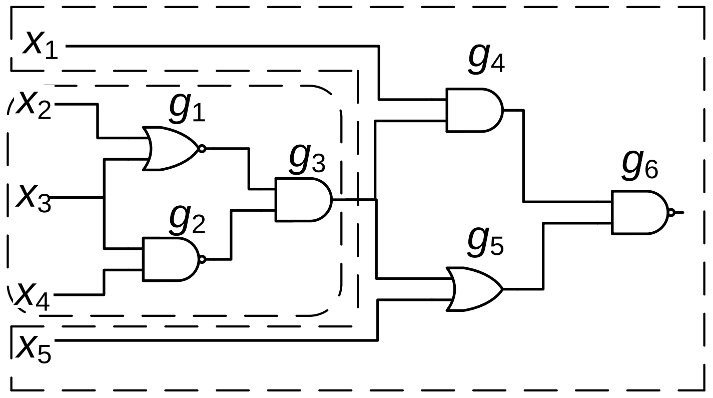


Figure 4.6. The circuit of Figure 4.4 with its two supergates marked by dashed lines.

Some supergates may have many inputs, *e.g.*, $r \geq 32$, and generating circuit PTMs for such cases may be infeasible. To guarantee that the size of all circuit PTMs is manageable, we apply a heuristic called "cone clustering" to each supergate for which $r \leq r_{max}$, where r_{max} is the maximum number of primary inputs in a cluster. The cone-clustering heuristic traverses a supergate *SG* from output to inputs. The initial cone *C* consists of the output gate *g* of *SG*. Gates connected to the inputs of *g* are added level by level to *C* until the number of primary inputs reaches r_{max}. The final inputs of *C* become the outputs of new cones, which are similarly formed. The heuristic is executed recursively until all gates of *SG* belong to at least one cone. An example is shown in Figure 4.7.

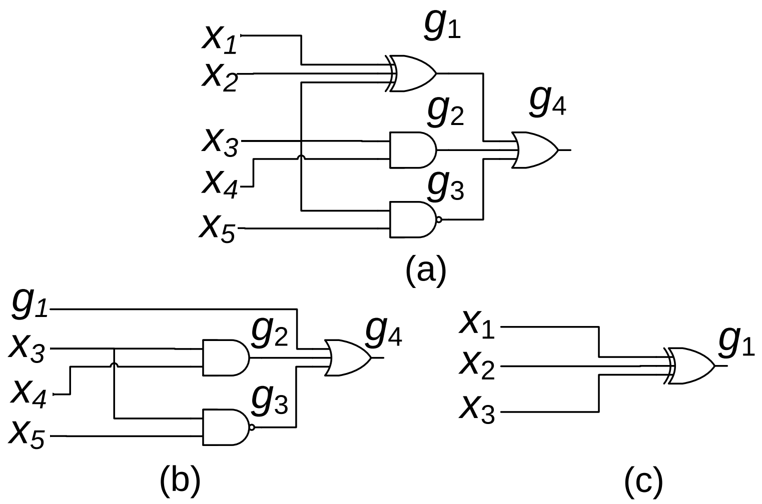


Figure 4.7. (a) Original circuit consisting of a single supergate; (b-c) the two sub-circuits resulting from cone clustering with $r_{max} = 4$.

Some internal clustered cones may contain incomplete fanout-reconvergence structures, which can lead to inaccurate probability calculations. In practice, however, very few supergates need to be reduced by cone-clustering. For example, in the case of the ISCAS-89 benchmarks, only 0.9% of supergates have $r \geq 10$. To determine the value of r_{max} that makes the best trade-off between accuracy and efficiency, we carried out simulation experiments with various r_{max} = 8, 10, 12, and 16 values, and measured their accuracy by comparing the results with signal probabilities obtained by MC sampling. According to these experiments, 10 inputs are accurate enough for most cases compared with r_{max} = 16 because the average inaccuracy of r_{max} = 10 is only 0.64% higher than that of r_{max} = 16. Our simulation data also show that, in most cases, we can maintain 98% accuracy over 100 simulated cycles.

4.5 Fault Simulation-based PTM construction

After supergate partitioning and (if necessary) cone clustering, we generate the PTM of each circuit partition. The basic PTM algorithm used in [34] provides a way to generate circuit PTMs by constructing and merging level PTMs, which preserves complete accuracy. Generally, the supergates are much smaller than the original circuit, thus making the basic construction procedure more efficient due to the fact that fewer level PTMs need to be calculated and merged in a supergate. However, many large matrix operations may be required for the construction process, and the computational complexity in terms of runtime and memory usage may be exponential in the circuit size. As a result, the basic PTM construction algorithm is still limited to relatively small circuits, even when the circuits are partitioned into supergates, so a speedup heuristic is needed to enhance the efficiency of matrix construction.

Since supergates and cones are single-output sub-circuits, their PTMs are two-column matrices. Constructing a two-column PTM is equivalent to calculating the circuit output error probabilities associated with all input vectors. In addition, recall that soft errors have very low occurrence probability, so we can assume that any observable circuit error is caused by a single erroneous gate, *e.g.*, a gate that receives a particle strike. In other words, the circuit error is often caused by a single erroneous gate at any time, so ignoring the probability of multiple errors has no significant impact on accuracy.

Finding all gates whose incorrect output values can flip the primary output is equivalent to identifying gates whose stuck-at-faults can propagate to the primary output. These may be called "critical" gates, where a gate is critical if their value change can be observed at primary outputs. For a given input vector, only some stuck-at-faults can propagate errors to the primary output, while others are blocked by logic masking effects. One method for identifying critical gates is critical-path tracing (CPT) proposed by Abramovici et al. [2], which tries to identify gates made critical by a given input vector.

Here, we introduce a CPT-based matrix construction heuristic called CPT-MC, which can efficiently generate circuit PTMs for cases with low gate-error probability. Let M be the PTM of a circuit partition. CPT-MC first identifies critical gates associated with each input vector by means of the CPT method. Then for each input vector v, the circuit output error probability $p_{err}(M|v)$ of M is computed as

$$p_{err}(M|v) = \sum_{i\in\Omega} p_{err}(g_i|v_i) \cdot (\Pi_{j\in\Omega,\, j\neq i}(1 - p_{err}(g_j|v_j))\,) \tag{4.4}$$

where Ω is the current set of critical gates, while $p_{err}(g_i|v_i)$ and $p_{err}(g_j|v_j)$ are the gate-error probabilities of g_i and g_j associated with the current input vectors v_i and v_j, respectively. For instance, the critical gates associated with the input vector v = 010 of the circuit in Figure 4.8 are g_4 and g_6, and the corresponding circuit output error probability is

$$p_{err}(M|v) = p_{err}(g_4|v_4)(1 - p_{err}(g_6|v_6)) + p_{err}(g_6|v_6)(1 - p_{err}(g_4|v_4)) = 1.52 \times 10^{-6}$$

CPT-MC is recursively applied until the circuit output error probabilities associated with all input vectors are obtained.

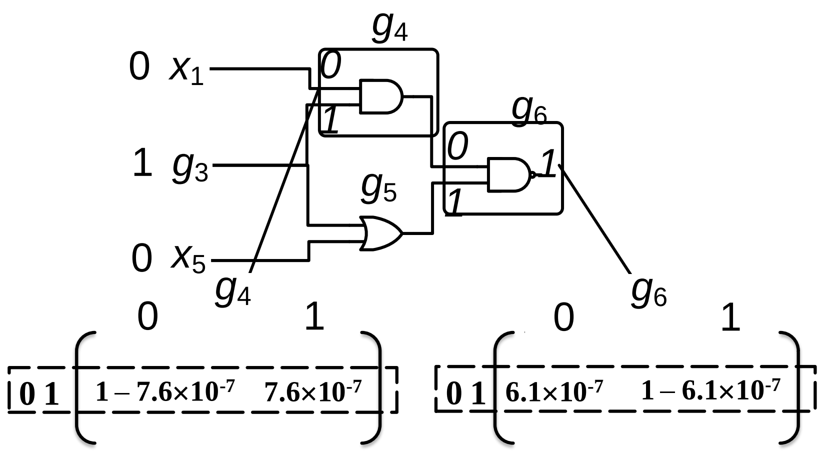


Figure 4.8. Supergate of g_6 with input vector (0, 1, 0) and critical gates g_4 and g_6.

4.6 Probabilistic Calculation Method

After calculating a circuit's PTM, its output signal probabilities are obtained by SAMPLE from the initial distribution of input signal probabilities and the PTMs of the partitioned circuit. SAMPLE first applies a given probability to each primary input. After the first cycle, the signal probabilities stored in flip-flops are updated to the corresponding secondary inputs at the beginning of the next cycle. Within each cycle, SAMPLE estimates the primary and secondary output signal probabilities of each sub-circuit in topological order by multiplying its input distribution vector by the circuit PTM according to (4.3). Finally, the signal probabilities of secondary outputs are stored in the corresponding flip-flops at the end of the cycle. Pseudo-code for the resulting probability estimation algorithm is shown in Figure 4.10.

Since a circuit PTM contains the output signal probability associated with each input combination, and SAMPLE preserves the probability of each flip-flop in every cycle, the circuit's probabilistic behavior can be fully expressed in terms of its PTMs and the corresponding input distributions. In other words, the behavior of the target circuit's transition function can be completely represented by the circuit PTMs, and all temporal correlations are implicitly examined by this estimation procedure.

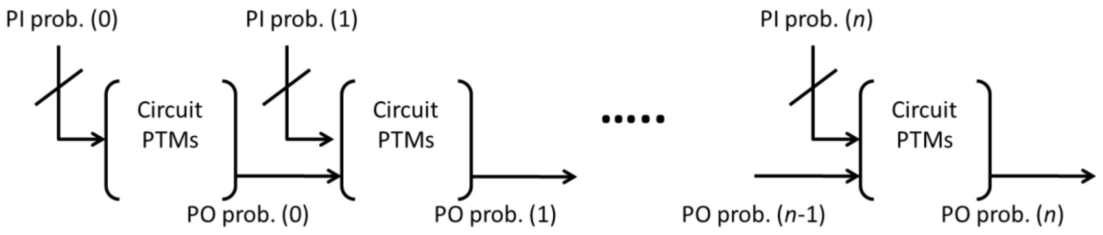


Figure 4.9. Probabilistic estimation procedure for $n + 1$ cycles; note that the same circuit PTMs are used for every cycle.

```
SAMPLE(sequential circuit SC, no. of simulation cycles n, gate PTM
configuration GP_conf, max. no. of inputs $r_{max}$ of a supergate)

    SG_List = Supergate_Partitioning(combinational part of SC)

    for each sg in SG_List

    if (no. of inputs >$r_{max}$)
        Rcone_List = Rcone_Clustering(sg)
        SG_List = SG_List – sg + Rcone_List

    PTM_List = CPT-MC(SG_List, GP_conf, $r_{max}$)

    Initialize input probability distribution

    for i= 1 to n
        for each PTM M in PTM_List in topological order
            Construct input distribution vector V of M
            Output signal probability $J = V \cdot M$

        Update flip-flop values at the corresponding secondary inputs

    END SAMPLE
```

Figure 4.10. Probability estimation algorithm used in SAMPLE.

4.7 SER Estimation

To evaluate the efficiency and accuracy of the proposed algorithm, we applied SAMPLE to the calculation of CEP for all the ISCAS-89 benchmark circuits. We believe that this is the first soft-error estimation algorithm capable of simulating all the ISCAS-89 circuits over 100 cycles.

Accurately calculating the impact of a soft error for all feedback loops of a circuit over several cycles is difficult. Prior work on this problem uses the $SE(1)$ model and various other heuristic simplifications to reduce the computational complexity. As discussed in Section 4.2, we measure the SER of a sequential circuit by its CEP, which can be directly obtained from its output error-free and erroneous signal probabilities. SAMPLE can efficiently calculate the signal probability even for very large sequential circuits. Moreover, SAMPLE is able to simulate any $SE(k)$ error model, and all logical masking effects are automatically and implicitly included in the calculations.

To evaluate the SER of the ISCAS-89 circuits using SAMPLE, we assigned two representative gate-error probabilities to the every input vector v, $p_{err} = 10^{-3}$ and 10^{-7}, and simulated $w = 100$ cycles. The first case demonstrates that SAMPLE can maintain high accuracy even with high gate-error probability. The second case simulates the situation where, due to soft errors, a gate has a very small probability 10^{-7} of producing an erroneous value in every cycle. The experiments were performed on an Intel Quad-Core, 2.35 GHz, 64-bit PC, with 4GB RAM. They produced the accuracy, runtime and memory data appearing in Table 4.1. In order to validate the accuracy of the SER evaluation results, we independently estimated the circuit error probabilities of all circuits by MC sampling, and compared the results with those of SAMPLE. For each circuit, we repeatedly apply randomly-generated input vectors with input signal probabilities of 0.5, and simulate the circuit until the output signal probabilities stabilize to fixed values.

Column 5 of Table 4.1 indicates that the maximum and average errors of the SER analysis with the *SE*(*w*) model over *w* = 100 cycles are 6.71% (S953) and 2.16% respectively. The maximum runtime and memory usage are 0.69 hours and 510 MB (S15850) for 100-cycle simulation.

Table 4.1. Performance results for the ISCAS-89 benchmark circuits (including the largest circuits).

Circuit	PTM runtime (s)		Memory usage(MB)	Error(%)
	Construction	Evaluation		
S298	0.21	<0.01	0.02	1.42
S510	3.79	0.04	0.06	1.09
S713	2.79	0.06	1.28	1.09
S838	0.62	0.02	0.44	1.83
S953	2.07	0.05	0.37	2.14
S953	2.06	0.02	0.65	6.71
S5378	6.59	0.16	5.24	0.43
S1423	18.22	0.20	7.82	0.53
S1238	37.57	0.05	2.03	0.93
S9234	1,075.16	0.74	24.70	4.71
S13207	138.24	2.82	115.26	2.31
S15850	614.92	10.70	510.86	4.31
S35932	987.80	4.42	7.43	2.63
S38417	1,796.79	83.57	55.40	2.97
S38584	2,155.85	352.87	171.04	3.72

Figures 4.2 and 4.11 show four examples of SER behavior for the *SE*(1) and *SE*(*w*) error models in representative sequential circuits, assuming primary input signal probabilities of 0.5 in every cycle. These two sets of results serve to illustrate two

probabilistic behaviors of a sequential circuit with different boundary conditions. As Figure 4.2 shows, the SER with the *SE*(1) model decreases dramatically in the first few cycles, which implies that most soft errors have a high chance of quickly escaping from the circuit. The SER with the *SE*(*w*) error model in Figure 4.2 rises in the first few cycles and grows much more slowly thereafter. Furthermore, the stable value (steady-state) with *SE*(*w*) is almost an order of magnitude higher than that with *SE*(1). This suggests that, in certain cases, the accumulated effect of soft errors has great impact, and should not be ignored. In other words, the *SE*(1) model may underestimate the SER after just a few cycles, and lead to unrealistic prediction of overall soft-error behavior.

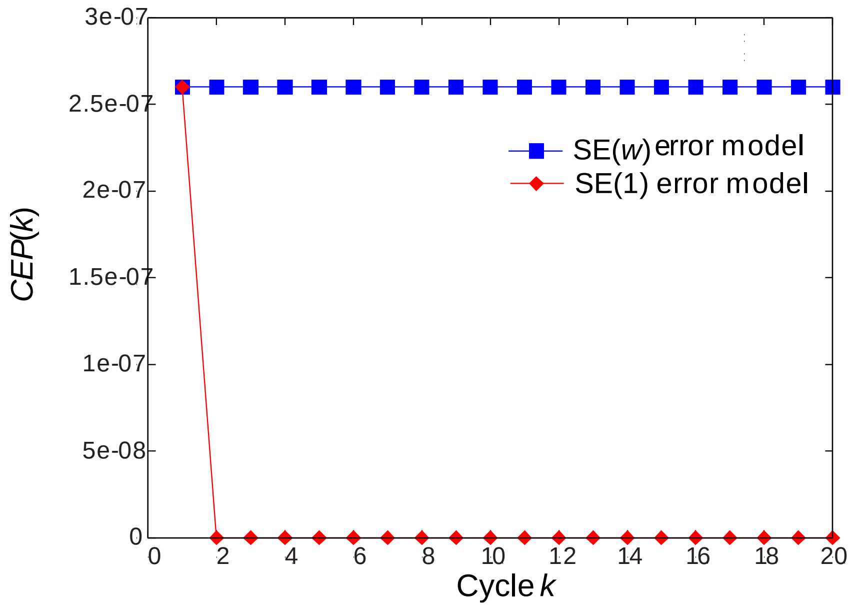


Figure 4.11. Circuit error probabilities for the S1238 benchmark assuming a gate-error probability of 10^{-7}.

Some pipeline-structured circuits such as S1238 contain no global feedback loops and so have different SER behavior from most other sequential circuits. SER results for

S1238 with the *SE*(1) and *SE*(*w*) models are shown in Figure 4.11. The SER with the *SE*(1) drops sharply to zero at the second cycle because no captured soft error can propagate to any secondary input in the next cycle. Therefore, any soft error that appears in a particular cycle will disappear in the next cycle. Similarly, since no global feedback is present in these circuits, the SER with the *SE*(*w*) model remains constant in each cycle. In other words, the SER behavior of such pipeline-structured circuits is similar to that of a combinational circuit.

4.8 Summary

We have presented a new probabilistic calculation tool for sequential circuits, SAMPLE, which can be used for accurate modeling of non-deterministic effects, and scales well to very large circuits. Prior work on this issue tends to be inaccurate and does not scale well. Unlike some prior methods, SAMPLE is capable of simulating fairly realistic error situations, and also provides a fast and accurate estimation of SER by combining supergate partitioning, critical-path tracing, and the PTM technique.

As shown in Chapter 1 and this chapter, the basic PTM method is powerful in the sense that it can accurately handle correlations associated with probabilistic calculations and logical masking effects. However, this method does not scale well to very large circuits even with the help of the ADD and other techniques, because it requires a single circuit PTM for the given circuit, and the problem of generating the circuit PTM may be intractable. SAMPLE solves this scalability problem by partitioning a circuit into smaller sub-circuits, and by generating PTMs for the sub-circuits. Therefore, it dramatically reduces the complexity of circuit PTM construction, and makes PTM size much more manageable. Moreover, SAMPLE can analyze a sequential circuit over hundreds of cycles without increasing memory usage significantly, which is useful for applications require long-term circuit simulation.

CHAPTER 5

Conclusions and Future Work

Due to the massive complexity of modern ICs, traditional deterministic EDA methods used to solve circuit design problems are no longer sufficient. Probabilistic approaches are needed that can account for complex process variations and soft-error effects. The quality of a probabilistic method largely depends on how error behavior and masking are modeled, and how signal correlations are handled. Our analysis of signal probability and soft errors has shown that existing probabilistic methods are limited to small circuits or tend to provide inaccurate results, usually because their models are oversimplified or their assumptions are unrealistic.

To deal with these issues, we have proposed some novel approaches to the following three problems: circuit sampling, probability modeling, and soft-error estimation for sequential circuits. First, we have proposed a way to use redundant signals for fast and accurate sampling-based calculation of signal probabilities. Second, we have applied trigonometric functions and Taylor expansions to obtain an accurate probability representation method that integrates signal and error probabilities. Finally, we have combined circuit partitioning, fault simulation, and matrix operations to create an efficient method for tracing soft errors in sequential circuits over many cycles.

The reminder of this chapter discusses our major contributions, and explores promising directions for future research.

5.1 Summary of Contributions

The main contributions of this thesis are as follows:

- An accurate circuit simulation methodology for signal probability calculation that is useful for validating EDA heuristics. Experimental results show that this technique is one to three orders of magnitude faster than conventional Monte Carlo (MC) methods, while maintaining the same accuracy level.
- A novel probability representation based on trigonometric functions and Taylor expansions whose calculation accuracy and efficiency can be fully controlled by setting the number of expansion terms.
- A method for improving the applicability of probabilistic transfer matrices (PTMs) to large combinational circuits by reducing the overhead of constructing circuit PTMs based on circuit partitioning and a fault simulation technique.
- A PTM-based and scalable simulation technique for estimating accumulated soft-error effects in sequential circuits, which can easily simulate the error behavior over many cycles without increasing the memory usage significantly.

Chapter 2 introduced an efficient and accurate sampling methodology, Reduced-Ordered Monte Carlo (ROMC), for signal probability calculation in logic circuits. ROMC reduces the sample variance by analyzing and exploiting partial redundancy among input signals. Based on redundancy analysis, ROMC (1) removes unnecessary sample sequences by identifying multiplexer-like structures, and (2) it prioritizes input signals based on their observability (influence). Our simulation results suggest that ROMC is robust because it always requires fewer samples than conventional MC methods for the same accuracy levels in all the benchmark circuits we considered.

In Chapter 3, we developed a probability calculation method, Trigonometric Probability Calculation (TPC), for modeling and analyzing unconditional errors in combinational circuits. TPC can accurately and efficiently estimate circuit reliability and gate susceptibility to soft errors. TPC is accurate because it simulates the impact of multiple soft errors; in contrast, existing methods only simulate circuits for the single-error cases. TPC is efficient because it employs trigonometric techniques to model the signal probabilities of gates as angles. This enables it to simulate the effect of a gate's error probability by a small rotation, which converts multiplications to additions.

In Chapter 4, we proposed a powerful probabilistic methodology, Scalable and Accurate Matrix-based Probabilistic Algorithm for Logic Signal Estimation (SAMPLE), for dealing with conditional error models in sequential circuits. Unlike the commonly-used unconditional error models, the error probabilities of conditional models can vary with respect to the input vectors, and are therefore more general. SAMPLE can simulate situations where soft errors occur at logic gates in one or more cycles by using PTMs to describe conditional errors. We have successfully improved the scalability of PTM-based calculations, which were previously confined to relatively small circuits. In particular, we applied supergate partitioning and CPT fault simulation methods to generate small PTMs that can account for most signal correlations, and can accurately model soft errors. Unlike prior work, SAMPLE is capable of simulating cases when soft errors can occur in any simulation cycle, and can trace the accumulated effects of errors through multiple cycles. In addition, it accounts for structural and temporal correlations, and therefore can process very large circuits over many cycles while maintaining relatively high accuracy.

5.2 Future Work

In this section, we discuss several potential extensions of our work. We first propose to use implications for further improving ROMC's simulation efficiency. Next, we propose to apply ROMC to SER estimation in circuits with unconditional error models. Finally, we propose to employ learning techniques to reduce the complexity of PTM construction.

5.2.1 Sampling with Implication

Although ROMC generally achieves high simulation accuracy with few samples, there are cases where its efficiency can be improved by identifying unnecessary samples using implications. An implication is present in a circuit when a partial or complete assignment *u* leads to a known output *r*; this is denoted by ($u \rightarrow r$). Consider the circuit *C* of Figure 2.5, which is reproduced in Figure 5.1. If inputs *ab* = 11, *z* immediately becomes 0, regardless of inputs *c* and *d*'s values. In other words, *ab* = 11 implies *z* = 0. Another example in *C* is ($ab = 01 \rightarrow z = 1$).

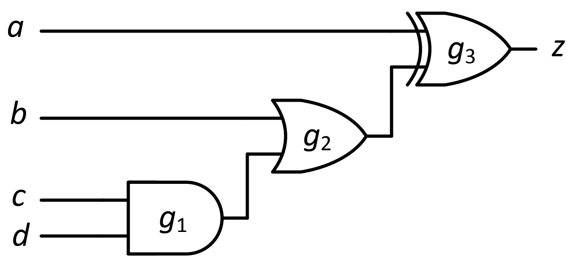


Figure 5.1. Four-input circuit.

Now we show how implication can discover unnecessary samples. If *C*'s sample size is $2^2 = 4$, ROMC will create a prioritized input order = *a*, *b*, *c*, *d*, and generate the four samples *ab* = 00, 01, 10, 11. However, the two implications, (*ab* = 11→ *z* = 0) and (*ab* = 01→ *z* = 1) show that inputs *c* and *d* are redundant with respect to *z* for the inputs *ab* = 01 and *ab* = 11. Hence, we can remove the two samples associated with *ab* = 01 and 11 without affecting the simulation accuracy.

Incorporating such implications effectively, however, remains a challenge because not all implications are useful for circuit sampling, e.g., (*abcd* = 0010 → *z* = 0). Identifying all useful implications may require an exhaustive search [15], which is impractical in large circuits.

Next, we suggest a way to discover many useful implications for circuit sampling. Given a circuit *C* with a sample size = 2^k, first determine the *k* most observable inputs. Then, recursively apply implication to identify cases where the output signals are fully determined by any one, two, …,*k* inputs, for the *k* most observable inputs. In Figure 5.1, since the two most observable inputs are *a* and *b*, this implication technique would first check if *z*'s value is determined by *a* or *b* alone. Since g_3 is an XOR gate, no single input can determine g_3's value. It would then deduce that output z is fully determined by ab = 01 or 11.

However, a problem with this method is that the number of cases that need to be examined for the k most observable inputs is $2^k - 1$, which grows exponentially with k. Therefore, how to reduce the complexity of discovering valuable implications is a key challenge in this area.

5.2.2 Soft-Error Estimation Using Sampling

Chapter 3 introduced a way to represent unconditional error probabilities by inserting additional XOR gates and input signals. This method simulates soft-error effects by converting a circuit C into an XOR-extended circuit C^e. This suggests that it should be possible to apply ROMC to evaluate soft-error effects in combinational logic. Although C^e is about twice the size of C, ROMC should be capable of handling C^e because it is several orders of magnitude faster than MC.

ROMC benefits from the use of signal redundancy and observability for reducing the size of the sample space and prioritizing inputs. However, since the signal probabilities of the inputs inserted into C^e and connected to XOR gates are not 0.5, ROMC cannot be directly applied to C^e without suitable modifications. This is because compatible inputs can share common sample sequences only if all their signal probabilities are 0.5 (as shown in Section 2.6), and observability estimation results can be inaccurate when input probabilities are not 0.5. When ROMC evaluates a signal s's observability, it implicitly assumes all input probabilities are 0.5, and calculates the chances of observing s and s' at primary output. Consider the circuit of Figure 5.2. If $p(x^e)$ = 0.5, the probabilities of observing y and y' at z are equal; however, when $p(x^e) = 10^{-9}$, the probabilities of observing $z = y$ and y' are $1 - 10^{-9}$ and 10^{-9}, respectively. This shows that y's observability at z is highly affected by x^e's input probability.

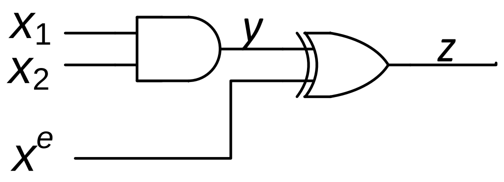


Figure 5.2. XOR-extended circuit for a two-input AND gate.

Jeavons et al. [32] propose a technique for converting arbitrary input signal probabilities to 0.5 in a circuit without affecting the circuit's output signal probabilities by using linear feedback shift registers (LFSRs). This suggests a way to enable ROMC to identify compatible signals, and accurately resolve signal observability in XOR-extended circuits. However, the size of the LFSRs grows linearly with decreasing probability values. Since gate-error probabilities are generally very small, how to reduce the size of inserted LFSRs without affecting accuracy significantly is a problem for future research.

There are other circuit properties that might be exploited to reduce simulation complexity without affecting accuracy significantly. For instance, if all gate-error probabilities are 10^{-9}, then the chance of having three errors is 10^{-27}, which is very unlikely. In addition, we can only consider situations where soft errors are highly observable. Figure 5.3(b) shows one such case where g_3 and g_4 are more observable than g_1 and g_2 in Figure 5.3(a). Therefore, developing a method to determine locations where errors have the most significant impact on circuit may be an interesting task for future research.

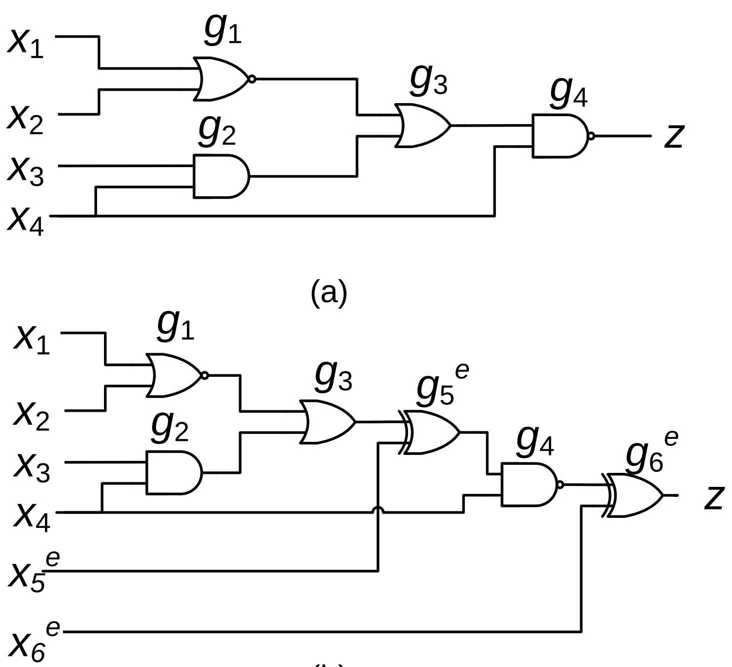


Figure 5.3. (a) Four-input circuit; (b) its XOR-extended circuit for simulating soft errors occurring at g_3 and g_4.

5.2.3 PTM Construction Using Learning

In this section, we consider how learning can be used to further speed up the PTM construction procedure discussed in Chapter 4. Learning is often an effective way to avoid unnecessary computation by continuously storing "valuable" intermediate results that can be frequently reused [35][53].

Consider the circuit *C* of Figure 5.4. *C* can be fully partitioned into two supergates, $SG(g_1) = \{x_1, x_2, g_2, g_3, g_4\}$ and $SG(g_4) = \{x_3, x_4, g_1\}$. After circuit partitioning, the critical-path-tracing (CPT) heuristic introduced in Section 4.5 can be used to examine the critical gates for individual input vectors of $SG(g_1)$ and $SG(g_4)$. For example, to evaluate the error probability associated with input vector $x_1x_2g_1 = 000$ of $SG(g_4)$, a typical circuit simulation is first applied to $SG(g_4)$, and g_4 is identified as its only critical gate. The next step is to evaluate the error probability associated with input vector $x_1x_2g_1 = 001$. Since these two input vectors $x_1x_2g_1 = 000$ and 001 differ in one bit g_1, only gates along g_1's propagation paths need to be "learned". Moreover, if all g_1's propagation paths are blocked due to logic masking, the simulation procedure can be terminated immediately.

In this case, only g_2's simulation result needs to be learned since g_2 is the only gate at the first level in g_1's propagation path. In the case of $x_1x_2g_1 = 001$, g_1 is blocked by g_2 because $x_2 = 0$. Therefore, the simulation procedure stops at the first level, and the values of the rest of the gates (g_2, g_3 and g_4) remain unchanged. This suggests that the simulation result associated with $x_1x_2g_1 = 000$ is worth learning because it can be reused when the $SG(g_4)$ is simulated with $x_1x_2g_1 = 001$.

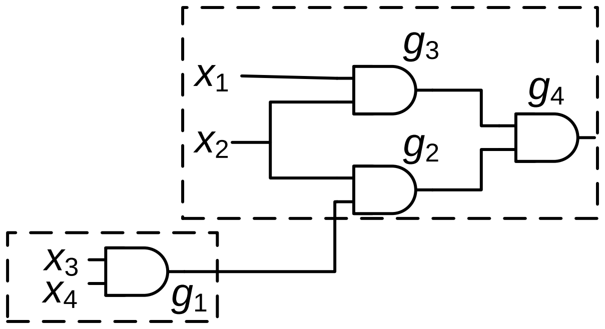


Figure 5.4. Four-input circuit; its two supergates are marked with dashed lines.

Now we suggest a possible way to apply learning to critical-gate identification for a set of input vectors. Suppose we are given a circuit and two input vectors v_{i-1} and v_i, that only differ in primary input bit x_j. If the simulation result associated with v_{i-1} is known, v_i

can be simulated by the following two steps. First, store the simulation result associated with v_{i-1}, and then identify the propagation paths of x_j. Second, re-examine the gates along with propagation paths level by level from the primary inputs to primary outputs. If all x_j's propagation paths are blocked at a particular level, the simulation procedure is terminated.

However, identifying simulation results that are useful for other simulation cases is not easy. Another challenge is to identify learned results that are no longer useful for cases not yet simulated. If k simulation results (associated with k different input vectors) are learned and stored in memory, any new simulation needs to compare its outcome to all k learned results, which can significantly affect the simulation efficiency if k is large. Therefore, developing a method that can identify and store reusable results, and automatically delete information that is no longer needed for better simulation efficiency and memory management is key to the success of applying the learning technique to PTM construction.

In closing, we have presented a set of methods for efficiently calculating signal probabilities and analyzing soft-error effects in logic circuits. Our research aims to characterize these circuits' statistical behavior and their vulnerabilities to various non-deterministic phenomena. We hope that this work proves useful for improving the performance and reliability of new ICs.

BIBLIOGRAPHY

[1] A. Abdollahi: “Probabilistic decision diagrams for exact probabilistic analysis,” *Proc. Intl. Conf. Comput.-Aided Design*, pp. 266-272, 2007.

[2] M. Abramovici *et al.*: "Critical path tracing: an alternative to fault simulation," *Proc. Design Automation Conf.*, pp. 214-220, 1983

[3] ACM/SIGDA: *Benchmarks newsletter*, http://www.cbl.ncsu.edu/benchmarks/PDWorkshop93/, 1993.

[4] K. Agarwal *et al.*: “Fast Characterization of threshold voltage fluctuation in MOS devices,” *IEEE Trans. Semicond. Manuf.*, vol. 21, pp. 526-533, 2008.

[5] S. B. Akers Jr.: “On a theory of Boolean functions,” *Jour. SIAM*, vol. 7, pp. 487-498, 1959.

[6] H. Ando *et al*.: "Validation of hardware error recovery mechanisms for the signal probability SARC64 V microprocessor," *Proc. Intl. Conf. Dependable Syst. & Networks*, pp. 62-69, 2008.

[7] G. Asadi and M. B. Tahoori: “An analytical approach for soft error rate estimation in digital circuits,” *Proc. Intl. Symp. Circuits & Syst.*, pp. 2991-2994, 2005.

[8] H. Asadi and M. B. Tahoori: "Soft error rating computation in sequential circuits," *Proc. Intl. Conf. Comput.-Aided Design*, pp. 497-501, 2006.

[9] Australian Transport Safety Bureau: *Aviation safety investigation & report*, http://www.atsb.com.au/publications/investigation_reports/2008/AAIR/ao-2008-070.aspx, 2008.

[10] R. I. Bahar *et al.*: “Algebraic decision diagrams and their applications,” *Proc. Intl. Conf. Comput.-Aided Design*, pp. 188-191, 1993.

[11] N. Battezzati *et al*.: “Monte Carlo analysis of the effects of soft error accumulation in SRAM-based FPGAs,” *IEEE Trans. Nucl. Sci.*, vol. 55, pp. 3381-3387, 2008.

[12] A. Bhanu *et al.*: “A more precise model of noise based PCMOS errors,” *Proc. Intl. Symp. Electron. Design, Test & Applicat.*, pp. 99-102, 2010

[13] D. Brelaz: “New methods to color the vertices of a graph,” *ACM Communications*, vol. 22, pp. 251-256, 1979.

[14] F. Brglez: "On testability analysis of combinational networks," *Proc. Intl. Symp. Circuits & Syst.*, pp. 221-225, 1984.

[15] M. L. Bushnell and V.D. Agrawal: *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*, Kluwer, 2000.

[16] R. E. Byrant: “Graph-based algorithms for Boolean function manipulation,” *IEEE Trans. Comput.*, vol. C-35, pp. 677-691, 1986.

[17] T. Chawla *et al*.: “Impact of intra-die random variations on clock tree,” *Proc. Physical Design & Rel. Issues in Nanoscale Analog CMOS Technologies*, pp. 1-4, 2009.

[18] C. H. Chen *et al.*: “Efficient approach for Monte Carlo simulation experiments and its applications to circuit systems design,” *Proc. Simulation Symp.*, pp. 65-71, 2011.

[19] M. R. Choudhury and K. Mohanram: “Accurate and scalable reliability analysis of logic circuits,” *Proc. Design, Automation & Test in Europe*, pp. 1 - 6, 2007.

[20] M. R. Choudhury and K. Mohanram: “Reliability analysis of logic circuits,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, pp. 392-405, 2009.

[21] J. Chung *et al*.: “Path criticality computation in parameterized statistical timing analysis using a novel operator,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, pp. 497-508, 2012

[22] M. A. Cirit: “Estimating dynamic power consumption of CMOS circuits,” *Proc. Intl. Conf. Comput.-Aided Design*, pp. 534-537, 1987.

[23] C. Constantinescu: “Trends and challenges in VLSI circuit reliability,” *IEEE Micro*, vol. 23, pp. 14-19, 2003.

[24] S. Dighe *et al*., “Within-die variation-aware dynamic-voltage-frequency scaling core mapping and thread hopping for an 80-core processor,” *Proc. Intl. Solid-State Circuits Conf.*, pp. 174-175, 2010.

[25] S. Ercolani *et al.*: "Estimate of signal probability in combinational logic networks," *Proc. European Test Symp.*, pp. 132-138, 1989.

[26] G. D. Hachtel and F. Somenzi: *Logic Synthesis and Verification Algorithms*, Kluwer, 1996.

[27] J. M. Hammersley and D. C. Handscomb: *Monte Carlo Methods*, Methuen, 1964.

[28] M. C. Hansen *et al*.: “Unveiling the ISCAS-85 bench-marks: a case study in reverse engineering,” *IEEE Design Test Comput.*, vol. 16, pp.72-80, 1999.

[29] T. Heijmen and A. Nieuwland: “Soft-error rate testing of deep-submicron integrated circuits,” *Proc. European Test Symp.*, pp. 247-252, 2006.

[30] International Technology Roadmap for Semiconductors: *ITRS 2009 edition*, http://www.itrs.net/Links/2009ITRS/Home2009.htm, 2009.

[31] J. Jaffari and M. Anis: “Timing yield estimation of digital circuits using a control variate technique,” *Proc. Intl. Symp. Quality Electron. Design*, pp. 382-287, 2009.

[32] P. Jeavons *et al*.: “Generating binary sequences for stochastic computing,” *IEEE Trans. Info. Theory*, vol. 40, pp. 716-720, 1994.

[33] J. Kahn *et al.*: “The influence of variables on Boolean functions,” *Proc. Symp. Found. Comput. Sci.*, pp. 68-80, 1988.

[34] S. Krishnaswamy *et al.*: "Probabilistic transfer matrices in symbolic reliability analysis of logic circuits," *ACM Trans. Design Autom. Electron. Syst.*, vol. 13, Article No. 8, 2008.

[35] W. Kunz and D. K. Pradhan: "Recursive learning: a new implication technique for efficient solutions to CAD problems–test, verification, and optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, pp. 1143-1158, 1994.

[36] J. L. Leray *et al.*: "Atmospheric neutron effects in advanced microelectronics, standards and applications," *Proc. Intl. Conf. Integrated Circuit Design & Technology*, pp. 311-321, 2004.

[37] A. Lesea *et al.*: "The Rosetta experiment: atmospheric soft error rate testing in differing technology FPGAs," *IEEE Trans. Device and Mater. Rel.*, vol. 5, pp. 317-328, 2005.

[38] J. J. Liou *et al.*: "False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation," *Proc. Design Automation Conf.*, pp. 566-569, 2002.

[39] G. De Micheli: *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.

[40] N. Miskov-Zivanov and D. Marculescu: "Modeling and optimization for soft-error reliability of sequential circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, pp. 803-816, 2008.

[41] A. Mondal *et al.*: "Statistical static timing analysis using symbolic event propagation," *IET Circuits, Devices & Syst.*, vol. 1, pp. 283-291, 2007.

[42] G. E. Moore: "Cramming more components onto integrated circuits," *Electronics*, vol. 38, pp. 8-11, 1965.

[43] F. N. Najm: "On the need for statistical timing analysis," *Proc. Design Automation Conf.*, pp. 764-765, 2005.

[44] S. Nassif *et al.*: "High performance CMOS variability in the 65nm regime and beyond," *Proc. Intl. Electron Devices Meeting*, pp. 569-571, 2007.

[45] M. Omana *et al.*: "A model for transient fault propagation in combinatorial logic," *Proc. Intl. On-Line Testing Symp.*, pp. 111-115, 2003.

[46] K. P. Parker and E. J. McCluskey: "Probabilistic treatment of general combinational networks," *IEEE Trans. Comput.*, vol. C-24, pp. 668-670, 1975.

[47] M. Pedram: "Power estimation and optimization at the logic level," *Jour. High Speed Electron. Syst.* vol. 5, pp. 179–202, 1994.

[48] T. Rejimon and S. Bhanja: "Scalable probabilistic computing models using Bayesian networks," *Proc. Midwest Symp. Circuits & Syst.*, pp. 712-715, 2005.

[49] S. M. Ross: *A Course in Simulation*, Prentice Hall, 1990.

[50] K. Roy *et al.*: "Test consideration for nanometer-scale CMOS circuits," *IEEE Trans. Design Test Comput.*, vol. 23, pp. 128-136, 2006.

[51] S. J. Russell and P. Norvig: *Artificial Intelligence*, Prentice-Hall, 2001.

[52] M. Sauer *et al.*: “Estimation of component criticality in early design steps,” *Proc. Intl. On-Line Testing Symp.* pp. 104–110, 2011.

[53] M. H. Schulz and E. Auth: “Improved deterministic test pattern generation with applications to redundancy identification,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 8, pp. 811–816, 1989.

[54] S. C. Seth and V. D. Agrawal: “A new model for computation of probabilistic testability in combinational circuits,” *Jour. VLSI Circuits*, vol. 7, pp. 49-75, 1989.

[55] SGI: *The SGI Solutions for Enterprises*, http://www.sgi.com/solutions/, 2012.

[56] A. Singhee and R. A. Rutenbar: “Why Quasi-Monte Carlo is better than Monte Carlo or Latin Hypercube sampling for statistical circuit analysis,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, pp. 1763-1776, 2010.

[57] S. Tasiran *et al.*: “A functional validation technique: biased-random simulation guided by observability-based coverage,” *Proc. Intl. Conf. Comput.-Aided Design*, pp. 82-88, 2011.

[58] T. Uchino *et al.*: "Switching activity analysis using Boolean approximation method," *Proc. Intl. Conf. Comput.-Aided Design*, pp.20-25, 1995.

[59] T. Uchino *et al.*: "Switching activity analysis for sequential circuits using Boolean approximation method," *Proc. Intl. Symp. Low Power Electron. & Design*, pp. 79-84, 1996.

[60] G. V. Varatkar *et al.*: “Stochastic networked computation,” *IEEE Trans. VLSI Syst.*, pp. 1421-1432, 2010.

[61] J. Von Neumann: “Probabilistic logics and synthesis of reliable organisms from unreliable components,” *Automata Studies*, pp. 43-98, Princeton University Press, 1956.

[62] N. H. E. Weste and D. M. Harris: *CMOS VLSI design: A Circuits and Systems Perspective*, Addison-Wesley, 2011.

[63] V. K. Wong and S. K. Teng: “Variation aware guard -banding for SOC static timing analysis,” *Proc. Intl. Symp. Quality Electron. Design*, pp.428-431, 2010.

[64] Q. Wu *et al.*: “A note on the relationship between signal probability and switching activity,” *Proc. Design Automation Conf.*, pp. 117-120, 1997.

[65] Y. Ye *et al.*: “Random variability modeling and its impact on scaled CMOS circuit,” *ACM J. of Comput. Electron.*, vol. 9, pp. 108-113, 2010.

[66] M. Yoshimi *et al.*: “FPGA implementation of a data-driven stochastic biochemical simulator with the next reaction method,” *Proc. Field Programmable Logic & Applicat.*, pp. 254-259, 2007.

[67] C. C. Yu and J. P. Hayes: “Scalable and accurate estimation of probabilistic behavior in sequential circuits,” *Proc. VLSI Test Symp.*, pp. 165-170, 2010.

[68] C. C. Yu and J. P. Hayes: “Trigonometric method to handle realistic error probabilities in logic circuits,” *Proc. Design, Automation & Test in Europe*, pp. 1-6, 2011.

[69] C. C. Yu *et al*.: "Scalable sampling methodology for logic simulation: reduced-ordered Monte Carlo," *Proc. Intl. Conf. Comput.-Aided Design*, 2012, to appear.

[70] K. M. Zick and J. P. Hayes: "High-level vulnerability over space and time to insidious soft errors," *Proc. High Level Design Validation & Test Workshop*, pp. 161-168, 2008.

[71] J. F. Ziegler et al.: "IBM experiments in soft fails in computer electronics (1978-1994)," *IBM Jour. Res. & Develop.*, pp. 3-18, 1996.