UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

FAST AND PRECISE POWER PREDICTION FOR COMBINATIONAL CIRCUITS CONSIDERING GLITCHING EFFECTS

A Dissertation

SUMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

HONGPING LI
Norman, Oklahoma
2003

Fast and Precise Power Prediction for Combinational Circuits Considering Glitching Effects

A Dissertation APPROVED FOR THE SCHOOL OF COMPUTER SCIENCE

BY

John K. Antonio

John K. Antonio, Committee Chair

[illegible]

Sudarshan Dhall

[illegible]

S. Lakshmivarahan

[illegible]

K. Thulasiraman

[illegible]

Victor DeBrunner

ACKNOWLEDGEMENTS

There are lots of people I would like to thank for a huge variety of reasons.

Firstly, I would like to thank my Supervisor, Dr. John K. Antonio. I could not have imagined having a better advisor and mentor for my Ph. D., and without his common-sense, knowledge, perceptiveness and cracking-of-the-whip I would never have finished. I would also like to thank Dr. Dhall and Dr. Varahan, who have helped immeasurably with contributions ranging from giving research lectures and suggestions to mentoring. Special thanks to Dr. K. Thulasiraman for giving me lots of suggestions. Thank you to all my examiners including Dr. DeBrunner (external), for managing to read the whole thing.

I would also like to thank all the rest of the academic and support staff of the School of Computer Science at The University of Oklahoma. Especially thanks to Barbara Bledsoe for finding lots of stuff for me to move around to keep my muscles strong. Thanks to Jim Summers for keeping my computer healthy and thanks to Sandy Johnson and Shanna Singleton for all they did for me.

Also thanks Dr. Su and Dr. Wei for putting up with me for almost three years. I really enjoyed the time we spent together especially fishing during the weekend. I would like to thank my family, my parents, have encouraged and supported me through years and years and years of school. I can't thank them enough for the loving environment they provided.

TABLE OF CONTENTS

List of Tables

LIST OF FIGURES

Abstract

Hongping Li (Ph.D. Computer Science)

Fast and Precise Power Prediction for Combinational Circuits Considering Glitching Effects

Directed by Dr. John K. Antonio, Director and Professor, The School of Computer Science, University of Oklahoma

The power consumed by a combinational circuit is dictated by the switching activities of all signals associated with the circuit. Analytical approaches, named MCP and MCPG algorithms, are proposed for calculating signal activities for combinational circuits, and the later considers glitching effects. Both approaches are based on a Markov chain signal model, and directly account for correlations present among the signals. The accuracy of the approaches is verified by comparing signal activity values calculated using the proposed approaches with corresponding values produced through simulation studies. Another approach (called the MMCP algorithm) is also proposed to calculate the total transition activities including glitching, and can be more accurate than the proposed MCPG algorithm. It is also demonstrated that the proposed approaches are computationally efficient.

CHAPTER 1

INTRODUCTION

Power consumption of integrated circuits (ICs) is of growing concern as more electronic devices are being deployed in mobile and portable applications, e.g., PDAs, mobile telephones, and other battery-powered electronic devices. As the functionality of such devices increases, so does the complexity and sophistication of the underlying circuits. More complexity and faster clock rates generally translate into higher power consumption for a given hardware implementation technology. Because battery technology has not improved at the same rate as IC technology, there is strong motivation to design circuits that are as power efficient as possible to extend battery life for portable devices.

Improvements in IC technologies (e.g., reduction in feature size) can reduce power requirements of a given circuit design. However, functionality and complexity of commercial devices generally increase from one generation to the next. So, the next generation device implemented with the next generation IC technology will generally have more functionality and complexity than the previous generation. Thus, the issue of architectural design of the underlying circuits to be power efficient remains important. Predicting and optimizing the power consumption during the design phase is critical for low power designs.

Power consumption in a CMOS circuit is primarily due to three types of current flow: switching current, leakage current, and short circuit current [11]. The first type of current flow is switching current for charging and discharging load capacitance due to switching activities. The short-circuit current within CMOS gates is caused by a brief short circuit that can occur when the state of the complimentary gates changes from on-to-off and off-to-on. This short circuit occurs when the complimentary MOSFETs are concurrently "on" for a brief transient period of time. The leakage current is associated with the imperfection of field effect transistors (FETs) that are used in CMOS devices.

The total power dissipation of a CMOS circuit is the sum of the three types of power consumption, i.e., switching current power dissipation, short-circuit current power consumption and leakage current power consumption. Because the dynamic power dissipation is by far the dominant component, almost all methods used to calculate power consumption in CMOS circuits are focused on estimation of dynamic power consumption [14, 15].

Because the power estimation is calculated at the gate level, assuming both the supply voltage and the capacitance are known, the power consumption can be estimated by calculating the switching activity for each circuit node.

Power dissipation is strongly dependent on the applied input signals to the circuit. Each applied input propagates through the circuit causing the internal nodes to perform transitions according the functionality and the interconnection of the circuit gates. The same circuit under different input scenarios may have totally different switching activities of the internal nodes, which will have different power dissipation. Thus the applied input must be taken into account.

The power estimation methodologies at the logic gate level can be divided into two general classes: the statistical-based and probabilistic-based methodologies. The statistical-based power estimation approaches use a large number of input vectors to simulate the circuit in order to achieve near real results and such simulations are highly dependent on the primary input vectors. This often makes statistical-based approaches impractical for large circuits and long input sequences. Several methods have been developed to overcome this drawback and the Monte-Carlo, the Advanced Sampling and the Vector Compaction methods are the most representative approaches [12, 14, 15].

Compared to the statistical-based approaches, probabilistic-based approaches compute switching activities in one run, which may result in much less computation time, but the accuracy may not be as good as statistical-based approaches. The goal of the research in this dissertation is to develop a signal model for a probabilistic-based approach that can achieve near statistical-based approaches' accuracy with much less time complexity.

Signals in a combinational logic circuit can be treated in a probabilistic sense, i.e., for signal x, the probability that x has logic value "1" is defined by $P(x) = P(x = 1)$. Let $x(t)$, $t \in (-\infty,+\infty)$, be a stochastic process that takes the values of logical "0" or logical "1", transitioning from one to the other at random times. Generally, a stochastic process is said to be strict-sense stationary (SSS) if its statistical properties are invariable to a shift of time origin. Based on the assumption of a SSS 0-1 mean-ergodic process, the probability of a signal $x(t)$ can be defined as the average fraction of time that the signal is high, and the activity can be defined as the average number of transitions in a time interval.

Several probabilistic-based approaches used to calculate signal probabilities, i.e. $P(x)$, of all signals in a circuit are developed in [2, 6, 7]. Although this probability calculation is not directly used in calculating a circuit's power consumption, it is a necessary component for signal models common to the activity approaches, which utilize both signal probability and signal activity parameters [3, 4].

The approaches of [2], [3], and [4] can have high computational complexities because the number of terms in the underlying equations/transformations can grow exponentially with the number of primary inputs to the circuit. In [7], a trade-off between computational complexity and resulting accuracy is illustrated in the context of the underlying equations/transformations introduced in [2]. In particular, an approximate approach is defined in [7] in which the transformations of [2] are applied in a "gate-by-gate" fashion. Thus, instead of deriving the transformation for a signal's probability parameter in terms of the circuit's primary inputs, it is derived in terms of the immediate inputs to the logic gate associated with the signal. This approach greatly reduces the computational complexity, but introduces error in the calculated probability parameters for circuits with re-convergent fan-out.

Similar trade-offs between computational complexity and accuracy are possible relative to the evaluation of activities associated with [3] and [4], respectively. Instead of deriving a signal's logic function in terms of the circuit's primary inputs, the parameters to the immediate inputs of the signal's logic gate can be used. Again, this type of "gate-by-gate" technique will generally introduce error because it does not account for correlations present among the internal signals that drive the gates within the circuit. The approach of [6] is a fast and accurate "gate-by-gate" technique for calculating a signal's probability parameter. It introduces the concept of a correlation factor to account for and appropriately adjust the transformation for correlated inputs to a gate.

Signals can be modeled by a Markov-Chain, having two states: state 0 and state 1, associated with two transition events: the transition event from state 0 to state 1 and the transition event from state 1 to state 0. It is shown that the proposed Markov chain model is equivalent to the two-parameter probability/activity signal model of [3] and [4]. The advantage of modeling signals with Markov chains is that it makes it possible to compute correlations between signals related to both probability and activity. Based on this Markov-chain signal modeling, we can develop a more efficient and more accurate algorithm (named MCP algorithm) by propagating signal parameters and correlation cofactors from the primary inputs through a "gate-by-gate" fashion. This MCP algorithm can achieve a very good accuracy and an $O(M^2)$ time complexity where M is the number of signals in the circuit.

Because the MCP algorithm assumes zero propagation delay through each gate, it will generate errors in real applications. In reality, gates have non-zero delays, which results in "signal glitching." In non-zero delay model, glitches will cause errors in general power consumption estimation algorithms and tools that assume a zero-delay model. So a MCPG algorithm is developed expanded from the MCP algorithm to take account of glitching effects. Compared to MCP algorithm, MCPG algorithm computes the glitching transitions caused by the associated input delays and propagates these glitching transitions to the next stage. Because it is assumed that the target circuit can be run ideally at infinite speed, every glitch may cause new glitches in the next stage, which gives us an upper bound of the activity of each node in the circuit.

Because circuits cannot run at an infinite frequency, MCPG algorithm will not generally give us an accurate result in real applications. To deal with this situation, another algorithm, named MMCP algorithm, is developed. It is shown that MCPG algorithm gives us a good prediction of the maximum activity of each node, which is an upper bound of the activity of each node, and the MMCP algorithm produces a closer prediction of activities of all signals in the circuit.

This thesis is organized as follows: In Chapter 2, three sources of power consumption will be discussed in detail and methodologies of estimation of power consumptions in CMOS circuits are also reviewed. Because we are focused on probabilistic approaches, we will briefly overview those past probabilistic-based approaches in Chapter 3. In Chapter 4, a Markov-Chain signal model is developed and based on this signal model, the MCP algorithm is proposed and analyzed in detail. In Chapter 5, the MCPG algorithm expanded from the MCP algorithm is developed to deal with glitching power consumption. To investigate the accuracy and efficiency of algorithms we developed, experimental setup and results are listed in Chapter 6. A more accurate glitching power consumption prediction algorithm is also introduced in Chapter 6, which is named MMCP algorithm. The final summary and future work is in Chapter 7.

CHAPTER 2

SOURCES OF POWER CONSUMPTIONS IN CMOS CIRCUITS

2.1 INTRODUCTION

In this chapter, three sources of power consumption will be addressed first. Then the estimation of power consumption problem is stated, followed by an overview of past methodologies for estimation of CMOS power consumption.

2.2 THREE SOURCES OF POWER CONSUMPTION

2.2.1 *BRIEF OVERVIEW*

Power consumption in a CMOS circuit is primarily due to three types of current flow: switching current, short circuit current and leakage current [11], which are summarized in the following equation:

$$P_{avg} = P_{switching} + P_{short-circuit} + P_{leakage} . \tag{2.1}$$

The first type of current flow, switching current, is due to charging and discharging of load capacitance associated with signal switching activities. The short-circuit current within CMOS gates is caused by a brief short circuit that can occur when the state of the complimentary gates changes from on-to-off and off-to-on. This short circuit occurs when the complimentary MOSFETs are concurrently "on" for a brief transient period of time. The leakage current is associated with the imperfection of field effect transistors (FETs) that are used in CMOS devices. These three components of power consumption are described in detail below.

2.2.2 *SWITCHING CURRENT POWER DISSIPATION*

The switching current power consumption (also called dynamic power dissipation), $P_{switching}$, is caused by the charging and discharging of capacitances in the circuit. To illustrate the computation of dynamic power dissipation in a CMOS circuit, we use an example of a CMOS inverter driving a load capacitor C_L, as shown in Figure 2-1a.

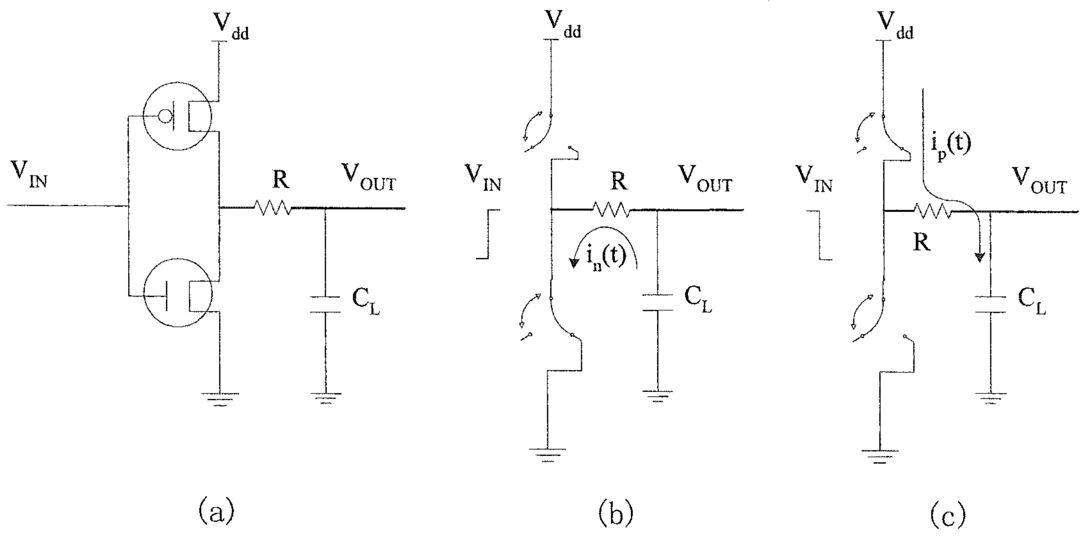


Figure 2-1. Operation of a CMOS inverter driving a load capacitor C_L: (a) inverter circuit model, (b) discharging phase, (c) charging phase.

As shown in Figure 2-1, the dynamic power consumption of an inverter is associated with power dissipated in charging and discharging of the load capacitor. To simplify the analysis, assume the input signal, V_{in}, is a square wave having a period T and that the rising and fall time of V_{in} is much less than the period T. The rise time and fall time of a signal are depicted in Figure 2-2 as T_r and T_f, respectively. Assume the circuit is initially in a steady state with input having a logic value "0" and thus the output has a logic value "1". In this state the output capacitor is charged and the output voltage is V_{dd}. When the input waveform undergoes a rising transition, the nMOS transistor conducts (ON) and the pMOS transistor turns OFF as shown in Figure 2-1b. Current is drawn from capacitor C_L, and the capacitor is discharged, resulting in an output voltage of zero. During this discharging process the average power dissipated can be expressed as

$$P_{\text{discharging}} = \frac{1}{T}\int_0^{\frac{T}{2}} i_n(t)V_{out}(t)dt, \tag{2.2}$$

where $i_n(t)$ is the current flowing from the capacitor through the nMOS to the ground as shown in Figure 2-1b.

As the input waveform goes from "1" to "0" (having a falling transition), the pMOS transistor will be ON and the nMOS will be OFF as shown in Figure 2-1c. In this charging phase, the current will flow from the power supply V_{dd} through the pMOS to the capacitor, and the average power consumed due to charging can be expressed as

$$P_{\text{charging}} = \frac{1}{T}\int_{\frac{T}{2}}^{T} i_p(t)(V_{dd} - V_{out}(t))dt \,. \tag{2.3}$$

Using the assumptions that $i_n(t) = -C_L \frac{dV_{out}(t)}{dt}$, $i_p(t) = C_L \frac{dV_{out}(t)}{dt}$, $V_{out}(0) = V_{dd}$, $V_{out}(T/2) = 0$, and $V_{out}(T) = V_{dd}$, the total power dissipation during charging and discharging can be expressed as[1]

$$P_{switching} = P_{charging} + P_{discharging} = \frac{C_L V_{dd}^2}{T} \,. \tag{2.4}$$

Assume f represents the frequency (switching frequency) of the input signal, and $f = 1/T$, then the above equation can be rewritten as

$$P_{switching} = C_L V_{dd}^2 f \,. \tag{2.5}$$

The dynamic power dissipation is the dominant factor compared with the other components of power dissipation in CMOS circuits. For current technologies, the dynamic power dissipation is about 80% of a circuit's total dissipation [14, 15]. Consequently, the majority of existing low power design and power estimation techniques focus on this dynamic component of dissipation.

Equation 2.5 shows that the dynamic power consumption in a CMOS circuit is proportional to the switching frequency, load capacitance and the square of the supply voltage. Based on this observation, the power reduction can be achieved by these methods:

---

[1] The V_{out} $(T/2) = 0$ and V_{out} $(T) = V_{dd}$ are based on the assumption that the RC time constant for the circuit satisfies $RC << T$.

- Reduction of output capacitance, C_L
- Reduction of power supply voltage, V_{dd}
- Reduction of the average switching frequency, f

Generally, power reduction can be achieved by the combination of one or more aforementioned methods. A very popular low power strategy aims at the reduction of the product of the load capacitance and the switching frequency, i.e., $C_L f$, which sometimes is called *effective capacitance*. It is noted here that a signal waveform is generally not a periodic regular signal like the "clock" signal assumed in this analysis. In general, f represents the "average" frequency of a signal, and determines this value for all signals in a circuit is the focus of this dissertation.

Another main low power reduction strategy, which is one of the most aggressive techniques, is the reduction of supply voltage because the power savings are significant due to the quadratic dependence of V_{dd} (as shown in Equation 2.5). The disadvantage of this technique is that it might decrease the performance of the circuits, specifically, the reduction of the power supply voltage leads to an increase to the delay propagation. Thus, reducing supply voltage leads to a trade-off between the power consumption and the circuit's speed.

2.2.3 *Short-Circuit Power Dissipation*

The short-circuit power consumption, $P_{short\text{-}circuit}$, is caused by the current flow directly from the power supply to the ground during the transition phase. Consider again the CMOS inverter shown in Figure 2-1. Assume initially that $V_{in} = 0$ and then starts to increase from zero to V_{dd}. When $V_{in} \geq V_{Tn}$ where V_{Tn} is the threshold of the nMOS, the nMOS starts to come out of cutoff and enters in conducting state. The load capacitor starts to discharge through nMOS and V_{out} begins to decrease. At this time because pMOS is not totally cutoff, there exists a conducting path for current to flow directly from the V_{dd} to the ground. This current flow directly from the power supply to the ground is called the short-circuit current. When V_{in} increases to the point of $V_{dd} - V_{in} < |V_{Tp}|$, where V_{Tp} is the threshold of the pMOS, the pMOS is totally cutoff. This process for V_{in} changing from V_{dd} to zero follows a similar sequence of events.

Figure 2-2 shows the short-circuit current behavior in an inverter. Exact analysis of the power dissipation due to short-circuit is complex. Here we give a simplified analysis which will give an upper bound of the power consumption due to short-circuit current in an inverter [12].

To simplify, consider a symmetric inverter (i.e., $V_{Tn} = V_{Tp}$) with a symmetric input signal V_{in} as shown in Figure 2-2. The rise and fall time of V_{in} are denoted by T_r and T_f. The time-averaged short-circuit current drawn from the power supply and the power dissipated due to this current of the symmetric inverter can be approximated by [12]

$$I_{short-circuit_{avg}} = \frac{k}{V_{dd}}(V_{dd} - V_T)^3 T_r f \qquad (2.6)$$

$$P_{short-circuit} = k(V_{dd} - V_T)^3 T_r f, \qquad (2.7)$$

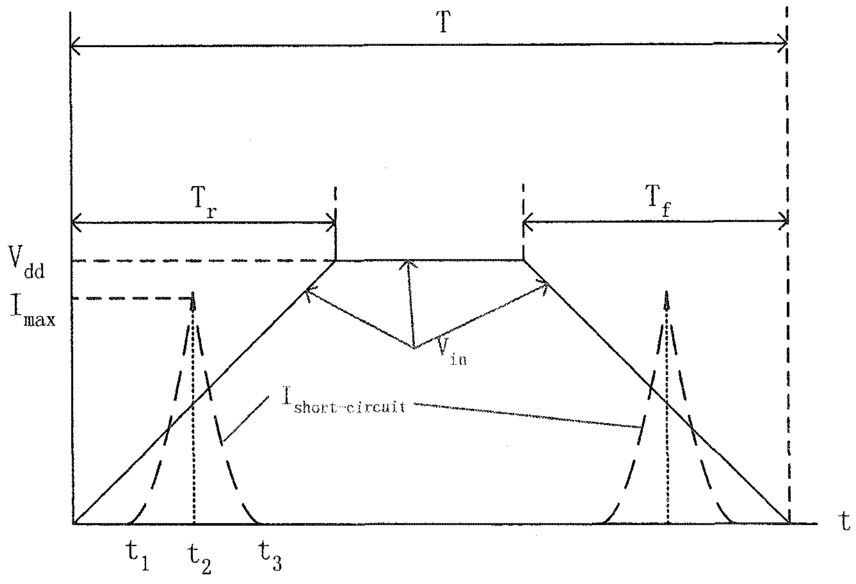


Figure 2-2. Short-circuit current in an inverter.

respectively, where *k* is a constant that depends on transistor sizes as well as technology, V_T is threshold voltage of the nMOS and pMOS transistors, T_r is the rise (or fall time) of the symmetric input signal, and *f* is the switching frequency.

Reduction in the short-circuit power dissipation can be achieved in different ways. From Equation 2.7, the power is proportional to the rising (or falling time) of the input signal and the switching frequency and therefore, reducing these input transition times decrease the short-circuit current. In addition, new technology will help to reduce the constant *k* value and the power supply voltage as a result to reduce the power dissipation due to short-circuit current in CMOS circuits.

2.2.4 *Leakage Current Power Dissipation*

The power dissipation due to leakage current (also called static power dissipation) is caused by the imperfection of the MOSFET devices. Consider the same inverter as shown in Figure 2-1, when the input signal is $V_{in} = 0$, the pMOS is ON and nMOS is in cutoff, and vise visa, when $V_{in} = V_{dd}$, the pMOS will be cutoff and the nMOS is ON. Hence, ideally, whenever the input V_{in} stays in "0" or V_{dd}, no current flows from the power supply to the ground. A very small amount of power dissipation, though, does take place. This small amount of power dissipation is due to the leakage currents flow from the power supply to the ground, which is also called static power consumption.

The static power dissipation can be expressed by $P_{static} = I_{leak}V_{dd}$ [13]. Compared to the other two types of power consumption in CMOS circuits, static power consumption is the smallest part and is often ignored in power consumption estimation. In our research, we mainly focus on power dissipation due to switching current to approximate the total power consumption by using

$$P_{avg} \approx P_{switching} = C_L V_{dd}^2 f \ . \tag{2.8}$$

If the parameters are given, such as C_L and V_{dd} are known, then by estimation of the signal switching frequency *f*, we can use Equation 2.8 to approximately calculate the average power consumption of CMOS circuits.

2.3 Methodologies of Estimation of Power Consumption

2.3.1 Brief Overview

The total power dissipation of a CMOS circuit is the sum of the three types of power consumption, i.e., dynamic power dissipation, short-circuit current power consumption and static power consumption. Because the dynamic power dissipation is by far the dominant component, almost all methods used to calculation power consumption in CMOS circuits are focused on estimation of dynamic power consumption. Considering that the power estimation is calculated at the gate level (as shown in dynamic power calculation in the previous section), the power consumption can be estimated by calculating the switching activity for each circuit node assuming the supply voltage and capacitance are specified.

In addition, power dissipation is strongly dependent on the characteristics of the applied input signals to the circuit. Each applied input propagates through the circuit causing the internal nodes to perform transitions according the functionality and the interconnection of the circuit gates. The same circuit under different input scenarios may have totally different switching activities of the internal nodes, which may result in different power dissipation. Thus the applied input must be taken into account.

Power consumption estimation means calculation of the average and/or worst case power consumption. Furthermore we assume that the time between two successive input vectors is enough to allow the circuit to reach in a steady state. Based on these assumptions, we state the problem of power estimation at the gate level as follows:

Power Estimation Problem: *"Given a gate netlist of a synchronous static CMOS circuit and provided with an associated input vector sequence, estimate the average power dissipation of the circuit by calculating the average switching activity of each circuit node."*

Therefore, the problem of estimation of the average power consumption of a given CMOS circuit is transferred into a problem of calculating the switching activity of each node in the circuit.

The power estimation methodologies at the logic gate level can be divided into two general classes: statistical-based and probability-based methodologies [5]. Figure 2-3 provides a general overview of these two methodologies [5]. The statistical-base methods (the upper flow) achieve power estimation by simulating the circuit with a large number of input vectors and averaging the large number of each internal signal waveform to get the average power consumption of the circuit. The probabilistic-based methods (the lower flow in Figure 2-3) first average the large number of input patterns to get probabilistic properties of input signals, then some analysis tools and/or techniques are used to predict the power consumption of the circuit.

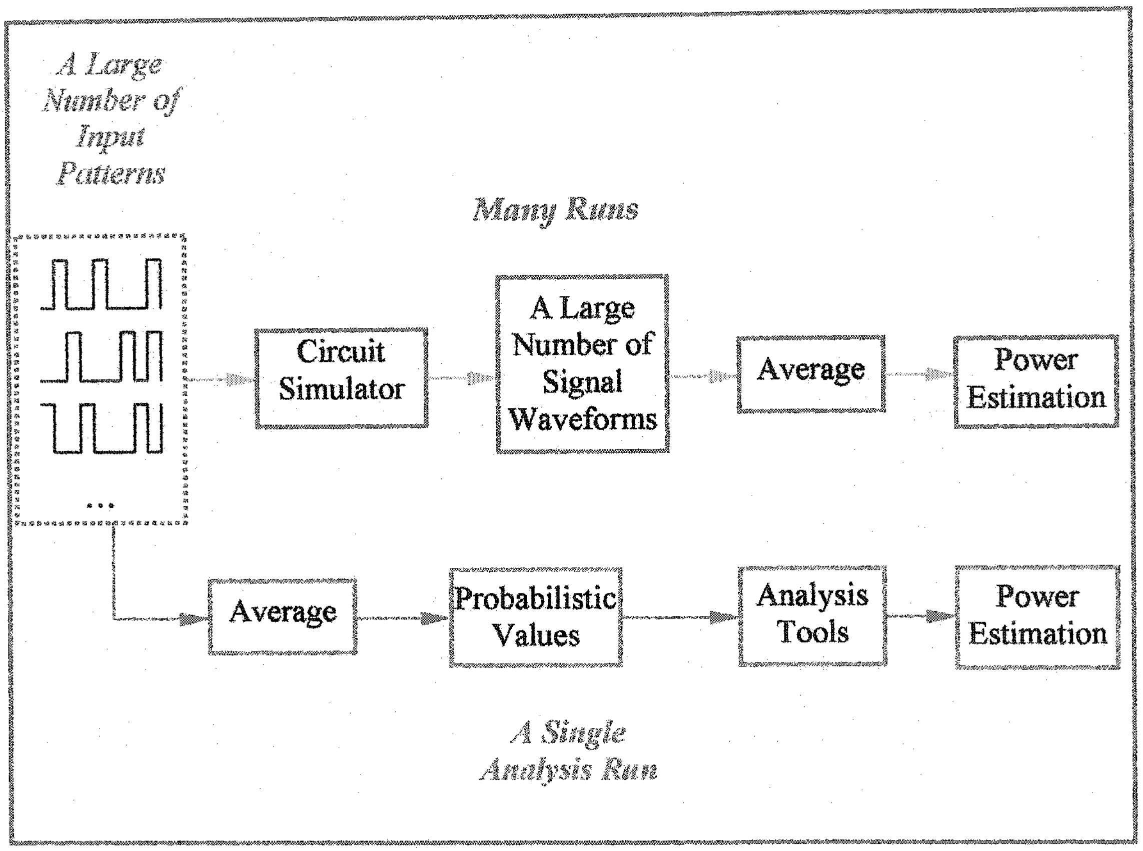


Figure 2-3. Two methodologies used to estimate the power consumption in CMOS circuits.

2.3.2 STATISTICAL-BASED POWER ESTIMATION

Because the simulation result is highly dependent on the primary input vectors, the statistical-based power estimation approach needs to use a large number of input vectors to simulate the circuit in order to achieve an accurate estimate of the circuit's behavior. This often makes this approach impractical for large circuits and long input sequences. Several methods have been developed to overcome this drawback; the Monte-Carlo, the Advanced Sampling and the Vector Compaction methods are the most representative methods [12]. Only the

Monte-Carlo approach will be described in this chapter, which is the most commonly used technique in statistical-based power estimation.

The block diagram in Figure 2-4 gives an overall view of this technique. The basic idea of Monte Carlo statistical technique is as follows [1]:

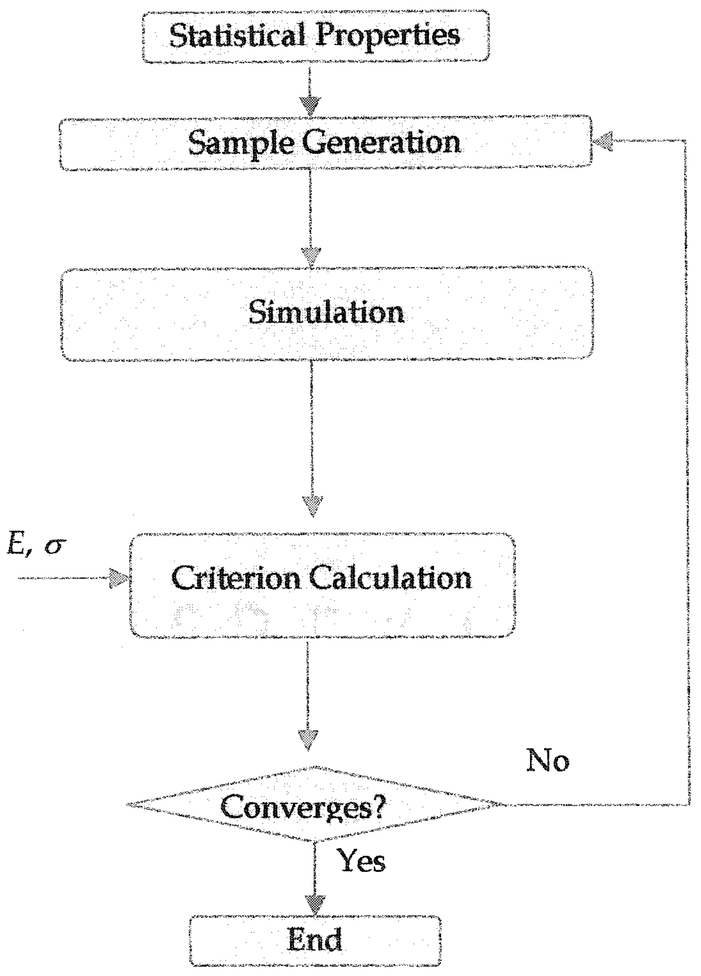


Figure 2-4. The block diagram of the Monte-Carlo method.

1. Input patterns are generated based on given input sequence statistical properties;

2. The number of transitions at each node is counted during a given period of duration, the power value (activity value) at the end of each simulation run is noted;
3. Decide whether to stop the process or to do another run. The decision is made based on the mean and standard deviation of the power values observed at the end of a number of successive iterations;
4. This process is repeated until it converges to the true result.

The main issue of this method is when to stop the simulation, which means a stopping criterion needs to be found. If the input patterns are independently generated as shown in step 1 in Monte-Carlo method, a large number of independent samples, represented as n independent samples, will be obtained by this measurement and the average will approach the desired average power for large n. In order to stop the simulation, the value of n needs to be found such that the average power is close enough to the true power, and this number of n is called the stopping criterion. This can be done by follows:

When we use a sample mean $\bar{a}$ to estimate the mean a of a population, there always exists an error and for large n, $\frac{\bar{a}-a}{\sigma/\sqrt{n}}$ is a value of a random variable having approximately the standard normal distribution, where σ is sample's standard deviation. We can assert with a probability of $1-\alpha$ that

$$-z_{\alpha/2} < \frac{\bar{a}-a}{\sigma/\sqrt{n}} < z_{\alpha/2} \text{ or } \frac{|\bar{a}-a|}{\sigma/\sqrt{n}} < z_{\alpha/2} \tag{2.9}$$

where $z_{\alpha/2}$ is such that the normal distribution curve area to its right equals $\alpha/2$. Using $E = |\bar{a} - a|$, we have

$$E < z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \,. \tag{2.10}$$

Equation 2.10 shows that if we estimate mean value by means of a random sample of size *n*, we can assert with a probability of $1-\alpha$ that the error, *E*, is less than $z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$, at least for large *n*.

Solving for *n* in 2.10, we have

$$n > \frac{z_{\alpha/2}^2\sigma^2}{E^2}. \tag{2.11}$$

So by given error *E*, standard deviation σ, and probability α, we can use Equation 2.11 to decide how many samples need to be generated in Monte-Carlo statistic simulation[2].

2.3.3 *Probabilistic-Based Power Estimation*

Compared to the statistical-based approaches, probabilistic-based approaches estimate the switching activities in one run, which may result in much less computation time. Figure 2-5 is a block diagram that shows how probabilistic-based approaches compute the switching activities in CMOS circuits.

[2] For detailed information of Monte-Carlo approaches, refer to [1].

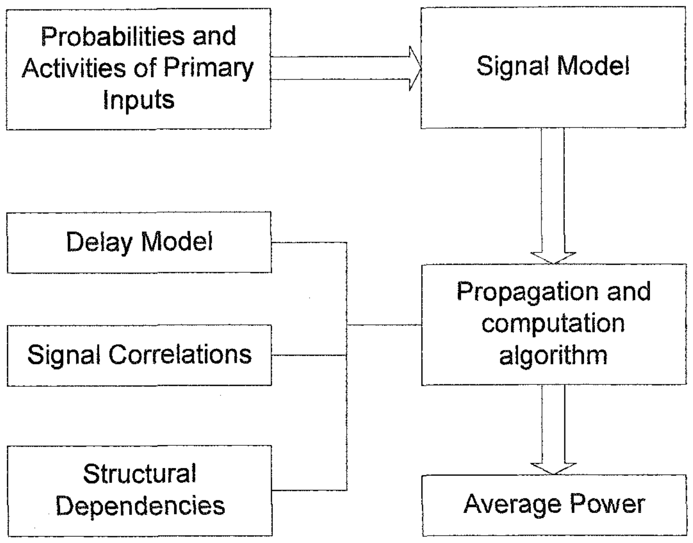

Probabilities and Activities of Primary Inputs
Signal Model
Delay Model
Signal Correlations
Structural Dependencies
Propagation and computation algorithm
Average Power

Figure 2-5. The typical diagram flow of probabilistic-based power estimation approaches.

According to the type of a circuit, probabilistic-based approach can be categorized into methods for combinational and sequential circuits. For combinational circuits, it can be further classified into zero-delay and non-zero-delay model. The detailed analysis of probabilistic-based approaches and their associated algorithms are provided in Chapter 3.

2.4 SUMMARY

There are three sources of power consumption in CMOS circuits: dynamic power dissipation, short-circuit power dissipation and static power dissipation. In these three power consumption components, dynamic power dissipation due to switching signal activities is dominant. Therefore, the problem of estimation of the average power consumption of a given CMOS circuit is transferred into a problem of calculating the switching activity of each node in the circuit.

Two general classes, the statistical-based and the probabilistic-based methodologies, exist in the power estimation methodologies at the logic gate level. The statistical-based approach, often represented by the Monte-Carlo approach, can provide accurate results, but generally, longer simulation time compared to probabilistic-based approaches.

A taxonomy of techniques used to estimate switching activities in CMOS circuits is shown in Figure 2-6. All colored blocks will be analyzed in this dissertation. Approaches represented by pink colored blocks represent our research contributions, which provide a solution with comparable accuracy to statistic-based simulation, but having probabilistic-based time complexity, will be introduced in followed chapters.

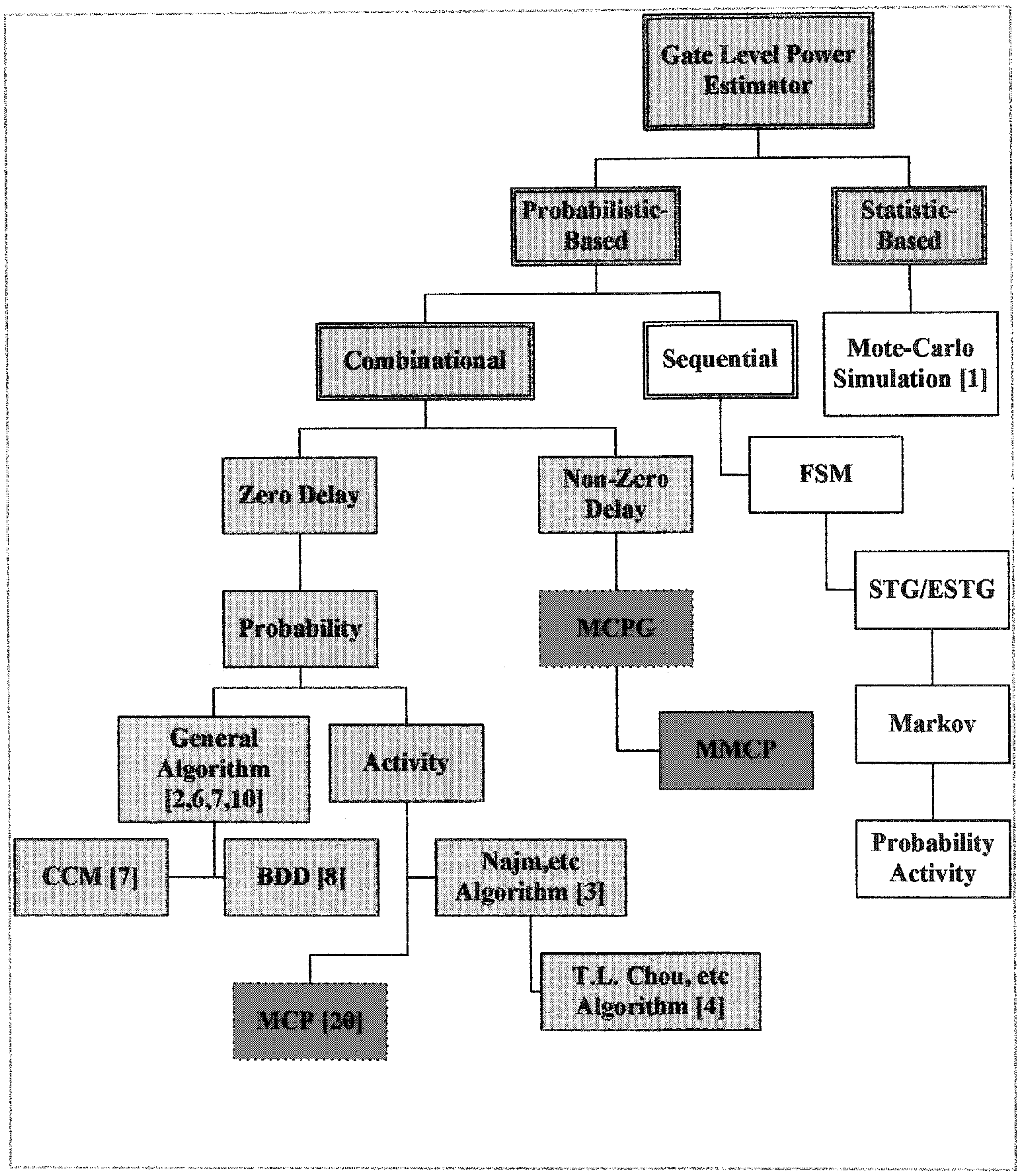


Figure 2-6. Summary of most techniques used to estimate switching activities in CMOS circuits.

CHAPTER 3

PROBABILISTIC-BASED POWER ESTIMATION

3.1 INTRODUCTION

Probabilistic-based approaches have the potential advantage of performing the switching activity computation in less time than competing statistical-based approaches. Therefore, more efficient, accurate and more practical algorithms have become a major concern in power estimation research. In this chapter, we first introduce a general signal model. Then based on this signal model, several probabilistic-based algorithms used to calculate signal probabilities will be reviewed, followed by algorithms for calculating signal activities. Finally, the detailed analysis of these algorithms including complexity and accuracy will be discussed.

3.2 MODELING OF SIGNALS

Signals in a combinational logic circuit can be treated in a probabilistic sense [1], i.e., for signal x, the probability that x has logic value "1" is defined by $P(x) = P(x = 1)$. Let $x(t)$, $t \in (-\infty,+\infty)$, be a stochastic process that takes the values of logical "0" or logical "1", transitioning from one to the other at random times.

Generally, a stochastic process is said to be strict-sense stationary (SSS) if its statistical properties are invariable to a shift of time origin. Based on the assumptions of a SSS 0-1 mean-ergodic process $x(t)$, the following definitions are derived from [3].

Definition 3.1 (Signal Probability) The probability of a logic signal $x(t)$ is the average fraction of time that the signal is high and is given by

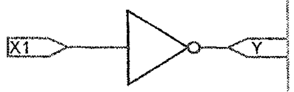
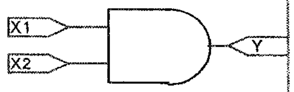
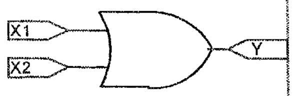
$$P(x) = \lim_{T \to \infty} \frac{1}{T} \int_{\frac{-T}{2}}^{\frac{+T}{2}} x(t)dt$$

Definition 3.2 (Signal Activity): The signal activity of a logic signal $x(t)$ is the average number of transitions, i.e., $n(T)$, in a time interval T and is given by

$$\alpha(x) \overset{\Delta}{=} \lim_{T \to \infty} \frac{n(T)}{T}$$

The analytical expressions of signal probability for some basic logic gates are defined in [2] and results are stated in Table 3-1.

Table 3-1. The output probability expression of some basic logic gates.

	P(Y)=1-P(x1)
	P(Y)=P(x1)P(X2)
	P(Y)=1-[1-P(x1)][1-P(x2)]

Clearly the input signals in Table 3-1 are assumed to be independent. But in general, signals might be correlated and can be separated into three types:

- *Temporally correlated*: a signal's value depends on the values that the signal has taken in the past
- *Spatially correlated:* a signal's value depends on the values of other signals
- *Spatiotemporal correlated:* a signal's value depends both on its own and other signal's value

The impact of the above three kind of correlations is critical in the probabilities and switching activities calculation. Figure 3-1 shows the impact on the results with different input vectors.

As shown in Figure 3-1 with zero delay assumption, three different input vector sequences, V_1, V_2 and V_3 are applied to the three primary inputs x, y and z. The input vector of sequence V_1 is generated by a random number generator, which means that the three primary inputs x, y and z are mutually independent in V_1. The signals of sequence V_2 is formed in such a way that the spatiotemporal correlations in input signals x, y and z are strong. In contrast to sequence V_2, the signals of sequence V_3 has a weak spatiotemporal correlations in inputs x, y and z. The number of transitions of each node corresponding to different input vector sequences is also listed in Figure 3-1. It illustrates that the number of transitions is affected by the correlations of the primary input signals.

V_1	V_2	V_3
xyz	xyz	xyz
001	000	010
100	001	101
110	010	010
111	011	011
011	100	100
101	101	101
010	110	010
000	111	101

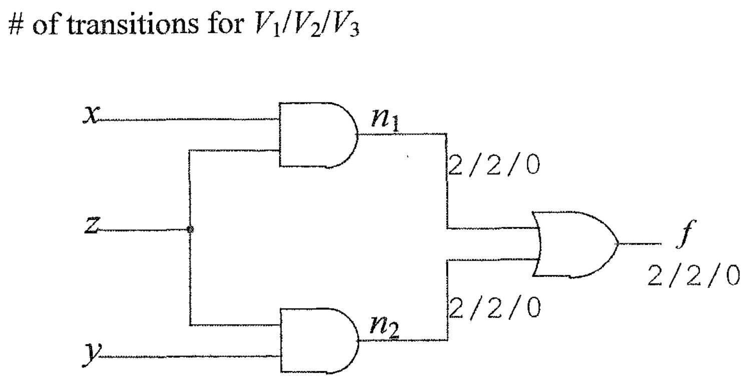


Figure 3-1. The impact on the number of transitions with different input vectors with different correlations derived from [13].

Besides correlations among the primary inputs, the structure of the circuit may cause additional dependency between signals, which is introduced due to reconvergent fanouts. Even if the primary inputs x, y and z are mutually independent in Figure 3-1, signal z fans out into two signals, this kind of correlation will also impact the resulting calculation. The detailed effect of this correlation and methods dealing with this kind of correlation in switching activity calculation will be described later in this chapter.

3.3 Signal Probability Calculation

Signal probability calculation is used for accurately estimating signal activity, which is necessary for power consumption estimation. Thus it is essential to estimate signal probability correctly for further use in signal activity calculation.

3.3.1 Early Algorithm

In [2], the concept of using probabilistic signal modeling for analysis of combinational circuits was first introduced. In this work, each signal is modeled with a single probabilistic parameter, $P(x)$, defining the probability of a signal having a logical value of one. The purpose is to calculate the probability parameter for all signals, given the probability parameters of the circuit's primary inputs. The motivation for this work originated from the area of pseudorandom testing, in which fault coverage and identification is achieved without resorting to exhaustive testing. Instead, by subjecting a circuit to a large number of randomly generated input signal vectors, one can deduce faults in the circuit by measuring the fraction of time that any given signal has logic value one. If any of the measured signal probabilities do not match calculated signal probabilities, then the possibility of a fault is present.

As mentioned above, signals in a combinational logic circuit are treated in a probabilistic sense in [2]. For signal x, the probability that x has logic value "1" is defined by $P(x) = P(x = 1)$. Two algorithms for calculating signal probabilities are introduced in [2] with an upper bound complexity of order 2^n where n is the number of circuit inputs. The second with less complexity is given below:

Early Algorithm: Compute signal probability of each signal in a circuit.

Input: Signal probabilities of all primary inputs to the circuit.

Output: Signal probabilities of all signals in the circuit.

1. For each input and gate output in the circuit, assign a unique variable;

2. Starting at the inputs and proceeding to the outputs, write the expression for the output of each gate as a function of its input expressions (using expressions in Table 3-1);
3. Suppress all exponents in a given expression to obtain the probability expression for that signal.

In this algorithm, the primary inputs are assumed to be mutually independent, and a Boolean function expression associated with each signal can be derived in terms of the primary inputs. However, the internal nodes of a circuit may be correlated due to reconvergent fan-out which can produce expressions having exponents greater than 1. Hence, Step 3 is used to handle signal correlations by suppressing exponents of variables in the Boolean function expressions.

To illustrate how to use this algorithm to calculate signal probabilities of a circuit, consider a simple circuit as shown in Figure 3-2.

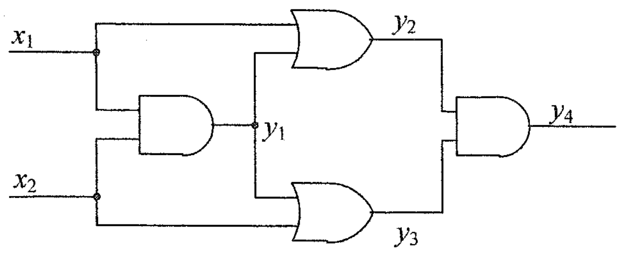


Figure 3-2. An example combinational circuit used to illustrate signal probability calculations (derived from [7]).

By using this Early Algorithm, the internal signal y_1 and signal y_2 can be expressed as

$$y_1 = x_1 x_2$$

$$y_2 = x_1 + y_1 - x_1 y_1 = x_1 + x_1 x_2 - x_1^2 x_2$$

Suppressing exponent of x_1, we have

$$y_2 = x_1 + x_1 x_2 - x_1 x_2 = x_1 .$$

Similarly,

$$y_3 = x_2 + y_1 - x_2 y_1 = x_2 + x_1 x_2 - x_1 x_2^2 = x_2 + x_1 x_2 - x_1 x_2 = x_2$$

$$y_4 = y_1 y_2 = x_1 x_2$$

3.3.2 *GENERAL ALGORITHM*

The Early Algorithm can solve the probabilities of all nodes in the circuit exactly when all primary inputs are assumed mutually independent. It results in exponential time complexity, though, due to simplification of Boolean functions associated with each node into Boolean functions expressed by primary inputs only. To reduce the time complexity, a computationally efficient algorithm for calculating signal probabilities is introduced in [6], named "General Algorithm," which operates by propagating probability values through the gates of circuit, thereby drastically reducing the size of the Boolean functions that must be evaluated. Specifically, the probability of the output of a gate is expressed in terms of the probability values for the inputs to that gate (instead of the primary inputs of the entire circuit, as required by the approach in [2]). This algorithm is an extension of the above Early Algorithm and is given below:

General Algorithm: Compute signal probability of each signal in a circuit.

<u>Input</u>: Signal probabilities of all primary inputs to the circuit.

<u>Output</u>: Signal probabilities of all nodes in the circuit.

1. For each input and gate output in the circuit, assign a unique variable;
2. Starting at the inputs and proceeding to the outputs, calculate the value of the output of each gate using expressions of Table 3-1.

This algorithm is simple and fast - it has a linear complexity in the number of gates - but is not accurate for all classes of circuits.

To illustrate the inaccuracies of General Algorithm, assume in Figure 3-2 that the probabilities of primary inputs x_1 and x_2 are both 0.5. By applying General Algorithm, the computed probabilities of the circuit's signals can be calculated and the results are provided in Table 3-2.

Table 3-2. Comparison of actual signal probabilities and those calculated using General Algorithm for the circuit of Figure 3-2 with $P(x_1) = P(x_2) = 1/2$.

	$P(y_1)$	$P(y_2)$	$P(y_3)$	$P(y_4)$
Actual	1/4	1/2	1/2	1/4
General Algorithm	1/4	5/8	5/8	25/64

The problem with the accuracy of the General Algorithm arises in circuits in which re-convergent fan-out signals are present. Re-convergent fan-out introduces functional dependencies and statistical correlations among the signals; however, the General Algorithm assumes statistical independence among the inputs to each gate. For example, signals y_2 and y_3 in Figure 3-2 both depend on signal x_1 due to re-convergent fan-out. Thus, applying the algorithm to calculate $P(y_4)$ under the assumption that signals y_2 and y_3 are independent results in an error in the value calculated for $P(y_4)$, as shown in Table 3-2. Similarly, the values calculated for $P(y_2)$ and $P(y_3)$ are also in error.

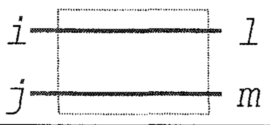
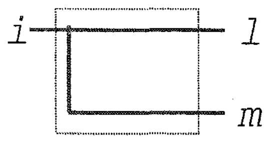
3.3.3 CCM ALGORITHM

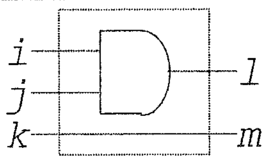
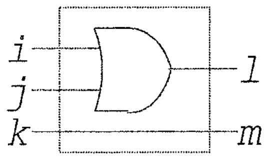
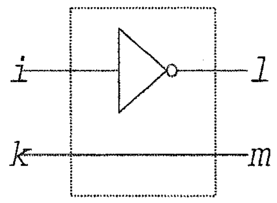
A method for accounting for signal probability correlations was developed in [6] named the correlation coefficient method (CCM). By defining the correlation coefficient of two events A and B as $C_{A,B}$ where

$$C_{AB} = \frac{P(AB)}{P(A)P(B)} = \frac{P(A/B)}{P(A)} = \frac{P(B/A)}{P(B)} ,$$

probabilities of output signals can be calculated by using these main rules as shown in Table 3-3 [6].

Table 3-3. Set of basic rules used to calculate the probability of output signals and correlation coefficients by given input signals' probability and correlation coefficients.

Rules	Probability	Probability Correlation Factors
Independent rule 	Same as input	$C_{lm} = C_{ij}$
Fan-out rule 	Same as input	$C_{lm} = \frac{1}{P(i)}$
AND rule	$P(l) = P(i)P(j)C_{ij}$	$C_{lm} = C_{ik}C_{jk}$

		
OR rule 	$P(l) = P(i) + P(j) - P(i)P(j)C_{ij}$	$C_{lm} = \dfrac{P(i)C_{ik} + P(j)C_{jk} - P(i)P(j)C_{ik}C_{jk}C_{ij}}{P(i)+P(j)-P(i)P(j)C_{ij}}$
NOT rule 	$P(l) = 1 - P(i)$	$C_{lm} = \dfrac{P(l/m)}{P(l)} = \dfrac{1-P(i)C_{ik}}{1-P(i)}$

By using this approach, the probability of the output of a two-input gate can be more accurately calculated, given the probabilities of the two inputs and an associated correlation factor associated with the two signals. In this algorithm, the correlation factor can also be calculated analytically by means of a set of basic propagation rules (as shown in Table 3-3). CCM algorithm is given as follows:

CCM algorithm: Compute signal probability of each signal in a circuit.

Input: Probabilities and correlation coefficients of primary input signals.

Output: Probabilities and correlation coefficients of all signals.

1. Compile the network transforming possible multiple inputs gates into a cascade of two input ones organizing the circuit into levels;
2. Initialize the correlation coefficients and the probabilities at primary inputs. Generally the probabilities of primary inputs are assumed to be

0.5 and are considered to be independent, thus the values for all correlation coefficients for the primary inputs are 1;

3. Calculate the node probabilities and signal correlation coefficients at each level by successively applying the rules, check that the calculated coefficients are within the bounds. If not, assign them the nearest bound values.

By applying this CCM algorithm to the circuit shown in Figure 3-2, the values of $P(y_1)$, $P(y_2)$, $P(y_3)$, and $P(y_4)$ are properly calculated and correspond to the actual values shown in Table 3-2. The time complexity of the CCM algorithm is $O(N^2)$ for a circuit with N gates.[3]

3.3.4 *BDD Algorithm*

Signal probabilities of any arbitrary Boolean expression can also be calculated using Binary Decision Diagrams (BDDs) [16, 17]. In general, each node of a circuit can be represented by a logic function and the functionality of a logic function can be graphically represented by Binary decision diagrams. Let us consider a Boolean function $f(x_1, x_2, \ldots, x_n)$, where variables $x_1, x_2, \ldots, x_n$ correspond to primary inputs. Function f can be represented using Shannon's expression [17] as follows:

[3] Sharper time complexity results can be obtained; for example, it can be shown that a circuit with $\sqrt{N}$ levels has a complexity of $O(N^{3/2})$

$$f = x_i \cdot f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n) + \overline{x_i} \cdot f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n). \tag{3.1}$$

The cofactors of Boolean function f with respect to x_i and $\overline{x_i}$ respectively are defined as

$$\begin{aligned} f_{x_i} &= f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n) \\ f_{\overline{x_i}} &= f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n). \end{aligned} \tag{3.2}$$

Thus functions f_{x_i} and $f_{\overline{x_i}}$ are obtained by replacing variable x_i with logic 1 and logic 0, respectively. Each node of the BDD represents an input x_i and the edges coming out of node x_i represent the value of input x_i either logic 1 or logic 0. By traversing the BDD from its root, one can determine the value of the function f by sequentially examining the values of the inputs.

As an example to illustrate the BDD representation, consider the Boolean function $f = x_1 \cdot x_2 + x_3$, which can be represented by the BDD shown in Figure 3-3. The leaf nodes represent the value of function f. For example, if one traverses the path of the graph by edges $x_1 = 1$, $x_2 = 0$, and $x_3 = 1$, then the function equals logic "1". The tree rooted to the left of x_1 represents function $f_{\overline{x_i}}$, while the tree rooted to the right of x_1 represents function f_{x_i}. We can see that the ordering of the nodes of the BDD has direct implications on the complexity of BDD.

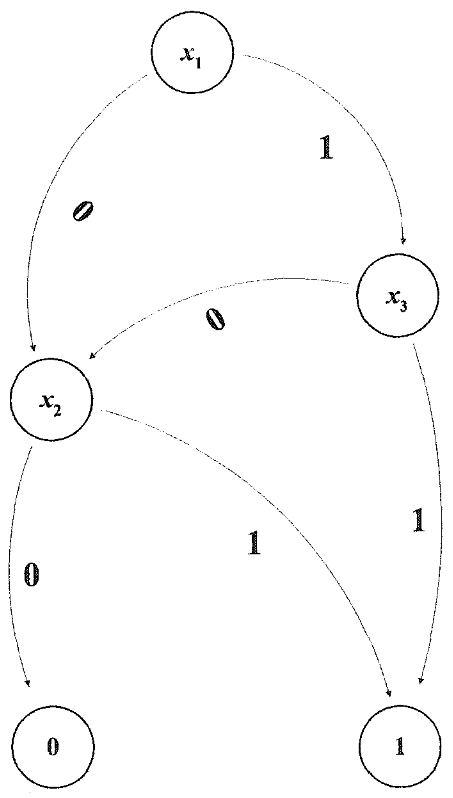


Figure 3-3. A BDD representation of Boolean function $f = x_1 \cdot x_2 + x_3$.

In general, let $y = f(x_1, x_2, \ldots, x_n)$ be Boolean function. If the primary inputs $x_1, x_2, \ldots, x_n$ are mutually independent, then the signal probability of y can be obtained in linear time (in the size of its BDD representation) as follows (using Equations 3.1 and 3.2):

$$\begin{aligned} p(y) &= p(x_1 \cdot f_{x_1} + \overline{x_1} \cdot f_{\overline{x_1}}) \\ &= p(x_1 \cdot f_{x_1}) + p(\overline{x_1} \cdot f_{\overline{x_1}}) \\ &= p(x_1) \cdot p(f_{x_1}) + p(\overline{x_1}) \cdot p(f_{\overline{x_1}}). \end{aligned} \tag{3.3}$$

The probability of y is stored in node x_1 as shown in Figure 3-3, and the probability of the cofactors are stored in node x_2 and x_3, respectively. The probability of the cofactors can now be represented in terms of its cofactors and so on. A depth-first traversal of BDD, with a post order evaluation of $p(.)$ at every node is required for evaluation of $p(y)$. This can be implemented using the "scan" function of the BDD package [5].

Another algorithm is proposed in [7] called the Weighted Averaging Algorithm (WAA), which generally achieves better accuracy than does the General Algorithm and has a comparable time complexity. However, the WAA still does not always produce correct values.

3.4 SIGNAL ACTIVITY CALCULATION

The average number of transitions per unit of time of a signal is defined as signal activity. The above-described approaches of [2], [6], and [7] are concerned with determining the probabilities of signal values, not the probabilities of signal transitions, i.e., activities, which are necessary for estimating power consumption. In general, there are two approaches for activity analysis, which are called the relative Boolean difference approach and the generalized Boolean difference approach. In this subsection, we will focus on the analysis of these two approaches.

3.4.1 RELATIVE BOOLEAN DIFFERENCE APPROACH

An early approach for estimating signal activities was developed in [3], in which signals of a circuit are modeled to be mutually independent strict-sense-stationary (SSS) mean-ergodic 0-1 processes. Under these assumptions, the activity of a signal y from a circuit with n-primary inputs can be expressed as

$$\alpha(y) = \sum_{i=1}^{n} P\left(\frac{\partial y}{\partial x_i}\right)\alpha(x_i) \tag{3.4}$$

where $\frac{\partial y}{\partial x_i}$ is the Boolean difference of function y with respect to x_i and is defined by

$$\begin{aligned}\frac{\partial y}{\partial x_i} = y\,|_{x_i=1} \oplus y\,|_{x_i=0} = {} & y(x_1,\cdots,x_{i-1},1,x_{i+1},\cdots,x_n) \\ & \oplus\, y(x_1,\cdots,x_{i-1},0,x_{i+1},\cdots,x_n).\end{aligned} \tag{3.5}$$

Intuitively, the Boolean difference $\frac{\partial y}{\partial x_i}$ defines whether a transition of signal x_i will cause a transition in output signal y. Specifically, if the Boolean difference function evaluates to one, then a transition of signal x_i causes a transition in y; if the Boolean difference function evaluates to zero, then a transition of signal x_i does not cause a transition in y. So, the probability of the Boolean difference function, $P\left(\frac{\partial y}{\partial x_i}\right)$, defines the probability that a change in y will occur given that there is a change in x_i. As an example of how to evaluate Equation 3.4, consider a simple case of a three-input AND function in which $y = x_1x_2x_3$.

$$\alpha(y) = \sum_{i=1}^{3} P\left(\frac{\partial y}{\partial x_i}\right)\alpha(x_i) \tag{3.6}$$

$$\frac{\partial y}{\partial x_1} = y\,|_{x_1=1} \oplus y\,|_{x_1=0} = (1\cdot x_2 \cdot x_3) \oplus (0 \cdot x_2 \cdot x_3) = x_2x_3$$

and similarly

$$\frac{\partial y}{\partial x_2} = x_1 x_3$$

$$\frac{\partial y}{\partial x_3} = x_1 x_2$$

Thus,

$$\alpha(y) = P(x_2 x_3)\alpha(x_1) + P(x_1 x_3)\alpha(x_2) + P(x_1 x_2)\alpha(x_3).$$

Because x_1, x_2, and x_3 are mutually independent, we can further simplify the probability terms as follows:

$$\begin{aligned}\alpha(y) = {} & P(x_2)P(x_3)\alpha(x_1) + P(x_1)P(x_3)\alpha(x_2) \\ & + P(x_1)P(x_2)\alpha(x_3)\end{aligned}. \tag{3.7}$$

The above expression is readily evaluated using the values of $P(x_i)$ and $\alpha(x_i)$, which are the known probabilities and activities of the primary input signals.

Although the calculation of the probability of the Boolean difference terms, i.e., $P\left(\frac{\partial y}{\partial x_i}\right)$, for the above example was relatively straightforward, this calculation can be complicated for large and complex circuits. In [3], the calculation of these terms is accomplished by first representing the nodes of the circuit with a binary decision diagram (BDD) [3, 5]. In practice, the BDD approach often achieves linear or near linear time complexity; however, in the worst case the complexity can grow exponentially with the number of gates.

3.4.2 *Generalized Boolean Difference Approach*

It is noted in [4] that Equation 3.4, i.e., the approach described in [3], fails to consider the effect of simultaneous switching of gate inputs. Figure 3-4 shows an example of how simultaneous switching of inputs to a logic gate affects the activity of the output node. As shown in the figure, if the two input signals always switch simultaneously, then the output signal of the XOR gate will have an activity of zero, even though the probability and activity terms in Equation 3.4 are nonzero [4]. This example is an extreme case, but is given to illustrate the importance of considering simultaneous switching.

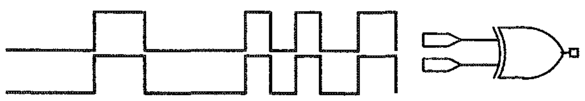


Figure 3-4. An example to illustrate the effect of simultaneous switching (derived from [4]).

Each Boolean difference term associated with Equation 3.4 describes an input-switching event in which exactly one of the inputs makes a transition. Thus, Equation 3.4 does not account for events involving simultaneous switching of two or more of the input signals. The concept of the generalized Boolean difference was introduced in [4] to account for simultaneous switching, and is denoted as follows:

$$\frac{\partial y^k \mid b_{i_1}, b_{i_2}, \ldots, b_{i_k}}{\partial x_{i_1} \partial x_{i_2} \ldots \partial x_{i_k}} = (y \mid x_{i_1} = b_{i_1}, x_{i_2} = b_{i_2}, \ldots x_{i_k} = b_{i_k}) \oplus (y \mid x_{i_1} = \overline{b_{i_1}}, x_{i_2} = \overline{b_{i_2}}, \ldots, x_{i_k} = \overline{b_{i_k}}), \tag{3.8}$$

where k is a positive integer, x_{i_j}, $j = 1, 2, ..., k$, are distinct mutually independent primary inputs of y, and b_{i_j} are binary values of "0" or "1". Note that if the generalized Boolean difference evaluates to one, then the simultaneous transitions of signals $(x_{i_1}, x_{i_2}, ..., x_{i_k})$ from $(b_{i_1}, b_{i_2}, ..., b_{i_k})$ to $(\overline{b_{i_1}}, \overline{b_{i_2}}, ..., \overline{b_{i_k}})$ or from $(\overline{b_{i_1}}, \overline{b_{i_2}}, ..., \overline{b_{i_k}})$ to $(b_{i_1}, b_{i_2}, ..., b_{i_k})$ will cause a transition at y.

Theorem 3.1 [11]: Assume that the primary inputs are mutually independent, and the logical signals can be modeled as *SSS* mean ergodic 0-1 discrete-time stochastic processes with logic modules having zero-delays. Also assume that signals can only transition at the leading edge of the clock cycle. Then the activity of a Boolean expression y with three primary inputs x_1, x_2 and x_3 (assumed mutually independent), i.e., $\alpha(y)$, can be expressed as

$$\begin{aligned} a(y) = & \sum_{i=1}^{3} Pc(\frac{\partial y}{\partial x_i})\left(a(x_i) \prod_{j \neq i, 1 \le j \le 3} [1 - a(x_i)] \right) \\ & + \frac{1}{2}\left\{ \sum_{1 \le i \le j \le 3} \left[Pc\left(\frac{\partial^2 y|_{00}}{\partial x_i \partial x_j} \right) + Pc\left(\frac{\partial^2 y|_{01}}{\partial x_i \partial x_j} \right) \right] \times \left(a(x_i) a(x_j) \prod_{l \in \{1,2,3\} - \{i,j\}} [1 - a(x_l)] \right) \right\} \\ & + \frac{1}{4}\left\{ \left[Pc\left(\frac{\partial^3 y|_{000}}{\partial x_1 \partial x_2 \partial x_3} \right) + Pc\left(\frac{\partial^3 y|_{001}}{\partial x_1 \partial x_2 \partial x_3} \right) + Pc\left(\frac{\partial^3 y|_{010}}{\partial x_1 \partial x_2 \partial x_3} \right) + Pc\left(\frac{\partial^3 y|_{011}}{\partial x_1 \partial x_2 \partial x_3} \right) \right] \times \prod_{l=1}^{3} a(x_l) \right\} \end{aligned} \tag{3.9}$$

where

$$Pc(\frac{\partial y}{\partial x_i}),\ Pc\left(\frac{\partial^2 y|_{00}}{\partial x_i \partial x_j} \right),\ \ldots,\ Pc\left(\frac{\partial^3 y|_{011}}{\partial x_1 \partial x_2 \partial x_3} \right)$$

are conditional probabilities under the condition that only the indicated primary inputs switch at the leading edge of the clock cycle and the rest do not.

Proof:

Because we assume that the module under consideration has zero delay and the primary inputs can switch only at the leading edge of the clock signal, the output signal will switch only at the leading edge of the clock signal. At any time t in which switching is possible, there will be only four kinds of events happening: none of the three inputs switching; one of the three inputs switching; two of the three inputs switching or all of the switching. The union of these four events is the sample space. To simplify the representation, assume x_i is the primary input, $i = 1, 2, 3$. Let event B_0 be the event with none of the three inputs switch. Let B_i be the event that only x_i switches. Let $B_{i,j}$, $i = 1, 2, 3$, $j = 1, 2, 3$, and $1 \le i < j \le 3$, be the events that only signal i and signal j switch at time t but the other signal does not switch. Finally let $B_{1,2,3}$ be the event that all three input signals switch at the same time. Based on the above definition, all the events are mutually exclusive; therefore, they form a partition of the sample space. Because signal x_1, x_2 and x_3 are mutually independent,

$$P(B_0) = [1-\alpha(x_1)][1-\alpha(x_2)][1-\alpha(x_3)]$$

Similarly,

$$P(B_1) = \alpha(x_1)[1-\alpha(x_2)][1-\alpha(x_3)]$$

$$P(B_2) = \alpha(x_2)[1-\alpha(x_1)][1-\alpha(x_3)]$$

$$P(B_3) = \alpha(x_3)[1-\alpha(x_1)][1-\alpha(x_2)]$$

$$P(B_{i,j}) = \alpha(x_i)\alpha(x_j)[1-\alpha(x_k)], i=1,2,3; j=1,2,3; k=1,2,3; i \neq j \neq k$$

and

$$P(B_{1,2,3}) = \alpha(x_1)\alpha(x_2)\alpha(x_3)$$

Because the probability of the union of a set of disjoint events is the sum of the probabilities of the individual events, and assuming event A represents the event that y is switching at time t, this leads to

$$\begin{aligned} P(A) = {} & P(A/B_0)P(B_0) + \sum_{i=1}^{3} P(A/B_i)P(B_i) \\ & + \sum_{1\le i<j\le 3} P(A/B_{i,j})P(B_{i,j}) + P(A/B_{1,2,3})P(B_{1,2,3}) \end{aligned} \tag{3.10}$$

We know that if none of the primary inputs switches at time t, the output signal y will not switch at time t, then $P(A/B_0) = 0$. If there is only one signal switching at time t, i.e., signal x_i is switching, then the conditional probability can be expressed as

$$P(A/B_i) = P_c\left(\frac{\partial y}{\partial x_i}\right) \tag{3.11}$$

When there are two signals switching at time t, i.e., signals x_i and x_j switch simultaneously, there are four possible cases: signals x_i and x_j both transition from low to high; both switch from high to low; x_i switches from low to high and x_j switches from high to low; or signal x_i switch from high to low and x_j switches from low to high. Because a rising transition at any node is always followed by a falling transition and vice versa, the conditional probability $P(A/B_{i,j})$ can be expressed as [4]

$$P(A/B_{i,j}) = \frac{1}{2}\left[Pc\left(\frac{\partial^2 y|_{00}}{\partial x_i \partial x_j}\right) + Pc\left(\frac{\partial^2 y|_{01}}{\partial x_i \partial x_j}\right)\right] \tag{3.12}$$

Similarly,

$$P(A/B_{1,2,3}) = \frac{1}{4}\left[\begin{array}{l} Pc\left(\dfrac{\partial^3 y|_{000}}{\partial x_1 \partial x_2 \partial x_3}\right) + Pc\left(\dfrac{\partial^3 y|_{011}}{\partial x_1 \partial x_2 \partial x_3}\right) \\ + Pc\left(\dfrac{\partial^3 y|_{010}}{\partial x_1 \partial x_2 \partial x_3}\right) + Pc\left(\dfrac{\partial^3 y|_{011}}{\partial x_1 \partial x_2 \partial x_3}\right) \end{array}\right]. \tag{3.13}$$

Q.E.D.

The conditional probability $Pc(x)$ can be calculated as follows: Assume signal x only switches at the leading edge of the clock signal, and t is some leading edge of the clock and T is the clock period. From the definition of generalized Boolean difference as shown in Equation 3.8, it is noted that x is actually an expression of primary input signals except those signals simultaneously switching at time t. To simplify the expression, we use $x(t-T)x(t)=1$ and $\bar{x}(t-T)\bar{x}(t)=1$ to represent that signal x does not switch at time t, i.e., from "1" to "1" and from "0" to "0" respectively. Similarly $x(t-T)\bar{x}(t)=1$ and $\bar{x}(t-T)x(t)=1$ represent that signal x does switch from "1" to "0" and from "0" to "1" respectively. Also assume that the probability and activity of signal x are $P(x)$ and $\alpha(x)$ respectively. Then we have

$$P(x \text{ is not switching at time } t) = P(x(t-T)x(t) + \bar{x}(t-T)\bar{x}(t)) = 1 - \alpha(x)$$

$$P(x \text{ is switching at time } t) = P(x(t-T)\bar{x}(t) + \bar{x}(t-T)x(t)) = \alpha(x)$$

Because we assume the signal is a SSS 0-1 process and mean-ergodic, the following equations hold:

$$P(x(t-T)x(t) + \bar{x}(t-T)\bar{x}(t)) = P(x(t-T)x(t)) + P(\bar{x}(t-T)\bar{x}(t))$$

$$P(x(t-T)\bar{x}(t) + \bar{x}(t-T)x(t)) = P(x(t-T)\bar{x}(t)) + P(\bar{x}(t-T)x(t))$$

$$P(x(t-T) = P(x(t)) = P(x)$$

$$P(\bar{x}(t-T)) = P(\bar{x}(t)) = 1 - P(x)$$

Since every transition from "1" to "0" will always be followed by a transition from "0" to "1" and verse visa, then we have

$$P(x(t-T)\bar{x}(t)) = P(\bar{x}(t-T)x(t)) = \frac{1}{2}\alpha(x) \tag{3.14}$$

In fact, $x(t) = x(t)x(t-T) + x(t)\bar{x}(t-T)$, then

$$\begin{aligned} P(x(t)) &= P(x(t)x(t-T) + x(t)\bar{x}(t-T)) \\ &= P(x(t)x(t-T)) + P(x(t)\bar{x}(t-T)) = P(x) \end{aligned} \tag{3.15}$$

Solving Equation 3.15 using Equation 3.14, results in

$$P(x(t-T)x(t)) = P(x) - \frac{1}{2}\alpha(x)$$

So the conditional probability of signal x being "1" while it does not switch at time t can be expressed as:

$$Pc(x) = P(x(t) = 1 \mid x \text{ does not switching at time } t)$$

Using the definition of conditional probability [8] that for two events A and B, the conditional probability $P(A/B)$ is defined as

$$P(A/B) = \frac{P(AB)}{P(B)}$$

Then

$$\begin{aligned} Pc(x) &= \frac{P(x(t)=1\ \&\ x \text{ does not switch at time } t\,)}{P(x \text{ does not switch at time } t)} \\ &= \frac{P(x(t)[x(t-T)x(t)+\overline{x}(x-T)\overline{x}(t)]=1)}{P([x(t-T)x(t)+\overline{x}(x-T)\overline{x}(t)]=1)} \\ &= \frac{P(x(t-T)x(t))}{1-\alpha(x)} = \frac{P(x)-\frac{1}{2}\alpha(x)}{1-\alpha(x)} \end{aligned}$$

So the conditional probability of signal x being "1" under the condition that x does not switch at time t is given by

$$Pc(x) = \frac{P(x)-\frac{1}{2}\alpha(x)}{1-\alpha(x)} \tag{3.16}$$

Equation 3-9 can be generalized to n-inputs and the proof is similar to that of the 3-input. Assume y is a Boolean expression and $x_i, i = 1,2,\ldots,n$, are mutually independent primary inputs of y, the activity of y can be expressed as [11]

$$\begin{aligned} \alpha(y) = &\sum_{i=1}^{n} Pc\left(\frac{\partial y}{\partial x_i}\right)\left(\alpha(x_i)\prod_{\substack{j\neq i\\ 1\le j\le n}}[1-\alpha(x_i)]\right) \\ &+\frac{1}{2}\left\{\sum\left[Pc\left(\frac{\partial^2 y|_{00}}{\partial x_i \partial x_j}\right)+Pc\left(\frac{\partial^2 y|_{01}}{\partial x_i \partial x_j}\right)\right]\left(\alpha(x_i)\alpha(x_j)\prod_{l\in\{1,2,\ldots,n\}-\{i,j\}}[1-\alpha(x_l)]\right)\right\}+\ldots \\ &+\frac{1}{2^{n-1}}\left[\begin{array}{l} Pc\left(\frac{\partial^n y|_{00\ldots0}}{\partial x_1 \partial x_2 \ldots \partial x_n}\right)+Pc\left(\frac{\partial^n y|_{00\ldots1}}{\partial x_1 \partial x_2 \ldots \partial x_n}\right) \\ +\ldots+Pc\left(\frac{\partial^n y|_{01\ldots1}}{\partial x_1 \partial x_2 \ldots \partial x_n}\right) \end{array}\right]\left(\prod_{l=1}^{n}\alpha(x_l)\right). \end{aligned} \tag{3.17}$$

where $Pc\left(\frac{\partial y}{\partial x_i}\right)$, $Pc\left(\frac{\partial^2 y|_{00}}{\partial x_i \partial x_j}\right)$, … , $Pc\left(\frac{\partial^n y|_{01\ldots1}}{\partial x_1 \partial x_2 \cdots \partial x_n}\right)$ are conditional probabilities of the generalized Boolean differences under the condition that only the indicated inputs simultaneously switch, and the rest do not.

We can see that the result obtained by considering the simultaneous switching activity is different with the result obtained by ignoring the simultaneous switching activity. Compare the results as shown in Equation 3.6, it is apparent that if the effect of simultaneous switching is neglected, $Pc(\partial y / \partial x_i) = P(\partial y / \partial x_i)$, and $\left\{\alpha(x_i)\prod_{j\neq i, 1\leq j\leq n}\left[1-\alpha(x_j)\right]\right\}$ is equal to $\alpha(x_i)$ and the above expression becomes identical to Equation 3.6.

Let's use an example to show how to apply Equation 3.9. Consider a simple logic expression as $y = x_1 x_2 \overline{x_3}$, with $x_i, i = 1,2,3$ input signals with probability and activity p_i and α_i, respectively. Then we have

$$\frac{\partial y}{\partial x_1} = (x_2 \overline{x_3}) \oplus 0 = x_2 \overline{x_3}$$
$$\frac{\partial y}{\partial x_2} = (x_1 \overline{x_3}) \oplus 0 = x_1 \overline{x_3}$$
$$\frac{\partial y}{\partial x_3} = 0 \oplus x_1 x_2 = x_1 x_2$$

$$P_c\left(\frac{\partial y}{\partial x_1}\right) = Pc(x_2 \overline{x_3}) = \frac{P(x_2 \overline{x_3}) - \frac{1}{2}\alpha(x_2 \overline{x_3})}{1-\alpha(x_2 \overline{x_3})} \tag{3.18}$$

$$P_c\left(\frac{\partial y}{\partial x_2}\right) = Pc(x_1 \overline{x_3}) = \frac{P(x_1 \overline{x_3}) - \frac{1}{2}\alpha(x_1 \overline{x_3})}{1-\alpha(x_1 \overline{x_3})} \tag{3.19}$$

$$P_c(\frac{\partial y}{\partial x_3}) = Pc(x_1x_2) = \frac{P(x_1x_2) - \frac{1}{2}\alpha(x_1x_2)}{1-\alpha(x_1x_2)} \tag{3.20}$$

and

$$\frac{\partial^2 y\,|_{00}}{\partial x_1 \partial x_2} = \overline{x_3} \oplus 0 = \overline{x_3}$$
$$\frac{\partial^2 y\,|_{01}}{\partial x_1 \partial x_2} = 0 \oplus 0 = 0$$

$$Pc(\frac{\partial^2 y\,|_{00}}{\partial x_1 \partial x_2}) + Pc(\frac{\partial^2 y\,|_{01}}{\partial x_1 \partial x_2}) = \frac{P(\overline{x_3}) - \frac{1}{2}\alpha(\overline{x_3})}{1-\alpha(\overline{x_3})}$$

$$\frac{\partial^2 y\,|_{00}}{\partial x_1 \partial x_3} = 0 \oplus 0 = 0$$
$$\frac{\partial^2 y\,|_{01}}{\partial x_1 \partial x_3} = 0 \oplus x_2 = x_2$$

$$Pc(\frac{\partial^2 y\,|_{00}}{\partial x_1 \partial x_3}) + Pc(\frac{\partial^2 y\,|_{01}}{\partial x_1 \partial x_3}) = \frac{P(x_2) - \frac{1}{2}\alpha(x_2)}{1-\alpha(x_2)}$$

$$\frac{\partial^2 y\,|_{00}}{\partial x_2 \partial x_3} = 0 \oplus 0 = 0$$
$$\frac{\partial^2 y\,|_{01}}{\partial x_2 \partial x_3} = 0 \oplus x_1 = x_1$$

$$Pc(\frac{\partial^2 y\,|_{00}}{\partial x_2 \partial x_3}) + Pc(\frac{\partial^2 y\,|_{01}}{\partial x_2 \partial x_3}) = \frac{P(x_1) - \frac{1}{2}\alpha(x_1)}{1-\alpha(x_1)}.$$

It is also the case that

$$\frac{\partial^3 y|_{000}}{\partial x_1 \partial x_2 \partial x_3} = 0 \oplus 0 = 0$$

$$\frac{\partial^3 y|_{001}}{\partial x_1 \partial x_2 \partial x_3} = 0 \oplus 1 = 1$$

$$\frac{\partial^3 y|_{010}}{\partial x_1 \partial x_2 \partial x_3} = 0$$

$$\frac{\partial^3 y|_{011}}{\partial x_1 \partial x_2 \partial x_3} = 0$$

To calculate the conditional probability as shown in Equation 3.18, $\alpha(x_2 \overline{x_3})$ needs to be calculated first. This can be done by deriving the activity calculation of the 2-input, based on the same conditional probability calculation. The method used to solve Equations 3.19 and 3.20 is similar to that for Equation 3.18 and the results are shown as following:

$$\alpha(x_2 \overline{x_3}) = (1 - p_3)\alpha_2(1 - a_3) + p_2\alpha_3(1 - a_2) + \frac{1}{2}\alpha_2\alpha_3\,,$$

$$\alpha(x_1 \overline{x_3}) = (1 - p_3)\alpha_1(1 - a_3) + p_1\alpha_3(1 - a_1) + \frac{1}{2}\alpha_1\alpha_3\,,$$

$$\alpha(x_1 x_2) = p_2\alpha_1(1 - \alpha_2) + p_1\alpha_2(1 - \alpha_1) + \frac{1}{2}\alpha_1\alpha_2\,.$$

The final symbolic analytical result is very complicated. To compare the result of using Equation 3.4 (without considering simultaneous switching effect) with the result by using Equation 3.9 (considering simultaneous switching effect), we assume values for probabilities and activities as $p_1 = 0.88$, $p_2 = 0.29$, $p_3 = 0.69$, $\alpha_1 = 0.1$, $\alpha_2 = 0.17$, and $\alpha_3 = 0.27$, then the activity of y of expression $y = x_1 x_2 \overline{x_3}$ is $\alpha(y) = 0.124$ by using Equation 3.4 and $\alpha(y) = 0.09345$ by using Equation 3.9.

Observe that the result obtained by considering the simultaneous switching activity is different than the result obtained by ignoring the simultaneous switching activity (the difference for the example considered is about 33%). The difference arises due to the generalized Boolean difference that accounts for simultaneous switching. In general, the approach of Equation 3.9 yields more accurate results than Equation 3.4. However, the overall complexity associated with evaluating Eq. 3.9 is generally much larger than that of Equation 3.4. This high complexity is due to a potentially large number of terms (exponential in the number of inputs) and the complexity associated with evaluating the conditional probabilities.

3.5 Summary

The signal model for the three approaches overviewed in this chapter is based on a single probability parameter [2, 6, 7]. Although this probability parameter is not directly used in calculating a circuit's power consumption, it is a necessary component for signal models common to the activity approaches which utilize both signal probability and signal activity parameters [3, 4].

The approaches of [2], [3], and [4] can have high computational complexities because the number of terms in the underlying equations/transformations can grow exponentially with the number of primary inputs to the circuit. In [7], a trade-off between computational complexity and resulting accuracy is illustrated in the context of the underlying equations/transformations introduced in [2]. In particular, an approximate approach is defined in [7] in which the transformations of [2] are applied in a "gate-by-gate" fashion. Thus, instead of deriving the transformation for a signal's probability parameter in terms of the circuit's primary inputs, it is derived in terms of the immediate inputs to the logic gate associated with the signal. This approach greatly reduces the computational complexity, but introduces error in the calculated probability parameters for circuits with re-convergent fan-out.

Similar trade-offs between computational complexity and accuracy are possible relative to the evaluation of Equation 3.4 and Equation 3.9 (associated with [3] and [4], respectively). Instead of deriving a signal's logic function in terms of the circuit's primary inputs, the parameters to the immediate inputs the signal's logic gate can be used. Again, this type of "gate-by-gate" technique will generally introduce error because it does not account for correlations present among the internal signals that drive the gates within the circuit.

The approach of [6] is a fast and accurate "gate-by-gate" technique for calculating a signal's probability parameter. It introduces the concept of a correlation factor to account for and appropriately adjust the transformation for correlated inputs to a gate.

CHAPTER 4

MARKOV-CHAIN SIGNAL MODEL AND MCP ALGORITHM

4.1 INTRODUCTION

In Chapter 3, signals are modeled as strict-sense stationary (SSS) 0-1 process and are assumed to be mean-ergodic. Based on this signal modeling, the probability of a logic signal $x(t)$ is defined as the average fraction of time that the signal is high, and the signal activity of a logic signal $x(t)$ is the average number of transitions, i.e., $n(T)$, in a time interval of length T. By defining a relative Boolean difference, signal activities can be derived by sum of products of activities of all primary inputs (assumed mutually independent) and the probabilities of their Boolean difference [3]. In this approach, simultaneous switching is ignored, thus introducing errors. Further more, the computational complexity is not efficient because the probabilities of the relative Boolean difference[4] must be calculated. In another approach, the generalized Boolean difference was introduced to account for simultaneous switching, and the activities of signals can be achieved by

[4] OBDD [5] can be used to calculate the probabilities of the Boolean difference, but construction the BDD diagram might result in an exponential time in worst case.

computing the sum of products of the activities of all primary inputs (also assumed to be mutually independent) and the conditional probabilities of enumeration of all possible generalized Boolean differences [4]. This approach gives us a more accurate solution but an exponential time complexity due to calculation of the conditional probabilities of the generalized Boolean difference that is impractical for large and complicated circuits.

In this chapter, we introduce a more efficient and more accurate algorithm (named MCP algorithm) based on Markov-chain signal modeling. By propagating signal parameters and correlation cofactors from the primary inputs through the circuit in a "gate-by-gate" fashion, our MCP algorithm can achieve a very good accuracy and an $O(M^2)$ time complexity where M is the number of signals in the circuit. The signal model we introduced here is based on a Markov chain having two event parameters. It is shown that the proposed Markov chain model is equivalent to the two-parameter probability/activity signal model of [3] and [4]. The advantage of modeling signals with Markov chains is that it makes it possible to compute correlations between signals related to both probability and activity.

The approach derived here can be viewed as a generalization of the approach in [6]. Instead of tracking a correlation factor for the single probability parameter model, transformations for correlation factors associated with the two parameters of the Markov model are derived. This ultimately leads to a fast and accurate "gate-by-gate" algorithm for calculating signal probabilities and activities.

4.2 MARKOV CHAIN SIGNAL MODEL

4.2.1 PRELIMINARIES

Assume a Markov chain consists of a set of states denoted as $U=\{s_1,s_2,\ldots,s_n,\ldots\}$, then two elements s_i and s_j are said to be in the same equivalence class if they can communicate to each other. The minimal elements (i.e., terminals) of the partial ordering of equivalence classes are called *ergodic sets*, the remaining elements are called *transient sets;* and the elements of an ergodic set are called *ergodic states*. A chain consisting of a single ergodic set is called *an ergodic chain*.

Let signal A(t) be strict-sense stationary (SSS) 0-1 process and mean-ergodic. Under the zero-delay model, signal A(t) can be modeled as a Markov chain process over the state set $\Omega = \{0,1\}$ with the transition matrix

$$Q = \begin{bmatrix} state & 0 & 1 \\ 0 & 1-P(A_1) & P(A_1) \\ 1 & P(A_2) & 1-P(A_2) \end{bmatrix}, \qquad (4.1)$$

where $P(A_1)$ and $P(A_2)$ denote the transition probability corresponding to probability of transition from state 0 to state 1 (i.e., event A_1) and probability of transition from state 1 to 0 (i.e., event A_2) respectively.

As illustrated in Figure 4-1, the proposed Markov chain signal model has two event parameters for the signal A. Events A_1 and A_2 are used to represent the events that there is a transition from state 0 to 1 and from state 1 to 0, respectively. There is no transient set because it is possible to go from any state to any other state. Hence there is a single ergodic set, and this Markov chain is an ergodic chain with one cyclic class.

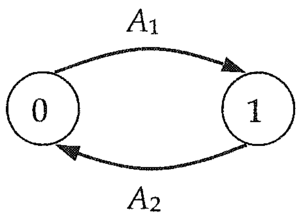


Figure 4-1. Proposed Markov chain signal model.

Hence, based on this Markov-chain model, signal probability and activity can be defined as:

Definition 4.1 (Signal Probability): The probability of a logic signal A, denoted by $P(A)$, is the probability of signal A being in state 1.

$$P(A) = P(A = 1). \tag{4.2}$$

Definition 4.2 (Signal Activity): The signal activity of a logic signal A is the sum of transition probability transition from state 1 to state 0 and transition from state 0 to state 1, and can be expressed as

$$\alpha(A) = P(A = 0)P(A_1) + P(A = 1)P(A_2). \tag{4.3}$$

Applying balance equations of Markov chain (flow-in equals to flow-out for state 1), we have

$$P(A=0)P(A_1)+P(A=1)(1-P(A_2))=P(A=1). \tag{4.4}$$

Solving Equation 4.4 results in

$$P(A)=\frac{P(A_1)}{P(A_1)+P(A_2)}. \tag{4.5}$$

Replacing $P(A)$ using Equation 4.5, Equation 4.3 becomes

$$\alpha(A)=\frac{2P(A_1)P(A_2)}{P(A_1)+P(A_2)}. \tag{4.6}$$

Thus, if the values of both the transition parameters associated with the proposed Markov model of a signal are known (i.e., $P(A_1)$ and $P(A_2)$), then the probability and the activity of the signal are completely determined by using Equation 4.5 and Equation 4.6. Likewise, knowing the probability and activity values of the signal fully determines the two transition parameters of the Markov chain model of the signal and can be expressed as

$$P(A_1)=\frac{\alpha(A)}{2(1-P(A))}, \tag{4.7}$$

$$P(A_2)=\frac{\alpha(A)}{2P(A)}. \tag{4.8}$$

4.2.2 *Definition of Correlation Cofactors*

In order to define correlations between two signals modeled with Markov chains, some basic definitions are needed. Let A and B denote two events and let $P(AB)$ denote the probability of both A and B occurring. From basic probability theory

[8], $P(AB) = P(A/B)P(B)$, where $P(A/B)$ represents the probability of A given B. Also, the correlation coefficient of two events A and B is defined as

$$\rho_{AB} = \frac{\sigma_{AB}}{\sigma_A \sigma_B},$$

where σ_{AB} is the covariance and σ_A and σ_B are the positive square roots of the variances of A and B. It can be shown [8] that

$$\rho_{AB} = \frac{P(AB) - P(A)P(B)}{\sqrt{P(A)(1-P(A))}\sqrt{P(B)(1-P(B))}}. \tag{4.9}$$

In order to simplify later derivations, it is convenient to define the correlation factor C_{AB} of two events A and B as

$$C_{AB} = \frac{P(AB)}{P(A)P(B)} = \frac{P(A/B)}{P(A)} = \frac{P(B/A)}{P(B)}. \tag{4.10}$$

By applying Eq. 4.9 to Eq. 4.10, the following relationship can be derived:

$$\rho_{AB} = \frac{P(A)}{\sqrt{P(A)(1-P(A))}} \frac{P(B)}{\sqrt{P(B)(1-P(B))}} (C_{AB} - 1). \tag{4.11}$$

Thus, C_{AB} is related to ρ_{AB} through scaling and shifting. The value of ρ_{AB}, by definition [8], is a real number in the interval [-1, 1]; therefore, according to Eq. 4.11, C_{AB} takes on real non-negative values. Also, $\rho_{AB} = 0$ corresponds to $C_{AB} =1$, and indicates that the events A and B are mutually independent. Similarly, $\rho_{AB} < 0$ (i.e., A and B are negatively correlated) corresponds to $0 \le C_{AB} < 1$, and $\rho_{AB} > 0$ (i.e., A and B are positively correlated) corresponds to $C_{AB} > 1$.

4.3 Markov Chain Model for Basic Logic Gates

The focus in this subsection is on deriving the Markov chain model for the output of a basic logic gate in which the Markov chain models of the input signals are known. The simple case of a NOT gate is considered first followed by the analysis of two-input basic logic gates.

For a NOT gate with input *A*, the Boolean output function is given by $Y = \overline{A}$. From Figure 4-1, it is clear that the Markov model for *Y* is given by

$$P(Y_1) = P(A_2),\ P(Y_2) = P(A_1). \qquad (4.12)$$

Consider now the case of a two-input basic logic gate, as shown in Figure 4-2. Assuming the Markov chain models of *A* and *B* are known, the objective is to derive the Markov chain model for output signal *Y*.

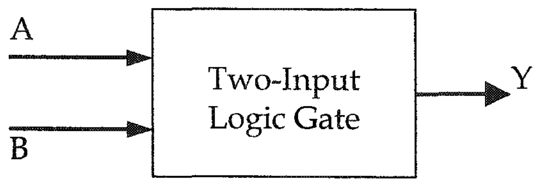


Figure 4-2. Generic two-input logic gate.

A key to deriving the Markov chain model for signal *Y* of Figure 4-2 is to represent the state transition diagram associated with the gate's two inputs, as shown in Figure 4-3. The four states in the figure correspond to the four input combinations for the two inputs. The first digit of each state label corresponds to the value of *A*, and the second to the value of *B*, e.g., the state labeled "01" corresponds to $A = 0$ and $B = 1$. Although not labeled on the figure, the directed edges represent transition events. To illustrate the notation to label transition events, "00→10" will be used to represent the event that input signal A transitions from 0 to 1 and signal B stays in state 0.

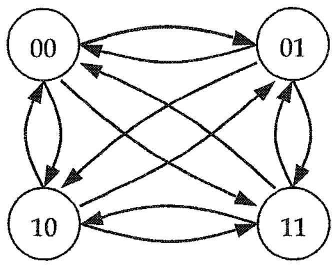


Figure 4-3. State transition diagram for inputs A and B of Figure 4-2.

The known parameters of the Markov chain models for signals A and B are given by $P(A_1)$, $P(A_2)$, $P(B_1)$, and $P(B_2)$. Also assumed to be known are the correlation factors for pairs of events associated with the Markov chain models for the inputs.[5] From Eq. 4.10 note that $P(AB) = P(A)P(B)C_{AB}$, where C_{AB} is the correlation factor associated with events A and B. Similarly, the correlation factor $C_{A_1B_2}$ enables the calculation of $P(A_1B_2)$ using the fact that $P(A_1B_2) = P(A_1)P(B_2)C_{A_1B_2}$. Recall from Eq. 4.11 that independent events correspond to a correlation factor of unity.

[5] Deriving transformations to determine correlations factors associated with pairs of signals will be discussed in next section; for purposes of the present section they are assumed to be known.

Given the Markov chain models for signals *A* and *B* (and the corresponding correlation factors) it is possible to derive the probability associated with every event shown in the state transition diagram of Figure 4-3. To illustrate, consider the probability of event 00→01:

$$\begin{aligned}
P(00 \rightarrow 01) &= P(\overline{A_1}B_1) \\
&= P(\overline{A_1} / B_1)P(B_1) \\
&= [1 - P(A_1 / B_1)]P(B_1) \\
&= P(B_1) - P(A_1 / B_1)P(B_1) \\
&= P(B_1) - P(A_1B_1) \\
&= P(B_1) - P(A_1)P(B_1)C_{A_1B_2}
\end{aligned}$$

Expressions for the probabilities of all events associated with the state transition diagram of Figure 4-3 can be derived similarly; a complete tabulation of these expressions are given in Table 4-1. For notational convenience and clarity, we will denote the value of *P*(*A*) as p_A (for the value of the probability of signal *A*) and the value of the activity $\alpha(A)$ as α_A (for the value of the activity of signal *A*) throughout the rest of the dissertation.

Table 4-1. Probability expressions of 16 transition edges associated with Figure 4-3.

Event	Probability
00→00	$P(00 \rightarrow 00) = 1 - \frac{\alpha_A}{2(1-p_A)} - \frac{\alpha_B}{2(1-p_B)} + \frac{\alpha_A}{2(1-p_A)}\frac{\alpha_B}{2(1-p_B)}C_{A_1B_1}$
00→01	$P(00 \rightarrow 01) = \frac{\alpha_B}{2(1-p_B)}\left(1 - \frac{\alpha_A}{2(1-p_A)}C_{A_1B_1}\right)$
00→11	$P(00 \rightarrow 11) = \frac{\alpha_A}{2(1-p_A)}\frac{\alpha_B}{2(1-p_B)}C_{A_1B_1}$
00→10	$P(00 \rightarrow 10) = \frac{\alpha_A}{2(1-p_A)}\left(1 - \frac{\alpha_B}{2(1-p_B)}C_{A_1B_1}\right)$

01→00	$P(01 \to 00) = \frac{\alpha_B}{2p_B}\left(1 - \frac{\alpha_A}{2(1-p_A)} C_{A_1B_2}\right)$
01→01	$P(01 \to 01) = 1 - \frac{\alpha_B}{2p_B} - \frac{\alpha_A}{2(1-p_A)}\left(1 - \frac{\alpha_B}{2p_B} C_{A_1B_2}\right)$
01→10	$P(01 \to 10) = \frac{\alpha_A}{2(1-p_A)} \frac{\alpha_B}{2p_B} C_{A_1B_2}$
01→11	$P(01 \to 11) = \frac{\alpha_A}{2(1-p_A)}(1 - \frac{\alpha_B}{2p_B} C_{A_1B_2})$
10→00	$P(10 \to 00) = \frac{\alpha_A}{2p_A}\left(1 - \frac{\alpha_B}{2(1-p_B)} C_{A_2B_1}\right)$
10→01	$P(10 \to 01) = \frac{\alpha_A}{2p_A} \frac{\alpha_B}{2(1-p_B)} C_{A_2B_1}$
10→10	$P(10 \to 10) = 1 - \frac{\alpha_A}{2p_A} - \frac{\alpha_B}{2(1-p_B)} + \frac{\alpha_A}{2p_A} \frac{\alpha_B}{2(1-p_B)} C_{A_2B_1}$
10→11	$P(10 \to 11) = \frac{\alpha_B}{2(1-p_B)}(1 - \frac{\alpha_A}{2p_A} C_{A_2B_1})$
11→00	$P(11 \to 00) = \frac{\alpha_A}{2p_A} \frac{\alpha_B}{2p_B} C_{A_2B_2}$
11→01	$P(11 \to 01) = \frac{\alpha_A}{2p_A}\left(1 - \frac{\alpha_B}{2p_B} C_{A_2B_2}\right)$
11→10	$P(11 \to 10) = \frac{\alpha_B}{2p_B}\left(1 - \frac{\alpha_A}{2p_A} C_{A_2B_2}\right)$
11→11	$P(11 \to 11) = 1 - \frac{\alpha_A}{2p_A} - \frac{\alpha_B}{2p_B} + \frac{\alpha_A}{2p_A} \frac{\alpha_B}{2p_B} C_{A_2B_2}$

After the sixteen transition edges have been derived, the 4-by-4 transition matrix is determined, then the probabilities of the four states as shown in Figure 4-3 can be derived as follows (based on the balance equations of the Markov chain):

$$P(00) = P(01)P(01 \to 00) + P(10)P(10 \to 00) + P(11)P(11 \to 00) + P(00)P(00 \to 00)$$

$$P(01) = P(01)P(01 \to 01) + P(10)P(10 \to 01) + P(11)P(11 \to 01) + P(00)P(00 \to 01)$$

$P(10) = P(01)P(01 \to 10) + P(10)P(10 \to 10) + P(11)P(11 \to 10) + P(00)P(00 \to 10)$

$P(11) = P(01)P(01 \to 11) + P(10)P(10 \to 11) + P(11)P(11 \to 11) + P(00)P(00 \to 11)$

and

$$P(00) + P(10) + P(01) + P(11) = 1\,.$$

Solving the above five equations (one balance equation is redundant) and using the results listed in Table 4.1, we can derive the solutions of the probabilities of the four states, i.e., $P(00)$, $P(01)$, $P(10)$ and $P(11)$. Because the probability expressions for these four states are very complicated, we introduce correlation cofactor of signal probabilities, denoted as C_{AB}, to simplify those expressions. C_{AB} is only used for simplification purpose, and it can be derived and expressed by $P(A_1)$, $P(A_2)$, $P(B_1)$, $P(B_2)$ and their correlations. Let

$$C_{AB} = \frac{\begin{pmatrix} P_{A_1}P_{B_1}C_{A_1B_1}(1-P_A-P_B)+ \\ P_B(P_{A_1}-P_{A_1}P_{B_2}C_{A_1B_2})+P_A(P_{B_1}-P_{A_2}P_{B_1}C_{A_2B_1}) \end{pmatrix}}{\begin{pmatrix} P_{A_1}+P_{A_2}+P_{B_1}+P_{B_2}-P_{A_1}P_{B_2}C_{A_1B_2} \\ -P_{A_2}P_{B_1}C_{A_2B_1}-P_{A_1}P_{B_1}C_{A_1B_1} \end{pmatrix}} \tag{4.13}$$

be the correlation cofactor, the probabilities of these four states can be simply expressed as

$$P(00) = 1 - p_A - p_B + p_A p_B C_{AB} \tag{4.14}$$

$$P(01) = p_B - p_A p_B C_{AB} \tag{4.15}$$

$$P(10) = p_A - p_A p_B C_{AB} \tag{4.16}$$

$$P(11) = p_A p_B C_{AB} \tag{4.17}$$

Deriving a Markov chain model for Y of Figure 4-2 depends on the particular function of the gate. To illustrate how to determine the Markov chain model for Y, consider the specific example of an AND gate, i.e., $Y = AB$. For an AND gate, the output takes on logic value "1" if and only if both inputs are "1". Thus,

$$P(Y) = P(11) = p_A p_B C_{AB}\,. \tag{4.18}$$

The event Y_1 is associated with three events from Figure 4-3, namely: 00→11, 01→11, and 10→11. Thus, equality can be established as follows:

$$\begin{aligned} P(\overline{Y})P(Y_1) = P(00)P(00 \to 11) + P(01)P(00 \to 11) \\ + P(01)P(00 \to 11) \end{aligned}\,. \tag{4.19}$$

Solving Eq. 4.19 for $P(Y_1)$ and using Eqs. 4.7 and 4.8 result in the following expression:

$$\begin{aligned} P(Y_1) = (\frac{1}{2}\lambda_A p_B \alpha_A + \frac{1}{2}\lambda_B p_A \alpha_B)/(1 - p_A p_B C_{AB}) \\ -\left[\frac{1}{4}\left(\lambda_A C_{A_1B_2} + \lambda_B C_{A_2B_1} - \lambda C_{A_1B_1}\right)\alpha_A \alpha_B /(1 - p_A p_B C_{AB})\right] \end{aligned} \tag{4.20}$$

where $\lambda_A = \dfrac{1 - p_A C_{AB}}{1 - p_A}$, $\lambda_B = \dfrac{1 - p_B C_{AB}}{1 - p_B}$ and $\lambda = \dfrac{1 - p_A - p_B + p_A p_B C_{AB}}{(1 - p_A)(1 - p_B)}$.

Derivation for $P(Y_2)$ follows in a similar fashion and can be expressed as

$$P(Y_2) = \frac{\alpha_A}{2p_A} + \frac{\alpha_B}{2p_B} - \frac{\alpha_A}{2p_A}\frac{\alpha_B}{2p_B}C_{A_2B_2}\,. \tag{4.21}$$

To use only two events associated with signal A and signal B, i.e., $P(A_1)$, $P(A_2)$, $P(B_1)$, $P(B_2)$, Equation 4.20 and Equation 4.21 can also be derived as follows. The event Y_2 is associated with two events from Figure 4-3, namely: 11→00, 11→01, and 11→10. Thus, equality can also be established as:

$$P(Y_2) = P(11 \to 10) + P(11 \to 00) + P(11 \to 01) \\ = 1 - P(11 \to 11) \quad (4.22)$$

Solving Equation 4.22 for $P(Y_2)$ and using Table 4-1 results in the following expression:

$$P(Y_2) = P(A_2) + P(B_2) - P(A_2)P(B_2)C_{A_2B_2}. \quad (4.23)$$

Using the fact that each signal's transition from "1" to "0" will always be followed by a transition from "0" to "1", means number of transitions from "1" to "0" is equal to the number of transitions from "0" to "1" in a long time period. Based on this fact, the following equation holds,

$$P(Y)P(Y_2) = P(\overline{Y})P(Y_1).$$

So derivation for $P(Y_1)$ can be obtained as

$$P(Y_1) = \frac{P(Y)P(Y_2)}{P(\overline{Y})} = \frac{P(Y)P(Y_2)}{1 - P(Y)}. \quad (4.24)$$

Using Equation 4.18 and Equation 4.23 results in

$$P(Y_1) = \frac{P(A_1)P(B_1)C_{AB}\left(P(A_2) + P(B_2) - P(A_2)P(B_2)C_{A_2B_2}\right)}{P(A_1)P(B_2) + P(A_2)P(B_1) + P(A_2)P(B_2)}. \quad (4.25)$$

Having the probabilities of the two events associated with signal Y, the probability and activity of output signal Y can be derived and expressed as

$$\alpha(Y) = 2P(Y)P(Y_2) = 2P(Y)\left(P(A_2) + P(B_2) - P(A_2)P(B_2)C_{A_2B_2}\right) \\ = 2\frac{P(A_1)}{P(A_1) + P(A_2)}\frac{P(B_1)}{P(B_1) + P(B_2)}C_{AB}\left(P(A_2) + P(B_2) - P(A_2)P(B_2)C_{A_2B_2}\right) \quad (4.26)$$

$$P(Y) = \frac{P(A_1)}{P(A_1) + P(A_2)}\frac{P(B_1)}{P(B_1) + P(B_2)}C_{AB} \quad (4.27)$$

Or by using activities and probabilities of inputs A and B:

$$\alpha(Y) = p_B\alpha_A C_{AB} + p_A\alpha_B C_{AB} - \frac{1}{2}\alpha_A\alpha_B C_{AB} C_{A_2B_2} \tag{4.28}$$

$$P(Y) = p_A p_B C_{AB} \,. \tag{4.29}$$

Derivations of $P(Y)$, $P(Y_1)$, and $P(Y_2)$ for two-input OR and XOR gates, i.e. $Y=A+B$ and $Y = A \oplus B$ respectively, are similar to the above derivation for the AND gate and the results are shown in Table 4-2. To reduce the notational burden, the formulas in Table 4-2 are expressed in terms of signal probabilities and activities instead of the Markov chain parameters (i.e., Eqs. 4.7 and 4.8 were applied).

Table 4-2. Formulas for computing Markov chain parameters for the output of basic gates.

Gate	$P(Y_1)$	$P(Y_2)$
NOT $Y = \overline{A}$	$P(A_2)$	$P(A_1)$
AND $Y = AB$	$\frac{1}{2}\frac{\lambda_A p_B\alpha_A + \lambda_B p_A\alpha_B)}{1-p_A p_B C_{AB}} - \frac{1}{4}\frac{\left(\lambda_A C_{A_1B_2} + \lambda_B C_{A_2B_1} - \lambda C_{A_1B_1}\right)\alpha_A\alpha_B}{1-p_A p_B C_{AB}}$	$\frac{\alpha_A}{2p_A} + \frac{\alpha_B}{2p_B} - \frac{\alpha_A}{2p_A}\frac{\alpha_B}{2p_B}C_{A_2B_2}$
OR $Y = A+B$	$\frac{\alpha_A}{2(1-p_A)} + \frac{\alpha_B}{2(1-p_B)} - \frac{\alpha_A}{2(1-p_A)}\frac{\alpha_B}{2(1-p_B)}C_{A_1B_1}$	$\frac{\left(1-p_A-p_B+p_A p_B C_{AB}\right)}{4(1-p_A)(1-p_B)\left(p_A+p_B-p_A p_B C_{AB}\right)} \times \frac{\alpha_A(1-p_B)+(1-p_A)\alpha_B - \alpha_A\alpha_B C_{A_1B_1}}{2(1-p_A)(1-p_B)\left(p_A+p_B-p_A p_B C_{AB}\right)}$

XOR $Y = A \oplus B$	$\frac{1}{2}\frac{\lambda - p_B\lambda + p_B C_{AB}}{1 - p_A - p_B + 2p_A p_B C_{AB}}\alpha_A + \frac{1}{2}\frac{\lambda - p_A\lambda + p_A C_{AB}}{1 - p_A - p_B + 2p_A p_B C}\alpha_B - \frac{1}{2}\frac{\lambda C_{A_1B_1} + C_{A_2B_2}}{1 - p_A - p_B + 2p_A p_B C_{AB}}\alpha_A\alpha_B$	$\frac{\frac{1}{2}p_B\lambda_A + \frac{1}{2}(1-p_B)\lambda_B}{p_A + p_B - 2p_A p_B C_{AB}}\alpha_A + \frac{\frac{1}{2}p_A\lambda_B + \frac{1}{2}(1-p_A)\lambda_A}{p_A + p_B - 2p_A p_B C_{AB}}\alpha_B - \frac{\frac{1}{2}\left(\lambda_A C_{A_1B_2} + \lambda_B C_{A_2B_1}\right)}{p_A + p_B - 2p_A p_B C_{AB}}\alpha_A\alpha_B$

$$C_{AB} = \frac{P_{A_1}P_{B_1}C_{A_1B_1}(1 - P_A - P_B) + P_B(P_{A_1} - P_{A_1}P_{B_2}C_{A_1B_2}) + P_A(P_{B_1} - P_{A_2}P_{B_1}C_{A_2B_1})}{P_{A_1} + P_{A_2} + P_{B_1} + P_{B_2} - P_{A_1}P_{B_2}C_{A_1B_2} - P_{A_2}P_{B_1}C_{A_2B_1} - P_{A_1}P_{B_1}C_{A_1B_1}}$$

$$\lambda_A = \frac{1 - p_A C_{AB}}{1 - p_A} \quad \lambda_B = \frac{1 - p_B C_{AB}}{1 - p_B} \quad \lambda = \frac{1 - p_A - p_B + p_A p_B C_{AB}}{(1 - p_A)(1 - p_B)}$$

Applying Eqs. 4.7 and 4.8, and using the parameter results listed in Table 4-2, the probability and activity values of the output signal Y of these two-input AND, OR and XOR gates and the NOT gate can be derived and the results are shown in Table 4-3.

Table 4-3. Probability and activity values of output signals of basic gates.

Gate	p_Y	α_Y
NOT $Y = \overline{A}$	$1 - p_A$	α_A
AND $Y = AB$	$p_A p_B C_{AB}$	$p_B\alpha_A C_{AB} + p_A\alpha_B C_{AB} - \frac{1}{2}\alpha_A\alpha_B C_{AB}C_{A_2B_2}$
OR $Y = A + B$	$p_A + p_B - p_A p_B C_{AB}$	$(1 - p_B)\lambda\alpha_A + (1 - p_A)\lambda\alpha_B - \frac{1}{2}\lambda\alpha_A\alpha_B C_{A_1B_1}$

XOR $Y = A \oplus B$	$p_A + p_B - 2p_A p_B C_{AB}$	$(\lambda_B + p_B(\lambda_A - \lambda_B))\alpha_A + (\lambda_A + p_A(\lambda_B - \lambda_A))\alpha_B - (\lambda_A C_{A_1B_2} + \lambda_B C_{A_2B_1})\alpha_A \alpha_B$
$C_{AB} = \frac{P_{A_1} P_{B_1} C_{A_1B_1}(1 - P_A - P_B) + P_B(P_{A_1} - P_{A_1} P_{B_2} C_{A_1B_2}) + P_A(P_{B_1} - P_{A_2} P_{B_1} C_{A_2B_1})}{P_{A_1} + P_{A_2} + P_{B_1} + P_{B_2} - P_{A_1} P_{B_2} C_{A_1B_2} - P_{A_2} P_{B_1} C_{A_2B_1} - P_{A_1} P_{B_1} C_{A_1B_1}}$ $\lambda_A = \frac{1 - p_A C_{AB}}{1 - p_A}$ $\lambda_B = \frac{1 - p_B C_{AB}}{1 - p_B}$ $\lambda = \frac{1 - p_A - p_B + p_A p_B C_{AB}}{(1 - p_A)(1 - p_B)}$		

4.4 CALCULATION OF CORRELATION FACTORS

The purpose of this section is to provide methods for calculating/propagating correlation factors through basic elements of a circuit. For two signals *A* and *B*, there are two kinds of correlations that need to be established: probability correlation factor denoted as C_{AB} (corresponding to correlation factor between signal *A* and signal *B* which is used to simplify the expressions of probabilities of the four states as shown in Figure 4-3) and transition correlation donated as $C_{A_iB_j}$ (corresponding to correlation factor between event A_i and event B_j), where $i, j \in \{1,2\}$ and A_i and B_j are transition events corresponding to signal *A* and signal *B* respectively as shown in Figure 4-1. For three events *A*, *B* and *C*, the correlations among these events are very complicated and are difficult to derive. Let's first denote correlation cofactor between event $A \cap B$, denoted as event *A* and B, and event C as $C_{AB,C}$. Using the definition of correlation cofactor as shown in Equation 4.10, $C_{AB,C}$ can be expressed as

$$C_{AB,C} = \frac{P(ABC)}{P(AB)P(C)}. \tag{4.30}$$

Replacing $P(AB)$ by using Equation 4.10,

$$C_{AB,C} = \frac{P(ABC)}{P(A)P(B)C_{AB}P(C)},$$

then we have

$$C_{AB,C}C_{AB} = \frac{P(ABC)}{P(A)P(B)P(C)}.$$

Similarly, $C_{AC,B}$ and $C_{BC,A}$ denote correlation cofactors of event $A \cap C$ and event B, event $B \cap C$ and event A, respectively, then the following equation holds,

$$C_{AB,C}C_{AB} = \frac{P(ABC)}{P(A)P(B)P(C)} = C_{AC,B}C_{AC} = C_{BC,A}C_{BC}. \tag{4.31}$$

The exact analytical expressions of $C_{AB,C}$, $C_{AC,B}$ and $C_{BC,A}$ are difficult to derive. Assume the correlation of correlation of two events to the third one can be neglected, then $C_{AB,C}$, $C_{AC,B}$ and $C_{BC,A}$ can be expressed as

$$C_{AB,C} = C_{AC}C_{BC} \tag{4.32}$$

$$C_{AC,B} = C_{AB}C_{BC} \tag{4.33}$$

$$C_{BC,A} = C_{AB}C_{AC}. \tag{4.34}$$

In general, this assumption is incorrect and will cause errors. To illustrate, consider an example as shown in Figure 4-4. Assume the primary inputs x_1, x_2, x_3

and x_4 are mutually independent, signals *A, B* and *C* are the three inputs to the AND gate that generates the output signal *Y*. From Figure 4-4, we get

$$A = x_1 x_2$$

$$B = x_1 x_3$$

$$C = x_1 x_4,$$

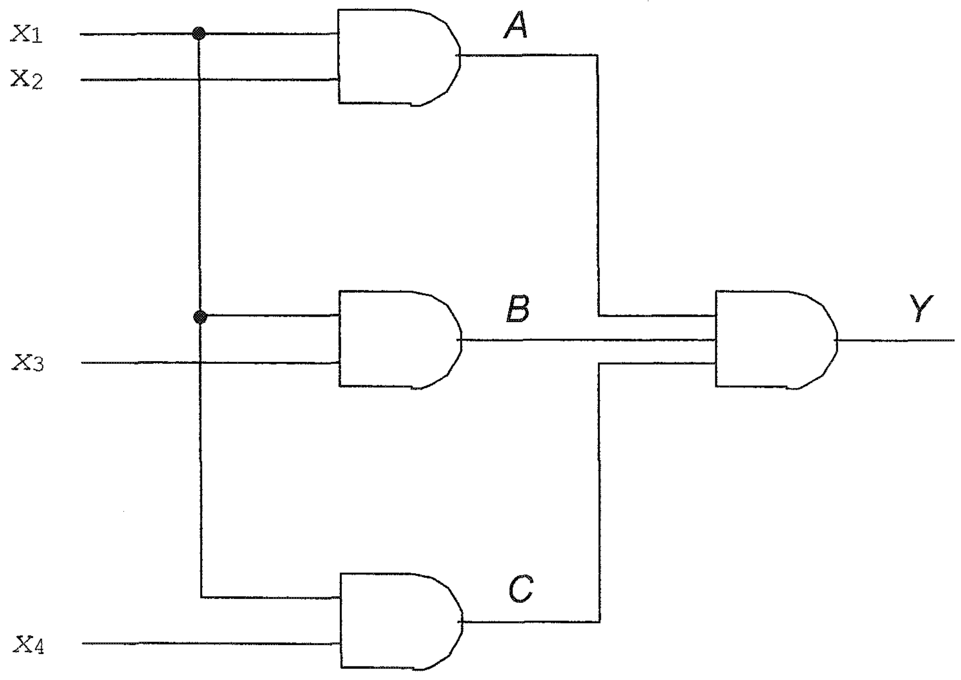


Figure 4-4. An Example to show the calculation of correlations among three signals.

and signal *A, B* and *C* are correlated. Using the definition of correlation factor as shown in Equation 4.10, the correlation cofactor of two signals can be derived as

$$C_{AB} = \frac{P(AB)}{P(A)P(B)} = \frac{P(x_1 x_2 x_1 x_3)}{P(x_1 x_2)P(x_1 x_3)} = \frac{1}{P(x_1)}$$

$$C_{AC} = \frac{P(AC)}{P(A)P(C)} = \frac{P(x_1 x_2 x_1 x_4)}{P(x_1 x_2)P(x_1 x_4)} = \frac{1}{P(x_1)}$$

$$C_{BC} = \frac{P(BC)}{P(B)P(C)} = \frac{P(x_1x_3x_1x_4)}{P(x_1x_3)P(x_1x_4)} = \frac{1}{P(x_1)}$$

The output signal Y can also be expressed as

$$Y = ABC = x_1x_2x_3x_4$$

Using Equation 4.30 and suppressing the exponent, we have

$$C_{AB,C} = \frac{P(ABC)}{P(AB)P(C)} = \frac{P(x_1x_2x_3x_4)}{P(x_1x_2x_3)P(x_4)} = \frac{1}{P(x_1)}.$$

So we can see that

$$C_{AC}C_{BC} = \frac{1}{[P(x_1)]^2} \neq C_{AB,C}. \tag{4.35}$$

To account for the above inequality (i.e., error), new concepts of conditional independence and signal isotropy were introduced in [13]. Two signals A, B are conditionally independent with respect to C when the following condition holds:

$$P(AB/C) = P(A/C)P(B/C). \tag{4.36}$$

Under this condition, the problem of handling correlations among three signals can be reduced to the problem of handling correlations of pairwise signals. If two of the three events are assumed to be conditionally independent, then we have the exact expression of $C_{AB,C}$, $C_{AC,B}$ and $C_{BC,A}$ as given in the following theorem.

Theorem 4-1: Given three events A, B and C,

$$\text{If } C_{AB} = 1 \text{ then } C_{AB,C} = C_{AC}C_{BC};$$

$$\text{If } C_{AC} = 1 \text{ then } C_{AC,B} = C_{AB}C_{BC};$$

$$\text{If } C_{BC} = 1 \text{ then } C_{BC,A} = C_{AB}C_{AC}.$$

Proof:

Given three events A, B and C, if $C_{AB} = 1$, which means events A and B are mutually independent, then we have

$$P(AB) = P(A)P(B)$$

thus

$$P((AB)/C) = P(A/C)P(B/C)$$

$$P(AB)C_{AB,C} = P(A)C_{AC}P(B)C_{BC}$$

$$P(A)P(B)C_{AB,C} = P(A)C_{AC}P(B)C_{BC}.$$

So

$$C_{AB,C} = C_{AC}C_{BC}. \tag{4.37}$$

Similarly, we can prove expressions for $C_{AC,B}$ and $C_{BC,A}$.

Q.E.D.

However, because the problem of finding a variable x such that the rest signals are conditionally independent is an NP-complete problem [13], the concepts of almost conditional independence and almost isotropy were proposed in [13]. A set of n signals $\{x_i\}$, $1 \le i \le n$, is called ε-isotropic if there exists some ε ($\varepsilon \ge 0$) such that

$$\left| \frac{\prod_{1\le j\le n, j\ne i} P(x_j \mid x_i)}{P(\prod_{1\le j\le n, j\ne i} x_j \mid x_i)} - 1 \right| \le \varepsilon \text{ for any } i = 1,2,\cdots,n. \tag{4.38}$$

The usefulness of the above result is twofold. First the small number ε is an upper bound of the relative error of the calculated correlation cofactor; and second, it is proved that it is not profitable to express a node with signals beyond some L predecessor levels, based on isotropy.

Neglecting the correlation of correlation of two events to a third one, correlation propagation rules can be established as follows:

The first rule to be established is the fan-out rule associated with the circuit diagram in Figure 4-5.

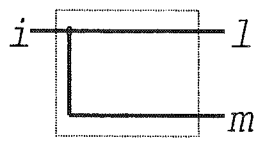


Figure 4-5. The circuit diagram associated with the fan-out rule.

Because signal l is the same signal as m,

$$P(l) = P(m) = P(i),$$

$$\because P(lm) = P(l/m)P(m) \ \ and \ \ P(l/m) = 1$$
$$\therefore C_{lm} = \frac{P(lm)}{P(l)P(m)} = \frac{1}{P(l)} = \frac{1}{P(i)}$$

$$P(l_1) = P(m_1) = P(i_1)$$

$$\because P(l_1/m_1) = 1$$
$$\therefore C_{l_1m_1} = \frac{P(l_1/m_1)}{P(l_1)} = \frac{1}{P(i_1)}$$
$$= \frac{2(1-p_i)}{\alpha_i}$$

Similarly,

$$C_{l_2m_2} = \frac{1}{P(i_2)} = \frac{2p_i}{\alpha_i}$$
$$C_{l_1m_2} = 0$$
$$C_{l_2m_1} = 0$$

The second rule is named AND rule and is associated with the circuit diagram in Figure 4-6.

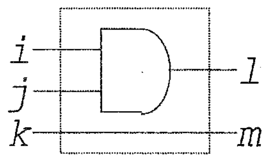


Figure 4-6. The circuit diagram associated with the AND rule.

Given correlation factors between input signals *i*, *j* and *k*, the correlation factors between output signals *l* and *m* can be derived by follows:

Because $P(lm) = P(l/m)P(m)$ and using the results in Table 4-2, $P(l) = P(i)P(j)C_{ij}$,

$$\begin{aligned} P(l/m) &= P(i/m)P(j/m)C_{ij} \\ &= P(i/k)P(j/k)C_{ij} \\ &= P(i)C_{ik}P(j)C_{jk}C_{ij} \\ &= P(l)C_{ik}C_{jk} \\ \therefore C_{lm} &= C_{ik}C_{jk} \end{aligned}$$

$$P(l_1m_1) = P(l_1)P(m_1)C_{l_1m_1} = P(l_1/m_1)P(m_1)$$

so

$$C_{l_1m_1} = \frac{P(l_1/m_1)}{P(l_1)}$$

$$\begin{aligned}
P(\bar{l})P(l_1) &= P(00)P(00 \to 11) + P(01)P(01 \to 11) + P(10)P(10 \to 11) \\
&= P(\bar{i}\bar{j})P(i_1 j_1) + P(\bar{i}j)P(i_1 \bar{j_2}) + P(i\bar{j})P(\bar{i_2} j_1) \\
&= \left(1 - P(i) - P(j) + P(i)P(j)C_{ij}\right)P(i_1)P(j_1)C_{i_1 j_1} \\
&+ \left(P(j) - P(i)P(j)C_{ij}\right)\left(P(i_1) - P(i_1)P(j_2)C_{i_1 j_2}\right) \\
&+ \left(P(i) - P(i)P(j)C_{ij}\right)\left(P(j_1) - P(i_2)P(j_1)C_{i_2 j_1}\right)
\end{aligned} \tag{4.39}$$

$$\begin{aligned}
P(\bar{l})P(l_1 / m_1) &= \left(1 - P(i) - P(j) + P(i)P(j)C_{ij}\right)P(i_1 / m_1)P(j_1 / m_1)C_{i_1 j_1} \\
&+ \left(P(j) - P(i)P(j)C_{ij}\right)\left(P(i_1 / m_1) - P(i_1 / m_1)P(j_2 / m_1)C_{i_1 j_2}\right) \\
&+ \left(P(i) - P(i)P(j)C_{ij}\right)\left(P(j_1 / m_1) - P(i_2 / m_1)P(j_1 / m_1)C_{i_2 j_1}\right) \\
&= \left(1 - P(i) - P(j) + P(i)P(j)C_{ij}\right)P(i_1)C_{i_1 m_1}P(j_1)C_{j_1 m_1}C_{i_1 j_1} \\
&+ \left(P(j) - P(i)P(j)C_{ij}\right)\left(P(i_1)C_{i_1 m_1} - P(i_1)C_{i_1 m_1}P(j_2)C_{j_2 m_1}C_{i_1 j_2}\right) \\
&+ \left(P(i) - P(i)P(j)C_{ij}\right)\left(P(j_1)C_{j_1 m_1} - P(i_2)C_{i_2 m_1}P(j_1)C_{j_1 m_1}C_{i_2 j_1}\right) \\
&= \frac{1}{2}\lambda_i p_j \alpha_i C_{i_1 m_1} + \frac{1}{2}\lambda_j p_i \alpha_j C_{j_1 m_1} \\
&- \frac{1}{4}\left(\lambda_i C_{i_1 m_1} C_{j_2 m_1} C_{i_1 j_2} + \lambda_j C_{i_2 m_1} C_{j_1 m_1} C_{i_2 j_1} - \lambda C_{i_1 m_1} C_{j_1 m_1} C_{i_1 j_1}\right)\alpha_i \alpha_j
\end{aligned} \tag{4.40}$$

So

$$C_{l_1 m_1} = \frac{P(l_1 / m_1)}{P(l_1)} = \frac{P(\bar{l})P(l_1 / m_1)}{P(\bar{l})P(l_1)} \ . \tag{4.41}$$

Solving Eq. 4.41 by applying Eqs. 4.39 and 4.40,

$$\begin{aligned}
C_{l_1 m_1} &= \frac{2}{\alpha(l)}\begin{pmatrix} \frac{1}{2}\lambda_i p_j \alpha_i C_{i_1 m_1} + \frac{1}{2}\lambda_j p_i \alpha_j C_{j_1 m_1} \\ -\frac{1}{4}\left(\lambda_i C_{i_1 m_1} C_{j_2 m_1} C_{i_1 j_2} + \lambda_j C_{i_2 m_1} C_{j_1 m_1} C_{i_2 j_1} - \lambda C_{i_1 m_1} C_{j_1 m_1} C_{i_1 j_1}\right)\alpha_i \alpha_j \end{pmatrix} \\
&= \lambda_i p_j C_{i_1 k_1} \frac{\alpha_i}{\alpha_l} + \lambda_j p_i C_{j_1 k_1} \frac{\alpha_j}{\alpha_l} \\
&- \frac{1}{2}\left(\lambda_i C_{i_1 k_1} C_{j_2 k_1} C_{i_1 j_2} + \lambda_j C_{i_2 k_1} C_{j_1 k_1} C_{i_2 j_1} - \lambda C_{i_1 k_1} C_{j_1 k_1} C_{i_1 j_1}\right)\frac{\alpha_i \alpha_j}{\alpha_l}
\end{aligned}$$

Other correlation factors (i.e., $C_{l_1 m_2}$, $C_{l_2 m_1}$, and $C_{l_2 m_2}$) can be obtained similarly:

$$C_{l_1m_2} = \frac{P(l_1/m_2)}{P(l_1)} = \frac{P(\bar{l})P(l_1/m_2)}{P(\bar{l})P(l_1)}$$

$$= \lambda_i p_j C_{i_1k_2}\frac{\alpha_i}{\alpha_l} + \lambda_j p_i C_{j_1k_2}\frac{\alpha_j}{\alpha_l}$$

$$-\frac{1}{2}\left(\lambda_i C_{i_1k_2}C_{j_2k_2}C_{i_1j_2} + \lambda_j C_{i_2k_2}C_{j_1k_2}C_{i_2j_1} - \lambda C_{i_1k_2}C_{j_1k_2}C_{i_1j_1}\right)\frac{\alpha_i\alpha_j}{\alpha_l}$$

$$\begin{aligned} P(l_2) &= P(11\rightarrow 00) + P(11\rightarrow 01) + P(11\rightarrow 10) \\ &= 1 - P(11\rightarrow 11) \\ &= P(i_2) + P(j_2) - P(i_2)P(j_2)C_{i_2j_2} \end{aligned}$$

$$\begin{aligned} P(l_2/m_1) &= P(i_2/m_1) + P(j_2/m_1) - P(i_2/m_1)P(j_2/m_1)C_{i_2j_2} \\ &= P(i_2)C_{i_2m_1} + P(j_2)C_{j_2m_1} - P(i_2)C_{i_2m_1}P(j_2)C_{j_2m_1}C_{i_2j_2} \end{aligned}$$

$$\begin{aligned} C_{l_2m_1} &= \frac{P(l_2/m_1)}{P(l_2)} \\ &= \frac{P(i_2)C_{i_2m_1} + P(j_2)C_{j_2m_1} - P(i_2)C_{i_2m_1}P(j_2)C_{j_2m_1}C_{i_2j_2}}{P(i_2) + P(j_2) - P(i_2)P(j_2)C_{i_2j_2}} \\ &= \frac{p_B\alpha_A C_{i_2k_1} + p_A\alpha_B C_{j_2k_1} - \frac{1}{2}\alpha_A\alpha_B C_{i_2k_1}C_{j_2k_1}C_{i_2j_2}}{p_B\alpha_A + p_A\alpha_B - \frac{1}{2}\alpha_A\alpha_B C_{i_2j_2}} \end{aligned}$$

$$\begin{aligned} C_{l_2m_2} &= \frac{P(l_2/m_2)}{P(l_2)} \\ &= \frac{P(i_2)C_{i_2m_2} + P(j_2)C_{j_2m_2} - P(i_2)C_{i_2m_2}P(j_2)C_{j_2m_2}C_{i_2j_2}}{P(i_2) + P(j_2) - P(i_2)P(j_2)C_{i_2j_2}} \\ &= \frac{p_B\alpha_A C_{i_2k_2} + p_A\alpha_B C_{j_2k_2} - \frac{1}{2}\alpha_A\alpha_B C_{i_2k_2}C_{j_2k_2}C_{i_2j_2}}{p_B\alpha_A + p_A\alpha_B - \frac{1}{2}\alpha_A\alpha_B C_{i_2j_2}} \end{aligned}$$

Derivations of correlation factors for OR and XOR gates follow in a similar fashion.

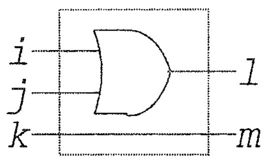


Figure 4-7. The circuit diagram associated with the OR rule.

Figure 4-7 shows the circuit diagram associated with the OR rule, and the correlation cofactors of the OR rule can be derived as follows:

$$P(l_1 m_1) = P(l_1)P(m_1)C_{l_1 m_1} = P(l_1 / m_1)P(m_1)$$

so

$$C_{l_1 m_1} = \frac{P(l_1 / m_1)}{P(l_1)}$$

$$\begin{aligned} P(l_1) &= P(00 \to 01) + P(00 \to 11) + P(00 \to 10) \\ &= 1 - P(00 \to 00) \\ &= P(i_1) + P(j_1) - P(i_1)P(j_1)C_{i_1 j_1} \end{aligned}$$

$$\begin{aligned} P(l_1 / m_1) &= P(i_1 / m_1) + P(j_1 / m_1) - P(i_1 / m_1)P(j_1 / m_1)C_{i_1 j_1} \\ &= P(i_1)C_{i_1 m_1} + P(j_1)C_{j_1 m_1} - P(i_1)C_{i_1 m_1}P(j_1)C_{j_1 m_1}C_{i_1 j_1} \\ &= P(i_1)C_{i_1 k_1} + P(j_1)C_{j_1 k_1} - P(i_1)C_{i_1 k_1}P(j_1)C_{j_1 k_1}C_{i_1 j_1} \end{aligned}$$

$$\begin{aligned} C_{l_1 m_1} &= \frac{P(l_1 / m_1)}{P(l_1)} \\ &= \frac{P(i_1)C_{i_1 k_1} + P(j_1)C_{j_1 k_1} - P(i_1)C_{i_1 k_1}P(j_1)C_{j_1 k_1}C_{i_1 j_1}}{P(i_1) + P(j_1) - P(i_1)P(j_1)C_{i_1 j_1}} \\ &= \frac{(1 - p_j)\alpha_i C_{i_1 k_1} + (1 - p_i)\alpha_j C_{j_1 k_1} - \frac{1}{2}\alpha_i \alpha_j C_{i_1 k_1} C_{j_1 k_1} C_{i_1 j_1}}{(1 - p_j)\alpha_i + (1 - p_i)\alpha_j - \frac{1}{2}\alpha_i \alpha_j C_{i_1 j_1}} \end{aligned}$$

$$C_{l_1m_2} = \frac{P(l_1/m_2)}{P(l_1)}$$

$$= \frac{(1-p_j)\alpha_i C_{i_1k_2} + (1-p_i)\alpha_j C_{j_1k_2} - \frac{1}{2}\alpha_i\alpha_j C_{i_1k_2}C_{j_1k_2}C_{i_1j_1}}{(1-p_j)\alpha_i + (1-p_i)\alpha_j - \frac{1}{2}\alpha_i\alpha_j C_{i_1j_1}}$$

$$\begin{aligned}
P(Y)P(Y_2) &= P(01)P(01 \to 00) + P(10)P(10 \to 00) + P(11)P(11 \to 00) \\
&= P(01)P(\overline{i_1} j_2) + P(10)P(i_2 \overline{j_1}) + P(11)P(i_2 j_2) \\
&= P(01)P(j_2) - P(01)P(i_1)P(j_2)C_{i_1j_2} \\
&\quad + P(10)P(i_2) - P(10)P(i_2)P(j_1)C_{i_2j_1} \\
&\quad + P(11)P(i_2)P(j_2)C_{i_2j_2}
\end{aligned}$$

$$= (p_j - p_ip_jC_{ij})\frac{\alpha_j}{2p_j} - (p_j - p_ip_jC_{ij})\left(\frac{\alpha_i}{2(1-p_i)}\frac{\alpha_j}{2p_j}\right)C_{i_1j_2}$$

$$+ (p_i - p_ip_jC_{ij})\frac{\alpha_i}{2p_i} - (p_i - p_ip_jC_{ij})\left(\frac{\alpha_j}{2(1-p_j)}\frac{\alpha_i}{2p_i}\right)C_{i_2j_1}$$

$$+ p_ip_jC_{ij}\frac{\alpha_i}{2p_i}\frac{\alpha_j}{2p_j}C_{i_2j_2}$$

$$= \frac{1}{2}(1-p_j)\lambda_j\alpha_i + \frac{1}{2}(1-p_i)\lambda_i\alpha_j - \frac{1}{4}\left(\lambda_iC_{i_1j_2} + \lambda_jC_{i_2j_1} - C_{ij}C_{i_2j_2}\right)\alpha_i\alpha_j$$
$$= \frac{1}{2}\alpha_l$$

$$\begin{aligned}
P(l)P(l_2/m_1) &= P(01)P(j_2/m_1) - P(01)P(i_1/m_1)P(j_2/m_1)C_{i_1j_2} \\
&\quad + P(10)P(i_2/m_1) - P(10)P(i_2/m_1)P(j_1/m_1)C_{i_2j_1} \\
&\quad + P(11)P(i_2/m_1)P(j_2/m_1)C_{i_2j_2} \\
&= P(01)P(j_2)C_{j_2m_1} - P(01)P(i_1)C_{i_1m_1}P(j_2)C_{j_2m_1}C_{i_1j_2} \\
&\quad + P(10)P(i_2)C_{i_2m_1} - P(10)P(i_2)C_{i_2m_1}P(j_1)C_{j_1m_1}C_{i_2j_1} \\
&\quad + P(11)P(i_2)C_{i_2m_1}P(j_2)C_{j_2m_1}C_{i_2j_2} \\
&= \frac{1}{2}(1-p_j)\lambda_j\alpha_iC_{i_2k_1} + \frac{1}{2}(1-p_i)\lambda_i\alpha_jC_{j_2k_1} \\
&\quad - \frac{1}{4}\left(\lambda_iC_{i_1j_2}C_{i_1k_1}C_{j_2k_1} + \lambda_jC_{i_2j_1}C_{i_2k_1}C_{j_1k_1} - C_{ij}C_{i_2j_2}C_{i_2k_1}C_{j_2k_1}\right)\alpha_i\alpha_j
\end{aligned}$$

$$C_{l_2m_1} = \frac{\frac{1}{2}(1-p_j)\lambda_j\alpha_i C_{i_2m_1} + \frac{1}{2}(1-p_i)\lambda_i\alpha_j C_{j_2m_1} - \frac{1}{4}\left(\lambda_i C_{i_1j_2}C_{i_1m_1}C_{j_2m_1} + \lambda_j C_{i_2j_1}C_{i_2m_1}C_{j_1m_1} - C_{ij}C_{i_2j_2}C_{i_2m_1}C_{j_2m_1}\right)\alpha_i\alpha_j}{\frac{1}{2}\alpha_l}$$

$$= (1-p_j)\lambda_j C_{i_2k_1}\frac{\alpha_i}{\alpha_l} + (1-p_i)\lambda_i C_{j_2k_1}\frac{\alpha_j}{\alpha_l}$$
$$-\frac{1}{2}\left(\lambda_i C_{i_1j_2}C_{i_1k_1}C_{j_2k_1} + \lambda_j C_{i_2j_1}C_{i_2k_1}C_{j_1k_1} - C_{ij}C_{i_2j_2}C_{i_2k_1}C_{j_2k_1}\right)\frac{\alpha_i\alpha_j}{\alpha_l}$$

$$C_{l_2m_2} = (1-p_j)\lambda_j C_{i_2k_2}\frac{\alpha_i}{\alpha_l} + (1-p_i)\lambda_i C_{j_2k_2}\frac{\alpha_j}{\alpha_l}$$
$$-\frac{1}{2}\left(\lambda_i C_{i_1j_2}C_{i_1k_2}C_{j_2k_2} + \lambda_j C_{i_2j_1}C_{i_2k_2}C_{j_1k_2} - C_{ij}C_{i_2j_2}C_{i_2k_2}C_{j_2k_2}\right)\frac{\alpha_i\alpha_j}{\alpha_l}$$

Figure 4-8 shows the Circuit diagram associated with the XOR rule, followed by the derivation of the correlation factors.

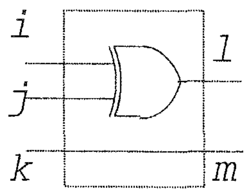


Figure 4-8. The circuit diagram associated with the XOR rule.

$$P(l) = P(i) + P(j) - 2P(i)P(j)C_{ij}$$

$$P(l/m) = P(i/m) + P(j/m) - 2P(i/m)P(j/m)C_{ij}$$
$$= P(i)C_{im} + P(j)C_{jm} - 2P(i)C_{im}P(j)C_{jm}C_{ij}$$

$$
\begin{aligned}
C_{lm} &= \frac{P(l/m)}{P(l)} \\
&= \frac{P(i)C_{im} + P(j)C_{jm} - 2P(i)C_{im}P(j)C_{jm}C_{ij}}{P(i) + P(j) - 2P(i)P(j)C_{ij}} \\
&= \frac{P(i)C_{ik} + P(j)C_{jk} - 2P(i)C_{ik}P(j)C_{jk}C_{ij}}{P(i) + P(j) - 2P(i)P(j)C_{ij}}
\end{aligned}
$$

$$
\begin{aligned}
P(\bar{l})P(l_1) &= P(00)P(00 \to 01) + P(00)P(00 \to 10) + P(11)P(11 \to 10) + P(11)P(11 \to 01) \\
&= P(00)\left(P(\overline{i_1}j_1) + P(i_1\overline{j_1})\right) + P(11)\left(P(\overline{i_2}j_2) + P(i_2\overline{j_2})\right) \\
&= \frac{1}{2}\left((1-p_j)\lambda\alpha_i + (1-p_i)\lambda\alpha_j - \lambda\alpha_i\alpha_j C_{i_1j_1}\right) + \frac{1}{2}\left(p_j\alpha_i + p_i\alpha_j - \alpha_i\alpha_j C_{i_2j_2}\right)C_{ij} \\
&= \frac{1}{2}\left((1-p_j)\lambda + p_jC_{ij}\right)\alpha_i + \frac{1}{2}\left((1-p_i)\lambda + p_iC_{ij}\right)\alpha_j - \frac{1}{2}\left(\lambda C_{i_1j_1} + C_{ij}C_{i_2j_2}\right)\alpha_i\alpha_j \\
&= \frac{1}{2}\alpha_l
\end{aligned}
$$

$$
\begin{aligned}
P(\bar{l})P(l_1/m_1) = {} & \frac{1}{2}\left((1-p_j)\lambda C_{i_1m_1} + p_jC_{ij}C_{i_2m_1}\right)\alpha_i + \frac{1}{2}\left((1-p_i)\lambda C_{j_1m_1} + p_iC_{ij}C_{j_2m_1}\right)\alpha_j \\
& - \frac{1}{2}\left(\lambda C_{i_1j_1}C_{i_1m_1}C_{j_1m_1} + C_{ij}C_{i_2j_2}C_{i_2m_1}C_{j_2m_1}\right)\alpha_i\alpha_j
\end{aligned}
$$

$$
\begin{aligned}
C_{l_1m_1} &= \frac{P(l_1/m_1)}{P(l_1)} \\
&= \frac{\frac{1}{2}\left((1-p_j)\lambda C_{i_1m_1} + p_jC_{ij}C_{i_2m_1}\right)\alpha_i + \frac{1}{2}\left((1-p_i)\lambda C_{j_1m_1} + p_iC_{ij}C_{j_2m_1}\right)\alpha_j - \frac{1}{2}\left(\lambda C_{i_1j_1}C_{i_1m_1}C_{j_1m_1} + C_{ij}C_{i_2j_2}C_{i_2m_1}C_{j_2m_1}\right)\alpha_i\alpha_j}{\frac{1}{2}a_l} \\
&= \left((1-p_j)\lambda C_{i_1k_1} + p_jC_{ij}C_{i_2k_1}\right)\frac{\alpha_i}{a_l} + \left((1-p_i)\lambda C_{j_1k_1} + p_iC_{ij}C_{j_2k_1}\right)\frac{\alpha_j}{a_l} \\
&\quad - \left(\lambda C_{i_1j_1}C_{i_1k_1}C_{j_1k_1} + C_{ij}C_{i_2j_2}C_{i_2k_1}C_{j_2k_1}\right)\frac{\alpha_i\alpha_j}{a_l}
\end{aligned}
$$

$$\begin{aligned}C_{l_1m_2} &= \frac{P(l_1/m_2)}{P(l_2)}\\ &= \left((1-p_j)\lambda C_{i_1k_2} + p_jC_{ij}C_{i_2k_2}\right)\frac{\alpha_i}{a_l} + \left((1-p_i)\lambda C_{j_1k_2} + p_iC_{ij}C_{j_2k_2}\right)\frac{\alpha_j}{a_l}\\ &\quad - \left(\lambda C_{i_1j_1}C_{i_1k_2}C_{j_1k_2} + C_{ij}C_{i_2j_2}C_{i_2k_2}C_{j_2k_2}\right)\frac{\alpha_i\alpha_j}{a_l}\end{aligned}$$

$$\begin{aligned}P(l)P(l_2) &= P(01)P(01\to 00) + P(01)P(01\to 11) + P(10)P(10\to 00) + P(10)P(10\to 11)\\ &= P(01)\left(P(\overline{i_1}j_2) + P(i_1\overline{j_2})\right) + P(10)\left(P(i_2\overline{j_1}) + P(\overline{i_2}j_1)\right)\\ &= \frac{1}{2}\alpha_l\end{aligned}$$

$$\begin{aligned}P(l)P(l_2/m_1) &= \left(\frac{1}{2}p_j\lambda_iC_{i_1m_1} + \frac{1}{2}(1-p_j)\lambda_jC_{i_2m_1}\right)\alpha_i + \left(\frac{1}{2}p_i\lambda_jC_{j_1m_1} + \frac{1}{2}(1-p_i)\lambda_iC_{j_2m_1}\right)\alpha_j\\ &\quad - \frac{1}{2}\left(\lambda_iC_{i_1j_2}C_{i_1m_1}C_{j_2m_1} + \lambda_jC_{i_2j_1}C_{i_2m_1}C_{j_1m_1}\right)\alpha_i\alpha_j\end{aligned}$$

$$\begin{aligned}C_{l_2m_1} &= \frac{\left(\frac{1}{2}p_j\lambda_iC_{i_1m_1} + \frac{1}{2}(1-p_j)\lambda_jC_{i_2m_1}\right)\alpha_i + \left(\frac{1}{2}p_i\lambda_jC_{j_1m_1} + \frac{1}{2}(1-p_i)\lambda_iC_{j_2m_1}\right)\alpha_j - \frac{1}{2}\left(\lambda_iC_{i_1j_2}C_{i_1m_1}C_{j_2m_1} + \lambda_jC_{i_2j_1}C_{i_2m_1}C_{j_1m_1}\right)\alpha_i\alpha_j}{\frac{1}{2}\alpha_l}\\ &= \left(p_j\lambda_iC_{i_1k_1} + (1-p_j)\lambda_jC_{i_2k_1}\right)\frac{\alpha_i}{\alpha_l} + \left(p_i\lambda_jC_{j_1k_1} + (1-p_i)\lambda_iC_{j_2k_1}\right)\frac{\alpha_j}{\alpha_l}\\ &\quad - \left(\lambda_iC_{i_1j_2}C_{i_1k_1}C_{j_2k_1} + \lambda_jC_{i_2j_1}C_{i_2k_1}C_{j_1k_1}\right)\frac{\alpha_i\alpha_j}{\alpha_l}\end{aligned}$$

$$\begin{aligned}C_{l_2m_1} &= \left(p_j\lambda_iC_{i_1k_2} + (1-p_j)\lambda_jC_{i_2k_2}\right)\frac{\alpha_i}{\alpha_l} + \left(p_i\lambda_jC_{j_1k_2} + (1-p_i)\lambda_iC_{j_2k_2}\right)\frac{\alpha_j}{\alpha_l}\\ &\quad - \left(\lambda_iC_{i_1j_2}C_{i_1k_2}C_{j_2k_2} + \lambda_jC_{i_2j_1}C_{i_2k_2}C_{j_1k_2}\right)\frac{\alpha_i\alpha_j}{\alpha_l}\end{aligned}$$

Finally, Figure 4-9 is the Circuit diagram associated with the NOT rule, followed by the correlation factor derivation.

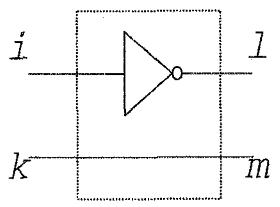


Figure 4-9. The circuit diagram associated with the NOT rule.

$$P(l_1) = P(i_2)$$
$$P(l_1 / m_1) = P(i_2 / m_1) = P(i_2)C_{i_2m_1}$$
$$C_{l_1m_1} = C_{i_2m_1} = C_{i_2k_1}$$

$$C_{l_1m_2} = C_{i_2m_2} = C_{i_2k_2}$$

$$P(l_2) = P(i_1)$$
$$P(l_2 / m_1) = P(i_1 / m_1) = P(i_1)C_{i_1m_1}$$
$$C_{l_2m_1} = C_{i_1k_1}$$

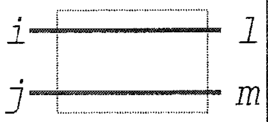
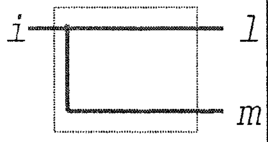
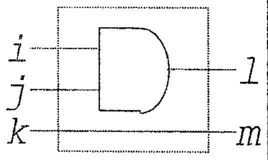
$$C_{l_2m_2} = C_{i_1k_2}$$

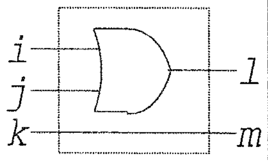
$$C_{l_1m_2} = C_{i_2k_2}$$

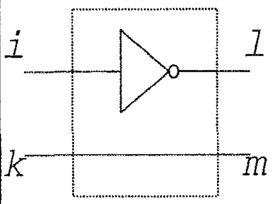
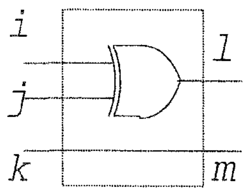
The results of these basic rules used to propagate correlation factors from the inputs to the output are listed in Table 4-4. These basic rules along with the transformations for determining the Markov chain parameters for the output of a logic function (Table 4-2) are the foundational components for the algorithm developed in the next section.

Table 4-4. Set of basic rules used to calculate the output correlation factors.

Rules	Probability Correlation Factors	Transition Correlation Factors

Independent rule 	$C_{lm} = C_{ij}$	$C_{l_1m_1} = C_{i_1j_1}$
Fan-out rule 	$C_{lm} = \frac{1}{P(i)}$	$C_{l_1m_1} = \frac{2(1-p_i)}{\alpha_i}$ $C_{l_2m_2} = \frac{2p_i}{\alpha_i}$ $C_{l_1m_2} = 0$ $C_{l_2m_1} = 0$
AND rule 	$C_{lm} = C_{ik}C_{jk}$	$C_{l_1m_1} = \lambda_i p_j C_{i_1k_1} \frac{\alpha_i}{\alpha_l}$ $+ \lambda_j p_i C_{j_1k_1} \frac{\alpha_j}{\alpha_l}$ $-\frac{1}{2}\begin{pmatrix} \lambda_i C_{i_1k_1} C_{j_2k_1} C_{i_1j_2} \\ + \lambda_j C_{i_2k_1} C_{j_1k_1} C_{i_2j_1} \\ - \lambda C_{i_1k_1} C_{j_1k_1} C_{i_1j_1} \end{pmatrix} \frac{\alpha_i\alpha_j}{\alpha_l}$ $C_{l_1m_2} = \lambda_i p_j C_{i_1k_2} \frac{\alpha_i}{\alpha_l}$ $+ \lambda_j p_i C_{j_1k_2} \frac{\alpha_j}{\alpha_l}$ $-\frac{1}{2}\begin{pmatrix} \lambda_i C_{i_1k_2} C_{j_2k_2} C_{i_1j_2} \\ + \lambda_j C_{i_2k_2} C_{j_1k_2} C_{i_2j_1} \\ - \lambda C_{i_1k_2} C_{j_1k_2} C_{i_1j_1} \end{pmatrix} \frac{\alpha_i\alpha_j}{\alpha_l}$ $C_{l_2m_1} = p_j C_{ij} C_{i_2k_1} \frac{\alpha_i}{\alpha_l}$ $+ p_i C_{ij} C_{j_2k_1} \frac{\alpha_j}{\alpha_l}$ $-\frac{1}{2} C_{ij} C_{i_2k_1} C_{j_2k_1} C_{i_2j_2} \frac{\alpha_i\alpha_j}{\alpha_l}$

		$C_{l_2m_2} = p_j C_{ij} C_{i_2k_2} \frac{\alpha_i}{\alpha_l} + p_i C_{ij} C_{j_2k_2} \frac{\alpha_j}{\alpha_l} - \frac{1}{2} C_{ij} C_{i_2k_2} C_{j_2k_2} C_{i_2j_2} \frac{\alpha_i \alpha_j}{\alpha_l}$
OR rule 	$\therefore C_{lm} = \frac{P(i)C_{ik} + P(j)C_{jk} - P(i)P(j)C_{ik}C_{jk}C_{ij}}{P(i)+P(j)-P(i)P(j)C_{ij}}$	$C_{l_1m_1} = \lambda(1-p_j)C_{i_1k_1}\frac{\alpha_i}{\alpha_l} + \lambda(1-p_i)C_{j_1k_1}\frac{\alpha_j}{\alpha_l} - \frac{1}{2}\lambda C_{i_1k_1} C_{j_1k_1} C_{i_1j_1}\frac{\alpha_i\alpha_j}{\alpha_l}$ $C_{l_1m_2} = \lambda(1-p_j)C_{i_1k_2}\frac{\alpha_i}{\alpha_l} + \lambda(1-p_i)C_{j_1k_2}\frac{\alpha_j}{\alpha_l} - \frac{1}{2}\lambda C_{i_1k_{21}} C_{j_1k_2} C_{i_1j_1}\frac{\alpha_i\alpha_j}{\alpha_l}$ $C_{l_2m_1} = (1-p_j)\lambda_j C_{i_2k_1}\frac{\alpha_i}{\alpha_l} + (1-p_i)\lambda_i C_{j_2k_1}\frac{\alpha_j}{\alpha_l} - \frac{1}{2}\begin{pmatrix} \lambda_i C_{i_1j_2} C_{i_1k_1} C_{j_2k_1} \\ + \lambda_j C_{i_2j_1} C_{i_2k_1} C_{j_1k_1} \\ - C_{ij} C_{i_2j_2} C_{i_2k_1} C_{j_2k_1} \end{pmatrix}\frac{\alpha_i\alpha_j}{\alpha_l}$ $C_{l_2m_2} = (1-p_j)\lambda_j C_{i_2k_2}\frac{\alpha_i}{\alpha_l} + (1-p_i)\lambda_i C_{j_2k_2}\frac{\alpha_j}{\alpha_l} - \frac{1}{2}\begin{pmatrix} \lambda_i C_{i_1j_2} C_{i_1k_2} C_{j_2k_2} \\ + \lambda_j C_{i_2j_1} C_{i_2k_2} C_{j_1k_2} \\ - C_{ij} C_{i_2j_2} C_{i_2k_2} C_{j_2k_2} \end{pmatrix}\frac{\alpha_i\alpha_j}{\alpha_l}$

NOT rule 	$C_{lm}=\frac{p(l/m)}{p(l)}=\frac{1-P(i)C_{ij}}{1-P(i)}$	$C_{l_1m_1}=C_{i_2k_1}$ $C_{l_2m_1}=C_{i_1k_1}$ $C_{l_1m_2}=C_{i_2k_2}$ $C_{l_2m_2}=C_{i_1k_2}$
XOR rule 	$C_{lm}=$ $\frac{P(i)C_{ik}+P(j)C_{jk}-2P(i)C_{ik}P(j)C_{jk}C_{ij}}{P(i)+P(j)-2P(i)P(j)C_{ij}}$	$C_{l_1m_1}=\begin{pmatrix}(1-p_j)\lambda C_{i_1k_1}\\+p_jC_{ij}C_{i_2k_1}\end{pmatrix}\frac{\alpha_i}{a_l}$ $+\begin{pmatrix}(1-p_i)\lambda C_{j_1k_1}\\+p_iC_{ij}C_{j_2k_1}\end{pmatrix}\frac{\alpha_j}{a_l}$ $-\begin{pmatrix}\lambda C_{i_1j_1}C_{i_1k_1}C_{j_1k_1}\\+C_{ij}C_{i_2j_2}C_{i_2k_1}C_{j_2k_1}\end{pmatrix}\frac{\alpha_i\alpha_j}{a_l}$ $C_{l_1m_2}=\begin{pmatrix}(1-p_j)\lambda C_{i_1k_2}\\+p_jC_{ij}C_{i_2k_2}\end{pmatrix}\frac{\alpha_i}{a_l}$ $+\begin{pmatrix}(1-p_i)\lambda C_{j_1k_2}\\+p_iC_{ij}C_{j_2k_2}\end{pmatrix}\frac{\alpha_j}{a_l}$ $-\begin{pmatrix}\lambda C_{i_1j_1}C_{i_1k_2}C_{j_1k_2}\\+C_{ij}C_{i_2j_2}C_{i_2k_2}C_{j_2k_2}\end{pmatrix}\frac{\alpha_i\alpha_j}{a_l}$ $C_{l_2m_1}=\begin{pmatrix}p_j\lambda_iC_{i_1k_1}\\+(1-p_j)\lambda_jC_{i_2k_1}\end{pmatrix}\frac{\alpha_i}{\alpha_l}$ $+\begin{pmatrix}p_i\lambda_jC_{j_1k_1}\\+(1-p_i)\lambda_iC_{j_2k_1}\end{pmatrix}\frac{\alpha_j}{\alpha_l}$ $-\begin{pmatrix}\lambda_iC_{i_1j_2}C_{i_1k_1}C_{j_2k_1}\\+\lambda_jC_{i_2j_1}C_{i_2k_1}C_{j_1k_1}\end{pmatrix}\frac{\alpha_i\alpha_j}{\alpha_l}$ $C_{l_2m_1}=\begin{pmatrix}p_j\lambda_iC_{i_1k_2}\\+(1-p_j)\lambda_jC_{i_2k_2}\end{pmatrix}\frac{\alpha_i}{\alpha_l}$ $+\begin{pmatrix}p_i\lambda_jC_{j_1k_2}\\+(1-p_i)\lambda_iC_{j_2k_2}\end{pmatrix}\frac{\alpha_j}{\alpha_l}$ $-\begin{pmatrix}\lambda_iC_{i_1j_2}C_{i_1k_2}C_{j_2k_2}\\+\lambda_jC_{i_2j_1}C_{i_2k_2}C_{j_1k_2}\end{pmatrix}\frac{\alpha_i\alpha_j}{\alpha_l}$

4.5 Markov Chain Propagation Algorithm

This section describes a proposed Markov Chain Propagation (MCP) algorithm [20] for determining the Markov chain models for all signals of a given combinational circuit. The Markov chain signal model proposed in the previous section is employed, and it is assumed that the parameters of the model are known for the circuit's primary inputs. The overall approach is to propagate signal information associated with the Markov chain model through the circuit in a "gate-by-gate" fashion. Recall that once the Markov chain model is determined for all signals, the signal activities and circuit power estimate are determined. It is assumed that the given circuit is specified at the level of basic logic gates.

MCP Algorithm: Compute signal probability and activity of every signal in a combinational logic circuit.

Input: Signal probabilities, activities and correlation cofactors of all primary inputs to the circuit.

Output: Signal probabilities, activities and correlation cofactors of all nodes in the circuit.

1. Represent the given combinational circuit as a directed acyclic graph (DAG);

   Vertices of the DAG correspond to basic gates and edges represent signals. Two extra vertices (a source and a sink) are included in the DAG to accommodate the *primary inputs and outputs of the circuit. An example of how to represent a circuit with the DAG model is illustrated by Figures 4-10(a) and 4-10(b).*

2. Perform a topological sort [10] on the DAG to obtain an ordering of the gates;

   See Figure 4-10(c).

3. Transformation to two-input basic logic gates;

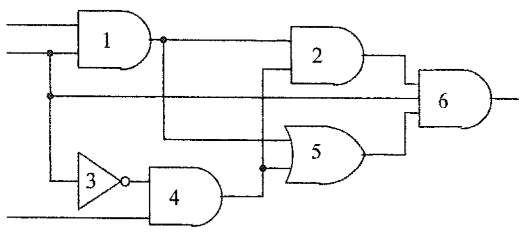
   As shown in Figure 4-10(d), replace all basic gates having more than two inputs with an equivalent sequence of two-input basic gates.

4. Partition the circuit into levels;

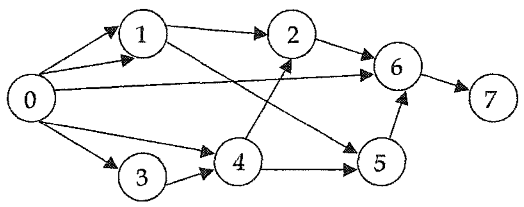
   As shown in Figure 4-10(e), levels are defined at the input and output of each basic gate. Note that there is at most one gate between any two consecutive levels.

5. Successively apply propagation rules at each level.

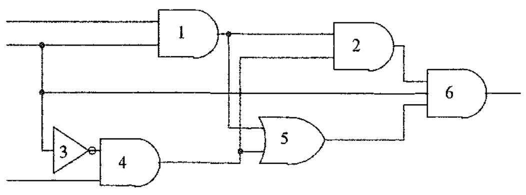
   Apply the propagation rules from Tables 4-2 and 4-4 for calculating the parameters of the Markov model for the basic gate outputs and the associated correlation factors.



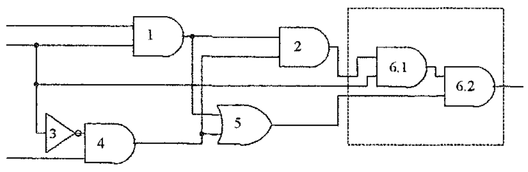
(a)



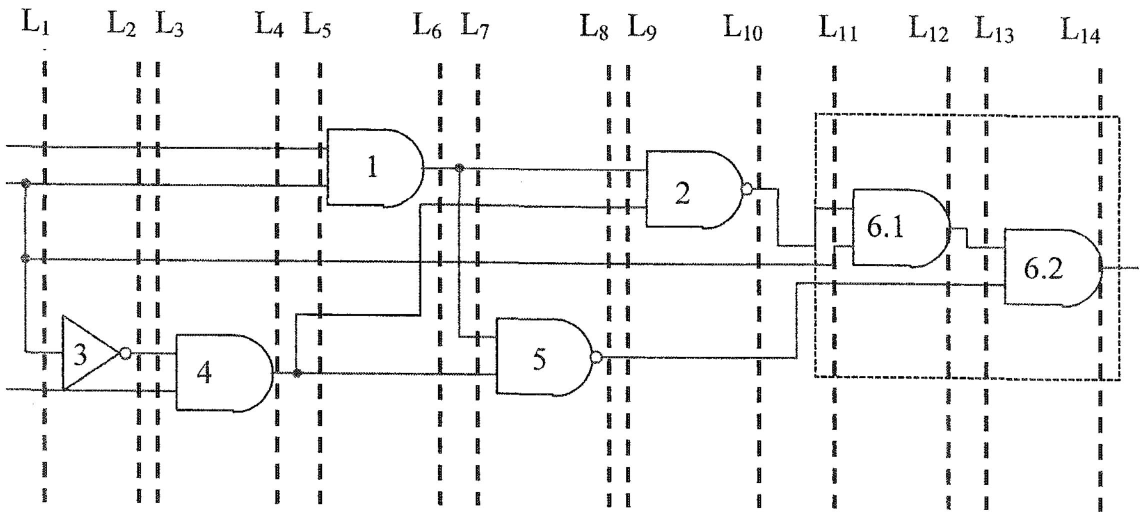
(b)



(c)



(d)



(e)

Figure 4-10. Illustration of the basic steps of the MCP Algorithm.

In deriving the time complexity of the MCP algorithm, let N denote the number of basic gates, M be the number of signals, and S the number of fan-out signals. Fan-out is associated with a signal that is broadcast (i.e., duplicated). To illustrate, for the circuit of Figure 4-10(a), N=6, M=16, S=7. Because two levels are associated with each gate (one is placed before the gate and the other after), there are less than $2M$ levels for a circuit with M total number of signals, which is 14 levels for the example shown in Figure 4-10(e).

Constructing the DAG (Step 1) from the given circuit requires $O(N+M)$ operations and it is shown in [10] that topological sort (Step 2) also requires $O(N+M)$ operations. Step 3 can be finished with no more than M operations and at most $2M$ operations are needed for Step 4.

For Step 5, there are two cases: from level L_i to level L_{i+1} and from level L_{i+1} to level L_{i+2},where i = 1, 3, ..., $2M$-1. For the first case, because there is only one gate (e.g., gate 1 when i = 5 as shown in Figure 4-10(e)) between level L_i and level L_{i+1}, the calculation needed is to propagate the inputs of the single gate to the output of that gate. As shown in Figure 4-10(e), when i = 5, the three parameters of the output signal of gate 1 can be obtained in a constant number of operations, denoted by C_1. The correlation factors between this output signal and other signals need to be calculated and inserted to the correlation factor table during this step. Because of the following three facts, it follows that the number of operations needed for this case of Step 5 can be expressed as $C_1 + 2MC_2$:

(i) only those signals having correlations with the input signals of the gate will have correlations with the output signal of the gate need to be calculated;

(ii) the maximum length of the correlation table of every entry is no more than M; and

(iii) the correlation factors between two signals can be done in a constant number of operations (assumed to be C_2) using basic rules shown in Table 4-4.

For the other case there isn't a gate between level L_{i+1} to level L_{i+2} (e.g., as shown in Figure 4-10(e), when $i = 5$, this corresponds to L_6 to L_7). The only calculation needed in this case is to calculate the correlation factors due to recovergent fan-outs. Assume there are k_i fan-outs from level L_{i+1} to level L_{i+2}. The needed number of operations is bounded by k_iC_2.

So the total number of operations in Step 5 is therefore

$$\sum_{j=1}^{2M}(k_{2j-1}C_2 + C_1 + 2MC_2) = 2MC_2 + 2MC_1 + 4M^2C_2 = O(M^2)$$

Combining the derived complexity results of Step 1 to Step 5, the time complexity of this MCP Algorithm is $O(M^2)$.

4.6 Summary

Signals can be modeled by a Marcov-chain having two event parameters. It is shown that the proposed Markov chain model is equivalent to the two-parameter probability/activity signal model of [3] and [4]. The advantage of modeling signals with Markov chains is that it makes it possible to compute correlations between signals related to both probability and activity. Based on this Markov-chain signal modeling, a more efficient and more accurate algorithm (named MCP algorithm) is developed. By propagating signal parameters and correlation cofactors from the primary inputs through the circuit in a "gate-by-gate" fashion, this MCP algorithm can achieve a very good accuracy and an $O(M^2)$ time complexity where M is the number of signals in the circuit. Simulation studies related to the accuracy of the MCP algorithm are provided in Chapter 6.

CHAPTER 5

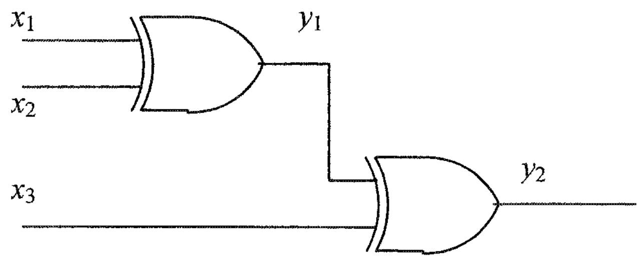
GLITCHING POWER CONSUMPTION ESTIMATION

5.1 INTRODUCTION

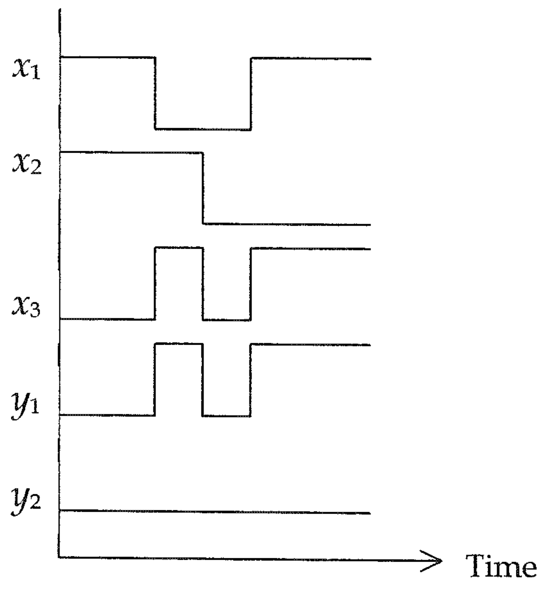
In Chapter 3, signals are modeled as strict-sense stationary (SSS) 0-1 process and mean-ergodic. Based on this signal model, signal activities can be calculated by two algorithms, which are proposed in [3] and [4] by using relative Boolean difference and generalized Boolean difference, respectively. In Chapter 4, we proposed a signal model using a Markov-chain process and the probabilities and activities of all signals in a circuit are calculated by the proposed MCP Algorithm. Both of these signal models and their associated algorithms assume zero propagation delay through each gate. In reality, gates have non-zero delays, which results in "signal glitching."

To illustrate how non-zero delays cause glitches, consider an example circuit as shown in Figure 5-1(a). Under the assumption of zero delay, the sample input signals x_1, x_2 and x_3 result in the output signals y_1 and y_2 shown in Figure 5-1(b). Notice that output signal y_2 experiences no transitions. For non-zero delays (assume the delay of each gate is d) the output signal y_2 for the same inputs is derived and shown in Figure 5-1(c), which has several "glitching" transitions.

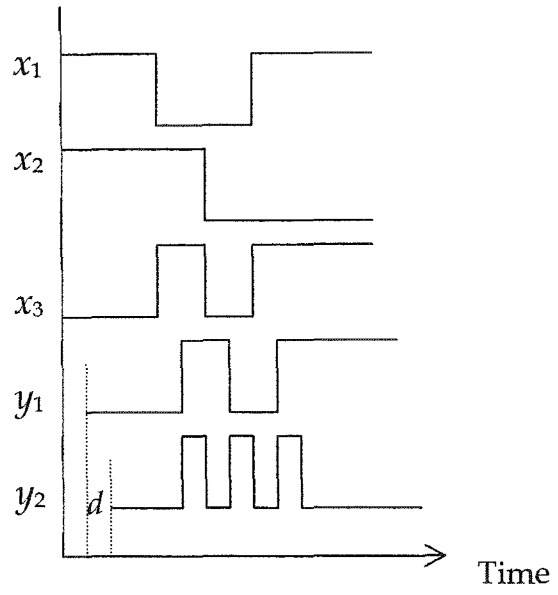
Power consumption is impacted by these signal glitches; thus, it is necessary to consider the effect of glitches due to non-zero propagation delays to achieve a better power estimation of a circuit for real applications.



(a)



(b)



(c)

Figure 5-1. An example used to show how non-zero delays cause glitches.

The power consumption due to glitching can be significant in some extreme cases, such as the example shown above. Hence, techniques used to estimate total power consumption in circuits need to take into account glitching power consumption. Currently, only statistical-based power estimation approaches are used to estimate glitching activities. The general ideal of statistical-based approaches is as follows: By using a logic or timing simulator, the estimator can efficiently estimate power dissipation due to both functional and spurious transitions. The technique in [19] gives an upper bound of glitches that can possibly occur, and a Monte-Carlo-based technique that can efficiently estimate glitches under different non-zero delay model is given in [11, 12]. In this chapter, we will propose a new probabilistic-based technique to calculate signal activities including glitches caused by gate delays.

5.2 Probabilistic Glitching Model

Under the zero-delay model assumption, the probability and activity of the output signal of a generic two-input logic gate can be calculated and the results are listed in Table 4-3. Assume that $\alpha_{zero-delay}(Y)$ denotes the activity of signal Y under zero-delay model, and $\alpha_{glitching}(Y)$ denotes the transition activity of signal Y due to glitching only under non zero-delay model. Because $\alpha_{zero-delay}(Y)$ has no contributions to $\alpha_{glitching}(Y)$, and also assume that $\alpha_{glitching}(Y)$ has no contributions to $\alpha_{zero-delay}(Y)$, the total activity of the output signal can be epressed as the sum of these two components, which is represented as

$$\alpha_{total}(Y) = \alpha_{zero-delay}(Y) + \alpha_{glitching}(Y). \tag{5.1}$$

Assume there are two input signals A and B, with signal A is ahead of signal B by d time units and the output signal is Y with a Boolean function of $Y = f(A,B)$. Because the activity under zero-delay model, denoted as $\alpha_{zero-delay}(Y)$ as shown in Equation 5.1, can be calculated by using MCP algorithm, we will focus on the activity derivation of the glitching part, denoted as $\alpha_{glitching}(Y)$ in Equation 5.1. Because three basic logical gates, i.e., AND gate, OR gate and XOR gate, are mainly used in CMOS circuits, we will focus on the derivation of $\alpha_{glitching}(Y)$ expression for these three basic logic gates.

The first is the AND gate, $Y=AB$. As shown in Figure 5-2, the glitching of output signal Y may happen only when signal A has a transition from state 0 to state 1 and signal B has a transition from state 1 to state 0 with d unit time lag. Table 5-1 shows those conditions for having a glitching at output signal Y.

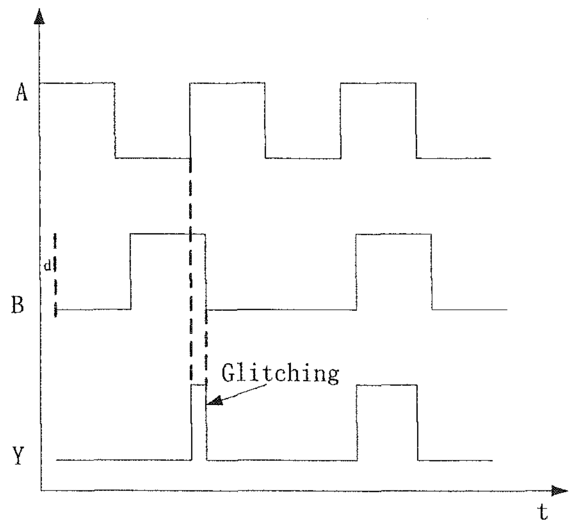


Figure 5-2. The d unit time delay causes glitching in an AND gate ($Y=AB$).

Table 5-1. The cases that cause glitching in an AND gate ($Y=AB$).

A	B	Glitching at Y
1→0	1→0	No
	0→1	No
0→1	1→0	Yes
	0→1	No

From Table 5-1, we can see that there is only one case that can cause glitching: signal A switches from state 0 to state 1 and signal B switches from state 1 to state

0, which can be represented by event 01→10. Hence the probability of the event 01→10, causing a glitch in signal Y, can be expressed as (using the results listed in Table 4-1)

$$P(01 \to 10) = \frac{\alpha_A}{2(1-p_A)} \frac{\alpha_B}{2p_B} C_{A_1B_2} . \tag{5.2}$$

Using the fact that each glitch has two transitions, i.e., from state 0 to state 1 followed by from state 1 to state 0, and vice versa, therefore, the probability of glitching can be calculated by

$$\begin{aligned}\alpha_{glitching}(Y) &= 2P(01)P(01 \to 10) = 2(p_B - p_A p_B C_{AB}) \frac{\alpha_A}{2(1-p_A)} \frac{\alpha_B}{2p_B} C_{A_1B_2} \\ &= \frac{1-p_A C_{AB}}{2(1-p_A)} \alpha_A \alpha_B C_{A_1B_2}\end{aligned} \tag{5.3}$$

The activity for output signal Y (without glitching) for an AND gate, as shown in Table 4-3, is expressed as

$$\alpha_{zero-delay}(Y) = p_B \alpha_A C_{AB} + p_A \alpha_B C_{AB} - \frac{1}{2} \alpha_A \alpha_B C_{AB} C_{A_2B_2} . \tag{5.4}$$

So combining these two Equations 5.3 and 5.4 together, and applying Equation 5.1, the total activity (considering glitching) of output signal Y can be derived and expressed as

$$\begin{aligned}\alpha_{total}(Y) &= p_B \alpha_A C_{AB} + p_A \alpha_B C_{AB} - \frac{1}{2} \alpha_A \alpha_B C_{AB} C_{A_2B_2} + \frac{1-p_A C_{AB}}{2(1-p_A)} \alpha_A \alpha_B C_{A_1B_2} \\ &= p_B \alpha_A C_{AB} + p_A \alpha_B C_{AB} - \frac{1}{2} \alpha_A \alpha_B \left\{ C_{AB} C_{A_2B_2} - \frac{1-p_A C_{AB}}{1-p_A} C_{A_1B_2} \right\} .\end{aligned} \tag{5.5}$$

For the case of Y=A+B that is an OR gate, as shown in Figure 5-3, the glitching of output signal Y may happen only when signal A has a transition from state 1 to state 0 and signal *B* has a transition from state 0 to state 1 with a *d* unit time lag. Table 5-2 shows the condition for having a glitch at output signal *Y* for an OR gate.

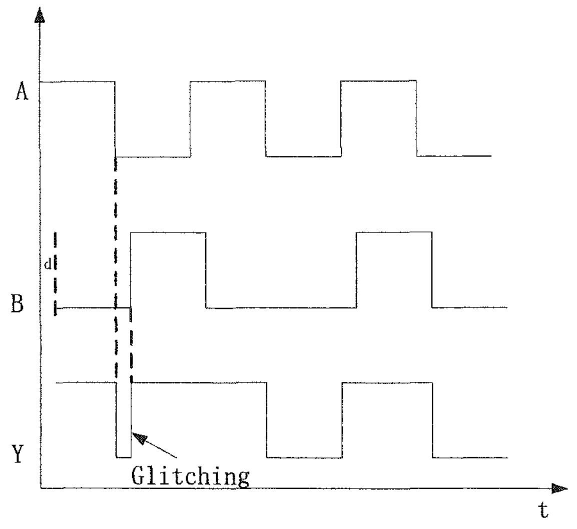


Figure 5-3. The *d* unit time delay causes glitching in an OR gate ($Y=A+B$).

Table 5-2. The cases that cause glitching for OR gate ($Y=A+B$).

A	B	Glitching at *Y*
1→0	1→0	No
	0→1	Yes
0→1	1→0	No
	0→1	No

So as shown in Table 5-2, the only case for causing glitching is at event 10→01. By using the results in Table 4-1, the probability of event 10→01 can be expressed as

$$P(10 \rightarrow 01) = \frac{\alpha_A}{2p_A} \frac{\alpha_B}{2(1-p_B)} C_{A_2B_1},$$

hence the probability of causing glitching can be expressed as

$$\begin{aligned}\alpha_{glitching}(Y) &= 2P(10)P(10 \to 01) \\ &= 2(p_A - p_A p_B C_{AB})\frac{\alpha_A}{2p_A}\frac{\alpha_B}{2(1-p_B)}C_{A_2B_1}.\end{aligned} \tag{5.6}$$

From Table 4-3, the activity of output signal Y for the zero-delay model is

$$\alpha_{zero-delay}(Y) = (1-p_B)\lambda\alpha_A + (1-p_A)\lambda\alpha_B - \frac{1}{2}\lambda\alpha_A\alpha_B C_{A_1B_1} \tag{5.7}$$

Therefore, combining Equations 5.6 and 5.7 together, the total activity of the output signal for an OR gate can be expressed as,

$$\begin{aligned}\alpha_{total}(Y) &= \alpha_{glitching} + \alpha_{zero-delay} \\ &= 2(p_A - p_A p_B C_{AB})\frac{\alpha_A}{2p_A}\frac{\alpha_B}{2(1-p_B)}C_{A_2B_1} \\ &\quad + (1-p_B)\lambda\alpha_A + (1-p_A)\lambda\alpha_B - \frac{1}{2}\lambda\alpha_A\alpha_B C_{A_1B_1} \\ &= \frac{1}{2}\lambda_B\alpha_A\alpha_B C_{A_2B_1} + (1-p_B)\lambda\alpha_A + (1-p_A)\lambda\alpha_B - \frac{1}{2}\lambda\alpha_A\alpha_B C_{A_1B_1} \\ &= (1-p_B)\lambda\alpha_A + (1-p_A)\lambda\alpha_B + \frac{1}{2}(\lambda_B C_{A_2B_1} - \lambda C_{A_1B_1})\alpha_A\alpha_B.\end{aligned} \tag{5.8}$$

When $Y = A \oplus B$ that is XOR gate, it will cause a glitch whenever there is a transition as shown in Figure 5-4 and Table 5-3. It should be noted that glitches are caused due to both transitions of signals A and B, not only signal A or B. So using the similar techniques as derived for gates AND and OR, the glitching activity of output signal Y can be derived and expressed as:

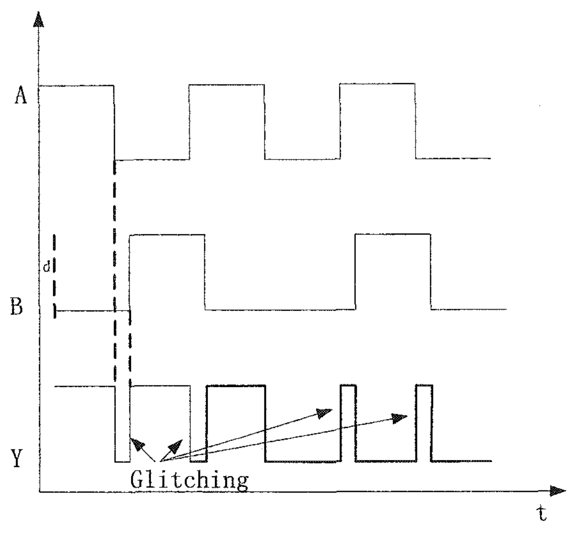


Figure 5-4. The d unit time delay causes glitching in a XOR gate ($Y = A \oplus B$).

Table 5-3. The cases that cause glitching for XOR gate ($Y = A \oplus B$).

A	B	Glitching at Y
1→0	1→0	Yes
	0→1	Yes
0→1	1→0	Yes
	0→1	Yes

$$\begin{aligned}
\alpha_{glitching} &= 2\begin{Bmatrix} P(11)P(11\to 00)+P(10)P(10\to 01) \\ +P(01)P(01\to 10)+P(00)P(00\to 11) \end{Bmatrix} \\
&= 2p_Ap_BC_{AB}\frac{\alpha_A}{2p_A}\frac{\alpha_B}{2p_B}C_{A_2B_2}+2(p_A-p_Ap_BC_{AB})\frac{\alpha_A}{2p_A}\frac{\alpha_B}{2(1-p_B)}C_{A_2B_1} \\
&\quad +2(p_B-p_Ap_BC_{AB})\frac{\alpha_A}{2(1-p_A)}\frac{\alpha_B}{2p_B}C_{A_1B_2} \\
&\quad +2(1-p_A-p_B+p_Ap_BC_{AB})\frac{\alpha_A}{2(1-p_A)}\frac{\alpha_B}{2(1-p_B)}C_{A_1B_1} \\
&= \frac{1}{2}\alpha_A\alpha_BC_{AB}C_{A_2B_2}+\frac{1}{2}\lambda_B\alpha_A\alpha_BC_{A_2B_1}+\frac{1}{2}\lambda_A\alpha_A\alpha_BC_{A_1B_2}+\frac{1}{2}\lambda\alpha_A\alpha_BC_{A_1B_1}
\end{aligned} \tag{5.9}$$

Without considering glitching effects in a combinational circuit, using the results listed in Table 4.3, the activity of output signal Y is

$$a_{zero-delay}(Y) = \left(\lambda_B + p_B(\lambda_A - \lambda_B)\right)\alpha_A + \left(\lambda_A + p_A(\lambda_B - \lambda_A)\right)\alpha_B \\ - \left(\lambda_A C_{A_1B_2} + \lambda_B C_{A_2B_1}\right)\alpha_A\alpha_B \tag{5.10}$$

Combining Equations 5.9 and 5.10 together and applying Equation 5.1, the total activity of output signal Y can be calculated as

$$\begin{aligned} &\alpha_{total}(Y) = \alpha_{glitching}(Y) + \alpha_{zero-delay}(Y) \\ &= \frac{1}{2}\alpha_A\alpha_B C_{AB} C_{A_2B_2} + \frac{1}{2}\lambda_B\alpha_A\alpha_B C_{A_2B_1} + \frac{1}{2}\lambda_A\alpha_A\alpha_B C_{A_1B_2} + \frac{1}{2}\lambda\alpha_A\alpha_B C_{A_1B_1} \\ &+ \left(\lambda_B + p_B(\lambda_A - \lambda_B)\right)\alpha_A + \left(\lambda_A + p_A(\lambda_B - \lambda_A)\right)\alpha_B - \left(\lambda_A C_{A_1B_2} + \lambda_B C_{A_2B_1}\right)\alpha_A\alpha_B \\ &= \left(\lambda_B + p_B(\lambda_A - \lambda_B)\right)\alpha_A + \left(\lambda_A + p_A(\lambda_B - \lambda_A)\right)\alpha_B \\ &- \frac{1}{2}\alpha_A\alpha_B(\lambda_A C_{A_1B_2} + \lambda_B C_{A_2B_1} - C_{AB}C_{A_2B_2} - \lambda C_{A_1B_1}). \end{aligned} \tag{5.11}$$

To show how glitching affects total activity of the output signal, consider an example $Y = A \oplus B$. Let the probability and activity values of inputs be $p_A = 0.5$, $p_B = 0.5$, $\alpha_A = 0.2$ and $\alpha_B = 0.2$ respectively. And assume the input signals are mutually independent, then the activity of output signal Y due to glitching is

$$\alpha_{glitching}(Y) = 2\alpha_A\alpha_B = 2 \times 0.2 \times 0.2 = 0.08.$$

And the activity without glitching can be calculated as

$$\alpha_{zero-delay}(Y) = \alpha_A + \alpha_B - 2\alpha_A\alpha_B = 0.32.$$

The total activity of signal Y is

$$\alpha_{total}(Y) = 0.08 + 0.32 = 0.4.$$

We can see that the glitching activity is around 20% of the total activity. An interesting fact is that the total output signal activity expression for two input signals (considering glitching) is the same as that shown in Chapter 3 using

Equation 3.4 without considering simultaneous switching if the input signals are assumed as mutually independent and have non-zero delay[6]. It is true because the simultaneous switching will not cause any glitching (no delay, no glitching). But Equation 3.4 is only suited for inputs with delays and assumed to be mutually independent. In general, this is not true because signals may be correlated and delays may or may not exist in signals.

If the primary inputs of a circuit are assumed to be mutually independent, by using the propagation rules listed in Table 4-4, the activities of the three basic gates can be listed in Table 5-4 as below:

Table 5-4. The activity calculation results for three basic gates, AND, OR and XOR, by given the probabilities, activities, correlation factors and delay times in the assumption that signal *B* has *d* unit time delay than signal *A*.

$Y = f(A,B)$	$\alpha_{total}(Y)$
$Y = AB$	$p_B\alpha_A C_{AB} + p_A\alpha_B C_{AB}$ $-\frac{1}{2}\alpha_A\alpha_B\left\{C_{AB}C_{A_2B_2} - \frac{1-p_AC_{AB}}{1-p_A}C_{A_1B_2}\right\}$
$Y = A + B$	$(1-p_B)\lambda\alpha_A + (1-p_A)\lambda\alpha_B$ $+\frac{1}{2}(\lambda_B C_{A_2B_1} - \lambda C_{A_1B_1})\alpha_A\alpha_B$
$Y = A \oplus B$	$(\lambda_B + p_B(\lambda_A - \lambda_B))\alpha_A$ $+(\lambda_A + p_A(\lambda_B - \lambda_A))\alpha_B$ $-\frac{1}{2}\alpha_A\alpha_B(\lambda_A C_{A_1B_2} + \lambda_B C_{A_2B_1} - C_{AB}C_{A_2B_2} - \lambda C_{A_1B_1})$

[6] For multi-input gates, if each input signal has delay to each other signal, then Eq. 3.4 still gives us a correct answer, otherwise it does not.

5.3 Markov Chain Propagation with Glitching Algorithm

In real applications, delay exists for every gate in a CMOS circuit. This delay can cause glitching and the activities of some signals in the circuit will increase. In this section, we will modify the MCP algorithm proposed in Chapter 4 to calculate probabilities and activities including glitching effects for all signals in the circuit, which is named as MCPG algorithm. It is assumed that all parameters of primary inputs and delays associated with each gate are known. The overall approach is similar to MCP algorithm, by propagating signal information associated with the Markov chain model through the circuit in a "gate-by-gate" fashion considering glitching activity. Recall that once the Markov chain model is determined for all signals, the signal activities and circuit power estimation are determined, and vice versa. It is also assumed that the given circuit is specified at the level of basic logic gates running in an ideal environment with no frequency limit.

MCPG Algorithm: Compute signal probability and activity of each signal in a circuit considering the glitching activity.

Input: Signal probabilities, activities, correlation cofactors, and time relations (time delay, general, it is assumed to be zero for primary inputs) of all primary inputs to the circuit. Also delay times for every gate in the circuit (to simplify, all assumed to be the same such as d unit times).

<u>Output</u>: Signal probabilities, activities and correlation cofactors of all nodes in the circuit.

1. Represent the given combinational circuit as a directed acyclic graph (DAG);

   Vertices of the DAG correspond to basic gates and edges represent signals. Two extra vertices (a source and a sink) are included in the DAG to accommodate the primary inputs and outputs of the circuit. An example of how to represent a circuit with the DAG model is illustrated by Figures 5-5(a) and 5-5(b).

2. Perform a topological sort [10] on the DAG to obtain an ordering of the gates;

   See Figure 5-5(c).

3. Partition the circuit into levels and assign a delay value to each signal;

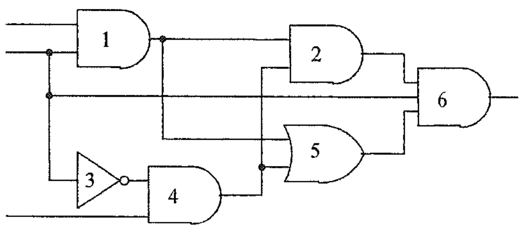
   As shown in Figure 4-10(e), levels are defined at the input and output of each basic gate. Note that there is at most one gate between any two consecutive levels. Based on the given inputs delay values, assign a delay value to each signal, the output signal delay value equals to the maximum delay value of the inputs associated to this gate plus the delay value corresponding to this gate.

4. Transformation to two-input basic logic gates;

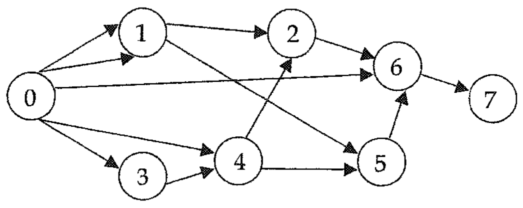
   As shown in Figure 5-5(d), replace all basic gates having more than two inputs with an equivalent sequence of two-input basic gates. In this step, the created intermediate two-input basic gate(s) will not have delay values, and the created *signals will assign delay values as the maximum delay of the associated two inputs only.*

5. Successively apply propagation rules at each level.

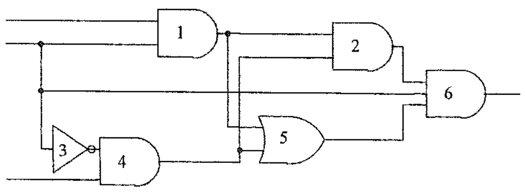
   Apply the propagation rules from Tables 4-2, Table 5-4 and Table 4-4 for calculating the parameters of the Markov model for the basic gate outputs and the associated correlation factors based on the given delay times corresponding to the two-input of the gate.



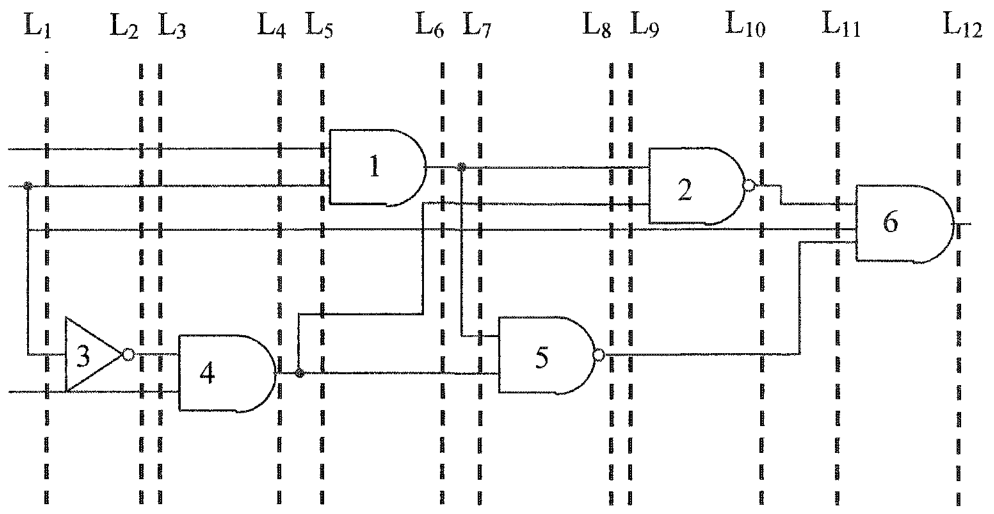
(a)



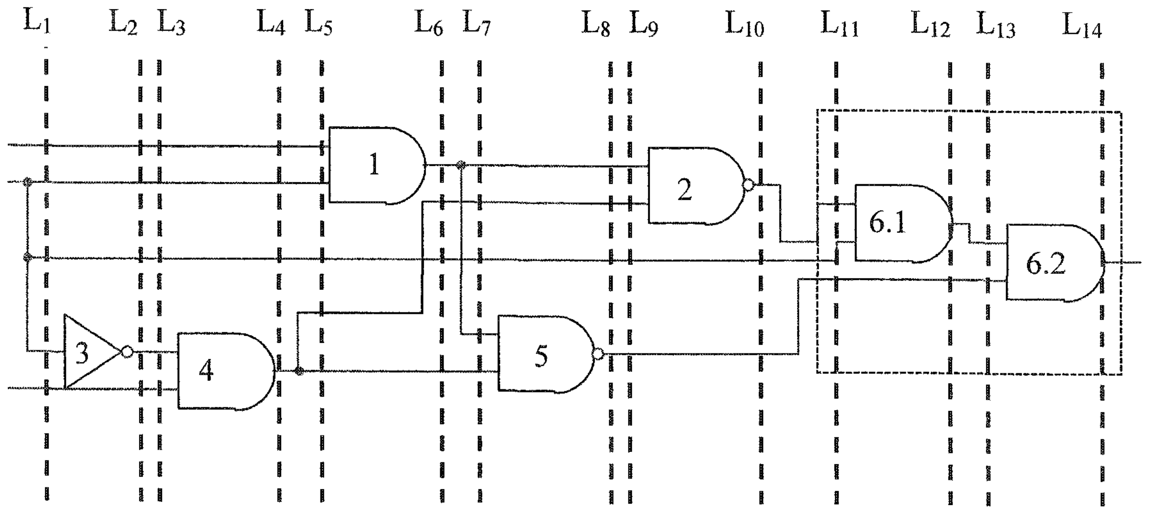
(b)



(c)



(d)



(e)

Figure 5-5. Illustration diagrams for basic steps of the MCPG Algorithm.

5.4 SUMMARY

In real applications, delay exists in logic gates that causes glitches. The power consumption due to glitching is significant in some extreme cases. The total activity for an output signal in a circuit is represented into two parts: the activity under zero-delay model and the activity caused by glitches. To calculate the activity part due to glitching, analytical expressions for three basic logic gates are developed. Based on these analytical expressions, a MCPG algorithm is also developed by expanding the MCP algorithm, which has the same time complexity.

CHAPTER 6

EXPERIMENTAL SETUP AND RESULTS

6.1 INTRODUCTION

After investigating the accuracy and efficiency of different approaches in previous chapters theoretically, experimental analysis should be carried out to support our theoretical analysis. Therefore, in this chapter, we will introduce some experiments implemented using the PSpice® simulator and our MCP simulator. The results obtained from simulations are compared to those from different approaches. In addition, we present here a new idea of using MCP algorithm to estimate power consumption considering glitching effects (named MMCP). It is shown that this new MMCP algorithm can gives us a more practical and more accurate solution for estimation power consumption in CMOS circuits.

6.2 EXPERIMENTAL SETUP

The experimental setup diagram is shown in Figure 6-1. It contains five components: Signal Generator, Circuit Description File, MCP Algorithm Simulator, PSpice Simulator, and Filter. The functionality and details of these five components are given below.

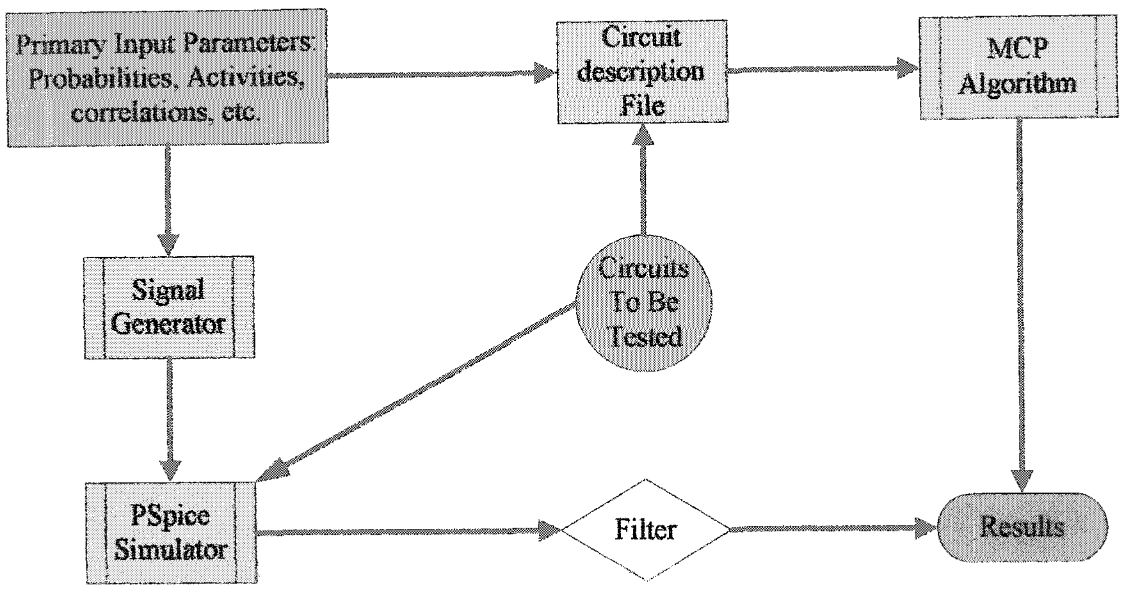


Figure 6-1. Components of experimental setup to test the accuracy and efficiency of different approaches.

6.2.1 Signal Generator

Signals that are primary inputs to PSpice simulator are generated based on given probabilities and activities. In Chapter 4, we model a signal A by a Markov chain with two transition states, i.e., state 0 and state 1, with two transition events, i.e., transition from state 0 to state 1 denoted as A_1 and transition from state 1 to state 0 denoted as A_2 as shown in Figure 6-2. Therefore, given probability and activity values of signal A, i.e., p_A and α_A, respectively, the probabilities of these two events can be calculated using Equations 6.1 and 6.2, which are the same as those equations shown in Chapter 4 (Equations 4.7 and 4.8).

$$P(A_1) = \frac{\alpha_A}{2(1-p_A)} \tag{6.1}$$

$$P(A_2) = \frac{\alpha_A}{2p_A} \tag{6.2}$$

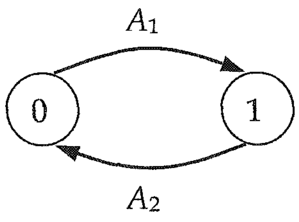


Figure 6-2. Signal A with two transition states 0 and 1, and two transition events A_1 and A_2.

For example, if the probability and activity of one primary input signal A are given to be $p_A = 0.6$ and $\alpha_A = 0.2$, then the probability value of event A_1 can be calculated using Equation 6.1 and results in $p_{A_1} = 0.25$. Similarly, the probability value for event A_2 is $p_{A_2} = 0.167$. Hence the probabilities for the signal to stay at state 0 and stay at state 1 are 0.75 and 0.833, respectively. Based on this result, the signal A can be generated as follows: each value is generated at the leading edge of the given clock (the frequency of the clock is assumed to be given); by using a random number generator that generates a value between "0" to "1", the first signal value is set to be "0" or "1" depends on the value generated by the random number generator. For the above example, if the output value of the random generator is less than 0.6, then the first value is set to be "0", otherwise it is set to be "1"; the next value is depend on the current signal value and the new data that the random number generator generated. For the above example, if the new output of the random number generator is larger than or equal to 0.25, and the current signal value is "0", then the signal value at this clock edge is set to be "1" (means that the signal transitions from state 0 to state 1). So by following the procedure illustrated above, mutually independent signals with given probabilities and activities can be generated. The pseudo-code for our signal generator is shown below:

```
input: PROB, ACTIVITY, SIZE;
      /SIZE=number of clock cycles needed
compute: S[1:SIZE+1]
   calculate PA1, PA2; /using Eqs.6.1 and 6.2
   r=rand();
   if (r <= PROB ) then S[1]=1 else S[1]=0 endif
   for I=1 to SIZE+1
       r=rand();
       if (S[I] == 0)
         if (r<=PA1) then S[I+1]=0 else S[I+1]=1 endif
       elseif (S[I] == 1)
         if (r<=PA2) then S[I+1]=1 else S[I+1]=0 endif
       endif
       I=I+1;
   endfor
```

6.2.2 *CIRCUIT DESCRIPTION FORMAT*

Circuits to be tested in our experiments are first transferred into predefined circuit description files as inputs to MCP Algorithm Simulator. This circuit description file is using ISCAS85 [21] netlist format. Below is an example of netlist format (ISCAS85 format) of the circuit c17[7] with its diagram shown in Figure 6-3.

[7] a six-NAND-gate circuit used to show netlist format for ISCAS85 benchmark.

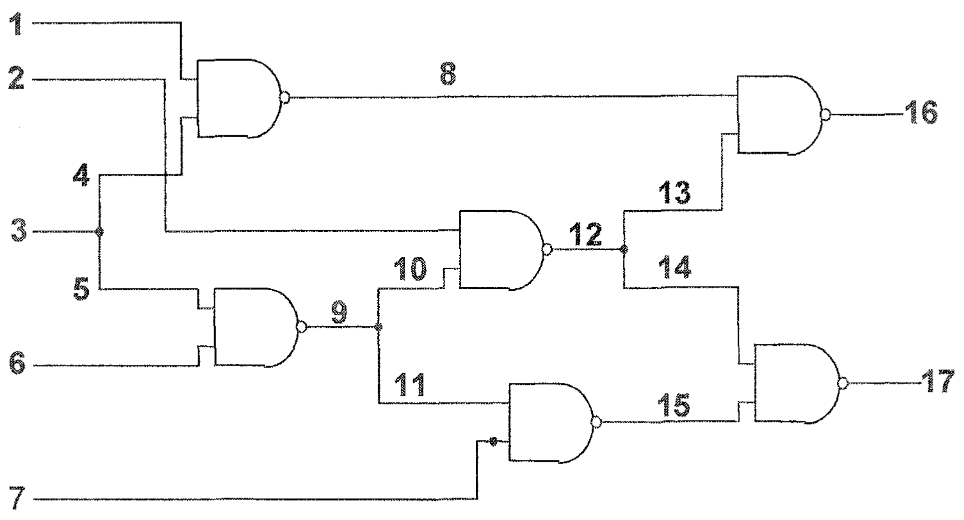


Figure 6-3. An example circuit named C17 used to illustrate the netlist format (ISCAS85 format).

The netlist of c17 is described as follows:

```
*c17 iscas example
*----------------------------------------------------
*
*
*  total number of lines in the netlist .............    17
*  simplistically reduced equivalent fault set size =    22
*        lines from primary input  gates .......     5
*        lines from primary output gates .......     2
*        lines from interior gate outputs ......     4
*        lines from **     3 ** fanout stems ...     6
*
*        avg_fanin  =  2.00,     max_fanin  =  2
*        avg_fanout =  2.00,     max_fanout =  2
*
*
*
*
*
    1      1gat inpt    1   0       >sa1
    2      2gat inpt    1   0       >sa1
    3      3gat inpt    2   0 >sa0 >sa1
    4      4fan from     3gat       >sa1
    5      5fan from     3gat       >sa1
    6      6gat inpt    1   0       >sa1
    7      7gat inpt    1   0       >sa1
    8      8gat nand    1   2       >sa1
    1      4
    9      9gat nand    2   2 >sa0 >sa1
    5      6
   10     10fan from    9gat       >sa1
```

```
11      11fan from    9gat       >sa1
12      12gat nand    1   2 >sa0 >sa1
 2      10
13      13gat nand    1   2      >sa1
 11       7
14      14gat nand    0   3 >sa0 >sa1
 8      12    13
```

In the above description, the line which begins with '*' is a comment line and is ignored during processing. Each line represents a line specification or the input lists of a gate. There are up to seven columns for each line:

1st column:	Line number.
2nd column:	The name of the line which is used for the connections.
3rd column:	The type of the line or gate which can be "inpt" or "from" or one of logic functions such as "and", "nand", "or", "xor", or "not". The "inpt" is a primary input of the circuit. The "from" is a fanout branch which is connected to a fanout stem specialized in the next column.
4th column:	The number of fanout branches except for the fanout branch line.
5th column:	The number of inputs (fanin) of the gate except for the fanout branch line. It is 0 for a primary input gate. If the type of the line is a logic function, the next row specifies the line numbers of the inputs of the gate.
6th column:	If ">SA0" is present, a stuck-at zero fault should be injected on the line.

7th column: If ">SA1" is present, a stuck-at one fault should be injected on the line.

Because the primary purpose of this circuit netlist description is for the area of pseudorandom testing, in which fault coverage and identification is achieved, fault information of the line is present in it, and it should be noted that in the last two columns, some of the entries are null indicating that those faults are equivalent to some other faults and need not be considered. Probabilistic properties of signals are mainly concerned in our research, hence, the 6th and 7th are ignored in our MCP simulator.

6.2.3 PSPICE SIMULATOR

PSpice circuit simulation software is a product of OrCAD, Inc. Circuits to be tested are designed in this simulator and the generated signals are saved as the input files for the tested circuit. The detailed information of PSpice can be found at: www.orcadpcb.com.

6.2.4 FILTER

The PSpice simulator simulates the tested circuit under realistic condition, which means it is impossible for the simulator to give us a zero-delay output results. Hence, a filter is used to filter out those transitions caused by glitching (delay in input signals to a gate will generate glitches in the output, and delays might be caused by gate delays). To identify those glitches, the clock for input signals are set to be very slow when they are generated, and a program was developed named Glitching Eliminator that is used to find the results for the zero-delay model.

6.2.5 *MCP Algorithm Simulator*

MCP algorithm simulator is a program developed using VC++6.0 under Microsoft Visio studio development environment. Microsoft Fundamental Classes (MFCs) are used for menu, I/O interface, documentation and message processing. This simulator takes the ISCAS85 netlist format (circuits description format) as inputs, and probabilities, activities and correlations between primary inputs are also accepted as support inputs to our MCP Algorithm simulator. The output is a text file containing the probability and activity of every signal in the circuit.

6.3 Experimental Results

6.3.1 *MCP Algorithm VS Other Algorithms*

To illustrate more efficiency and more accuracy of our MCP algorithm compared to other approaches, consider a two-input multiplier as shown in Figure 6-3 as the first test case. Assume $p_A = p_B = p_C = 0.5$ and $\alpha_A = \alpha_B = \alpha_C = 0.2$ for all three mutually independent primary inputs, and also assume the circuit running in a zero-delay model. Designed in PSpice Capture® and took a logical simulation run in PSpice, the results are compared to those taken from different approaches and is listed in Table 6-1.

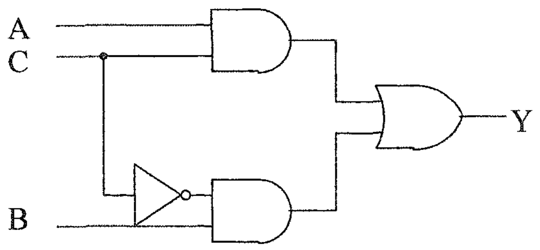


Figure 6-4. A two-input multiplexer for the first test case.

Table 6-1. Results of output signal *Y* of a two-input multiplexer using different algorithms.

Algorithms / Output signal Y	Najm etc. Algorithm [3]	Roy etc. Algorithm [4]	MCP Algorithm [20]	PSpice Simulator
Activity	0.3	0.28	0.236	0.235
Probability	0.475	0.475	0.5	0.498
Error (%)	28%	19%	1%	NA

From Table 6-1, we can see that the MCP algorithm gives us a very close result compared to PSpice simulation. The results for the other two algorithms are not as accurate as the MCP algorithm due to correlations between internal signals. It should be noted that it takes much longer time to obtain the results for Roy's algorithm than that used by MCP and Najm's Algorithm.

6.3.2 MCP ALGORITHM (ZERO-DELAY MODEL)

To test the accuracy and efficiency of the MCP algorithm in zero-delay model, several circuits are used including one ISCAS85 benchmark circuit [21] called

C432[8], in which has the most correlation effects. We use the same two-input multiplier as shown in Figure 6-4 for the first test case. Assuming the input values are $p = 0.5$ and $\alpha = 0.2$ for all three primary inputs, MCP simulator gives $p(y) = 0.5$ and $\alpha(y) = 0.236$. The results from the PSpice logic simulation run, showing the correct convergent behavior at the output y, are shown in Figure 6-5. The horizontal axis in Figure 6-5 is the number of clocks elapsed during the simulation run, and the vertical axis is the corresponding activity and probability values of the output node y. The two horizontal dashed lines are the values of activity and probability computed by MCP simulator. From Figure 6-5 we can see that 4000 clock cycles are long enough to obtain a correct convergent result for output signal y. It might not true for some other internal nodes, though. To obtain better results, several test cases should be taken including some internal nodes and output nodes in every logic simulation runs. Even then, it is practically impossible to examine the activity plot for every node to determine whether the run is long enough for it to converge. Based on several test circuits, however, it is found that an average of 4000 clock cycles per input node seems to be enough to approximate most node activities. Such logic simulation runs were performed on all tested circuits in our experiments.

[8] C432 is a 27-channel interrupt controller with 36 inputs, 7 outputs and 160 gates.

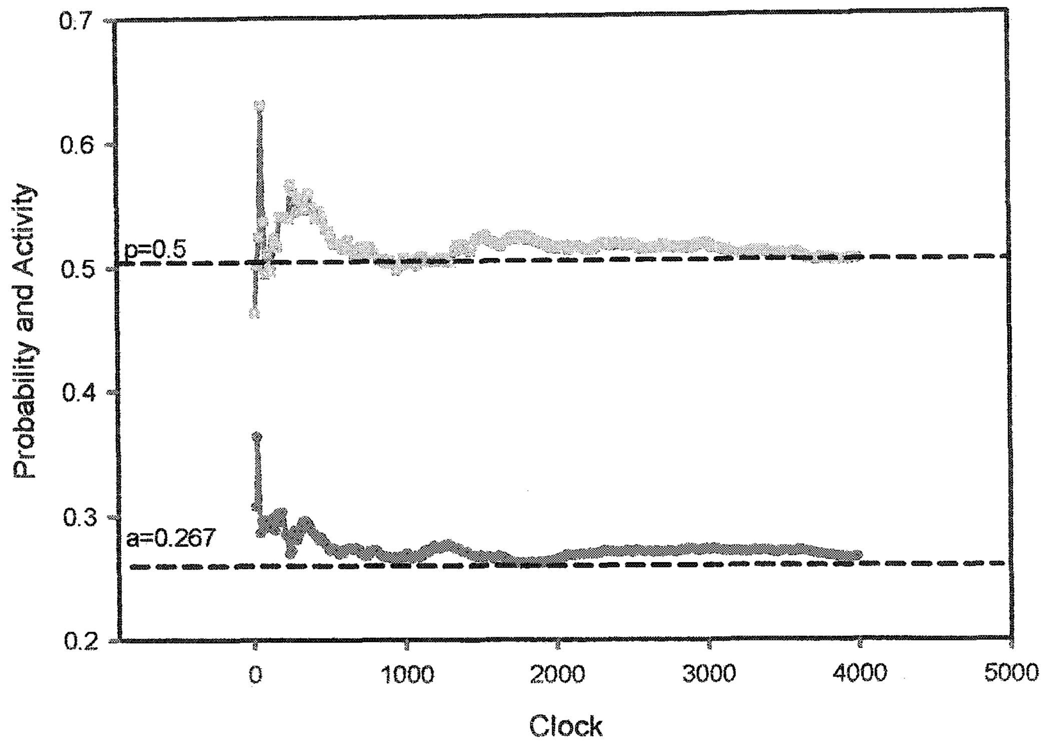


Figure 6-5. Activity and probability convergence plot at output node *y*.

Our second test case is a ISCAS85 circuit called C17, which is shown in Figure 6-2. We use values $p = 0.5$ and $\alpha = 0.2$ for all primary inputs and test the probabilities and activities for all internal and output nodes. Also all primary inputs are assumed to be mutually independent. The comparison between probability and activity values produced by the MCP Algorithm and those produced through PSpice simulation are provided in Table 6-2. From Table 6-2 we can see that the results obtained from our simulator is almost the same as the results obtained from PSpice simulation.

Table 6-2. Results obtained from C17 circuit.

node	MCP Algorithm		Pspice Simulation	
	Probability	Activity	Probability	Activity
8	0.750	0.180	0.749	0.178
9	0.750	0.180	0.747	0.182
12	0.625	0.222	0.626	0.224
15	0.625	0.222	0.630	0.223
16	0.563	0.247	0.556	0.249
17	0.563	0.254	0.560	0.255

The MCP Algorithm was also evaluated using a circuit named C432 from the ISCAS-85 Benchmark Set. For this circuit there are a total of 145 distinct signals, not including the primary inputs. (Note that there are a total of 432 physical signals, which includes fan-out signals.) Table 6-3 shows the distribution of absolute differences between activity values computed by the MCP Algorithm and those derived through simulation. These results indicate that the MCP Algorithm produces very accurate predictions of signal activities.

Table 6-3. Results from MCP Algorithm and Simulation Studies for Circuit C432 from the ISCAS-85 Benchmark Set.

Absolute Diff. Range	Number of Signals
[0, 0.01]	70
(0.01, 0.02]	35
(0.02, 0.03]	19
(0.03, 0.04]	10
(0.04, 0.05]	10
(0.05, 0.06]	1
(0.06, 1]	0

Relative Error Range (%)	Number of Signals
[0, 1]	43
(1, 2]	41
(2, 5]	31
(5, 10]	25
(10, 20]	3
(20, 50]	2
>50	0

6.3.3 *Glitching Power Prediction*

Power consumption due to glitches may play a critical role in power consumption prediction algorithms and tools. Figure 6-6 is a special example in which glitching is an extreme factor. In this example, the activity for the output signal, named O15 in Figure 6-6, is extraordinarily different when considering glitches to that when no gitching is taken into account. Allowing for glitching, the activity of the output signal is around 2.5 compared to around 0.5220 when no glitching is allowed in PSpice simulation (under the same simulation conditions when all primary inputs are assumed to be mutually independent and the probabilities and activities are all set to be $p = 0.5, \alpha = 0.2$, respectively).

Table 6-4. The activity values for output signals for Figure 6-6 considering glitching and without considering glitching.

Signal Name	MCPG Algorithm Simulation	PSpice Simulation (nonzero-delay)	PSice Simulation (zero-delay)
O1	0.4	0.3348	0.3348
O2	0.6	0.4990	0.4030
O3	0.8	0.6700	0.4348
O4	1.0	0.8168	0.4523
O5	1.2	0.9900	0.4763
O6	1.4	1.1500	0.4798
O7	1.6	1.3020	0.4940
O8	1.8	1.4500	0.5050
O9	2.0	1.6068	0.4990
O10	2.2	1.7700	0.4970
O11	2.4	1.9165	0.5010
O12	2.6	2.0600	0.4970
O13	2.8	2.2080	0.5030
O14	3.0	2.3600	0.5140
O15	3.2	2.500	0.5220

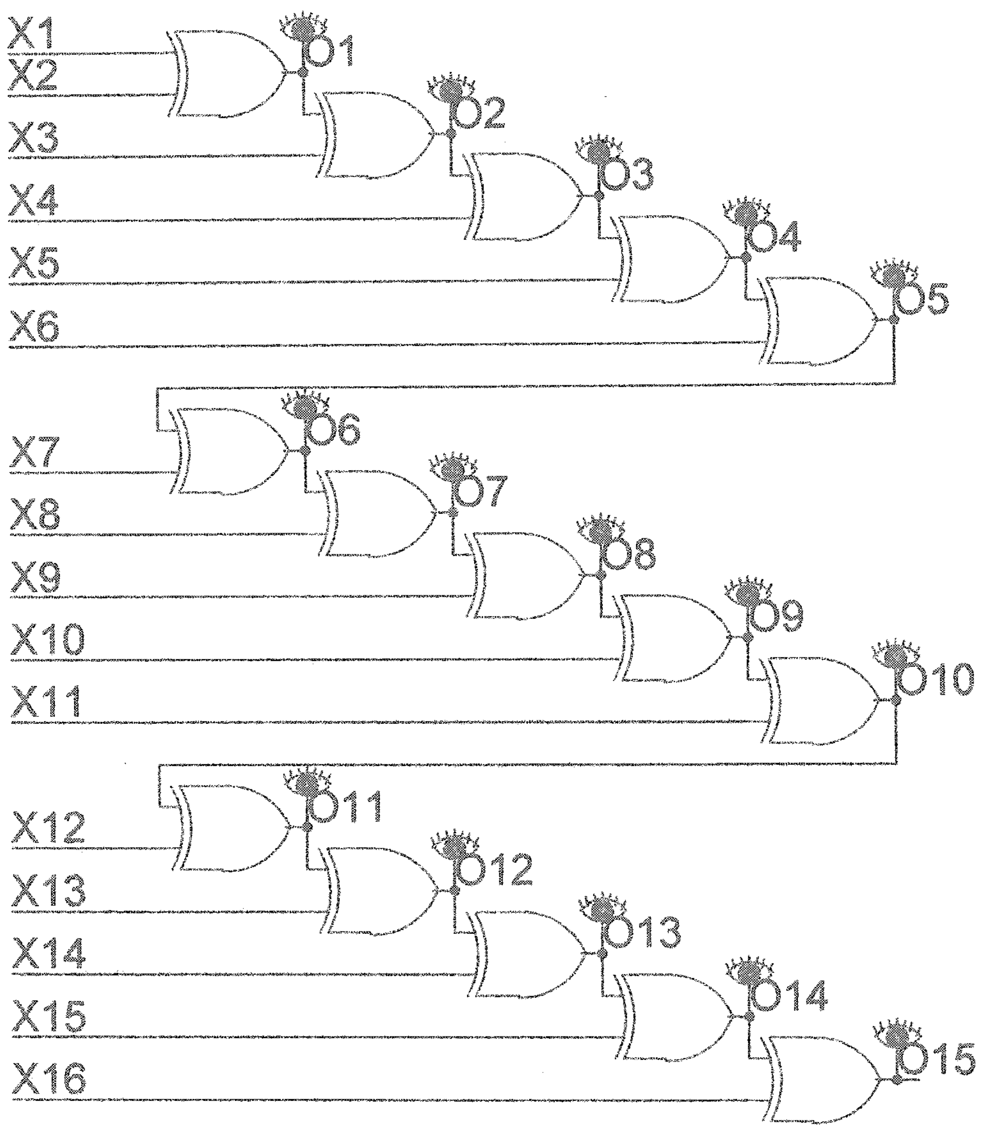


Figure 6-6. A special example to show the large difference of the activity of the output signal O15 between considering glitching and without considering glitching.

This example implies that glitching may produce much more power consumption in VLSI CMOS circuits, which is very important and should be taken into account in our power consumption prediction algorithms and tools. The MCPG algorithm introduced in Chapter 5 is for this purpose. Table 6-4 shows some results obtained by implementing MCPG algorithm to the circuit shown in Figure 6-6.

From Table 6-4 we can see that the results obtained from MCPG algorithm are much better compared to the PSpice results without considering glitching. However, the errors are still very large. The reason for the large errors is that MCPG algorithm simulates an ideal running environment for the logic circuit based on the assumption that the circuit can run as fast as possible. PSpice simulation, however, mimics a real logic running environment that has a limited clock frequency. Hence for the MCPG case, glitches caused in the previous stage will generate new glitches in current stage whenever there are delays between the input signals at this stage, and these new generated glitches will cause new glitches in the next stage, which will propagate through all gates in the circuit. For PSpice simulation, it mimics the real logic run environment that has limited circuit speed. Due to this speed limitation, some glitches caused in previous stage might not generate any new glitches in the next stage. Figure 6-7 is one of the results from PSpice logic simulation run of the circuit shown in Figure 6-6. It shows that some glitches will generate some new glitches in the next stage but some do not cause any new glitches in the next stage.

An interesting feature of the result shown in Figure 6-7 is the glitch width that determines whether this glitch will cause new glitches or not in the future stages. If the glitch width is equal or less than the width of the shortest clock frequency (corresponding to the speed limit of the test circuit), then no further glitches are generated by this glitch. In other words, not all transitions of the signal will cause new transitions if the circuit doesn't have enough time to respond to this transition. It illustrates the point that no signals can run faster than the circuit speed limit. This leads us to a new idea of how to calculate glitch activities by using MCP algorithm (we call this new algorithm as MMCP algorithm).

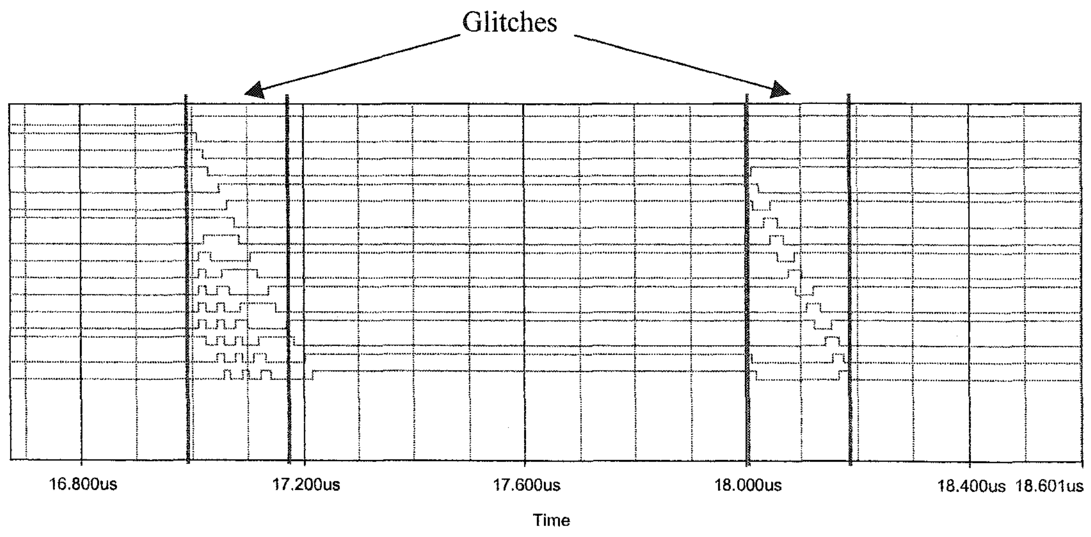


Figure 6-7. A result from PSpice simulation run of the circuit in Figure 6-6, which illustrates glitches effect the next stage.

Assume we know that the highest frequency of the circuit to be tested is no more than F_{max} (the circuit can not run faster than F_{max}). By normalizing the activities of primary input signals of the circuit with this frequency F_{max}, the activity value of each primary input becomes $1/F_{max}$ of the original value. By doing this, we can mimic the running environment of the circuit to a running environment with frequency F_{max}. It results that no glitches will be generated in the simulation run, and this new running environment satisfies the requirements of MCP algorithm.

This idea can be illustrated by using Figure 6-6 as an example. Suppose the circuit runs in a frequency of 10MHz, and the maximum frequency (maximum speed) the circuit can afford is 100MHz. Assume the activities and probabilities of the three primary inputs are all assigned to be 0.2 and 0.5, respectively, and the input signals are assumed to be mutually independent. Also suppose that every gate in the circuit has the same delay time. First, normalizing all input signals in 100MHz frequency to mimic a new running environment. Hence, by normalizing 0.2 into $0.2 \times 10/100$ = 0.02, the MCP algorithm will take the activity input values of 0.02 for all primary inputs. Second, run MCP algorithm and then the activities of all signals normalized by frequency F_{max} can be produced. For instance, it will produce the activity value of 0.1772 for the output signal O8. The final result for signal O8 is 1.772 by normalizing it back to original running frequency, which is 10MHz, by normalizing 0.1772 into $0.1772 \times 100/10$ = 1.772.

Because we couldn't determine the exact speed limit of the PSpice simulator, we run the MMCP simulation for the circuit in Figure 6-6 by setting the speed limit to different values and the results are shown in Figure 6-8. In Figure 6-8, the red line with up-triangle symbols is the data obtained from MCPG simulation in which infinite circuit speed is assumed. The pink line with down-triangle symbols is the result from MCP simulation with zero-delay model assumed. The blue line with circle symbols is the results from PSpice simulation considering glitching effects that is the actual activities for the output signals. Let F be the ratio of the possible fastest frequency PSpice could afford to the original running frequency in the tested circuit, the other three lines are the results from the MMCP simulator by setting $F = 100$, $F = 10$ and $F = 5$, respectively. It is shown from Figure 6-8 that $F = 10$ is a reasonable value that gives us a very close result to the actual activities.

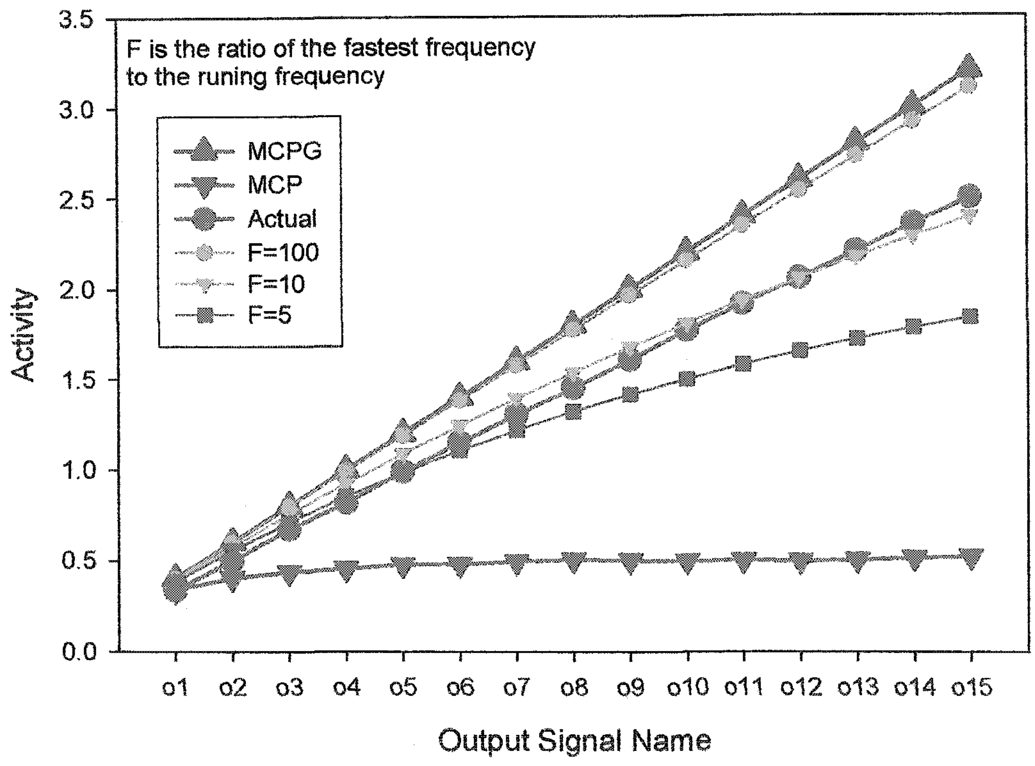


Figure 6-8. Results produced by MMCP algorithm running under different *F* values.

We also tested the C432 circuit using MMCP algorithm by setting the frequency ration *F* to be 10. The average error of all signals for frequency ratio 10 is about 9%, and the maximum relative error is about 48%, which shows that our MMCP algorithm can handle glitching effects very well. Table 6-5 shows the results for some significant (i.e. largest) error signals in circuit C432.

Table 6-5. Results of some most significant (i.e., largest) error signals in circuit C432.

Signal Name	MMCP ($F = 10$)	PSpice (Actual)	MCP (zero-delay)
136	0.1980	0.2000	0.1800
164	0.1693	0.1330	0.1050
183	0.3606	0.2815	0.2460
219	0.1980	0.1900	0.1900
229	0.1435	0.1480	0.1300
256	0.6895	0.4660	0.3360
275	0.8132	0.6320	0.3900
293	0.1980	0.1860	0.1860
302	0.1257	0.1670	0.1000
311	0.8180	0.8300	0.3500
322	0.2613	0.2570	0.2280
336	0.4357	0.4100	0.2600
340	0.4578	0.4770	0.2200
349	0.3409	0.2690	0.1600
384	0.7356	0.6060	0.3600
387	0.3312	0.2710	0.1500
391	0.3164	0.2440	0.1200
392	0.7362	0.6810	0.3600
396	0.2960	0.2440	0.1200
397	0.7616	0.6330	0.3600
399	0.8564	0.6690	0.3650
400	0.8564	0.6690	0.3650
401	0.7537	0.6220	0.3600
402	0.7874	0.6670	0.2700
403	0.8196	0.6870	0.3000

CHAPTER 7

SUMMARY

The total power dissipation of a CMOS circuit is the sum of the three types of power consumption, i.e., dynamic power dissipation, short-circuit current power consumption and static power consumption. Because the dynamic power dissipation is by far the dominant component, thus almost all methods used to calculate power consumption in CMOS circuits are focused on estimation of dynamic power consumption in CMOS circuits. Considering that the power estimation is calculated in the gate level, and both the supply voltage and the capacitance have already been determined at design steps, the power consumption in gate level can be estimated by calculating the switching activity for each circuit node.

The power estimation methodologies at the logic gate level can be divided into two general classes: statistical-based and probabilistic-based methodologies. Because the simulation result is highly dependent on the primary input vectors, the statistical-based power estimation, represented by the Monte-Carlo approach, needs to use a large number of input vectors to simulate the circuit in order to achieve a near real result of the circuit. This makes it impractical for large circuits and long input sequences. Compared to the statistical-based approach, probabilistic-based approaches compute the switching activities in one run, which can result in much less time. According to the type of the circuit, probabilistic-based approach can be categorized into methods for combinational and sequential circuits. Combinational circuits can be further classified into zero-delay and non-zero-delay model.

Several probability estimation approaches are overviewed, which are based on a single probability parameter signal model. Although this probability parameter is not directly used in calculating a circuit's power consumption, it is a necessary component for signal models common to the activity approaches that utilize both signal probability and signal activity parameters.

The approaches of [2], [3], and [4] can have high computational complexities because the number of terms in the underlying equations/transformations can grow exponentially with the number of primary inputs to the circuit. In [7], a trade-off between computational complexity and resulting accuracy is illustrated in the context of the underlying equations/transformations introduced in [2]. In particular, an approximate approach is defined in [7] in which the transformations of [2] are applied in a "gate-by-gate" fashion. Thus, instead of deriving the transformation for a signal's probability parameter in terms of the circuit's primary inputs, it is derived in terms of the immediate inputs to the logic gate associated with the signal. This approach greatly reduces the computational complexity, but introduces error in the calculated probability parameters for circuits with re-convergent fan-out.

Similar trade-offs between computational complexity and accuracy are possible relative to the evaluation of Equation 3.4 and Equation 3.9 (associated with [3] and [4], respectively). Instead of deriving a signal's logic function in terms of the circuit's primary inputs, the parameters to the immediate inputs the signal's logic gate can be used. Again, this type of "gate-by-gate" technique will generally introduce error because it does not account for correlations present among the internal signals that drive the gates within the circuit.

The approach of [6] is a fast and accurate "gate-by-gate" technique for calculating a signal's probability parameter. It introduces the concept of a correlation factor to account for and appropriately adjust the transformation for correlated inputs to a gate.

A Markov-chain model can also be used to model signals. It is shown that the proposed Markov chain model is equivalent to the two-parameter probability/activity signal model. A more efficient and more accurate algorithm (named MCP algorithm) based on Markov-chain signal modeling is present. By propagating signal parameters and correlation cofactors from the primary inputs through the circuit in a "gate-by-gate" fashion, this MCP algorithm can achieve a very good accuracy and an $O(M^2)$ time complexity where M is the number of signals in the circuit. The advantage of modeling signals with Markov chains is that it makes it possible to compute correlations between signals related to both probability and activity.

In non-zero delay model, glitches will cause large errors in general power consumption estimation algorithms and tools. Thus a MCPG algorithm is developed that is expanded from the MCP algorithm to take account of glitching effects. Compared to MCP algorithm, MCPG algorithm computes the glitching transitions caused by the associated delays and propagates this glitching transitions to the next stage. Because it is assumed that the target circuit can run in ideally infinite speed, every glitch may cause new glitches in the next stage. Thus the MCPG algorithm gives us an upper bound of the activity of each node in the circuit. Another new idea named MMCP algorithm is also developed, which deals with the real situation of a circuit's fastest response. Based on the assumption that the highest speed of the target circuit can run is given, activities of the primary inputs are normalized in this highest speed frequency and feed into MCP algorithm to calculate activities of every node. The final results are set back by renormalizing them into the original frequency.

An MCP algorithm (running in MCP, MCPG and MMCP algorithms) is developed using VC++6.0 under Microsoft Visio studio development environment. Microsoft Fundamental Classes (MFCs) are used for menu, I/O interface, documentation and message processing. This simulator takes the ISCAS85 netlist format (circuit description format) as inputs, and probabilities, activities and correlations between primary inputs are also accepted as support inputs to our MCP Algorithm. The output is a text file containing the probability and activity of every signal in the circuit. To investigate the accuracy and efficiency of the results produced by the MCP algorithm, PSpice® circuit simulations are performed on several test circuits. In the simulation studies, time-series realizations from the assumed Markov chain model for each primary input are used to drive the circuit simulation. Estimates of signal probabilities were derived from the simulations by counting the fraction of time each signal took on a value of unity. Estimates of signal activities are derived from the simulations by counting signal transitions. It is shown that the MCP algorithm will give us a very close result compared to other approaches in a zero-delay model. For non-zero-delay model, our simulation shows that the MCPG algorithm gives us a good prediction of the activity of each node, which is an upper bound of the activity of each node, and the MMCP algorithm produces a closer prediction of activities of all signals in the circuit, provided that the proper scaling frequency is determines.

The above techniques do not apply directly to sequential circuits. Future work includes extending the MCP algorithm to simulate circuits with feedbacks, such as sequential circuits.

REFERENCE

[1] R. Burch, F. N. Najm, P. Yang, and T. Trick, "A Monte Carlo Approach for Power Estimation", *IEEE Trans. VLSI Systems*, Vol. 1, No. 1, Mar. 1993, pp. 63-71.

[2] K. P. Parker and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks," *IEEE Trans. Computers*, Vol. C-24, No. 6, June 1975, pp. 668-670.

[3] F. N. Najm, "Transition Density: A New Measure of Activity in Digital Circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 2, Feb. 1993, pp. 310-323.

[4] T.-L. Chou and K. Roy, "Estimation of Activity for Static and Domino CMOS Circuits Considering Signal Correlations and Simultaneous Switching," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 10, Oct. 1996, pp 1257-1265.

[5] F. N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Trans. on VLSI Systems*, Vol. 2, No. 4, Dec. 1994, pp. 446-455.

[6] S. Ercolani, M. Favalli, M. Damiani, P. Olovo, and B. Ricco, "Estimate of Signal Probability in Combinational Logic Networks," *Proc. IEEE European Test Conference*, April 1989, pp. 132-138.

[7] B. Krishnamurthy and I. G. Tollis, "Improved Techniques for Estimating Signal Probabilities," *IEEE Trans. Computers*, Vol. 38, No. 7, July 1989, pp. 1041-1045.

[8] J. B. Thomas, "An Introduction to Applied Probability and Random Processes", *Krieger Publishing*, Huntington, NY, 1981.

[9] M. J. M. Smith, "Application-Specific Integrated Circuits", *Addison Wesley*, Reading, MA, 1997.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms", *McGraw-Hill New York*, NY, 2001.

[11] Kaushik Roy, and Sharat C. Prasad, "Low-Power CMOS VLSI Circuit Design", *Wiley-Interscience New York*, NY, 2000.

[12] Dimitrios Soudris, Christian Piguet, and Costas Goutis, "Designing CMOS Circuits for Low Power", *Kluwer Academic Publishers*, Boston, 2002.

[13] Radu Marculescu, Diana Marculescu, and Massoud Pedram, "Probabilistic Modeling of Dependencies During Switching Activity Analysis", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, No. 2, Feb. 1998.

[14] J. Rabaey and M. Pedram, "Low Power design Methodologies", *Kluwer Academic Publishers*, 1996.

[15] A. Chadrakasan and R. W. Brodersen, "Low Power Digital CMOS Design", *Kluwer Academic Publishers*, 1995.

[16] C. Lee, "Representing of Switching Circuits by Binary Decision Diagrams", *Bell System Technology Journey*, pp. 985-999, July 1959.

[17] R. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Tans. on CAD*, pp. 77-691, Aug. 1986.

[18] M. Xakellis and F. Najm, "Statistical Estimate of the Switching Activity in Digital Circuits", *ACM/IEEE Design Automation Conference*, pp. 728-733, 1994.

[19] F. N. Najm and M. Y. Zhang, "Extreme Delay Sensitivity and the Worst-Case Switching Acivity in VLSI Circuits", *ACM/IEEE Design Automation Conference*, pp. 623-627, 1995.

[20] Hongping Li, John K. Antonio, and Sudarshan K. Dhall, "Fast and Precise Power Prediction for Combinational Circuits," *Proceedings of the IEEE Symposium on VLSI*, sponsor: IEEE, Tampa, FL, pp. 254-259, Feb 2003.

[21] Brglez, F., and Fujiwara, H. "A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran". *In 1985 International Symposium on Circuits And Systems*, 1985.