Probabilistic Analysis of Power Dissipation in VLSI Systems

Radu Marculescu

CENG 98-29

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213-740-4458)
August 1998

PROBABILISTIC ANALYSIS OF POWER DISSIPATION IN VLSI SYSTEMS

by

Radu Marculescu

A Dissertation Presented to the

FACULTY OF THE GRADUATE SCHOOL

UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the

Requirements for the Degree

Doctor of Philosophy

(ELECTRICAL ENGINEERING)

August, 1998

To my parents, my wife Diana and son Andrei with gratitude and love.

Acknowledgments

Coming to an end with this thesis, I would like to thank to those people who influenced my career and made this endeavor possible. I want to apologize, in advance, for this rather impersonal way of acknowledging their real contribution to this journey started exactly five years ago. My gratitude and appreciation goes far beyond these lines.

I am deeply indebted to my advisor, Professor Massoud Pedram, for his guidance and continuous encouragement throughout my Ph.D. I will always treasure the open and stimulating relationship I have had with him throughout these difficult years. He was always open to new ideas and gave me complete freedom to pursue my own research interests. He taught me many things not only about CAD (and good quality research) but also about life.

I would like to thank Professors Sandeep Gupta and Michael Arbib for serving in my thesis committee and for taking from their time to review this thesis. They had always time to listen to me and gave me good advice. I would also like to thank Professors John Choma Jr., Giovanni De Micheli and Srinivas Devadas for helping me during my last year as doctoral student.

I am grateful to all my current and past colleagues in THE Low-Power group for helping me out in many occasions and for creating such a friendly environment; special thanks to Drs. Chi-Ying Tsui and Chih-Shun Ding for interesting and stimulating discussions. I wish also to thank NSF, DARPA and SRC for their financial support that made this work possible.

Before coming to US, many dedicated individuals have influenced my career and stimulated my passion for research: my elementary school teacher, Ms. Coter, introduced

me to the wonderful world of mathematics; the high-school teachers S. Maftei, S. Oprita and T. Jaluba, with whom I had the privilege to work in Dorohoi, have trained me for a technical profession; Professors Liviu Goras and Nicolae Gheorghiu (whom I met in Iasi) opened my eyes to research and offered me an analytical perspective of the real (not necessarily engineering) world. Later, when I joined the CS department at the "Politehnica" University in Bucharest, I had the chance of meeting Professors Irina Athanasiu, Serban Petrescu and Cristian Giumale who helped and encouraged me to pursue an academic career.

My friends and colleagues have played an important role in my life and profession. In Iasi, Cristian Chitaru was always close to me in discovering the (new) world. (Thank you Chitaras for your wonderful friendship and those pure moments we have spent together.) I also remember Ioan Rosca with whom I have had (and still have) many interesting discussions. After finishing my Master degree, I joined the CAD group at ICCE, Bucharest. There I met Marius Garbea, Tudor Niculiu, and Sorin Cotofana with whom I discussed on any possible subject that we could imagine. Even if we are not in the same CAD group anymore, I am happy that we are still enjoying doing research on our own. Outside my technical training and profession, Dan Condurache offered me the acute perspective of life (or being alive) and created for me an unforgettable Macondo. Monica & Gigi, helped me to come in US and have constantly encouraged me to move forward. My friends in LA, Haru, Araxi, Adi, Alex, Margaret & Irv, have made my life easier and more enjoyable in many occasions.

Finally, I am deeply grateful to my family for everything they did for me to make this adventure possible. My parents, Elena-Graziela and Vasile, made me what I am, gave me the sense of right and wrong and taught me what love is. My wife, Diana, shared with me everything in a perfect balance of love and understanding. She is just my other half. And our son Andrei simply is.

Table of Contents

List of Figures

List of Tables

Abstract

Computer-Aided Design tools play an important role in the efficient design of high-performance digital systems. In the past, time, area and testability were the main concerns of the design community. With the growing need for low-power electronics, power analysis and low-power synthesis have also become major design concerns.

The objective of this thesis is to provide the theoretical foundations of an integrated framework for power analysis of digital CMOS circuits. To this end, the problem of power estimation is addressed from a probabilistic viewpoint and the following contributions are made:

1) For the class of *static* power estimation techniques, the previous work done on switching activity estimation is extended to explicitly account for complex spatiotemporal correlations which occur at the primary inputs when the target circuit receives data from real applications. More precisely, using lag-one Markov Chains, two new concepts - conditional independence and signal isotropy - are brought into attention and based on them, sufficient conditions for exact analysis of complex dependencies are given. It is shown that the relative error in calculating the switching activity of a logic gate using only pairwise probabilities can be upper-bounded. Relying on the concept of signal isotropy, approximate techniques with bounded error are proposed for estimating the switching activity.

2) For the class of *dynamic* power estimation techniques, an approach which reduces the simulation time by orders of magnitude is proposed using the paradigm of sequence compaction. Given an initial input sequence, we compact this sequence into a much shorter one such that the new data represents a good approximation as far as total power consumption is concerned. To this end, a hierarchical Markov model is introduced as a flexible framework for capturing not only complex spatiotemporal correlations, but also the dynamic changes in the sequence characteristics. Furthermore, a family of variable-order dynamic Markov models is presented to handle the case of sequential circuits. These Markov models provide an effective way for accurate modeling of external input sequences that affect the behavior of finite state machines.

Chapter 1 Introduction

1.1 Motivation

In the past, the common view on integrated circuits performance was centered around the concept of *speed* of the circuit. This perception was motivated by the continuous demand for increased processing power which is normally required in most applications. Generally speaking, the improvements in performance can be obtained at the expense of the silicon area. As a consequence, using sophisticated CAD tools, the VLSI designer has traditionally explored the area-time implementation space, attempting to find a reasonable balance between the two (often conflicting) objectives.

Despite the fact that power consumption is another metric which can quantify the performance of a circuit, until recently, power concerns were totally secondary in the design process. Of course, in certain ultra low-power applications (e.g. pacemakers, wrist watches, etc.), power dissipation has always played a major part in the design cycle. However, in the case of high-performance microprocessors or general ASICs, we have to note that these systems were just designed for maximum performance, regardless of the side effects on power consumption.

In the recent years, however, this picture has changed drastically. Driven by increased levels of device integration and complexity, together with higher device speed, power dissipation has become a crucial design concern, limiting the number of devices that can be put on a chip and dramatically affecting the packaging and cooling costs associated with ASICs. Power dissipation is even a bigger concern for the class of battery-powered personal computing devices and wireless communication systems. To give some figures, for a portable multi-media terminal, the projected power budged, when implemented using

off-the-shelf components not designed for low-power operation, is around 40W. With modern Nickel-Cadmium batteries offering around 20W-hours/pound, such a terminal would require 20 pounds of batteries for 10 hours of operation between recharges [38]. Taking a completely different perspective, contemporary high-performance processors dissipate as much as 30W. In the near future, it is projected that a $10cm^2$ microprocessor, clocked at 500MHz, would consume around 315W [38]. As a consequence, the analysis and synthesis of low-power VLSI systems has become a very active area of research in the VLSI community. Because excessive power dissipation reduces the reliability and increases the cost of cooling and packaging systems, it is obvious that the successful research on low-power systems can be beneficial for the development of both, high-performance and portable systems.

In this thesis, we focus our attention on the issue of power analysis and provide the theoretical foundations of an integrated framework for power estimation in digital circuits. CAD tools have always been the major requirement for designing state-of-the-art VLSI circuits and low-power VLSI systems are no exception to this rule. We do need a new generation of CAD tools targeting power optimization and, to this end, a methodology for accurate power analysis is a must.

1.2 Sources of power dissipation

In this section we will review the main sources of power dissipation in CMOS digital circuits with particular emphasis on *average power* consumption. *Peak power* dissipation is also an important concern but, in this research, we assume that the operation of the circuit has been ensured for worst-case conditions.

Let us consider the CMOS inverter in Figure 1.1. Under normal operation, the average power consumed by this inverter can be described by the following simple equation [47] which basically reflects one static (P_{static}) and one dynamic component ($P_{short} + P_{leakage} + P_{dynamic}$):

$$P_{avg} = P_{static} + P_{short} + P_{leakage} + P_{dynamic} \tag{1.1}$$

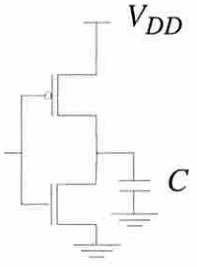


Figure 1.1 A CMOS inverter for power analysis

Ideally, CMOS circuits dissipate no static (DC) power since, in the steady state, there is no direct path from V_{DD} to ground. Of course, this scenario can never be realized in practice since in reality the MOS transistor is not a perfect switch. Thus, there will always be leakage currents and substrate injection currents, which will give rise to a static component of CMOS power dissipation. Fortunately, for current technology, the actual contribution from this static component is several orders of magnitude below other contributors and therefore, this component is practically absent for well-designed CMOS circuits. However, we note that this component may become significant under some circumstances and consequently, logic families which experience static power dissipation should be avoided in low-power design.

The dynamic component of power dissipation arises from the transient switching behavior of the CMOS device. At some point during the switching, both NMOS and PMOS devices in Figure 1.1 will be turned on and, during this time, a short-circuit exists between V_{DD} and ground and currents are allowed to flow. Fortunately, in well-designed

CMOS circuits, the short-circuit power dissipation is usually a small fraction (10-15%) of the total power dissipation then it is reasonable to expect that, in the design for high-performance, short-circuit power dissipation is not a major concern.

$P_{leakage}$ is mainly caused by reverse saturation currents in diffusion regions of the PMOS and NMOS transistors as well as the subthreshold leakage current of transistors that are normally off. Unfortunately, despite the fact that in today's circuits $P_{leakage}$ is still a small fraction (less than 10%) of the total power dissipation, the subthreshold leakage is becoming increasingly important as power voltages decrease.

Finally, the dominant source of power dissipation (80-90% of total power) is represented by the dynamic dissipation due to capacitance charging and discharging. This situation is modeled in Figure 1.1 where all capacitances are lumped at the output capacitance *C*. Assuming a gate-level implementation of the target circuit, one can calculate the average dynamic power consumption by using the formula:

$$P_{dynamic} = \left(\frac{f_{clk}}{2} V_{DD}^2\right) \sum_x C(x) sw(x) \tag{1.2}$$

where f_{clk} is the clock cycle frequency, V_{DD} is the supply voltage, $C(x)$ and $sw(x)$ represent the load capacitance and the switching activity, respectively, at the output of any gate x in the circuit. The product $C(x)sw(x)$ is often referred to as the *switched capacitance* of the circuit. As we can see from equation (1.2), power dissipation is linearly dependent on the switched capacitance. Generally speaking, calculation of the switched capacitance is difficult as it depends on static and dynamic parameters of the circuit which are not readily available or precisely characterized. Among the factors which have an important role in accurate estimation of the switched capacitance we mention: physical

capacitance, circuit structure, input pattern dependence, the delay model, statistical variation of circuit parameters. In this thesis, we assume that the physical capacitance is either calculated or taken from the library and therefore concentrate on average switching activity estimation.

Summarizing our discussion about sources of power dissipation in CMOS circuits, we conclude that, for current technology, the dominant fraction of average power is attributed to the dynamic component caused by the switching activity at the gate outputs. For this reason, the focus of this thesis is on estimating the dynamic power consumption.

1.3 Basic issues in power estimation

Existing techniques for power estimation at gate- and circuit-level can be divided in two main classes: static and dynamic [37], [25]. *Static techniques* rely on probabilistic information about the input stream (e.g. switching activity of the inputs, signal correlations, etc.) to estimate the internal switching activity of the circuit. These techniques generally provide sufficient accuracy with low computational overhead. However, they cannot accurately capture factors such as slew rates, glitch generation and propagation. In addition, a major challenge in probabilistic power estimation approaches is the ability to account for internal dependencies due to the reconvergent fan-out in the target circuit. This problem, which we will refer to as the *"circuit problem"*, is by no means trivial. Indeed, a whole set of solutions have been proposed, ranging from approaches which build the global OBDDs [1] and therefore capture all internal

dependencies, to efficient techniques which partially account for dependencies in an incremental manner [32], [45], [46].

Dynamic techniques [19], [18] explicitly simulate the circuit under a "typical" input stream. Consequently, their results depend on the simulated sequence, and the required number of simulated vectors is usually high. These techniques can provide sufficient accuracy, but the price to be paid is too high. Switching activity information can be extracted by doing exhaustive simulation on small circuits; it is, however, unrealistic to rely on simulation results for large circuits. To address this problem, a Monte Carlo simulation technique was proposed in [7]. This technique uses an input model based on a Markov process to generate the input stream for simulation. The simulation is performed in an iterative fashion and the iteration terminates when some stopping criterion is met. The approach has two deficiencies. First, the required number of samples (which directly impacts the simulation run time) is approximately proportional to the ratio between the sample variance and the square of the sample mean value. For certain sequences, this ratio becomes large, thus significantly increasing the simulation run time. Second, if the sample distribution significantly deviates from the normal distribution, the simulation may terminate prematurely. Difficult distributions that cause premature termination are bi-modal, multi-modal, and distributions with long or asymmetric tails [11]. Existing statistical techniques for power estimation in sequential circuits require a large number of simulated vectors. Basically, their efficiency is lower than that for combinational circuits [8], [34].

In this thesis, we investigate both alternatives that is, we provide possible solutions for static and dynamic power estimation. This is because, in our vision, the designer should benefit from both types of techniques in order to asses the potential of a new design from the power consumption perspective. In addition to this, the unprecedented complexity of the current digital systems demands a combined set of tools for power estimation rather than the use of a single tool.

1.4 Thesis contribution

The methodology presented in this thesis offers a possible solution to the critical need for CAD tools to estimate power dissipation during the design process in order to meet the power budget without having to go through a costly redesign effort. The results presented in this thesis advance the state-of-the-art by providing the following key contributions.

- *An analytical model for spatio-temporal correlations in digital circuits*

To estimate the power consumption one has to calculate the signal and transition probabilities of the internal nodes of the target circuit. Besides delay and circuit structure, the signal and transition probabilities at the internal nodes are very sensitive to the statistics of the input patterns that feed the circuit. In Chapter 3, we propose the first analytical model which accounts for spatiotemporal correlations under the zero-delay hypothesis. As the main theoretical contribution, we extend the previous work done on switching activity estimation to explicitly account for complex spatio-temporal correlations which occur at the primary inputs when the target circuit receives data from real applications.

The model based on spatio-temporal correlations clearly demonstrates, qualitatively and quantitatively, the importance of being accurate node-by-node (not only for the total power consumption) and identifies potential drawbacks in previous approaches when patterns feeding the inputs become highly correlated. The practical value of this model becomes relevant during optimization and synthesis for low-power when accurate power estimation is the critical step.

Based on this model, we developed a set of efficient techniques for power estimation in large combinational modules. The whole idea behind efficiency is to calculate and propagate transition correlation coefficients from the circuit inputs toward the circuit outputs, and hence, eliminate the need for building the global function of the intermediate nodes. As quantitative illustration, these techniques can analyze combinational modules with 10,000 gates in about 200 CPU sec., yet producing power estimates with less than 10-15% error, compared to circuit-level simulation.

Going deeper into details, the model of spatiotemporal correlations is based on the following ideas introduced in this thesis.

a) Uniform modeling of spatial and temporal correlations using time-homogeneous Markov Chains

> Temporal correlations for the values of some signal x in two successive clock cycles are considered through a Markov Chain with only two states; spatial correlations for a pair of signals (x,y) are modeled by a four-state Markov Chain. By using this formalism, we were able to consider in a systematic way different kinds of dependencies in large combinational modules for both pseudorandom and highly correlated input streams.

b) Using the conditional independence and signal isotropy concepts to develop an incremental propagation mechanism based on binary decision diagrams (BDDs)

> Based on conditional independence and signal isotropy concepts, we provide sufficient conditions for exact analysis of complex dependencies. More precisely, we demonstrate that the relative error in calculating the switching activity of a logic gate using only pairwise probabilities can be upper-bounded. In the general case, the conditional independence problem has been shown to be NP-complete and thus appropriate approximate techniques (with bounded error) are presented to estimate the switching activity. From a practical point of view, the use of conditional independence and signal isotropy concepts significantly speed-up the estimation process (the larger the module, the more substantial is the speed-up that can be achieved) while the quality of estimates is improved by one order of magnitude, on average, compared to other proposed approaches that ignore the input correlations.

In summary, the model of spatio-temporal correlations improves the state-of-the-art in two ways: theoretically, by providing a deep insight about the relationship between the logic and probabilistic domains, and practically, by offering a sound mathematical framework and an efficient technique for power analysis.

- *A class of dynamic Markov models for input data modeling*

As we have already noted, existing techniques for switching activity estimation are either simulative or non-simulative in nature. Simulative techniques are in general more accurate, but tend to be significantly slower compared to the other ones. It is simply impractical to simulate a large circuit using millions or even thousands of vectors. Yet,

this is currently necessary to get accurate power estimates using circuit-level or switch-level simulators. It is thus useful to develop a vector compaction technique that reduces the input vector size while preserving the statistical behavior of the original set (and hence the average power dissipation). This can make simulators a viable option for power analysis even for very large circuits as the size of the input vector set can be greatly reduced. To be more precise, having characterized the input statistics (using signal probabilities and pairwise spatiotemporal correlations), one has to generate a minimum cardinality set of input data that exhibits the same statistical behavior. To this end, having an initial sequence (assumed representative for some target circuit), we target *lossy compression*, that is the process of transforming an input sequence into a smaller one, such that the new body of data represents a *good approximation* as far as total power consumption is concerned.

In Chapter 4 and Chapter 5, we provide different solutions for vector compaction problem which reduces the gap between simulative and nonsimulative approaches used in power estimation. The mathematical foundation of these techniques is probabilistic in nature and they make extensive use of concepts and results from the theory of finite Markov chains. One important feature of these techniques is the independence from the actual implementation of the target circuit, therefore one can use any of these techniques early in the design cycle when the structure of the circuit is not determined yet.

As the results demonstrate, large compaction ratios of few orders of magnitude can be obtained without significant loss (less than 5% on average) in the accuracy of power estimates. These achievements are possible by developing the following three models.

a) The flat model

The basic approach is presented in Chapter 4; it relies on *adaptive* (*dynamic*) *modeling* of binary input streams as first-order Markov sources of information; it is applicable to combinational and, under some circumstances, to sequential circuits. The model introduced in Chapter 4 is inspired from the literature on data compression [9]. However, the original model in [9] is not completely satisfactory for our purpose. We thus extend the initial formulation to manage not only correlations among adjacent bits that belong to the same input vector, but also correlations between successive input patterns. This model gives good results when used for homogeneous input sequences that is, input sequences which contain statistically similar vectors. This is because homogeneous sequences exercise the circuit such that the value of average power converges rapidly. However, non-homogeneous input sequences (that is, input sequences where the stimuli may contain a mixture of vectors, each one very different as far as average switching activity per bit is concerned) are very common in practice and they may determine erroneous power estimates depending on which component (with low or high activity) is visited more often. Two refinements of this model are the hierarchical and variable-order models.

b) The hierarchical model

In Chapter 4, we introduce the hierarchical Markov model as an effective way to structure the input space into a hierarchy of *macro-* and *micro-states*: at the first (high) level in the hierarchy we have a Markov chain of macrostates; at the second (low) level, each macrostate is in turn characterized by a Markov chain for all its constituent microstates. Our primary motivation for constructing this hierarchical structure is to enable a better modeling of the different stochastic levels that are present in

sequences that arise in practice. Another important property of these models is the ability to capture different operating modes of the circuit using the first level in the hierarchical Markov model. This provides high adaptability to different operating modes of the circuit.

c) The variable-order model

This model is introduced in Chapter 5 and it is fully applicable to sequential circuits because it considers temporal correlations longer than one time step. Relying on the concept of *block entropy*, we present a technique for identifying the actual *order* of variable-order Markov sources of information that is, sources of information that can be *piecewise* modeled by Markov chains of different orders. Using the concept of dynamic Markov chain modeling, we propose an effective approach to compact an initial sequence into a much shorter one such that the steady state and transition probabilities (and therefore the total power consumption) in the target finite state machine (FSM) are preserved.

We note that the vector compaction procedures presented in Chapter 4 and Chapter 5 perform much better than classical statistical sampling techniques proposed for power estimation.

1.5 Thesis outline

We first provide a general background in the theory of finite order Markov chains. Far from being exhaustive, we restrict ourselves to only those concepts that are needed for a better understanding of the following chapters. We then start the main part of the thesis by first exploring, from a probabilistic point of view, the issue of switching activity estimation in combinational circuits. In Chapter 3, based on lag-one Markov chains, we

propose a new analytical model which accounts for spatio-temporal correlations under the zero-delay hypothesis. Evaluations of the model and a comparative analysis on common benchmark circuits show that node-by-node switching activities are strongly pattern dependent and therefore, accounting for dependencies is mandatory if accuracy is a major concern.

In the second part of this thesis, that is in Chapter 4 and Chapter 5, we present the theory and practice of sequence compaction for power estimation. We first note that the paradigm of sequence compaction looks very attractive because it represents a first step towards reducing the gap between static and dynamic techniques currently used in power estimation. As the main theoretical contribution, under stationary conditions, we formulate and prove the correctness of vector compaction problem and then introduce a family of dynamic Markov trees that can be used effectively in sequence compaction. More precisely, in Chapter 4, we introduce the hierarchical modeling of Markov chains as a flexible framework for capturing not only spatio-temporal correlations, but also the dynamic changes in the sequence characteristics such as different operating modes or power distributions. In Chapter 5, we deal with the special case of sequential circuits and introduce the variable-order dynamic Markov model as an effective way for accurate modeling of external input sequences that affect the behavior of the target FSM. As the experimental show, for both combinational and sequential circuits, large compaction ratios of orders of magnitude can be obtained without significant loss in accuracy for power estimates.

Finally, Chapter 6 contains a summary of the theoretical and practical results presented in this thesis and a discussion of directions for future research.

Chapter 2 Preliminaries

The purpose of this chapter is to provide a short background for the main concepts that will be used in the following chapters. The reader interested in switching theory and synthesis and optimization of digital circuits is referred to standard books published in the last few decades [17], [24], [5], [30], [10], [16]. In what follows, we present only the basic definitions and notations from the theory of finite-order Markov chains that will be used throughout this thesis. Far from being exhaustive, we restrict our attention to only those concepts that are required by our working hypotheses. For a complete documentation on the subject, the reader is referred to monographs [22], [20], [35], [44].

2.1 Finite order Markov chains

A *stochastic process* is defined as a family of random variables $\{x(t), t \in \mathrm{T}\}$ defined on a given probability space and indexed by the parameter t, where t varies over the index set T. The stochastic process is said to be *stationary* when it is invariant under an arbitrary shift of the time origin. In this case, the values assumed by the random variable $x(t)$ are called *states*, and the set of all possible states forms the *state space* of the process.

A *Markov process* $\{x(t), t \in \mathrm{T}\}$ is a stochastic process whose dynamic behavior is such that, given the present value, the future values of the process are independent of the past values. This is the so called "Markov property" and it defines a fundamental subclass of stochastic processes. We shall assume that the transitions out of state $x(t)$ are independent of time and, in this case, the Markov process is said to be *time-homogeneous*.

If the state space of a Markov process is *discrete*, the Markov process is referred to as a *Markov chain* (MC). In what follows, we consider only MCs with finite state space. If

we assume that the index set T is also discrete, then we have a *discrete-parameter* MC. We may assume, without loss of generality, that T = {1, 2, ...} and denote the MC by $\{x_n\}_{n\geq 1}$.

Definition 2.1 (Lag-one MC) A discrete stochastic process $\{x_n\}_{n\geq 1}$ is said to be a lag-one MC if at any time step $n \geq 2$ and for all states x_n:

$$p(x_n = \alpha_n | x_{n-1} = \alpha_{n-1} x_{n-2} = \alpha_{n-2} \ldots x_1 = \alpha_1) = p(x_n = \alpha_n | x_{n-1} = \alpha_{n-1}) \quad (2.1)$$

The conditional probabilities $p(x_n = \alpha_n | x_{n-1} = \alpha_{n-1})$ are called *single-step transition probabilities* and represent the conditional probabilities of making a transition from state x_{n-1} to state x_n at time step n. In homogeneous MCs these probabilities are independent of n and consequently written as $p_{ij} = p(x_n = j | x_{n-1} = i)$ for all $n \geq 2$.

Note: Latter in this thesis, we will model the stochastic behavior of binary sequences by MCs. For any two consecutive vectors v_i, v_j, we can write symbolically that $p_{ij} = p(next\_vector = j | present\_vector = i)$; that is, the probability p_{ij} of a transition to vector v_j given that the present vector is v_i.

The matrix Q, formed by placing p_{ij} in row i and column j, for all i and j, is called the *transition probability matrix*. We note that Q is a *stochastic matrix* because its elements satisfy the following two properties: $0 \leq p_{ij} \leq 1$ and $\sum_j p_{ij} = 1$.

Although conditional transition probabilities can be used as a rough approximation to the transition probabilities, in general we need to know the probability of a transition independent of the present state. These are so-called *total transition probabilities* $p_{i \to j}$

and can be calculated knowing the *state probabilities*, that is $p_{i \to j} = \pi_i \cdot p_{ij}$, where π_i represents the probability that the MC is in a given state *i*.

An equivalent description of the MC can be given in terms of its *state transition graph* (STG). Each node in the STG represents a state in the MC, and an edge labelled p_{ij} (from node *i* to node *j)* implies that the one-step transition probability from state *i* to state *j* is p_{ij}.

Example 2.1 Let S_1 and S_2 be two 2-bit sequences, of length 48, as shown in Figure 2.1a. These two sequences, have exactly the same set of first-order temporal statistics that is, they cannot be distinguished as far as wordwise one-step transition probabilities are concerned. In fact, in Figure 2.1b we provide the wordwise transition graph for these two sequences. Each node in this graph is associated to a distinct pattern that occurs in S_1 and S_2 (the upmost bit is the most significant one, e.g. in S_1, v_1 = '1', v_2 = '2', v_3 = '3',..., v_{48} = '1'). Each edge represents a valid transition between any two valid patterns and has a nonzero probability associated with it. For instance, the pattern '3' in S_1 and S_2 is always followed by '1' (thus the edge between nodes '3' and '1' has the probability 1) whereas it is equally likely to have either '0', '2' or '1' after pattern '1'.

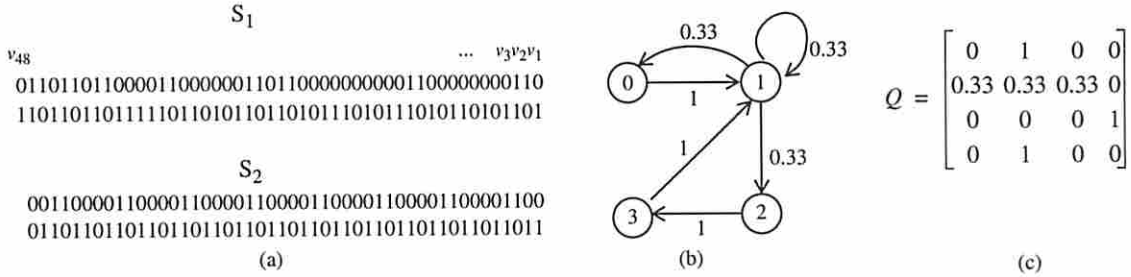


Figure 2.1 Sequence characterization with STG and transition matrix

Starting with different initial states and using a random number generator we may, of course, generate other sequences equivalent with S_1 and S_2 as far as the one-step transition probabilities are concerned. We can then see the graph in Figure 2.1b as a compact, canonical, characterization of sequences S_1 and S_2. Suppose that we want to compute the occurrence probability of string v = '01 10' that is, the probability that transition $1 \rightarrow 2$ is taking place in S_1. To this effect, we just use $p(v) = p(v_1 v_2) = p(v_1) \cdot p(v_2|v_1)$ which gives us the value of 1/6. If we are interested in finding the two-step transition probability $0 \rightarrow 1 \rightarrow 1$ in S_2, then using $p(v) = p(v_1 v_2 v_3) = p(v_1) \cdot p(v_2|v_1) \cdot p(v_3|v_2 v_1)$, we get the value of 1/6. The matricial representation, equivalent with the STG, is given in Figure 2.1c. We can easily verify that the sum of all elements on each row is 1, Q thus being indeed a stochastic matrix.

Definition 2.2 (Lag-k MC) A discrete stochastic process $\{x_n\}_{n \geq 1}$ is said to be a lag-k MC if at any time step $n \geq k+1$:

$$\begin{aligned} &p(x_n = \alpha_n | x_{n-1} = \alpha_{n-1} x_{n-2} = \alpha_{n-2} \ldots x_0 = \alpha_0) = \\ &= p(x_n = \alpha_n | x_{n-1} = \alpha_{n-1} x_{n-2} = \alpha_{n-2} \ldots x_{n-k} = \alpha_k) \end{aligned} \tag{2.2}$$

It should be noted that any lag-k MC can be reduced to a lag-one MC based on the following result.

Proposition 2.1 [35]. If $\{u_n\}_{n \geq 1}$ is a lag-k MC then $\{v_n\}_{n \geq 1}$, where $v_n = (u_n, u_{n+1}, \ldots, u_{n+k-1})$, is a multivariate first-order MC.

As a consequence, the study of lag-k MCs is practically reduced to study the properties satisfied by lag-one MCs. We will refer subsequently only to lag-one MCs but, by virtue of Proposition 2.1, all results easily translate to lag-k MCs.

2.2 State classification, decomposability, probability distribution

One of the fundamental problems in MCs theory is related to the state classification process. According to whether or not a state occurs infinitely often two types of states can be distinguished.

Definition 2.3 (Recurrent State) A state in the MC is called *recurrent* if the probability of returning in this state after $n \geq 1$ steps is greater than zero. Otherwise, the state is called *transient*. If the greatest common divisor over all such integers n is $d = 1$, then the state is also called *aperiodic*.

A finite MC has at least one recurrent state. If there are no transient states, then the MC is called *recurrent*. In our subsequent discussion, we will consider that *all states* are *recurrent* since all transient states vanish after a small number of steps. Indeed, it can be proved [20] that if a MC starts in a transient state then, with probability 1, it stays in the set of transient states for just a finite number of steps after which it enters a recurrent class where it remains forever. On account of this property a MC with transient states will also be called *absorbing*.

Definition 2.4 (Nondecomposable MC) A Markov chain is said to be *nondecomposable* if every state can be reached from every other state in a finite number of steps. Otherwise,

the chain is called *decomposable* and its transition matrix can be written as a block-diagonal matrix:

$$Q = \begin{bmatrix} Q_1 & 0 & \dots & 0 \\ 0 & Q_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & Q_r \end{bmatrix}$$, where the r sets of states contain only recurrent states and do not communicate with each other.

Here the stochastic matrices $Q_1, Q_2, ..., Q_r$ are the stochastic matrices associated with the recurrent classes. If $r > 1$ no communication is possible between these classes and, in fact, we have r distinct MCs that can be studied separately.

Note: The STG in Figure 2.1b is nondecomposable; we note that, in general, the STG associated to any input sequence of vectors is also nondecomposable.

Definition 2.5 *(*Periodic MC). A nondecomposable Markov chain is called *periodic* if and only if there is some periodic state with period $d > 1$. In this case, the d-th power of Q (that is, Q^d) can be written as a block-diagonal matrix with d matrices on the main diagonal. Otherwise, the Markov chain is said to be *aperiodic*.

A state that is recurrent and aperiodic is said to be *ergodic*. If all the states of a MC are ergodic, then the MC itself is said to be ergodic.

Changes of states over $n > 1$ time steps are ruled by probability rules simply expressed in terms of p_{ij}. Let us denote by p_{ij}^n the probability of transition from state i to state j in exactly n steps, namely: $p_{ij}^n = p(x_{m+n} = j | x_m = i)$, whatever the integer m. It is easily

recognized that probabilities p_{ij}^{n} (which are called the *n-step transition probabilities*) represent the entries of the Q^n matrix (called the *n-step transition matrix)*, $n \geq 1$. The Q^n matrix itself is still a stochastic matrix and satisfies the identity $Q^{m+n} = Q^m \cdot Q^n$, $m, n \geq 0$ (Q^0 is by definition the unit matrix I) or just the system of equations $p_{ij}^{m+n} = \sum_k p_{ik}^{m} \cdot p_{kj}^{n}$, known as the *Chapman-Kolmogorov* equations [35], [20]. In words, to go from i to j in $(m+n)$ steps, it is necessary to go from i to an intermediate state k in m steps and then from k to j in the remaining n steps. By summing over all possible intermediate states k, we consider all possible distinct paths leading from i to j in $(m+n)$ steps.

Theorem 2.1 [35] For a nondecomposable MC, the equation $\pi \cdot Q = \pi$ has a unique solution that represents the *stationary distribution* of the MC. ❑

Note: If the Markov chain is aperiodic, then Q^n converges to a stable matrix when $n \to \infty$, and π can be found to be any row of the limiting matrix. Otherwise, Q^n is no longer convergent in the usual sense, but in Cesaro sense [22]; that is, the limit $\lim_{n \to \infty} \frac{I + Q + \ldots + Q^{n-1}}{n}$ exists and it is equal to $\frac{I + Q + \ldots + Q^{d-1}}{d} \cdot A = A \cdot \frac{I + Q + \ldots + Q^{d-1}}{d}$, where $A = \lim_{n \to \infty} Q^{n \cdot d} = \begin{bmatrix} A_1 & \ldots & 0 \\ & \ldots & \\ 0 & \ldots & A_d \end{bmatrix}$; each A_i is a stable stochastic matrix. In this case, the solution of the equation in Theorem 2.1 is given by $\pi = \frac{1}{d} \cdot [\pi_1 \ldots \pi_d]^T$, where π_i is any row of the stable matrix A_i, for each $i = 1$,

2,..., d.

Note: Intuitively, in the case of binary sequences, the concept of stationary distribution implies that, as the observation time increases, that probability that the MC characterizing the sequence reaches each of its states converges to a stationary set of real numbers.

Finding the steady-state behavior is a fundamental problem in CAD and, generally speaking, it is possible to find STGs for which the state probabilities do not exist. However, in this thesis, we will not go beyond the requirements of Theorem 2.1 for existence and uniqueness of the state probability vector.

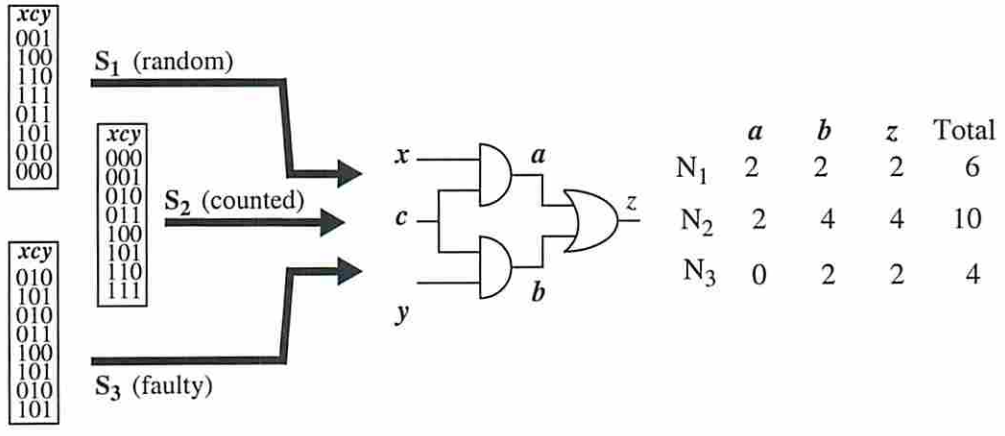
Chapter 3 Circuit Dependent Techniques

3.1 Introduction

As we have already seen, having a gate-level implementation of the target circuit, one can calculate the average dynamic power consumption by using the well-known formula $P_{dynamic} = \left(\frac{f_{clk}}{2} V_{DD}^2\right) \sum_x C(x) sw(x)$, where f_{clk} is the clock cycle frequency, V_{DD} is the supply voltage, $C(x)$ and $sw(x)$ represent the load capacitance and the switching activity, respectively, at the output of any gate x in the circuit. Estimating the capacitance at the technology independent phase of the logic synthesis process is difficult as it requires estimation of the load capacitances from structures which are not mapped yet to gates in a cell library. We assume, however, that the capacitances are either estimated or taken from a library and, in addition, we ignore the interconnect capacitances. As consequence, we concentrate on estimating the average switching activity per node (gate) which is the key parameter that needs to be correctly determined because charging and discharging different load capacitances is by far the most important source of energy dissipation in digital CMOS circuits.

Common digital circuits exhibit many dependencies; the most known one is the dependency due to reconvergent fan-out among different signal lines, but even structurally independent lines may have dependencies (induced by the sequence of inputs applied to the circuit) which cannot be neglected. Accounting for all kinds of dependencies is impossible even for small circuits; consequently, for real-size circuits, only some of the dependencies have been considered and even then, only heuristics have been proposed [37]. This is a consequence of the difficulty in managing complex data dependencies at acceptable levels of computational work.

Besides dependencies described above (called also *spatial dependencies*), another type of correlations, namely *temporal* may appear in digital circuits. Let us consider a simple case to illustrate these issues. The circuit in Figure 3.1 is fed successively by three input sequences, S_1, S_2 and S_3; S_1 is an exhaustive pseudorandom sequence, S_2 is also an exhaustive sequence but generated by a 3-bit counter and S_3 is obtained by a 'faulty' 3-bit counter.



	a	*b*	*z*	Total
N_1	2	2	2	6
N_2	2	4	4	10
N_3	0	2	2	4

Figure 3.1 An illustration of spatiotemporal correlations

All three sequences have the same signal probability on lines x, y and c ($p = 0.5$), but even intuitively they are completely different. There are two other measures which differentiate these sequences, namely *transition* and *conditional* probabilities, and switching activity calculations should rely on them. In fact, these sequences exercise the circuit such that the number of transitions N_1, N_2, N_3 on each signal line a, b, z (and hence the total number of transitions) becomes quite different once we feed S_1, S_2, or S_3 respectively. To accurately compute the number of transitions, we should undoubtedly account for the influence of reconvergent fanout, specifically in the previous figure, a and b cannot be considered independent signal lines. This problem could be solved (but only

for small circuits) by expressing the function of each signal in terms of primary inputs, but even then, neglecting the correlations among primary inputs can lead to incorrect results. As we see in this example, assuming input independence for sequences S_2 and S_3, is an unrealistic hypothesis because the patterns in each of them are temporally correlated (for example, each pattern in sequence S_2 is obtained from the previous one by adding a binary 1). Even more than this, transitions such as $0 \rightarrow 1$ or $1 \rightarrow 0$ on apparently independent signal lines (e.g. x and c in Figure 3.1) are correlated and a detailed analysis on these input streams can reveal a strong spatial relationship. Consequently, to accurately compute the switching activity, one has to account for both spatial and temporal dependencies starting from the primary inputs and continuing throughout the circuit.

In this chapter, we propose a new analytical model which accounts for spatiotemporal correlations under the zero-delay model. Its mathematical foundation consists of using lag-one Markov Chains to capture different kinds of dependencies in combinational circuits [27]. Temporal correlations for any signal x are considered through a Markov Chain with only two states whereas spatial correlations for any pair of signals (x,y) are modeled with a four-state Markov Chain. To summarize, the basic assumptions we use throughout this chapter are:

- The target circuit is combinational and the logic value of any signal line x can only be 0 or 1.
- Under a zero-delay model, any signal line x can switch at most once within each time step.

Under these hypotheses, we present theoretical and practical evidences showing that *conditional independence* is a concept powerful enough to overcome difficulties arising from structural dependencies and external input dependencies [28]. More precisely, based on conditional independence and *signal isotropy*, we give a formal proof showing that the statistics taken for pairwise correlated signals are sufficient enough to characterize larger sets of dependent signals.

The practical value of these results becomes particularly evident during optimization and synthesis for low-power [4]; a detailed analysis presented here illustrates the importance of being accurate *node-by-node* (not only for the total power consumption) and identifies potential drawbacks in previous approaches when patterns feeding the inputs become highly correlated. To support the potential impact of this research, experimental results are presented for common benchmark circuits.

The chapter is organized as follows. First, we review the prior work relevant to our research. In Section 3.3 we present in detail the analytic model for switching activity estimation which accounts for spatiotemporal correlations. In Section 3.4 we present global and incremental propagation mechanisms for transition probabilities and transition coefficients calculation. We also discuss the algorithmic complexity for the proposed propagation mechanisms. In Section 3.5 we improve the results given in Section 3.4 by providing an enhanced propagation mechanism based on conditional independence and signal isotropy. In Section 3.6 we give some practical considerations and report our results on common benchmark circuits. Finally, we summarize our main contribution.

3.2 Prior work

Most of the existing work in pseudorandom testing and power estimation relies on probabilistic methods and signal probability calculations. One of the earliest works in computing the signal probabilities in combinational circuits is presented in [36]. While the algorithm is simple and general, its worse case time complexity is exponential. For tree circuits which consist of simple gates, the exact signal probabilities can be computed during a single post-order traversal of the network [15]. An algorithm, known as the cutting algorithm, which computes lower and upper bounds on the signal probability of reconvergent nodes is presented in [39]. The bounds are obtained by cutting the multiple-fanout reconvergent input lines and assigning an appropriate probability range to the cut lines and then propagating the bounds to all the other lines of the circuits by using propagation formulas for trees. The algorithm runs in polynomial time in the size of the circuits. Ercolani et al. present in [12] a procedure for propagating the signal probabilities from the circuit inputs toward the circuit outputs using only pairwise correlations between circuit lines and ignoring higher order correlations. The signal probability of a product term is estimated by breaking down the implicant into a tree of 2-input AND gates, computing the correlation coefficients of the internal nodes and hence the signal probability at the output. Similarly, the signal probability of a sum term is estimated by breaking down the implicate into a tree of 2-input OR gates.

People working in power estimation have also considered the issue of signal probability estimation. An exact procedure based on Ordered Binary-Decision Diagrams (OBDDs) [6] which is linear in the size of the corresponding function graph (the size of the graph, of

course, may be exponential in the number of circuit inputs) can be found in [33]. Using an event-driven simulation-like technique, the authors describe a mechanism for propagating a set of probability waveforms throughout the circuit. Unfortunately, this approach does not take into account the correlations that might appear due to reconvergent fan-out among the internal nodes of the circuit. The authors in [1] use symbolic simulation to produce exact boolean conditions for switching at a particular node of the circuit. However, this approach is expensive in terms of computational cost (time and space requirements).

Recently, a few approaches which account for correlations have been proposed. Using an event-driven probabilistic simulation technique, Tsui et al. account in [45] only for first-order spatial correlations among probabilistic waveforms. Kapoor in [21] suggests an approximate technique to deal with structural dependencies, but on average the accuracy of the approach is modest. In [40] the authors rely on lag-one Markov Chains and account for temporal correlations; unfortunately, they assume independent transition probabilities among the primary inputs and use global OBDDs to evaluate the switching activity (severely limiting the size of the circuits they can process).

In what follows, we introduce a new model which extends over the previous work by taking into account spatiotemporal correlations at the primary inputs of the target circuit and by providing a general mechanism for handling them inside the circuit.

3.3 An analytical model for dependencies

We adopt the conventional probability model which consists of the triplet (Ω, Σ, p), where Ω represents the sample space, Σ denotes the class of events of interest and p is the probability measure associated to Σ.

3.3.1 Temporal correlations

Let us consider first a combinational logic module fed in turn by the input vectors v_1, v_2,..., v_n.

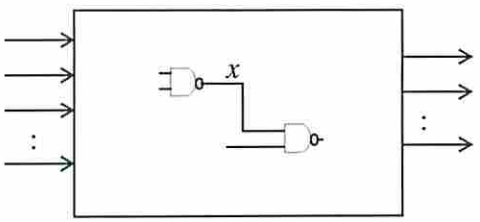


Figure 3.2 Temporal effects on any signal line x

While the input vectors v_1, v_2,..., v_n are applied at the primary inputs of the circuit at time steps 1, 2,..., n, the logic value of any line x may be 0 or 1. Hence, under the zero-delay model, x may switch at most once during each clock cycle. Let x_n be a random variable which describes the state of line x at any time n. If $\{x_n\}_{n\geq 1}$ is modeled as a lag-one Markov chain (Figure 3.3), then its behavior, over the state set $\Omega = \{0,1\}$, can be described through the transition matrix Q [35]:

$$Q = \begin{bmatrix} p^x_{0,0} & p^x_{1,0} \\ p^x_{0,1} & p^x_{1,1} \end{bmatrix} \tag{3.1}$$

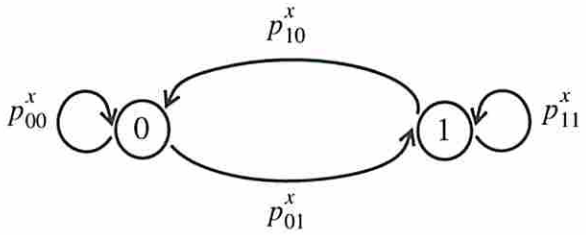


Figure 3.3 A lag-one Markov Chain describing temporal effects on line x

Every entry p^x_{ij} in the Q matrix represents the conditional probability of signal line x and may be viewed as the one-step transition probability to state j at step n from state i at step n-1.

Definition 3.1 (Conditional Probabilities) We define the *conditional probabilities* of any signal line x as:

$$p^x_{i,j} = p((x_n = j)\big|(x_{n-1} = i)) = \frac{p((x_n = j) \cap (x_{n-1} = i))}{p(x_{n-1} = i)} \quad \forall i, j = 0, 1 \tag{3.2}$$

where $n \geq 2$.

We note that Q is a *stochastic* matrix that is, every column adds to unity:

$$p^x_{0,0} + p^x_{0,1} = 1 \quad p^x_{1,0} + p^x_{1,1} = 1 \tag{3.3}$$

A lag-one Markov Chain has the property that one-step transition probabilities do not depend on the 'history', i.e. they are the same irrespective of the number of previous steps. The process $\{x_n\}_{n\geq 1}$ is *homogeneous* and *stationary*: indeed, because any combinational circuit is a memoryless device, having a homogeneous and stationary distribution at the primary inputs is a sufficient condition for homogeneity and stationarity to hold throughout

the circuit [35]. Because the process $\{x_n\}_{n\geq 1}$ is homogeneous, then the probability distribution of the chain π may be expressed as:

$$\pi = (Q)^n \pi_0 \tag{3.4}$$

where π_0 is the initial distribution vector. Because the process $\{x_n\}_{n\geq 1}$ is also stationary, then equation (3.4) becomes:

$$\pi = Q\pi \tag{3.5}$$

Theorem 3.1 The signal probabilities may be expressed in terms of conditional probabilities as follows:

$$p(x=0) = \frac{p^x_{1,0}}{p^x_{1,0} + p^x_{0,1}} \qquad p(x=1) = \frac{p^x_{0,1}}{p^x_{1,0} + p^x_{0,1}} \tag{3.6}$$

Proof: We may write equation (3.5) explicitly as:

$$\begin{bmatrix} p(x=0) \\ p(x=1) \end{bmatrix} = \begin{bmatrix} p^x_{0,0} & p^x_{1,0} \\ p^x_{0,1} & p^x_{1,1} \end{bmatrix} \begin{bmatrix} p(x=0) \\ p(x=1) \end{bmatrix}$$

or $p(x=0) = p^x_{0,0}p(x=0) + p^x_{1,0}p(x=1)$ and $p(x=1) = p^x_{0,1}p(x=0) + p^x_{1,1}p(x=1)$, where $p(x=1)$ represents the signal probability of line x. On the other hand we have that $p(x=0) = 1 - p(x=1)$, respectively $p(x=1) = 1 - p(x=0)$ and then equation (3.6) follows immediately. ■

Definition 3.2 (Transition Probabilities) We define the *transition probabilities* of any signal line x as:

$$p(x_{i\to j}) = p((x_n = j) \cap (x_{n-1} = i)) \quad \forall i, j = 0, 1 \tag{3.7}$$

Signal, conditional and transition probabilities associated to any signal line x are not independent measures. The following two theorems describe quantitatively the relationship between them.

Theorem 3.2 Transition probabilities may be expressed in terms of conditional probabilities as:

$$p(x_{0\to 0}) = \frac{p^x_{1,0}p^x_{0,0}}{p^x_{1,0}+p^x_{0,1}} \qquad p(x_{0\to 1}) = \frac{p^x_{1,0}p^x_{0,1}}{p^x_{1,0}+p^x_{0,1}} \tag{3.8}$$

$$p(x_{1\to 0}) = \frac{p^x_{1,0}p^x_{0,1}}{p^x_{1,0}+p^x_{0,1}} \qquad p(x_{1\to 1}) = \frac{p^x_{1,1}p^x_{0,1}}{p^x_{1,0}+p^x_{0,1}}$$

Proof: Using equation (3.2) and stationarity of the process, we have: $p(x_{i\to j}) = p(x=i)p^x_{i,j}$ for any values $i, j = 0,1$. From equation (3.6), the above formulas are straightforward. ■

Theorem 3.3 Conditional probabilities may be expressed in terms of transition probabilities as:

$$p^x_{0,0} = \frac{p(x_{0\to 0})}{p(x_{0\to 0})+p(x_{0\to 1})} \qquad p^x_{0,1} = \frac{p(x_{0\to 1})}{p(x_{0\to 0})+p(x_{0\to 1})} \tag{3.9}$$

$$p^x_{1,0} = \frac{p(x_{1\to 0})}{p(x_{1\to 0})+p(x_{1\to 1})} \qquad p^x_{1,1} = \frac{p(x_{1\to 1})}{p(x_{1\to 0})+p(x_{1\to 1})}$$

Proof: It suffices to use the following two identities obtained from equation (3.8):

$$p(x_{0\to 0})+p(x_{0\to 1}) = \frac{p^x_{1,0}}{p^x_{1,0}+p^x_{0,1}} \qquad p(x_{1\to 0})+p(x_{1\to 1}) = \frac{p^x_{0,1}}{p^x_{1,0}+p^x_{0,1}}$$ ■

Example 3.1 Suppose that the signal line x takes a set of binary values described by the following string "*aababaaabb*", where *a*, *b* $\in \{0, 1\}$. To compute the signal, conditional and transition probabilities, we have to do the following calculations:

• Signal probability calculations

$p(x_n = a) = \frac{6}{10}$, $p(x_n = b) = \frac{4}{10}$ (because, out of 10 symbols, *a* is occurring 6 times and *b* is occurring 4 times).

• Conditional probability calculations

$p(x_n = a | x_{n-1} = a) = p^x_{a,a} = \frac{3}{6}$ (symbol *a* appears 6 times and half of the time it is followed by another *a*).

$p(x_n = b | x_{n-1} = a) = p^x_{a,b} = \frac{3}{6}$ (symbol *a* appears 6 times and half of the time it is followed by *b*).

$p(x_n = b | x_{n-1} = b) = p^x_{b,b} = \frac{1}{4}$ (symbol *b* appears 4 times and once it is followed by another *b*).

$p(x_n = a | x_{n-1} = b) = p^x_{b,a} = \frac{3}{4}$ (symbol *b* appears 4 times and 3 times it is followed by *a*).

Based on these probabilities we can define the stochastic matrix Q as: $Q = \begin{bmatrix} \frac{3}{6} & \frac{3}{6} \\ \frac{1}{4} & \frac{3}{4} \end{bmatrix}$.

• Transition probability calculations

$p(x_{a\to b}) = p(x=a)\cdot p^x_{a,b} = \frac{6}{10}\cdot\frac{3}{6} = \frac{3}{10} \neq p^x_{b,a}$ (follows immediately from equation (3.8) and equation (3.6))

$p(x_{b\to a}) = p(x=b)\cdot p^x_{b,a} = \frac{4}{10}\cdot\frac{3}{4} = \frac{3}{10} \neq p^x_{a,b}$.

Relying on Theorem 3.1, Theorem 3.2, and Theorem 3.3, the relationship between signal, conditional and transition probabilities can be illustrated as in Figure 3.4.

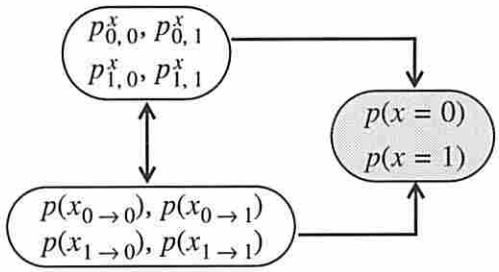


Figure 3.4 The paradigm of signal, conditional and transition probabilities

As we can see, we need less information to compute the signal probabilities, but the ability to derive anything else is severely limited. On the other hand, once we get either conditional or transition probabilities we have all we need to characterize that particular signal.

Definition 3.3 (Switching Activity) For any signal line x, the *switching activity* is defined as:

$$sw(x) = p(x_{0\to 1}) + p(x_{1\to 0}) = 2\frac{p^x_{1,0}p^x_{0,1}}{p^x_{1,0} + p^x_{0,1}} \tag{3.10}$$

As we can see, the switching activity depends only on two consecutive time steps an thus, using lag-one Markov chains is sufficient for estimating the switching activity.

Note: we should point out that equation (3.10) reduces to the well-known formula $sw(x) = 2 \cdot p(x = 1) \cdot [1 - p(x = 1)]$ *only if* the events are *temporally uncorrelated.* As long as we deal with temporally correlated signals, the exact relationship in equation (3.10) should be used. For instance, in Example 3.1, $sw(x) = 3/10 + 3/10 \neq 2 \cdot (6/10) \cdot (4/10)$.

3.3.2 Spatial correlations

This type of correlations has two important sources:

- *Structural dependencies* due to reconvergent fanout in the circuit;
- *Input dependencies* that is, spatial and/or temporal correlations among the input signals which are the result of the actual input sequence applied to the target circuit.

Referring to the combinational module in Figure 3.5, the lines x and y are obviously correlated because of the reconvergent fanout; on the other hand, even independent signal lines like the primary inputs of this module may also become correlated due to a particular input sequence (as is the case with sequences S_2 and S_3 in Figure 3.1 when the structurally independent lines x and c become correlated).

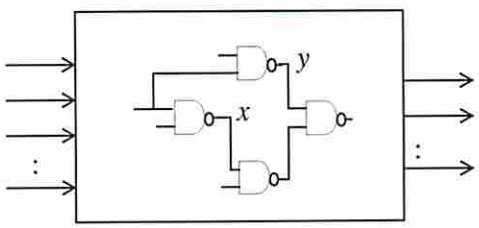


Figure 3.5 Two spatially correlated signal lines x and y

To take into account the exact correlations is practically impossible even for small circuits. To make this problem more tractable, we allow only *pairwise correlated signals*,

which is undoubtedly an approximation, but provides good results in practice. Consequently, we consider the correlations for all 16 possible transitions of a pair of signals (x,y) and model them as a lag-one Markov Chain with 4 states (denoted by 0, 1, 2, 3 which stand for the encoding 00, 01, 10, 11 of (x,y)).

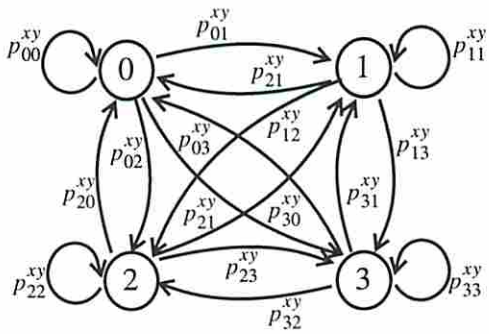


Figure 3.6 A lag-one Markov Chain for spatial correlations between x and y

Definition 3.4 (Pairwise Conditional Probabilities) We define the *conditional probability of pair of signals* (x,y) as:

$$p_{a,b}^{xy} = p(x_n = k \cap y_n = l \,|\, (x_{n-1} = i \cap y_{n-1} = j)) \tag{3.11}$$

where $a, b = 0, 1, 2, 3$, a being encoded as ij and b as kl. It basically describes the probability that the pair of signals (x, y) goes from state ij at time n-1 to state kl at time step n.

Ercolani et al. consider in [12] structural dependencies between any two signals in a circuit, through the *signal correlation coefficients* (*SCs*):

$$SC_{ij}^{xy} = \frac{p(x = i \cap y = j)}{p(x = i)p(y = j)} \tag{3.12}$$

where $i, j = 0,1$. Assuming that higher order correlations of two signals to a third one can be neglected, the following approximation is used:

$$SC_{ijk}^{xyz} = \frac{p(x = i \cap y = j \cap z = k)}{p(x = i)p(y = j)p(z = k)} = SC_{ij}^{xy} SC_{ik}^{xz} SC_{jk}^{yz} \tag{3.13}$$

Proposition 3.1 For every pair of signals (x,y) the following equations hold:

$$\sum_{j=0,1} SC_{ij}^{xy} p(y=j) = 1 \qquad \forall i = 0, 1 \tag{3.14}$$

$$\sum_{i=0,1} SC_{ij}^{xy} p(x=i) = 1 \qquad \forall j = 0, 1.$$

The set of 4 equations and 4 unknowns SC_{ij}^{xy} $i, j = 0, 1$ is indeterminate; the matrix of the system has the rank ≤ 3 in all non-trivial cases (i.e. when none of the signal probabilities is 1).

Proof: From the definition of SC, we get:

$$\sum_{j=0,1} SC_{ij}^{xy} p(y=j) = \frac{1}{p(x=i)} p((x=i) \sum_{j=0,1} (y=j)) = 1 \quad .$$

The second equation follows in a similar manner. ■

Our approach is more general: to capture the spatial correlations between signals, for each pair of signals (x,y) and for all possible transitions, we consider instead *transition correlation coefficients* (*TCs*).

Definition 3.5 (Transition Correlation Coefficients) We define the *TCs* for any two signals x, y as

$$TC_{ij,kl}^{xy} = \frac{p(x_{i \to k} \cap y_{j \to l})}{p(x_{i \to k}) p(y_{j \to l})} \tag{3.15}$$

where $i, j, k, l = 0, 1$.

Note: If the signals a and b in Figure 3.1 are spatially correlated then, based on *TCs* defined above, we have: $p(a_{0 \to 1} b_{1 \to 0}) = p(a_{0 \to 1}) p(b_{1 \to 0}) TC_{01,10}^{ab}$.

Definition 3.6 We define the *TCs* among three signals x, y, z as:

$$TC^{xyz}_{ijk,\,lmn} = \frac{p(x_{i\to l}\, y_{j\to m} z_{k\to n})}{p(x_{i\to l})\, p(y_{j\to m})\, p(z_{k\to n})} \tag{3.16}$$

where $i, j, k, l, m, n = 0,1$.

Neglecting higher order correlations, we therefore assume that the following relation holds for any signals x, y, z and any values $i, j, k, l, m, n = 0, 1$:

$$TC^{xyz}_{ijk,\,lmn} = TC^{xy}_{ij,\,lm} TC^{yz}_{jk,\,mn} TC^{xz}_{ik,\,ln} \tag{3.17}$$

Proposition 3.2 For every pair of signals (x,y) the following equations hold:

$$\sum_{j,\,l\,=\,0,\,1} TC^{xy}_{ij,\,kl}\, p(y_{j\to l}) = 1 \qquad \forall i, k = 0, 1 \tag{3.18}$$

$$\sum_{i,\,k\,=\,0,\,1} TC^{xy}_{ij,\,kl}\, p(x_{i\to k}) = 1 \qquad \forall j, l = 0, 1\,.$$

The above set of 8 equations and 16 unknowns $TC^{xy}_{ij,kl}$ $i, j, k, l = 0, 1$ is indeterminate; the matrix of the system has the rank ≤ 7 in all non-trivial cases (i.e. none of the transition probabilities is 1).

Proof: Similar to the proof for Proposition 3.1, but using the definition of *TCs*. ■

The last two propositions are very important from a practical point of view. The set of equations involving *SCs* may be solved knowing only SC^{xy}_{11} for example, and this was the approach taken in [12] (although, no similar analysis appeared in their original paper). In the more complex case involving *TCs*, we need to know at least 9 out of 16 coefficients in order to deduce all other values.

3.4 Propagation mechanisms

Having already described the analytical model for dependencies, we present subsequently the mechanism for propagating spatiotemporal correlations from the primary inputs throughout the target circuit. To this end, in what follows we ignore the higher order correlations; that is, correlations between any number of signals are expressed only in terms of pairwise correlation coefficients.

Definition 3.6 and equation (3.17) may be easily extended to any number of signals. Based on the above assumption, we use an OBDD-based procedure for computing the transition probabilities and for propagating the *TCs* throughout the network. The main reason for using the OBDD representation for a signal is that it is a canonical representation of a Boolean function and it offers a disjoint cover which is essential for our purposes. Depending on the set of signals with respect to which we represent a node in the boolean network, two approaches may be used:

- a *global approach*: for each node, we build the OBDD in terms of the primary inputs of the circuit;
- an *incremental approach*: for each node, we build the OBDD in terms of its immediate fanin and propagate the transition probabilities and the *TCs* through the boolean network.

The first approach is more accurate, but requires much more memory and run time; indeed, for large circuits, it is nearly impractical. The second one offers good results whilst being more efficient as far as memory requirements and running time are concerned. However, the propagation mechanisms we present subsequently are equally applicable to both global and incremental approaches.

3.4.1 Computation of transition probabilities

Let f be a node in the boolean network represented in terms of n (immediate fanin or primary input) variables $x_1, x_2, \ldots, x_n$; f may be defined through the following two sets of OBDD paths:

- $\prod_1$ - the set of all OBDD paths in the ON-set of f;

- $\prod_0$ - the set of all OBDD paths in the OFF-set of f;

Some of the approaches reported in the literature (e.g. [1]), use the XOR-OBDD of f at two consecutive time steps to compute the transition probabilities. We consider instead only the OBDD of f and through a dynamic programming approach, we compute the transition probabilities more efficiently.

Based on the above representation, the event 'f switching from value i to value j' ($i, j = 0, 1$), may be written as:

$$f_{i \to j} = \bigcup_{\pi \in \Pi_i} \bigcup_{\pi' \in \Pi_j} \bigcap_{k=1}^{n} x_{k_{i_k \to j_k}} \tag{3.19}$$

where i_k, j_k are the values of variable x_k on the paths π and π' respectively (that is $x_k = i_k$ for path π, $x_k = j_k$ for path π', where $i_k, j_k = 0, 1, 2$, and 2 stands for don't care values) for each $k = 1, 2, \ldots, n$. In other words, this event basically represents the *union* over all possible switchings from a path $(i_1, i_2, \ldots, i_n)$ to a path $(j_1, j_2, \ldots, j_n)$. Thus, the probability of the event 'f switches from value i to value j' ($i, j = 0, 1$) may be expressed as:

$$p(f_{i \to j}) = p\left(\bigcup_{\pi \in \Pi_i} \bigcup_{\pi' \in \Pi_j} \bigcap_{k=1}^{n} x_{k_{i_k \to j_k}} \right) \tag{3.20}$$

Applying the property of disjoint events (which is satisfied by the collection of paths in the OBDD), the above formula becomes:

$$p(f_{i \to j}) = \sum_{\pi \in \Pi_i} \sum_{\pi' \in \Pi_j} p(\bigcap_{k=1}^{n} x_{k_{i_k \to j_k}}) \tag{3.21}$$

Since the variables x_k may *not* be spatially independent of one another, the probability of a path to 'switch' from $(i_1, i_2, \ldots, i_n)$ to $(j_1, j_2, \ldots, j_n)$ *cannot* be simply expressed as the product of transition probabilities of individual variables. Instead, we will use the following result which holds if we neglect higher order correlations.

Proposition 3.3 If equation (3.17) is true for any three signals in the set $\{x_1, x_2, \ldots, x_n\}$, then:

$$p(\bigcap_{k=1}^{n} x_{k_{i_k \to j_k}}) = \prod_{k=1}^{n} \left(p(x_{k_{i_k \to j_k}}) \prod_{1 \le k < l \le n} TC^{x_k x_l}_{i_k i_l, j_k j_l} \right) \tag{3.22}$$

Proof: Follows directly from equation (3.17) by induction on the number of variables. ■

According to this result, the transition probability of the signal f for any values $i, j = 0, 1$ satisfies the following:

Theorem 3.4 The transition probability of a signal f from state i to state j $(i, j = 0, 1)$ is:

$$p(f_{i \to j}) = \sum_{\pi \in \Pi_i} \sum_{\pi' \in \Pi_j} \prod_{k=1}^{n} \left(p(x_{k_{i_k \to j_k}}) \prod_{1 \le k < l \le n} TC^{x_k x_l}_{i_k i_l, j_k j_l} \right) \tag{3.23}$$

Proof: Follows immediately by applying Proposition 3.3 to equation (3.21). ■

Though this expression seems to be very complicated, its complexity is within reasonable bounds. We will show that it is not necessary to enumerate all *pairs* of paths in the OBDD (which would provide a quadratic complexity in the number of paths in the

OBDD), but for a fixed path in Π_i the computation may be done in linear time in terms of the OBDD-nodes.

While for the global approach equation (3.23) can be applied knowing only the *TCs* of the primary inputs, for the incremental approach, we need a mechanism not only for computing the transition probabilities, but also for propagating the *TCs* through the boolean network. For a given node in the circuit, it is only necessary to propagate the *TC*s of the output with respect to the signals on which the inputs depend.

3.4.2 Propagation of transition correlation coefficients

Let f be a node with immediate inputs $x_1, x_2, \ldots, x_n$ and x a signal on which at least one of the inputs $x_1, x_2, \ldots, x_n$ depends. According to the definition of the *TCs*, for every $i, j, p, q =$ 0, 1 possible values of f and x respectively, we have:

$$TC^{fx}_{ip,\,jq} = \frac{p(f_{i\to j}x_{p\to q})}{p(f_{i\to j})p(x_{p\to q})} \tag{3.24}$$

Since the transition probabilities for f and x are already computed at this point, the only problem is to compute the probability of both f and x switching from i to j and from p to q respectively. We get the following important result:

Theorem 3.5 The *TC* between signals f and x, for any values $i, j, p, q = 0, 1$ may be expressed as:

$$TC^{fx}_{ip,\,jq} = \frac{\sum_{\pi' \in \Pi_i}\sum_{\pi'' \in \Pi_j}\prod_{k=1}^{n}\left(TC^{x_k x}_{i_k p,\,j_k q}\,p(x_{k_{i_k\to j_k}})\prod_{1\le k<l\le n}TC^{x_k x_l}_{i_k i_l,\,j_k j_l}\right)}{p(f_{i\to j})} \tag{3.25}$$

Proof: Using the representation of the event '*f* switches from *i* to *j*' given in equation (3.19), we obtain the following for the event '*f* switches from *i* to *j* and *x* switches from *p* to *q* *simultaneously*':

$$f_{i \to j} x_{p \to q} = \left(\bigcup_{\pi \in \Pi_i} \bigcup_{\pi' \in \Pi_j} \bigcap_{k=1}^{n} x_{k_{i_k \to j_k}} \right) x_{p \to q}$$

and:

$$p(f_{i \to j} x_{p \to q}) = p\left(\bigcup_{\pi' \in \Pi_j} \bigcup_{\pi' \in \Pi_j} \left\{ x_{p \to q} \bigcap_{k=1}^{n} x_{k_{i_k \to j_k}} \right\} \right)$$

Applying the disjointness property of the paths, we get:

$$p(f_{i \to j} x_{p \to q}) = \sum_{\pi \in \Pi_i} \sum_{\pi' \in \Pi_j} p(x_{p \to q} \bigcap_{k=1}^{n} x_{k_{i_k \to j_k}}) \tag{3.26}$$

Since the variables x_i may not be independent and, furthermore, some of them may depend on x, we need to apply the result provided by Proposition 3.3 for the set of n+1 variables $\{x_1, x_2, \ldots, x_n, x\}$:

$$p(f_{i \to j} x_{p \to q}) = \sum_{\pi \in \Pi_i} \sum_{\pi' \in \Pi_j} p(x_{p \to q}) \prod_{k=1}^{n} \left(TC^{x_k x}_{i_k p, j_k q} p(x_{k_{i_k \to j_k}}) \prod_{1 \le k < l \le n} TC^{x_k x_l}_{i_k i_l, j_k j_l} \right) \tag{3.27}$$

Thus, the *TC* between *f* and *x* in equation (3.25) follows immediately. ■

In the incremental approach, equation (3.23) and equation (3.25) are applied in a recursive manner until all probabilities and *TCs* become known.

3.4.3 Complexity issues

In order to assess the complexity claimed in Section 3.4.1, let us define $f_{\pi \to j} = \bigcup_{\pi' \in \Pi_j} \bigcap_{k=1}^{n} x_{k_{i_k \to j_k}}$, such that $f_{i \to j} = \bigcup_{\pi \in \Pi_i} f_{\pi \to j}$ ($i, j = 0, 1$ and i_k, j_k are the values of variable x_k on paths π, π' respectively). Using the disjointness property of the paths in the OBDD, the corresponding probability is:

$$p(f_{\pi \to j}) = \sum_{\pi' \in \Pi_j} p(\bigcap_{k=1}^{n} x_{k_{i_k \to j_k}}) \tag{3.28}$$

Since the path π is fixed, the above probability may be computed using the OBDD in the same way as a signal probability. The idea is that, using Shannon decomposition, the signal probability (and hence the above probability) may be computed in linear time in the number of the OBDD nodes. Indeed, $f_{\pi \to j}$ may be decomposed as follows:

$$f_{\pi \to j} = x_{k_{i_k \to 0}} f_{\pi \to j}^{\bar{x}_k} + x_{k_{i_k \to 1}} f_{\pi \to j}^{x_k} \tag{3.29}$$

where $f_{\pi \to j}^{\bar{x}_k}$, $f_{\pi \to j}^{x_k}$ are the cofactors with respect to $\bar{x}_k$ and x_k, respectively. Based on this recursive decomposition, we may also write a similar relation for the corresponding probabilities, taking also into account the possible existing correlations:

$$p(f_{\pi \to j}) = p(x_{k_{i_k \to 0}}) p(f_{\pi \to j}^{\bar{x}_k}) \prod_{k < l \le n} TC_{i_k i_l, 0 j_l}^{x_k x_l} + p(x_{k_{i_k \to 1}}) p(f_{\pi \to j}^{x_k}) \prod_{k < l \le n} TC_{i_k i_l, 1 j_l}^{x_k x_l} \tag{3.30}$$

Having computed this probability for each path π, we get immediately the corresponding transition probabilities and hence the switching activity. Thus, for a fixed path π, the complexity is $O(n^2 N)$, where n is the number of variables and N is the number of nodes in the OBDD. The n^2 factor comes from the necessity of taking into account the

correlations: besides the transition probabilities, we also have to keep track of the *TCs* involved on each path. There is a number of $\binom{n}{2}$ factors in the product, thus the complexity is quadratic in the number of variables.

Hence, overall, for all the paths in Π_i, the time complexity is $O(n^2NP)$, where P is the number of paths in the OBDD. In the incremental approach, this is within reasonable limits since usually n does not exceed 3 or 4 variables in the immediate fanin of the node.

Example 3.2 Let's consider the following function: $f = x_1 \oplus x_2 \oplus x_3$ and its OBDD representation from Figure 3.7. Suppose $i = 0, j = 1$ and $\pi = (0\ 1\ 1)$ is a fixed path in the OFF-set Π_0 of f. We can compute the probability given in equation (3.30) (that is, the probability of the event '$f = x_1 \oplus x_2 \oplus x_3$ switches to value 1 from the path $\pi = (0\ 1\ 1)$ in the OFF-set Π_0') by using a bottom-up parsing of the OBDD from the leaf labelled 1 to the root. We adopt a dynamic programming approach in which at each level we use the results computed at lower levels. For each node, the partial results are shown in Figure 3.7. For instance, in the case of node A (which corresponds to cofactoring f with respect to $\bar{x}_1 x_2$ or $x_1 \bar{x}_2$), variable x_3 must change from 1 to 0, and therefore node A is labeled with $p(x_{3_{1\to 0}})$. At node B (corresponding to cofactoring f with respect to $\bar{x}_1$), we have two alternatives: either x_2 switches from 1 to 1 and x_3 from 1 to 0, or x_2 switches from 1 to 0 and x_3 from 1 to 1. Because these transitions are not independent, for each alternative we have to use the corresponding *TC*s as shown in Figure 3.7. The same operations are performed for any other path in Π_0, thus allowing us to compute in the same manner all

the transition probabilities and hence the switching activity (with equation (3.10)). A similar approach can be further used to propagate the *TCs* between f and any other signal x.

$$p(f_{\pi\to j}) = p(x_{1_{0\to 0}})p(x_{2_{1\to 1}})p(x_{3_{1\to 0}})TC_{11,10}^{x_2x_3}TC_{01,00}^{x_1x_3}TC_{01,01}^{x_1x_2} +$$
$$p(x_{1_{0\to 0}})p(x_{2_{1\to 0}})p(x_{3_{1\to 1}})TC_{11,01}^{x_2x_3}TC_{01,01}^{x_1x_3}TC_{01,00}^{x_1x_2} +$$
$$p(x_{1_{0\to 1}})p(x_{2_{1\to 0}})p(x_{3_{1\to 0}})TC_{11,00}^{x_2x_3}TC_{01,10}^{x_1x_3}TC_{01,10}^{x_1x_2} +$$
$$p(x_{1_{0\to 1}})p(x_{2_{1\to 1}})p(x_{3_{1\to 1}})TC_{11,11}^{x_2x_3}TC_{01,11}^{x_1x_3}TC_{01,11}^{x_1x_2} +$$

$$p(f_{\pi\to j}^{\overline{x_1}}) = p(x_{2_{1\to 1}})p(x_{3_{1\to 0}})TC_{11,10}^{x_2x_3} + p(x_{2_{1\to 0}})p(x_{3_{1\to 0}})TC_{11,01}^{x_2x_3}$$

$$p(f_{\pi\to j}^{x_1}) = p(x_{2_{1\to 0}})p(x_{3_{1\to 0}})TC_{11,00}^{x_2x_3} + p(x_{2_{1\to 1}})p(x_{3_{1\to 1}})TC_{11,11}^{x_2x_3}$$

$$p(f_{\pi\to j}^{x_1\overline{x_2}}) = p(f_{\pi\to j}^{\overline{x_1}x_2}) = p(x_{3_{1\to 0}})$$

$$p(f_{\pi\to j}^{x_1x_2}) = p(f_{\pi\to j}^{\overline{x_1}\,\overline{x_2}}) = p(x_{3_{1\to 1}})$$

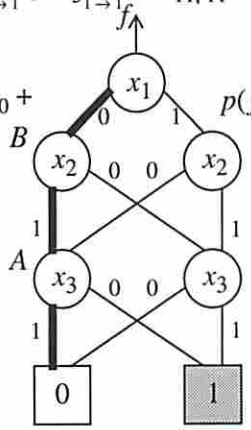


Figure 3.7 Probability calculations for $f = x_1 \oplus x_2 \oplus x_3$

3.5 An axiomatic approach to conditional probability

In this section we discuss some practical limitations regarding the mechanism described in Section 3.4. In particular, we introduce two new concepts, that is conditional independence and signal isotropy, which will help to overcome these limitations.

3.5.1 Issues in performance management

In real examples, we may have to estimate power consumption in large circuits like C6288, C7552, 32-bit multipliers, etc. where global approaches are totally impractical; in such cases, incremental approaches based on correlation coefficients are still applicable,

despite the significant amount of CPU time they need for switching activity analysis [27]. Surprisingly enough, there are other circuits, much simpler as gate count and internal structure, which raise a lot of problems in terms of the run time. In these cases, the incremental approaches need a large number of backtracks in order to compute the correlations among different signals and in some sense they "degenerate" to global approaches; that is, they tend to behave almost alike as far as the running time is concerned.

To begin with, let us consider ordinary tree circuits with k primary inputs consisting of common type gates (2-input AND, OR, XOR, etc.). At each level j ($1 < j \leq \log_2(k)$) we need to compute for each gate $(4^j - 1) / 3$ correlation coefficients, which adds up to a total of $\theta(k^2)$ calculations for the entire circuit. The running time for tree circuits is thus about 4-5 times that of non-tree circuits with the same number of gates and circuit inputs. This worst-case computation requirement is not present in non-tree circuits.

A second issue is related to the degree in which the signals are correlated. The degree of correlation is reflected in the actual values of correlation coefficients; for instance, given $TC^{xy}_{ij,kl} = 1$, $TC^{zt}_{ij,kl} = 4$ and $TC^{uv}_{ij,kl} = 256$, then we may say that the pairs (x, y), (z, t) and (u, v) are uncorrelated, slightly correlated and highly correlated, respectively. In general, large values for the correlation coefficients cause a lot of problems in the propagation mechanisms of the coefficients, due to the approximate formulae used throughout the calculations. Highly correlated signals may arise everywhere in the circuits, even starting at the primary inputs; for example, suppose we want to compute the

dot-product between two vectors x and y using the following piece of code (x and y are assumed to contain 10 random components between 0 and 1000):

```
for(i=0; i<10; i++) z=z+x[i]*y[i];
```

This instruction has been translated into assembly code as follows:

```
CPU Pentium
#TEST1#24: z=z+x[i]*y[i];
cs: 006A 8BDE    mov bx, si
cs: 006C 03DB    add bx, bx
cs: 006E 8D46EA lea ax, [bp-16]
cs: 0071 03D8    add bx, ax
:
:
cs: 0086 46      inc si
cs: 0087 83FE0A comp si, 000A
cs: 008A 7CDE    jl #TEST1#24 (006A)
```

We went deeper into details and monitored the actual values at the primary inputs of a virtual 16-bit adder (Figure 3.8a) which would perform the **add** operation (e.g. instructions at addresses 006C, 0071 and so on). We give in Figure 3.8b the values at the primary inputs of this adder only for the first iteration in this loop:

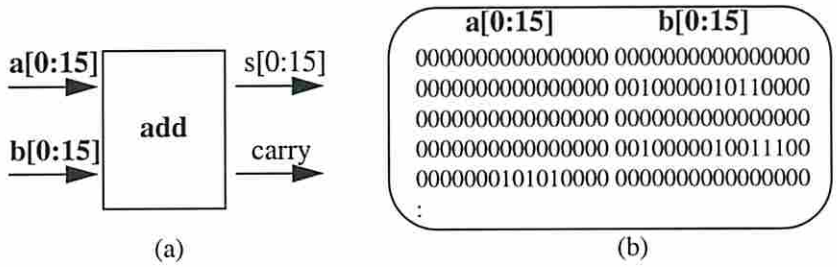


Figure 3.8 An example involving highly correlated signals

Analyzing the whole sequence coming from all 10 iterations we found that it is a highly correlated one. Assuming input independence is therefore incorrect and we give in Figure 3.9 the overestimation of switching activity per node one would make by ignoring the actual input statistics.

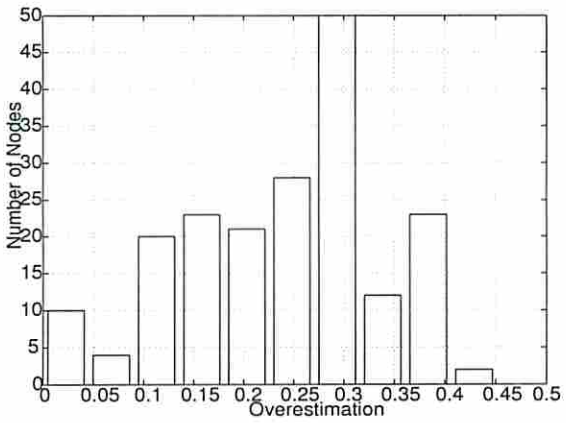


Figure 3.9 Switching activity overestimation by ignoring input statistics

As we can see, more than 70% of the nodes are significantly overestimated. This is a typical case which may arise in practice; therefore, in order to "make the common case accurate", we need a really good mechanism for controlling the error level throughout the circuit. Unfortunately, the spatiotemporal hypothesis alone does not provide a bounding value for the error.

These two limitations, namely *accuracy degradation* for highly correlated signals and *excessive running time* for tree-like circuits, stimulated us to further investigate stronger concepts able to overcome these drawbacks.

3.5.2 Conditional independence and signal isotropy

Definition 3.7 (Conditional Independence) Let (Ω, Σ, p) be a discrete probability space and let A, B and C be three events; the events A and B are *conditionally independent* with respect to C iff

$$p(A \cap B|C) = p(A|C) \cdot p(B|C) \tag{3.31}$$

The above definition may be extended to any number of digital signals as follows:

Definition 3.8 (Conditional Independence for Boolean Variables) Given the set of n signals $\{x_1, x_2,..., x_n\}$ and an index i $(1 \leq i \leq n)$, we say that the subset $\{x_1, x_2,..., x_{i-1}, x_{i+1},..., x_n\}$ is conditionally independent with respect to x_i if the following holds:

$$p\left(\bigcap_{1 \leq j \leq n,\, j \neq i} x_j \middle| x_i\right) = \prod_{1 \leq j \leq n,\, j \neq i} p(x_j | x_i) \tag{3.32}$$

Note: It should be pointed out that if the set $\{x_1, x_2,..., x_{i-1}, x_{i+1},..., x_n\}$ is conditionally independent with respect to x_i, it might not be conditionally independent with respect to $\bar{x}_i$. However, the corresponding set in which *any* variable (or subset of variables) is complemented, is still conditionally independent with respect to x_i if the conditions from Definition 3.8 are met.

Using the notion of support of a boolean function (i.e. the set of variables on which the function depends), we give the following definition:

Definition 3.9 (Logic Independence) Two boolean functions f and g are said to be *logically independent* (denoted by $f \perp g$) iff $Sup(f) \cap Sup(g) = \varnothing$; if they are not logically independent then f and g must share at least one common input variable.

Note: It can be seen from the above definition of f and g that logic independence is a *functional* notion and does not use any information about the statistics of the inputs.

For boolean functions, we give the following property:

Proposition 3.4 Let f and g be two boolean functions and f^c, g^c the cofactors of f and g with respect to a common variable c; if $f^c \perp g^c$ and the variables in their support sets are independent, then f and g are conditionally independent with respect to c that is:

$$p(f \cdot g|c) = p(f|c) \cdot p(g|c) \tag{3.33}$$

Proof: Shannon's decomposition for f and g gives $f = c\, f^c + \bar{c}\, f^{\bar{c}}$, $g = c\, g^c + \bar{c}\, g^{\bar{c}}$ respectively; consequently, $f\, g = c\, f^c\, g^c + \bar{c}\, f^{\bar{c}}\, g^{\bar{c}}$. One can calculate $p\,(f\, g \mid c)$ as:

$p\,(f\, g \mid c) = p\,(f\, g\, c) / p\,(c) = p\,(c\, f^c\, g^c) / p\,(c) = p\,(f^c\, g^c) = p\,(f^c) p\,(g^c)$

because $Sup\,(f^c) \cap Sup\,(g^c) = \varnothing$. We have also:

$p\,(f \mid c) = p\,(f\, c) / p\,(c) = p\,(c\, f^c) / p\,(c) = p\,(f^c)$

$p\,(g \mid c) = p\,(g\, c) / p\,(c) = p\,(c\, g^c) / p\,(c) = p\,(g^c)$

therefore $p\,(f \mid c)\, p\,(g \mid c) = p\,(f^c) p\,(g^c)$ and this concludes our demonstration. ■

Example 3.3 In Figure 3.10, the signals a, b are conditionally independent with respect to c; indeed, we have:

$p\,(a\, b \mid c) = p\,(a\, b\, c) / p\,(c) = p\,(x\, c\, y\, c\, c) / p\,(c) = p\,(x\, y\, c) / p\,(c) = p\,(x)\, p\,(y)$

$p\,(a \mid c)\, p\,(b \mid c) = p\,(a\, c)\, p\,(b\, c) / p^2(c) = p\,(x\, c)\, p\,(y\, c) / p^2(c) = p\,(x)\, p\,(y)$

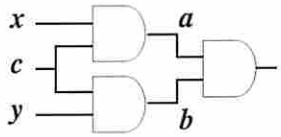


Figure 3.10 An example to illustrate conditional independence

It is worthwhile to note that, in order to compute $p\,(a\, b\, c)$ if a and b are conditionally independent with respect to c, we may use only pairwise signal probabilities. Indeed, we may deduce by simple manipulations:

$p\,(a\, b\, c) = p\,(a\, b \mid c)\, p\,(c) = p\,(a \mid c)\, p\,(b \mid c)\, p\,(c) = p\,(a\, c)\, p\,(b\, c) / p\,(c)$

which reduces the problem of evaluating the probability of three correlated signals to the one of considering only pairwise correlated signals (if the hypothesis of conditional independence is satisfied).

As a conclusion, the conditional independence concept can lead to efficient computations even in very complex situations. In fact, Proposition 3.4 gives a *sufficient* condition for conditional independence and this is very useful from a practical point of view, because all events appearing in digital logic are somehow logically correlated. However, the general problem, to determine a variable x_i from a set of n signals $\{x_1, x_2,..., x_n\}$ such that the remaining set of $(n - 1)$ signals is conditionally independent with respect to x_i is a complex problem; we will prove in the following that this is actually a **NP**-complete problem.

Theorem 3.6 (Conditional Independence Problem - CIP) Given a set of n boolean functions $\{x_1,x_2,...,x_n\}$, an index i and $k \leq n - 1$, deciding whether there are at least k signals from the remaining subset conditionally independent with respect to x_i, is a **NP**-complete problem.

Proof: First, we have to prove that CIP is in **NP**. Indeed, given a particular instance of the problem, it may be verified in polynomial time whether the requirements are met.

We will prove that CIP is **NP**-complete using a reduction from the Set Packing Problem [13]: *given a collection C of finite sets $\{S_1, S_2,..., S_n\}$, a positive integer $k \leq |C|$, deciding whether C contains at least k mutually disjoint sets is* **NP***-complete.*

Let C and k be as above, $n = |C| + 1$ and x be a boolean function such that $Sup(x) \cap S_j = \varnothing$ for every $j = 1, 2,..., n - 1$ where $S_j \in C$. We build the following boolean functions x_j, $j = 1, 2,..., n - 1$:

$$x_j = xf_j + \bar{x}g_j$$

where f_j and g_j are boolean functions such that $Sup\ (f_j) = S_j$ and g_j is an arbitrary boolean function. We can see that there is a subset of at least k signals from $x_1, x_2,..., x_{n-1}$ which are conditionally independent with respect to x iff there exists a permutation $i_1, i_2,..., i_{n-1}$ of $1, 2,..., n - 1$ such that the following is true:

$$p\left(\bigcap_{j=1}^{K} x_{i_j} \middle| x\right) = \prod_{j=1}^{K} p(x_{i_j}|x) \qquad \text{where } K \geq k.$$

Using the definition of conditional probability and the expression of x_j, we get

$$\frac{p(x \bigcap_{j=1}^{K} f_{i_j})}{p(x)} = \prod_{j=1}^{K} \frac{p(xf_{i_j})}{p(x)}.$$

The construction of x_j's was done such that x and f_j's have disjoint supports then, according to Definition 3.9, they are logically independent; equivalently, we get

$p(\bigcap_{j=1}^{K} f_{i_j}) = \prod_{j=1}^{K} p(f_{i_j})$ which is true iff f_{i_j} are logically independent i.e. their supports are mutually disjoint: $Sup(f_{i_j}) \cap Sup(f_{i_l}) = \varnothing$ or $S_{i_j} \cap S_{i_l} = \varnothing$ for any $j, l \leq K$.

Thus, the set of signals built above has at least k signals conditionally independent with respect to x iff C has at least k mutually disjoint sets. To conclude, CIP is NP-complete. ■

One may extend the notion of conditional independence with respect to a single signal to that with respect to a subset of signals. The disadvantage is that, even if we find such a set, we may not express the probability of complex events in terms of probabilities of pairs of events as it is the case with conditional independence with respect to a single signal. Thus, from a computational point of view, this does not seem to be useful. In the following, we will use instead an approximation of conditional independence which holds for correlated inputs.

Definition 3.10 (Signal Isotropy) Given the set of n signals $\{x_1, x_2,..., x_n\}$, we say that the conditional independence relation is *isotropic*, if it is true for all signals $x_1, x_2,..., x_n$; more precisely, taking out all x_i's, one at a time, the subset of the remaining $(n - 1)$ signals is conditionally independent with respect to the taken x_i.

Returning to our example in Figure 3.10, given the set of signals $\{a, b, c\}$ we have that $\{a, b\}$ is conditionally independent with respect to c, but the sets $\{a, c\}$ or $\{b, c\}$ are not conditionally independent with respect to b, or a, respectively; it follows that conditional independence is not isotropic in this particular case. Intuitively, the concept of isotropy as defined above, is restrictive by its very nature and it is hardly conceivable that a set of signals taken randomly from a target circuit will satisfy Definition 3.10. Our goal, however, is not to use this concept as it is, but to make it more practical and therefore we propose the following approximation.

Definition 3.11 (ε-Isotropy) The property of conditional independence for a set of n signals $\{x_j\}_{1 \le j \le n}$ is called ε-*isotropic* if there exists some ε ($\varepsilon \ge 0$) such that:

$$\left| \frac{\prod_{1 \le j \le n,\, j \ne i} p(x_j | x_i)}{p\left(\bigcap_{1 \le j \le n,\, j \ne i} x_j \middle| x_i \right)} - 1 \right| \le \varepsilon \text{ for any } i = 1, 2, \ldots, n \tag{3.34}$$

Differently stated, ε-isotropy is an approximation of pure isotropy within given bounds of relative error. A natural question may arise now: how often it is appropriate to consider ε-isotropy as an approximation of pure isotropy? To answer this question, we consider in Figure 3.11 several common situations involving the set of signals $\{u, v, w\}$ and the relative position of their logic cones (each cone illustrates the dependence of signals u, v, w on the primary inputs). Whilst the isotropy is completely satisfied only in (*b*), the ε-isotropy concept is applicable in all other cases; more precisely, the conditional independence relation is partially satisfied in (*a*) with respect to w, in (*c*) with respect to u and v and in (*d*) with respect to u and v.

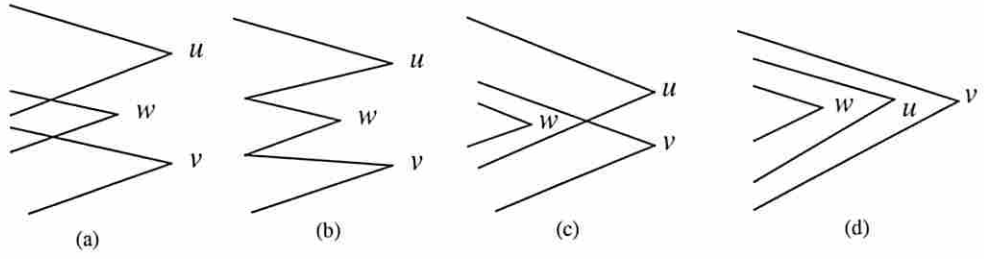


Figure 3.11 An example to illustrate pure and ε-isotropy

Based on the previous definition, we get the following:

Theorem 3.7 Given an ε-isotropic set of signals $\{x_j\}_{1 \le j \le n}$, the probability of the composed signal $p(\bigcap_{j=1}^{n} x_j)$ can be estimated within ε-relative error as:

$$p(\bigcap_{j=1}^{n} x_j) = \frac{\left(\prod_{1 \le i < j \le n} p(x_i x_j)\right)^{\frac{2}{n}}}{\left(\prod_{i=1}^{n} p(x_i)\right)^{\frac{n-2}{n}}} \tag{3.35}$$

Proof: From the definition of ε-isotropy, we get ($\forall\ i = 1, 2,..., n$):

$\left| \frac{\prod_{1 \le j \le n,\ j \ne i} p(x_j | x_i)}{p(\bigcap_{1 \le j \le n,\ j \ne i} x_j | x_i)} - 1 \right| \le \varepsilon$ which may be re-written using the definition of conditional probability as:

$$1 - \varepsilon \le \frac{\frac{\prod_{1 \le j \le n,\ j \ne i} p(x_i x_j)}{p^{n-2}(x_i)}}{p(\bigcap_{j=1}^{n} x_j)} \le 1 + \varepsilon \qquad \text{for each } i = 1, 2,..., n$$

Multiplying all inequalities we get exactly the above claim. ■

This proposition provides a strong result: given that n signals are ε-isotropic, the probability of their conjunction may be estimated within ε-relative error using only the probabilities of pairs of signals, thus reducing the problem complexity from exponential to quadratic. This is similar to the concept of pairwise correlation coefficient introduced in [12] and generalized in [27]. However, the approach in [27] does not provide sufficient accuracy for highly correlated signals as we shall see later.

3.5.3 Computation of transition probabilities using ε-isotropic signals

If the ε-isotropy property is satisfied, Theorem 3.7 may be easily extended to boolean functions represented by OBDDs. Let f be a boolean function of n variables $x_1, x_2,..., x_n$

which may be defined through the ON- and OFF-sets as in Section 3.4. In the global approach, f is represented in terms of the primary inputs, while in the incremental approach it depends only on its immediate fanin variables. Based on this representation, we give the following result.

Theorem 3.8 Given f a boolean function of variables $x_1, x_2, ..., x_n$, the following hold:

a) If the set $\{x_j\}_{1 \le j \le n}$ (where each variable is either direct or complemented) is ε-isotropic, then the *signal probability* $p\ (f = i)$ with $i = 0, 1$ may be expressed within ε-relative error as:

$$p(x = i) = \sum_{\pi \in \Pi_i} \frac{\left(\prod_{1 \le k < l \le n} p(x_k = i_k \cap x_l = i_l) \right)^{\frac{2}{n}}}{\left(\prod_{k=1}^{n} p(x_k = i_k) \right)^{\frac{n-2}{n}}} \tag{3.36}$$

where i_k is the value taken by variable x_k in the cube $\pi \in \Pi_i$.

b) If the set $\{x_{j_{k \to l}}\}_{1 \le j \le n,\, k,\, l = 0, 1}$ is ε-isotropic, then the *transition probability* $p\ (f_{i \to j})$ (with $i, j = 0, 1$) may be expressed within ε-relative error as:

$$p(f_{i \to j}) = \sum_{\pi \in \Pi_i} \sum_{\pi' \in \Pi_j} \frac{\left(\prod_{1 \le k < l \le n} p(x_{k_{i_k \to j_k}} \cap x_{l_{i_l \to j_l}}) \right)^{\frac{2}{n}}}{\left(\prod_{k=1}^{n} p(x_{k_{i_k \to j_k}}) \right)^{\frac{n-2}{n}}} \tag{3.37}$$

where i_k, j_k are the values taken by the variable x_k in the cubes $\pi \in \Pi_i$. and $\pi' \in \Pi_j$.

Proof: a) We may define the event 'f takes the value i' as:

$$(f = i) = \bigcup_{\pi \in \Pi_i} \bigcap_{k=1}^{n} (x_k = i_k)$$

In the probabilistic domain, this becomes:

$$p(f = i) = \sum_{\pi \in \Pi_i} p(\bigcap_{k=1}^{n} (x_k = i_k))$$

because the paths $\pi \in \Pi_i$ are disjoint. Since the input variables are ε-isotropic, we may use Theorem 3.7 to express the probability of each cube, getting exactly equation (3.36).

b) Proof similar to case a) but considering instead the event 'f switches from i to j' expressed as:

$$f_{i \to j} = \bigcup_{\pi \in \Pi_i} \bigcup_{\pi' \in \Pi_j} \bigcap_{k=1}^{n} x_{k_{i_k \to j_k}} . \blacksquare$$

The above result may be reformulated using signal and transition correlation coefficients; it can be used in signal probability and switching activity estimation, given that the ε-isotropy conditions are met.

Corollary 3.1 Given a set of signals $\{x_j\}_{1 \le j \le n}$ as in Theorem 3.8 and a boolean function f of variables $\{x_j\}_{1 \le j \le n}$, the following hold within ε-relative error:

$$\text{a)}\ p(f = i) = \sum_{\pi \in \Pi_i} \left(\prod_{1 \le k < l \le n} SC_{i_k i_l}^{x_k x_l} \right)^{\frac{2}{n}} \prod_{k=1}^{n} p(x_k = i_k) \tag{3.38}$$

$$\text{b)}\ p(f_{i \to j}) = \sum_{\pi \in \Pi_i} \sum_{\pi' \in \Pi_j} \left(\prod_{1 \le k < l \le n} TC_{i_k i_l, j_k j_l}^{x_k x_l} \right)^{\frac{2}{n}} \prod_{k=1}^{n} p(x_{k_{i_k \to j_k}}) \tag{3.39}$$

Proof: a) and b) Using the definition of transition probability and Theorem 3.8, we easily get the above inequalities. ■

For the incremental approach, this result can be extended to the calculation of correlation coefficients (*SCs* or *TCs*) between two signals in the circuit. In practice, this

becomes an important piece in the propagation mechanism of probabilities and coefficients through the boolean network.

3.5.4 Computation of transition correlation coefficients using ε-isotropic signals

Theorem 3.9 Given a set of signals $\{x_j\}_{1 \le j \le n}$, a boolean function f of variables $\{x_j\}_{1 \le j \le n}$, and x a signal from the circuit, if $\{x_1, x_2, ..., x_n, x\}$ is a set as in Theorem 3.8, then the correlation coefficients (*SCs* and *TCs*) can be expressed within ε-relative error[1] as:

$$\text{a) } SC_{ij}^{fx} = \frac{\sum_{\pi \in \Pi_i} \left(\prod_{1 \le k < l \le n} SC_{i_k i_l}^{x_k x_l} \right)^{\frac{2}{n+1}} \prod_{k=1}^{n} \left(p(x_k = i_k)(SC_{i_k j}^{x_k x})^{\frac{2}{n+1}} \right)}{p(f = i)} \tag{3.40}$$

$$\text{b) } TC_{ip,\,jq}^{fx} = \frac{\sum_{\pi \in \Pi_i} \sum_{\pi' \in \Pi_j} \left(\prod_{1 \le k < l \le n} TC_{i_k i_l,\, j_k j_l}^{x_k x_l} \right)^{\frac{2}{n+1}} \prod_{k=1}^{n} \left(p(x_{k_{i_k \to j_k}})(TC_{i_k p,\, j_k q}^{x_k x})^{\frac{2}{n+1}} \right)}{p(f_{i \to j})}$$

where $i, j, p, q = 0, 1$.

Proof: a) and b) follow directly from the definition of *SC*s and *TCs* and using the events '$f = i$ and $x = j$ simultaneous' and 'f switches from i to j and x from p to q simultaneously', respectively. ■

These results lead to a new heuristic algorithm for signal and transition probability estimation under input streams which exhibit spatiotemporal correlations. We may thus see equation (3.39) as the improvement of equation (3.23) by using the notions of conditional independence and signal isotropy. Compared to the heuristic proposed in

1. This ε is the maximum over all values that occur during the incremental propagation process.

Section 3.4, this new approach based on conditional independence has also the advantage of supplying bounds for error estimation provided that the input signals are ε-isotropic. This bounding value could not be justified by using the spatiotemporal hypothesis alone. Finally, the model introduced above provides a way to improve the run time requirement as shown below.

Proposition 3.5 If C_j is a correlation coefficient (*SC* or *TC*) at level j (given by a topological order from inputs to outputs of the circuit), then it is related to C_{j-l} $(0 < l < j)$ by a proportionality relationship expressible as $C_l \propto (C_{j-l})^{\left(\frac{2}{n+1}\right)^l}$, where n represents the average fan-in value in the circuit. Moreover, if $l \to \infty$, then $(C_{j-l})^{2/(n+1)^l} \to 1$ (the signals on level *j-l* behave as uncorrelated).

Proof: First part follows from Theorem 3.9 and Corollary 3.1 if the conditions required in Theorem 3.8 are satisfied. The second part follows immediately, more precisely, $(C_{j-l})^{\left(\frac{2}{n+1}\right)^l} \to 1$ when $l \to \infty$ and this represents the condition of uncorrelated signals in our approach. ■

In other words, we do not need to compute the coefficients which are beyond some level l in the circuit; instead, we may assume them equal to 1 without decreasing the level of accuracy. Also, *the larger the average fanin n of the circuit, the smaller value for l* may be used. It is worthwhile to note that the conditional independence relationship, more specifically the concept of ε-isotropy, is essential for this conclusion. The approach based on spatiotemporal correlations *only,* does not provide a sufficient rationale for such

a limitation. This is actually a very important heuristic to use in practice and its impact on run time is huge; limiting the number of calculations for each node in the boolean network to a fixed amount (which depends on the value set as threshold for *l*) reduces the problem of coefficients estimation from *quadratic* to *linear* complexity.

3.6 Practical considerations and experimental results

All experiments are performed using the SIS [42] environment on an Ultra SPARC 2 workstation with 64 Mbytes of memory; the working procedure is shown below.

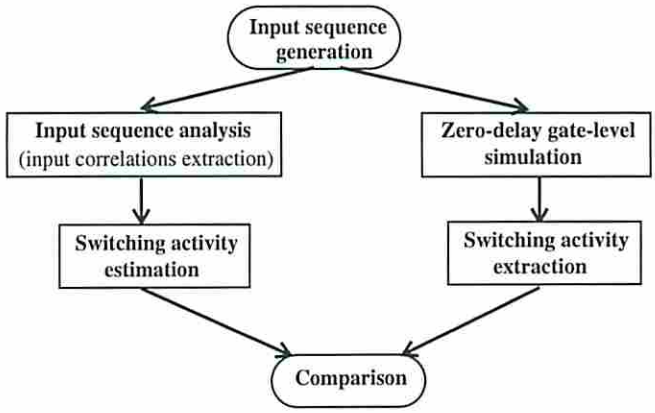


Figure 3.12 The experimental setup

To generate pseudorandom inputs we have used as input generator a maximal-length linear feedback shift register modified to include the all-zero pattern; these registers are based on primitive polynomials that is, they randomly generate all distinct patterns that correspond to a given polynomial before repeating the sequence. Purely random generators do not exist, therefore the primitive polynomials used, give us multiple correlations among primary inputs. The length of the input register was set equal to the number of inputs of the circuit under analysis, thereby creating a pseudorandom source; when the length of this register became too large, we tried to keep the time/space requirements at a reasonable level

and hence, for these cases we generated only a significant part of the exhaustive sequence (up to 2^{20} input patterns).

As the standard measure for power estimation, we have used the average switching activity at each node of the circuit calculated as in equation (3.10). In our experiments, we were mainly interested in measuring the accuracy of the model in estimating the switching activity locally (at each internal node of interest) and globally (for the entire circuit), given a set of inputs with spatiotemporal correlations. The analysis part of the experiment may be skipped if the user specifies directly the characteristics of the input stream (transition probabilities and correlation coefficients).

To illustrate the main features of our approach, we consider in Figure 3.13 the ISCAS circuit C17, fed by the sequence generated with the primitive polynomial $p(x) = 1 \oplus x \oplus x^3$. Due to the deterministic way in which we generate the input sequence, structurally independent signal lines become correlated as is the case with the inputs 1 & 2, 2 & 3, 3 & 6, 6 & 7; in turn, the fan-out points on the input lines add additional correlations. For an accurate analysis of the switching activity, we have to account for all these dependencies. In Table 3.1 we list the transition probability coefficients for this particular input sequence; we mention that in this case, all signal correlation coefficients (*SC*s) are equal to 1, therefore they cannot make a difference alone. In Table 3.2 we present the estimated and exact values of the switching activity per clock cycle. In Figure 3.13, the color code is used

to reflect the switching activity at the output of the gates, i.e. darker gates are more active (in particular, gate 23 is the most active one).

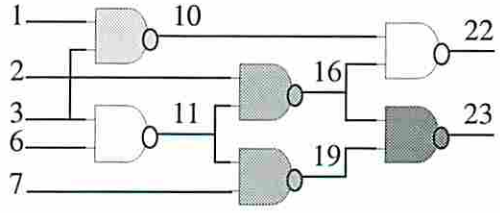


Figure 3.13 Switching activities distribution in C17 (pseudorandom inputs)

Table 3.1. C17: *TC*s for pseudorandom inputs

ij, kl	1&2	2&3	3&6	6&7
00,00	2.0000	2.0000	2.0000	1.7143
00,01	0.0000	0.0000	0.0000	0.4444
00,10	2.0000	2.0000	2.0000	1.7778
00,11	0.0000	0.0000	0.0000	0.0000
01,00	2.0000	2.0000	2.0000	2.2857
01,01	0.0000	0.0000	0.0000	0.0000
01,10	2.0000	2.0000	2.0000	1.7778
01,11	0.0000	0.0000	0.0000	0.0000
10,00	0.0000	0.0000	0.0000	0.0000
10,01	2.0000	2.0000	2.0000	1.7778
10,10	0.0000	0.0000	0.0000	0.4444
10,11	2.0000	2.0000	2.0000	1.7143
11,00	0.0000	0.0000	0.0000	0.0000
11,01	2.0000	2.0000	2.0000	1.7778
11,10	0.0000	0.0000	0.0000	0.0000
11,11	2.0000	2.0000	2.0000	2.2857

Table 3.2. C17: *sw_act* for pseudorandom inputs

Node	Estimated sw_act	Exact sw_act
1	0.5000	0.5000
2	0.5000	0.5000
3	0.5000	0.5000
6	0.5000	0.5000
7	0.5625	0.5625
10	0.3750	0.3907
11	0.2500	0.2649
16	0.5000	0.5236
19	0.5687	0.5236
22	0.2978	0.3125
23	0.6006	0.5625

It should be pointed out that the actual values for all coefficients in Table 3.1 represent the characteristics of the input stream; if we select another primitive polynomial to generate the inputs, we may obtain a completely different set of transition correlation coefficients. This dependency is even more salient if we consider 'biased inputs' (i.e. the switching activity is not 0.5). To generate such sequences, we used a simple functional generator

based on the *random* function in C language. For each bit, we set up a specific threshold $t \in [0,1]$ and generated a set of random numbers in [0,1]. If these numbers exceeded the threshold t, then the output of the generator is set to 1; otherwise the output is 0. We give in Table 3.3 ($t = 0.25$ for all bits) and Table 3.4 ($t \in [0.1, 0.5]$), the values obtained for two such sequences, and in Figure 3.14 the new distribution of switching activity among the internal nodes of the circuit. As we can see, gates 22 and 19 become in turn the most active gates in these two experiments. We note that in these cases we have spatial dependencies among all primary inputs.

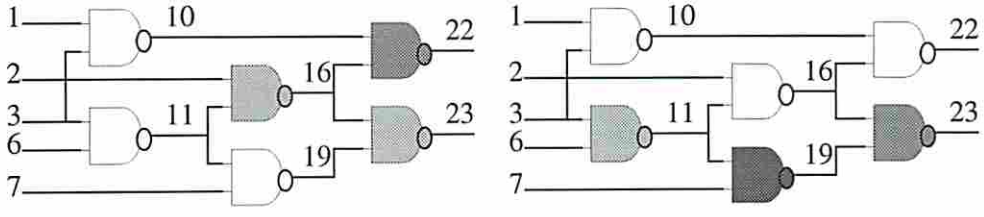


Figure 3.14 Switching activities distribution in C17 for two biased input sequences

Table 3.3. C17: sw_act for biased inputs

Node	Estimated sw_act	Exact sw_act
1	0.5625	0.5625
2	0.3750	0.3750
3	0.3125	0.3125
6	0.5625	0.5625
7	0.5000	0.5000
10	0.3125	0.3125
11	0.3125	0.3125
16	0.4569	0.5000
19	0.3367	0.3125
22	0.4960	0.5000
23	0.4527	0.4375

Table 3.4. C17: sw_act for biased inputs

Node	Estimated sw_act	Exact sw_act
1	0.0625	0.0625
2	0.1250	0.1250
3	0.2500	0.2500
6	0.5000	0.5000
7	1.0000	1.0000
10	0.1250	0.1250
11	0.2500	0.2500
16	0.1295	0.1250
19	0.7262	0.7500
22	0.2040	0.1250
23	0.4597	0.5000

To further assess the impact of correlations, we considered the benchmark *f51m* and the following two scenarios:

a) *Low Correlations:* the input patterns are generated by a linear feedback shift register which implements the primitive polynomial: $p(x) = 1 \oplus x \oplus x^2 \oplus x^7 \oplus x^8$.

b) *High Correlations:* the input patterns are generated using the state lines of an 8-bit counter.

In order to do a fair comparison between the existing estimation techniques (including the ones which use global OBDDs) and our technique, we had to choose a small sized circuit.

The estimated values in both cases were compared against the exact values of switching activity obtained by exhaustive simulation; all internal nodes and primary outputs have been taken into consideration (see Figure 3.15)[1].

1. In Figure 3.15, on the x axis, we report the absolute error of switching activity that is, $|sw_{exact} - sw_{estimated}|$.

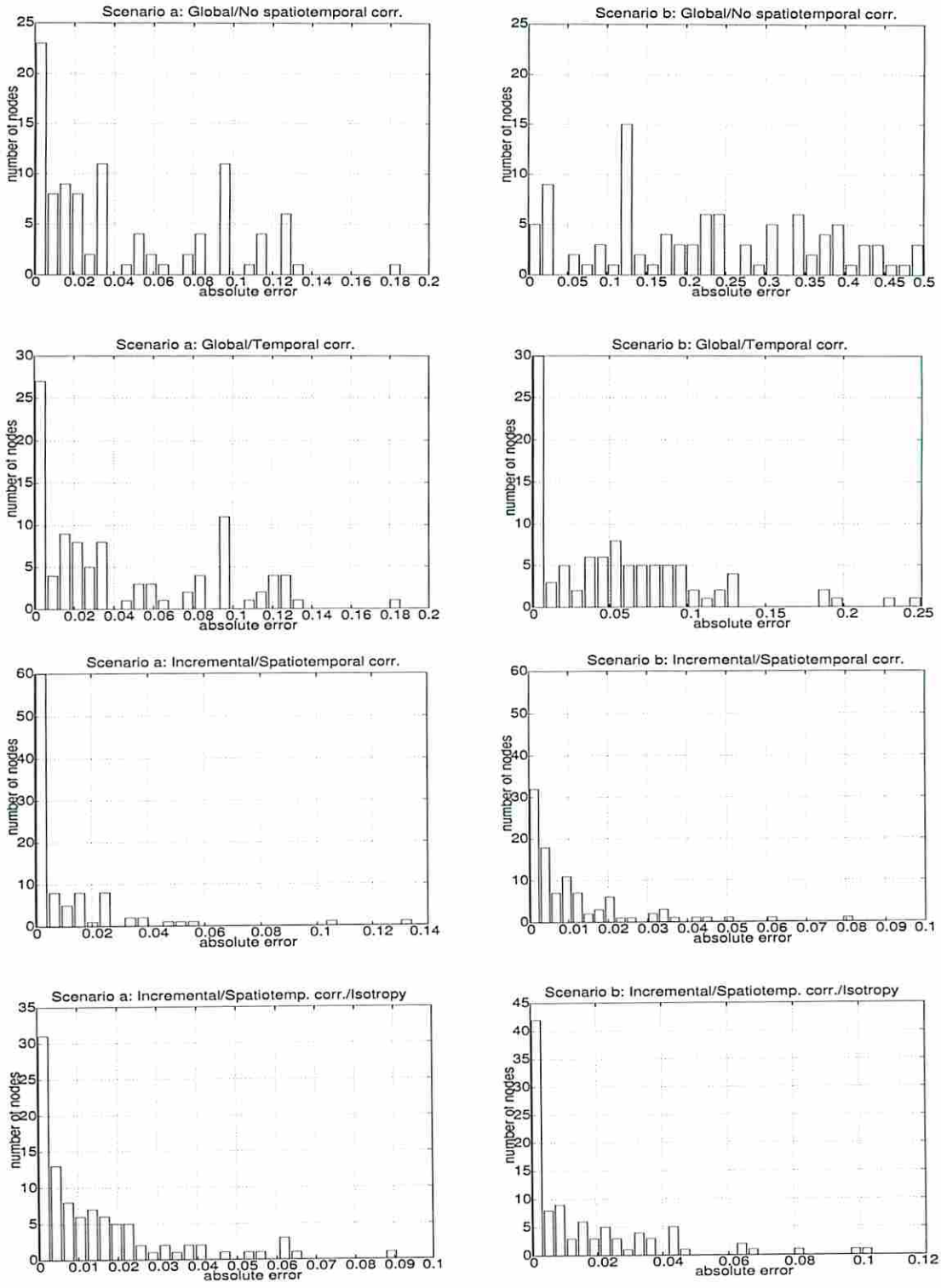


Figure 3.15 The impact of the correlation level on *sw_act* estimation in *f51m*

In Scenario (a), all approaches are quite accurate but we point that only considering spatiotemporal correlations and signal isotropy ensures the highest accuracy (100% of the nodes estimated with error less than 0.1). As the results of Scenario (b) show, the level of correlation on the primary inputs strongly impacts the quality of estimation. Specifically, it makes completely inaccurate the global approach based on input independence (despite the fact that internal dependencies due to reconvergent fan-out are accounted by building the global OBDD). As expected, this is visible mostly in Scenario (b), where less than 20% of the nodes are estimated with precision higher than 0.1. On the other hand, even if temporal correlations are taken into account, but the inputs are assumed to be spatially uncorrelated, only 80% of the nodes are estimated with error less than 0.1. Accounting for spatiotemporal correlations provides excellent results for highly correlated inputs (100% nodes estimated with precision 0.1), but the mean error in the hypothesis of conditional independence is anyway smaller (90% of the nodes are estimated with error less than 0.05).

These results clearly demonstrate that power estimation is a strongly pattern dependent problem, therefore accounting for dependencies (at the primary inputs and internally, among the different signal lines) is mandatory if accuracy is important. From this perspective, accounting for spatiotemporal correlations and using the conditional independence and signal isotropy concepts seems to be the best candidate to date.

Using some ISCAS'85 benchmarks, we further performed the following experiments:

a) One set of experiments to validate the model based only on spatiotemporal correlations without conditional independence.

b) Another one to assess the impact of the limiting technique from Proposition 3.5.

c) The last one to validate the model based on spatiotemporal correlations with conditional independence and signal isotropy.

Once again, the switching activity values and power consumption were estimated at *each* internal node and primary output and compared with the ones obtained from logic simulation. We found that power estimation for the entire circuit is not a real measure to use in low-power design and power optimization where the switching activity at *each* node has to be accurately estimated.

a) Experiments to validate the model based only on spatiotemporal correlations

In the following, we give the error values for pseudorandom inputs, using the incremental approach without conditional independence. In reporting the error, we compared our switching activity estimates with the results of logic simulation at every internal node and primary output. All benchmarks were mapped with *lib2genlib*.

Table 3.5. Pseudorandom inputs; no limit[a] ($l \rightarrow \infty$) in *TC*s calculation

Circuit	MAX	MEAN	RMS	STD	Time (sec.)
C17	0.0565	0.0119	0.0238	0.0226	0.04
C432	0.0716	0.0133	0.0222	0.0179	30.17
C499	0.0334	0.0047	0.0072	0.0055	88.17
C880	0.1131	0.0158	0.0306	0.0264	45.38
C1355	0.0393	0.0027	0.0051	0.0044	61.76
C1908	0.0353	0.0044	0.0082	0.0069	67.15
C3540	0.1765	0.0276	0.0429	0.0318	491.19
C6288	0.2046	0.0204	0.0443	0.0396	616.22

a. l is the limit used in Proposition 3.5.

b) Experiments concerning run time improvement

Heuristic proposed in Section 3.5.3 is important in practice not only for substantially reducing the run time but also for keeping the same level of accuracy as the case when the threshold limit is set to infinity (that is, we did not use any limitation in *TC*s calculation). In the following, we present a detailed analysis for the benchmark *duke2* which exhibits a typical behavior; in the first case the limit was set to infinity, in the second one the limit was 4. To report error, we used standard measures for accuracy: maximum error (MAX), mean error (MEAN), root-mean square (RMS) and standard deviation (STD); we excluded deliberately the relative error from this picture, due to its misleading prognostic for small values.

Table 3.6. *duke2* - Speed-up vs. accuracy

	Low Correlations		High Correlations	
Error	NO limit ($l \rightarrow \infty$)	Limit $l = 4$	NO limit ($l \rightarrow \infty$)	Limit $l = 4$
MAX	0.0744	0.0710	0.0299	0.0299
MEAN	0.0133	0.0161	0.0056	0.0055
RMS	0.0223	0.0269	0.0085	0.0083
STD	0.0182	0.0219	0.0065	0.0063
Time (sec)	131.08	40.83	130.55	42.14

As we can see, the quality of estimation is practically the same in both cases whilst the run time is significantly reduced in the second approach. It should be pointed out, that this limiting technique works fine also for partitioned circuits which is an essential feature in hierarchical analysis for power estimation. Running extensively our estimation

tool on circuits of various sizes and types (ISCAS'85 benchmarks, adders, multipliers), we observed the following general tendency for the speed-up.

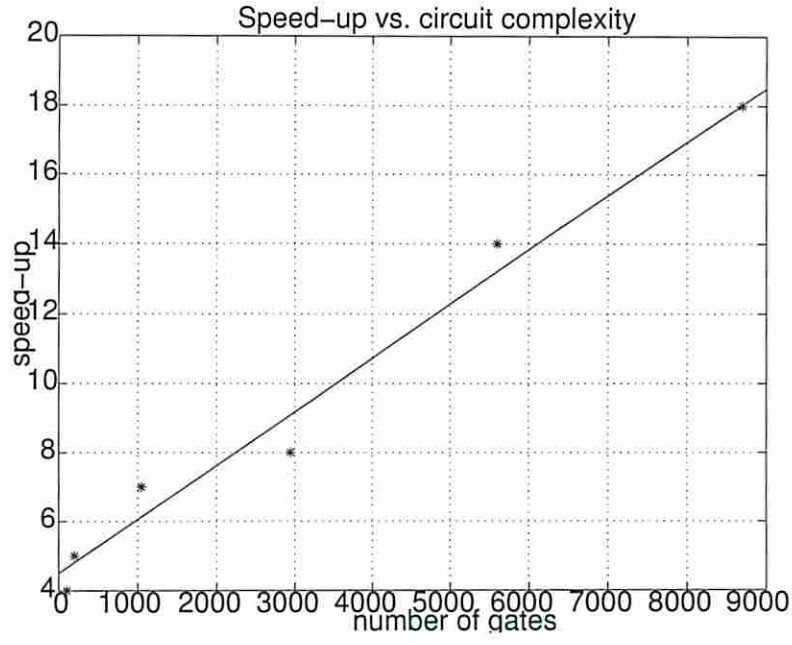


Figure 3.16 Speed-up factor vs. circuit complexity

We can see that the speed-up is about $3 \div 5$ times for less complex circuits, but it may become $20 \div 30$ times for large examples; we estimated the power consumption for multipliers on 16 bits (2048 gates) and 32 bits (9124 gates) and the run times were 43.90 and 195.13 sec.

We present in Table 3.7 the results obtained for the set of ISCAS'85 benchmarks using the limit $l = 4$ in *TC*s calculation. By comparing these results with those given in Table 3.5, we can see that the quality of the estimates remained basically the same while the run time was significantly improved.

Table 3.7. Pseudorandom inputs; $l = 4$ in TCs calculation (μW @ 20MHz; V_{DD}=5V)

Circuit	MAX	MEAN	RMS	STD	Total Power	Time (sec.)
C432	0.1916	0.0281	0.0465	0.0374	3372.57	11.82
C499	0.0624	0.0134	0.0184	0.0126	7645.56	10.57
C880	0.0691	0.0135	0.0211	0.0164	6391.83	18.35
C1355	0.0225	0.0041	0.0051	0.0030	6797.92	4.24
C1908	0.1315	0.0091	0.0206	0.0185	7435.34	12.69
C3540	0.2010	0.0307	0.0509	0.0407	16356.82	60.86
C6288	0.0890	0.0142	0.0241	0.0196	46846.48	29.10

c) Experiments to validate the conditional independence and signal isotropy

The experiments were performed on large ISCAS'85 examples using pseudorandom and highly correlated inputs (obtained from counted sequences of length 2^{20}); all results reported here, have been derived using the value $l = 4$ as the limit for coefficients calculations. To report error, all estimations were verified against exhaustive simulation performed with SIS logic simulator. To show the impact of using signal isotropy concept, for high-correlation scenario, we also present in Table 3.8 the results obtained if signal isotropy is not used.

Table 3.8. High-correlations on primary inputs (μW @ 20MHz; V_{DD}=5V)

	WITH conditional independence (ε-isotropy)					WITHOUT conditional independence				
Circuit	MAX	MEAN	RMS	STD	Total power	MAX	MEAN	RMS	STD	Total power
C432	0.2538	0.0225	0.0585	0.0545	306.88	0.8499	0.1058	0.2274	0.2032	1390.07
C499	0.1566	0.0421	0.0760	0.0634	2283.03	0.4254	0.0387	0.0933	0.0851	2543.99
C880	0.0175	0.0013	0.0040	0.0038	263.13	0.7853	0.0471	0.1630	0.1571	482.60
C1355	0.1930	0.0227	0.0520	0.0469	1865.81	0.4722	0.0516	0.1252	0.1144	1961.76
C1908	0.3907	0.0294	0.0868	0.0820	3156.83	0.4903	0.0459	0.1000	0.0892	3201.10
C3540	0.0279	0.0279	0.0030	0.0030	166.25	0.5463	0.0280	0.0365	0.0365	207.38
C6288	0.1773	0.0231	0.0521	0.0471	8843.72	0.5639	0.1092	0.1995	0.1685	19428.91

As we can see, by using conditional independence and signal isotropy, the accuracy for node-by-node analysis improves, on average, by an order of magnitude; on the other hand,

by not using conditional independence at all, the total power consumption is overestimated by 100% on average.

To further asses the impact of the correlation level, we considered the following experiment. We estimated the total power consumption for a set of ISCAS circuits using random inputs (the low correlation scenario in Table 3.9). After that, we sorted the input vectors (therefore we kept the same signal probability on the inputs) and we applied the resulting file to the same set of circuits (the medium correlation scenario). Finally, we considered as input a counted sequence with the same length and we estimated once again the total power consumption (the high correlation scenario in Table 3.9). All the values of power consumption are reported in μW at 20 MHz.

Table 3.9. Total Power Consumption (μW @ 20MHz; V_{DD}=5V)

Circuit		Low Correlations	Medium Correlations	High Correlations
Circuit 1:	C432	3372.57	2046.93	306.88
Circuit 2:	C499	7645.56	5068.32	2283.03
Circuit 3:	C880	6391.83	4781.47	263.13
Circuit 4:	C1355	6797.92	4272.74	1865.81
Circuit 5:	C1908	7435.34	4307.76	3156.83
Circuit 6:	C3540	16356.82	8464.05	166.25
Circuit 7:	C6288	46846.48	40748.45	8843.72

As we can see, there is a significant difference in all cases: not only the switching activities at each internal node were completely different as the level of inputs correlation changes, but also the values of total power consumption. For example, for C432, the total power estimated under low correlated inputs was 3372.57 μW, while this value for strongly correlated inputs was 306.88 μW (there is a factor of 11 between the two). The same behavior has been observed for other circuits. To give a more meaningful measure to this experiment, we give in Figure 3.17 the error made in medium and high correlation

scenarios by ignoring the correlations on the circuit inputs (o-medium correlations and *-high correlations).

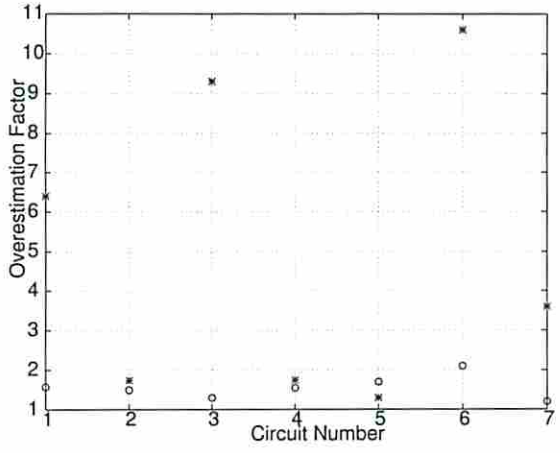


Figure 3.17 Overestimation factor by ignoring input correlations

As we can see, once again the level of correlations has a significant impact on our predictions; whilst for medium correlations one may overestimate in general about twice, for high correlations the overestimate is about 10 (e.g. Circuit 6: C3540, Circuit3: C880).

To conclude, input pattern dependencies (in particular highly correlated inputs) is an extremely important issue in power estimation. From this perspective, power analysis needs analytical models to overcome this difficulty.

3.7 Summary

In this chapter, we proposed an original approach for switching activity estimation in combinational logic modules under pseudorandom or highly biased input streams. Using the zero-delay hypothesis, we have derived a probabilistic model based on lag-one Markov Chains which supports spatiotemporal correlations among the primary inputs and internal lines of the circuit under consideration. The main feature of our approach is the

systematic way in which we can deal with complex dependencies that may appear in practice. From this perspective, the new concepts of conditional independence and signal isotropy are used in a uniform manner to fulfill practical requirements for fast and accurate power estimation. Under general assumptions, the conditional independence problem has been shown to be **NP**-complete; consequently, efficient heuristics have been provided for transition probabilities and correlation coefficients calculation. As a distinctive feature, the approach presented in this chapter improves the state-of-the-art in two ways: theoretically by providing a deep insight about the relationship between the logical and probabilistic domains, and practically by offering a sound mathematical framework and an efficient technique for power analysis.

Chapter 4 Circuit Independent Techniques: The Combinational Case

4.1 Introduction

As we have seen in the previous chapter, the key element in the process of power estimation is accurate and fast estimation of the average switching activity. *Static techniques* (like the probabilistic power estimation approach presented in Chapter 3) are in general faster, but less accurate than those based on exhaustive simulation (also called *dynamic techniques*). Usually, the input correlations and the reconvergent fan-out in the target circuit make things very complicated and simplifying assumptions (like input independence or the zero-delay model) become mandatory. On the other hand, general simulation techniques provide sufficient accuracy, but at high computational cost; it is simply expensive to simulate large circuits using millions or even thousands of input vectors and therefore, the length of the simulation sequence is another important issue that must be considered. From this perspective, the research presented in this chapter shifts the focus from the *"circuit problem"* to the *"input problem"* and proposes an original solution for power estimation based on the paradigm of *sequence compaction*. This kind of technique is appealing because it is practically independent of the actual implementation of the target circuit and it can therefore be used early in the design cycle when the structure of the circuit has not been determined yet.

The basic idea is illustrated in Figure 4.1. Given an input sequence L_0 (Figure 4.1a), to evaluate the total power consumption of a target circuit, we first derive the Markov model of the input sequence and then, having this compact representation, we generate a much

shorter sequence L, equivalent with L_0, which can be used with any available simulator to derive accurate power estimates (Figure 4.1b).

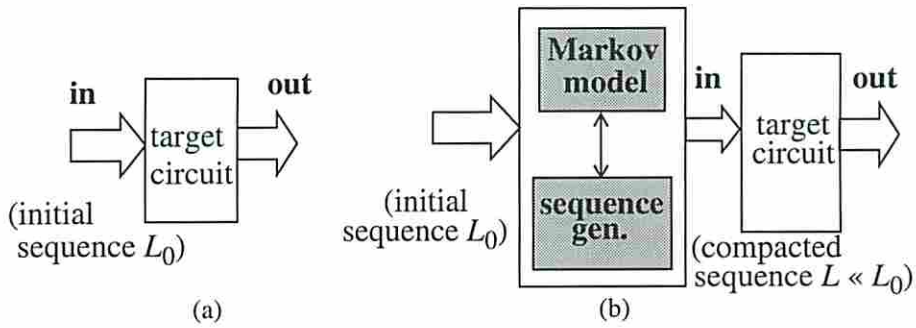


Figure 4.1 Data compaction for power estimation

The key element in this schema is the actual Markov model used to represent the initial input sequence. The construction of this Markov model relies on *adaptive* (*dynamic*) *modeling* of binary input streams as first-order Markov sources of information. Once this first-order Markov model is constructed, the sequence compaction procedure we propose subsequently is applicable to combinational and, under some circumstances, sequential circuits.

The adaptive modeling technique itself (best known as Dynamic Markov Chain or DMC modeling) was introduced recently in the literature on data compression [3], [9] as a candidate to solve various compression problems. Traditionally, data compression techniques were aimed to lossless compression, that is the ability to encode a body of data, transmit it eventually over a communication line and finally, decode it uniquely, without any loss of information during this process. Our objective in this chapter is slightly different: having an initial sequence (assumed representative for some target circuit), we target *lossy compression* [43], that is the process of transforming an input sequence into a

shorter one, such that the new body of data represents a *good approximation* as far as total power consumption is concerned.

From the very beginning, the DMC modeling technique looked very promising and indeed, in most practical situations, has been more effective than any other compression technique available to date. However, the original model introduced in [9] is not completely satisfactory for our purpose. In this chapter, we thus extend the initial formulation to manage not only correlations among adjacent bits that belong to the same input vector, but also correlations between successive input patterns.

The results presented in this chapter represent an important step toward reducing the gap between dynamic and static techniques commonly used in power estimation. We also point out that the present approach can be used without any difficulty to generate benchmark data for power estimation, that is, input sequences with different lengths that satisfy a set of user-prescribed characteristics in terms of word-level transition or conditional probabilities.

This chapter is organized as follows: first, we review the prior work relevant to our research. Section 4.3 formalizes the power-oriented vector compaction problem and discusses different parameters which makes this approach effective in practice. In Section 4.4 we introduce the basic concepts of DMC modeling technique and present a DMC-based procedure for vector compaction. To provide a better adaptability of the basic DMC modeling technique to the dynamic changes in the sequence characteristics, in Section 4.5, we introduce the hierarchical modeling of Markov chains as a flexible framework for sequence compaction. Finally, we conclude by summarizing our main contribution.

4.2 Prior work

As described in the introductory part of this chapter, a number of issues are important for power estimation and low-power synthesis: the *input statistics* which must be properly captured and the *length of the input sequences* which must be applied are two such issues. Generating a minimal-length sequence of input vectors that satisfies these statistics in not trivial. More precisely, LFSRs which have traditionally found use in testing or functional verification [2], are of little or no help here. The reason is that more elaborate set of input statistics must be preserved or reproduced during sequence generation for use by power simulators. One such attempt is [31] where authors use deterministic FSMs to model user-specified input sequences. Since the number of states in the FSM is equal to the length of the sequence to be modeled, the ability to characterize anything else but short input sequences is limited. A more elaborate and effective technique was presented in [26] where, based on stochastic sequential machines (SSMs), the authors succeed in compacting large sequences without significant loss in accuracy.

In what follows, we present a new approach based on DMC modeling. This new framework is specifically tailored for power-oriented data compaction.

4.3 A first-order probabilistic model

We use the theory of discrete-parameter time-homogeneous Markov chains to derive a probabilistic model suited for combinational circuits. As shown in Figure 4.2, we model the 'tuple' (*input_sequence, target_circuit*) by the 'tuple' (*Markov_chain, target_circuit*),

where *Markov_chain* represents the Markov chain that models the *input_sequence* and *target_circuit* is the combinational circuit where power consumption has to be determined.

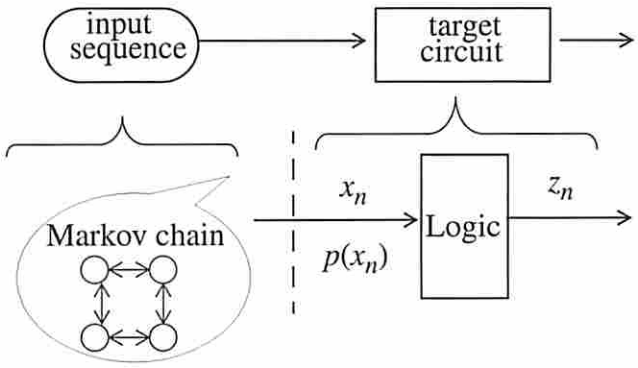


Figure 4.2 The tuple (*Markov-Chain, target_circuit*)

Let x_n denote a random variable associated to primary inputs of the circuit shown in Figure 4.2; $p(x_n)$ is then the probability that the input is x_n at time step n. We are interested in defining the probabilities $p(x_n)$ and $p(x_n x_{n-1})$ because, as we can see in Figure 4.2, they completely capture the characteristics of the input data that feeds the target circuit.

4.3.1 Problem formulation

Using the probabilities $p(x_n)$ and $p(x_n x_{n-1})$ in Figure 4.2, the vector compaction problem can be formulated as follows: given a sequence of length L_0, find another sequence of length $L < L_0$ (consisting of a subset of the initial sequence), such that the a*verage transition probability* on the primary inputs is preserved *wordwise,* for two consecutive time steps. More formally, the following condition should be satisfied:

$$\left| p(x_n x_{n-1}) - p^*(x_n x_{n-1}) \right| < \varepsilon \tag{4.1}$$

where p and p^* are the probabilities in the original and compacted sequences, respectively. ❑

This condition simply requires that the joint transition probability for the primary inputs is preserved within a given level of error, for any two consecutive vectors. We want to prove now that indeed, by having satisfied equation (4.1), it is guaranteed to produce a new sequence which is asymptotically close to the original one.

4.3.2 Proof of correctness

As usual, Q represents the stochastic matrix associated to the original input sequence, i.e. $p_{ij} = p(v_j|v_i)$, where v_i, v_j are any two consecutive vectors in the initial sequence. To produce an equivalent sequence from a reference one, one should preserve the word-level transition probabilities. This essentially becomes the problem of preserving both conditional and state probabilities because $p_{i \to j} = \pi_i \cdot p_{ij}$, where π_i is the i'th component of the state probability vector (that is, $p(s_i)$) and $p_{i \to j}$ represents the transition probability of going from vector v_i to vector v_j. (From Theorem 2.1, it can be seen that π is the left eigenvector that corresponds to the eigenvalue $\lambda = 1$ in the general equation $\pi \cdot Q = \lambda \cdot \pi$).

To complete the proof, we note that every stochastic matrix has 1 as simple eigenvalue and all other eigenvalues have absolute values less than one. This is in fact a consequence of the Perron-Frobenius theorem [41] which states that for every matrix with non-negative entries, there exists a simple[1], positive eigenvalue greater than the absolute value of any

1. This means that the multiplicity of the root $\lambda = 1$ in the equation $\pi \cdot Q = \lambda \cdot \pi$ is one.

other eigenvalue. This result is very important because it makes it possible to analyze the effect of perturbation of matrix Q on the eigenvectors that correspond to the eigenvalue 1. To this end, let's assume that the newly generated sequence is characterized by the matrix $Q^* = [p_{ij}^*]$ where $p_{ij}^* = p_{ij} + \varepsilon_{ij}$ (ε_{ij} represents the error induced by perturbations) and $|\varepsilon_{ij}| < 1$. We can write $Q^* = Q + \varepsilon \cdot B$ where $\varepsilon = max|\varepsilon_{ij}|$ and $b_{ij} = \frac{\varepsilon_{ij}}{\varepsilon}$. Because Q^* characterizes a sequence of vectors, it is also a stochastic matrix and therefore, as stated in Theorem 2.1, it has an eigenvalue $\lambda^* = 1$. But, from the theory of algebraic functions [48] we have that, for any eigenvector π of Q corresponding to the simple eigenvalue $\lambda = 1$, there exists an eigenvector π^* of Q^* corresponding to the simple eigenvalue $\lambda^* = 1$, such that $\|\pi - \pi^*\| = 0(\varepsilon)$ (read as '*zero of epsilon*'), where $0(\varepsilon)$ is any power series in ε (convergent for sufficiently small ε) having the form $k_1\varepsilon + k_2\varepsilon^2 + \ldots$.

As a consequence, since $|p_{ij} - p_{ij}^*| = 0(\varepsilon)$, it is easy to see that $|p_{i \to j} - p_{i \to j}^*| = 0(\varepsilon)$. ■

Summarizing, we have that:

Corollary 4.1 If the stochastic matrix Q is properly preserved during the compaction process, then the transition probabilities of the newly generated sequence are *asymptotically close* to the original ones, that is $|p_{i \to j} - p_{i \to j}^*| = 0(\varepsilon)$. ❑

Proof: Follows immediately from the above discussion. ■

We have thus proved that we can *asymptotically reproduce* an initial sequence by preserving its matrix Q. From a practical point of view, let's see the implications of the

above corollary on total power consumption in a target circuit when the original input sequence is approximated by a new one.

Corollary 4.2 If P and P^* are the values of the total power consumption which are obtained for two sequences satisfying the conditions in Corollary 4.1, then we have that $|P - P^*| = 0(\varepsilon)$. ❑

Proof: We have that $P = \frac{f_{clk}}{2} \cdot V_{DD}^2 \cdot \sum_{i,j,k} p_{i \to j} \cdot C_k \cdot n_{ij}^k$, where C_k is the output capacitance of gate k and n_{ij}^k is the number of transitions at the output of gate k when vector v_i, followed by vector v_j, is applied at the input of the circuit. Assuming that the input sequence is approximated by another input sequence such that the new set of transition probabilities satisfies $\left|p_{i \to j} - p^*_{i \to j}\right| = 0(\varepsilon)$, then the error made in the value of total power consumption is given by

$$|P - P^*| \leq \frac{f_{clk}}{2} \cdot V_{DD}^2 \cdot \sum_{i,j,k} \left|p_{i \to j} - p^*_{i \to j}\right| \cdot C_k \cdot n_{ij}^k = 0(\varepsilon) \tag{4.2}$$

■

Corollary 4.2 basically shows that, if the new sequence is asymptotically close to the original one, then the same type of asymptotic relationship holds for the total power values. We have, therefore, proved that a first-order probabilistic model is *sufficient* to perform sequence compaction for power estimation in combinational circuits. The remaining portion of this section describes how can we efficiently do compaction in practice.

4.4 Flat models

Without loss of generality, in what follows we restrict ourselves to finite binary strings, that is, finite sequences consisting only of 0's and 1's. The set of events of interest is the set S of all finite binary sequences on b bits. A particular sequence S_1 in S consists of vectors v_1, v_2,..., v_n (which may be distinct or not), each having a positive occurrence probability. Indices 1, 2,..., n represent the discrete time steps when a particular vector is applied to a target circuit.

4.4.1 Dynamic Markov models

Imposing a total ordering among bits, such a sequence may be conveniently viewed as a binary tree (called DMT_0 from *Dynamic Markov Tree of order zero*) where nodes at level j correspond to bit j ($1 \leq j \leq b$) in the original sequence; each edge that emerges from a node is labelled with a positive count (and therefore with a positive probability) that indicates how many times the substring from the root to that particular node, occurred in the original sequence. For clarity, let's consider the following example.

Example 4.1 For the following 4-bit sequence consisting of 8 non-distinct vectors: (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8) = (0000, 0001, 1001, 1100, 1001, 1100, 1001, 1100) the construction of the tree DMT_0 is shown step-by-step in Figure 4.3a.

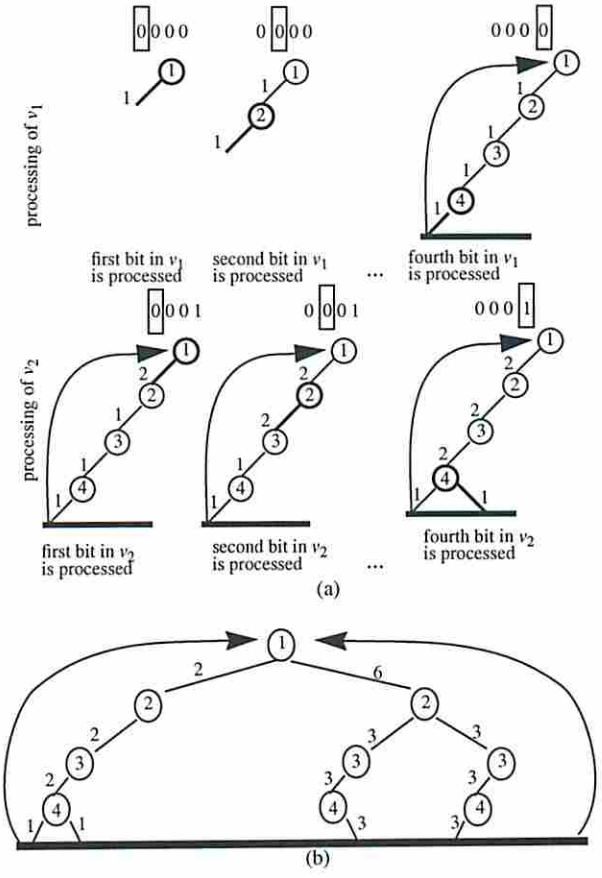


Figure 4.3 The tree DMT_0 for the sequence in Example 4.1

Obviously, the whole Markov tree that models this sequence must have four levels because the original sequence is a 4-bit sequence. Without loss of generality, we assume a left-to-right order among bits that is, the leftmost bit in any vector v_1 to v_8 is considered as being bit number one (and consequently represented at level one in DMT_0 as shown in Figure 4.3a), the next bit is considered as being bit number two and so on. Every time a vector is completely scanned (this corresponds to reaching the level four in the tree), we

come back to the root and start again with the next vector in the sequence. While the input sequence is scanned, the actual counts on the edges are dynamically updated such that, for this particular example, they finally become as indicated in Figure 4.3b.

The Markov tree in Figure 4.3b contains in a compact form all the spatial information about the original sequence v_1, v_2,..., v_8. We point out that this sparse structure is possible only by using the *dynamic (adaptive)* fashion of growing the tree DMT_0 just illustrated. Another approach would have been to consider a static binary tree capable to model any 4-bit sequence and just to update the counts on the edges while scanning the original sequence. By doing so, we would end up with the obvious disadvantage of having 15 instead of 9 nodes in the structure for the same amount of information; this reason alone is sufficient for considering from now on only dynamically grown Markov models.

Proposition 4.1 If p is a probability function from S to [0,1], then:

$$p(v) = \sum_{x \in S} p(vx) \tag{4.3}$$

for all v in S, where vx represents the event corresponding to the joint occurrence of the strings v and x.

The above condition, simply states that the sum of the counts attached to the immediate successors of node v equals its own value $p(v)$. As we can easily see in Figure 4.3, equation (4.3) is satisfied at every node in this representation[1]. In addition, based on the counts of the terminal edges, we may easily compute the probability of occurrence for a particular vector in the sequence. For instance, the probability of occurrence for string

1. This is actually similar to Kirchoff's law for currents.

'1001' is 3/8 (because the count on the terminal edge that corresponds to '1001' is 3 and the length of the sequence is 8) while the occurence probability of the string '1111' is zero, '1111' being a 'forbidden' vector for this particular sequence.

For power estimation purposes, it follows that not only a particular vector v_i in a given sequence is important, but also its relative position in that sequence matters. More precisely, different permutations of vectors belonging to the same initial set (v_1, v_2,..., v_n), define completely different input sequences. Coming back to the structure just introduced, we observe that DMT_0 alone cannot capture this property; we say that DMT_0 has *no memory* and therefore the relative order of vectors in the initial sequence is irrelevant in the construction of DMT_0. In Figure 4.3b for instance, the value of 3/8 is the probability that we find the particular string (state) '1001' in the original sequence, but this gives us no indication about the sequencing of this vector relative to another one, say '0001'.

To properly solve the vector compaction problem, we refine now the above structure by incorporating *first-order memory effects*. Specifically, we consider a more intricate structure, namely a tree called DMT_1 (*Dynamic Markov Tree of order 1*).

Example 4.2 For the same sequence in Example 4.1, suppose we want to construct its corresponding tree DMT_1. We begin as in DMT_0 and for each leaf that represents a valid combination in the original sequence, we construct a new tree (having the same depth as

DMT_0) which is meant to preserve the *context* in which the next combination occurs (Figure 4.4).

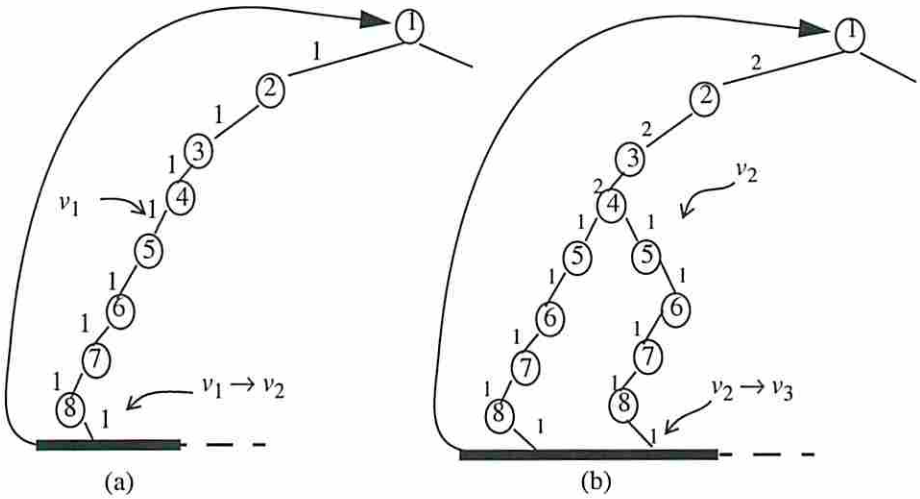


Figure 4.4 Construction of the tree DMT_1

For instance, the vector v_2 = 0001 follows immediately after v_1 = 0000; consequently when we reach the node that corresponds to v_1 (the leftmost path in Figure 4.4a), instead of going back to the root and therefore 'forgetting' the context, we start building a new tree (rooted at the current leave of DMT_0) as indicated in Figure 4.4a. The newly constructed tree will preserve the context in which v_2 = 0001 occurred that is, immediately after v_1 = 0000 (this is denoted by $v_1 \rightarrow v_2$). After processing the pair (v_1,v_2), we come back to the root and continue with the pair (v_2,v_3) as shown in Figure 4.4b.

In fact, all vectors except the first and the last are processed exactly twice, once in the upper DMT_0 and next in the lower subtree. What is important to note here is that *all* vector pairs in the original sequence are processed, that is, none of them is skipped during the construction of DMT_1. This is the theoretical basis for accurate modeling of the input

sequences as first-order Markov sources of information. Similarly, continuing this process for all leaves in DMT_0, we end up by building the whole tree DMT_1 as shown in Figure 4.5.

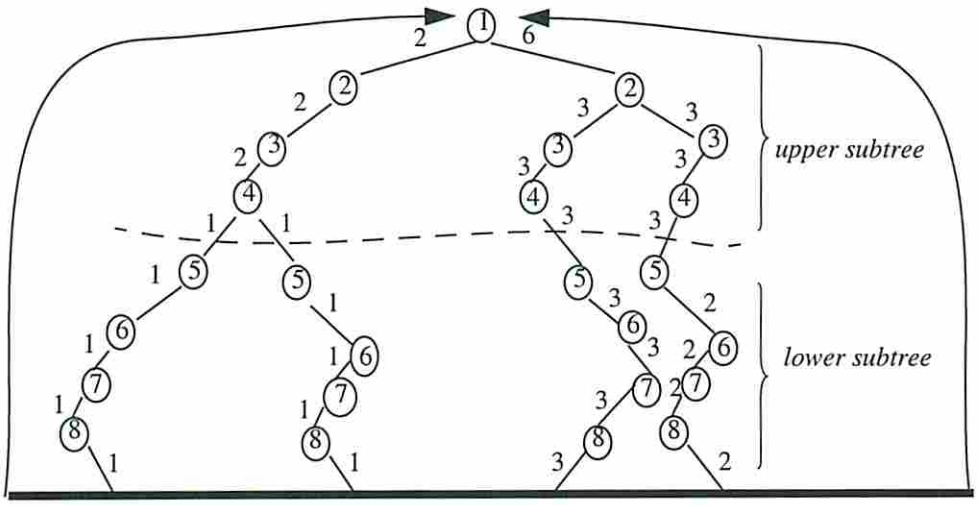


Figure 4.5 The tree DMT_1 for the sequence in Example 4.1

In Figure 4.5, the upper subtree (levels 1 to 4) represents DMT_0, that is, it sets up the state probabilities for the sequence; the lower subtrees (levels 5 to 8), give the actual sequencing between any two successive vectors. To keep the counts in these subtrees consistent, while we traverse the lower subtrees and update the counts on their edges, we also accordingly increment the counts on the paths in the upper subtree. In practice the counts of these two subtrees may differ by one (as it can be seen on the rightmost path at the border between the upper and lower subtree in Figure 4.5), due to the finite length of the sequences. A practical solution to this issue is to consider the input sequence as being cyclic.

Obviously, DMT_1 provides more information than DMT_0. To give an example, the string '1001' can follow only after '0001' or '1100', information that cannot be gathered

by analyzing DMT_0 alone. We also note that counts on the edges allow a quick calculation of the transitions probabilities that characterize any particular sequence. To this end, we use the following result:

Proposition 4.2 [35] We write the probability of a vector string $v = v_1 v_2 ... v_n$ as follows:

$$p(v) = p(v_1) \cdot p(v_2 | v_1) \cdot \ldots \cdot p(v_n | v_1 v_2 \ldots v_{n-1}) \tag{4.4}$$

where the conditional probabilities are uniquely defined by: $p(x|v) = p(vx) / p(v)$.

For example, if we want to calculate the transition probability '1001' → '1100' we have $p(v) = p(v_1 v_2) = p(v_1) \cdot p(v_2 | v_1) = 3/8$, which is exactly the count on the path '10011100' in the tree DMT_1 divided by the sequence length.

Theorem 4.1 Any sequence in S can be modeled as a first-order Markov source using the tree DMT_1 and the probability function p. We call this process *Dynamic Markov Chain* (DMC) modeling.

Proof: If $v = v_1\ v_2$ is a string in the structure DMT_1 such that v_1 is in the upper tree and v_2 is in the lower tree, then $p\ (v_2 \mid v_1) = p\ (v) / p\ (v_1)$. Thus, the parameters stored on the edges of DMT_1 structure provide the conditional probabilities that characterize the lag-one Markov chain for the sequence in S. ■

Theorem 4.2 The structure DMT_1 and parameters p are equivalent to a stochastic sequential machine.

Sketch of proof: DMT_1 defines a Markov source (based on Theorem 4.1). Any Markov source is characterized by a stochastic matrix. According to the decomposition Theorem 1

given in [26], this matrix is uniquely associated to a stochastic machine (a finite-state machine with randomly generated inputs). Thus, DMT_1 is equivalent to a SSM. ■

Generally speaking, the theory of stochastic sequential machines is far more developed than the theory of DMC modeling. However, the DMC modeling technique based on DMT_1 seems to be more effective as it offers a much more compact structure and generally outperforms the compaction techniques based on stochastic machines.

The structure DMT_1 just introduced is general enough to capture completely spatial correlations and first-order temporal correlations. Indeed, the recursive construction of DMT_1 by considering successive bits in the upper and lower subtrees completely captures the word-level (spatial) correlations for each individual input vector in the original sequence. Furthermore, cascading lower subtrees for each path in the upper subtree, gives the actual sequencing (temporal correlation) between two successive input patterns.

4.4.2 A practical procedure

A practical procedure to construct DMT_1 and generate the compacted sequence is given in Figure 4.6a. During a one-pass traversal of the original sequence (when we extract the bit-level statistics of each individual vector $v_1,v_2...,v_n$ and also those statistics that correspond to pairs of consecutive vectors (v_1v_2), $(v_2v_3),...,(v_{n-2}v_{n-1}),(v_{n-1}v_n))$, we grow simultaneously the tree DMT_1. We continue to grow DMT_1 as long as the number of nodes in the Markov model is smaller than a user-specified threshold (*model_size*), otherwise we just generate the new sequence up to that point and discard (flush) the model. A new

Markov model is started again and the process is continued up to the end of the original sequence.

```
procedure DMC (input_file, ratio, model_size) {
  initial_state = new_state ();
  symbol = read_input (input_file);
  update_tree (symbol, upper_tree, initial_state);
  crt_state = last_state (symbol, upper_tree);
  while (!EOF (input_file)) {
    symbol = read_input (input_file);
    if (number_of_states < model_size) {
      update_tree (symbol, lower_trees, crt_state);
      update_tree (symbol, upper_tree, initial_state);
      crt_state = last_state (symbol, upper_tree);
    }
    else {
      generate_seq (upper_tree, lower_trees, ratio);
      flush_model (upper_tree, lower_trees);
      initial_state = new_state ();
      symbol = read_input (input_file);
      update_tree (symbol, upper_tree, initial_state);
      crt_state = last_state (symbol, upper_tree);
    }
  }
  generate_seq (upper_tree, lower_trees, ratio);
}
```

(a)

```
procedure generate_seq (upper_tree,
lower_trees, ratio) {
  crt_symbol = generate_random ();
  lower_tree_node = last_node (upper_tree,
crt_symbol);
  upper_tree_node = root (upper_tree);
  do {
    for each bit in the current vector {
      generate '0' or '1' to maximize the decrease in
      absolute error;
      if ('0' is generated) {
        lower_tree_node = left (lower_tree_node);
        upper_tree_node = left (upper_tree_node);
      }
      else {
        lower_tree_node = right (lower_tree_node);
        upper_tree_node = right (upper_tree_node);
      }
    }
    lower_tree_node = upper_tree_node;
    upper_tree_node = root (upper_tree);
  }
  while there are still vectors to be generated;
```

(b)

Figure 4.6 The vector compaction procedure

The *generate_seq* procedure called by the DMC program is detailed in Figure 4.6b. Each generation phase is driven by the user-specified compaction parameter *ratio*; that is, in order to generate a total of $L = L_0/ratio$ vectors, we have to keep the same compaction ratio for every dynamically grown Markov model. For generation, we use a modified version of the *dynamic weighted selection algorithm* [14]. In that approach, a similar structure with DMT_0 is built; more precisely, a full tree having on the leaves the symbols that need to be generated. The counts on the edges are dynamically changed and the symbols are generated according to their probability distribution. For this, a single random number generator is required in order to divide the interval [0,1] into subintervals that

correspond to symbols' probabilities. At each level, the random number is compared to the left probability: if lower, a zero value is generated; if greater, a one value is generated and the number is decreased by the left probability. We use this strategy only to generate the first vector. After that, to ensure a minimal level of error, we use an *error controlling mechanism* in a greedy fashion. More precisely, at each level in the lower Markov tree, in order to decide whether a zero or one has to be generated, we compute the transition probabilities for both alternatives and choose the one that minimizes the absolute error accumulated up to that point. Simultaneously, the upper tree is parsed from the root to the leaves, according to the bits generated in the lower subtree. The procedure is then resumed until the needed number of vectors is generated.

Example 4.3 Assume that we are given the following 3-bit sequence consisting of 17 non-distinct vectors: $(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}, v_{16}, v_{17})$ = (001, 100, 001, 110, 111, 111, 101, 110, 011, 000, 101, 001, 100, 000, 110, 110, 011); our objective is to compact this sequence with a compaction ratio of 2.

We start building the Markov model that characterizes the initial sequence. For clarity, the construction of the tree DMT_1 is shown in Figure 4.7 and Figure 4.8 for two different scenarios. First, in Figure 4.7, we assume that the parameter *model_size* is set by the user

to the value 35; this means that the model can be grown dynamically (without any need for flushing) until this limit is reached.

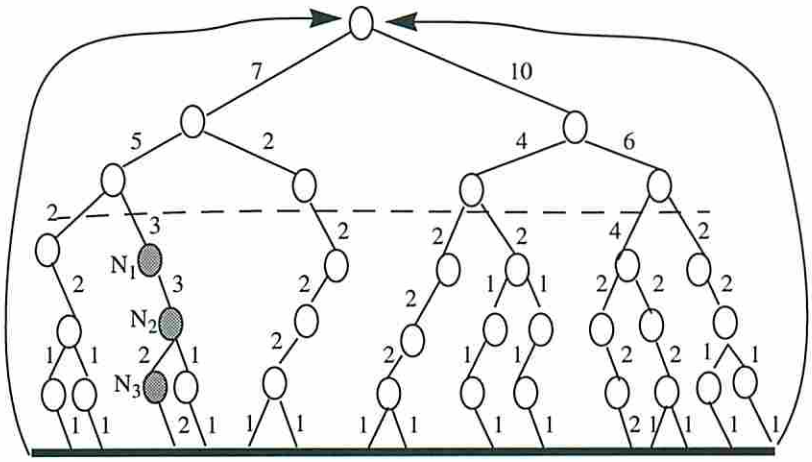


Figure 4.7 DMT_1 for sequence in Example 4.3 (*model_size* = 35)

Once we built the Markov tree in Figure 4.7, we start the procedure *generate_seq* with parameter *ratio* = 2 and generate a subset of 8 vectors which best approximate the original sequence. If we assume that $x = 0.23$ is the first randomly generated number, based on the tree in Figure 4.7, since $0.23 < 7/17$ we take the left edge, generate a value 0 and x remains unchanged. At the second level, $x = 0.23 < 5/17$ so again we generate a '0' and leave x unchanged. Now $x = 0.23 > 2/17$ so a '1' is generated and x becomes $x = 0.23 - 2/17 = 0.11$. For the lower subtree rooted at the node denoted by the vector '001' (that is, we parse the upper subtree according to the already generated bits 0, 0, 1), to produce the second vector, we use the error controlling mechanism. Specifically, at node N_1 in Figure 4.7, the only choice is to take the right edge, generating a '1'. Next, at node N_2, the absolute error made for the transition probabilities becomes $|2/17-1/8| + |1/17 - 0| = 0.066$ if we take the left edge, and $|2/17 - 0| + |1/17 - 1/8| = 0.183$ if we take the right edge (8 is the length of the sequence to be obtained). The first choice is preferred and therefore a '0'

is generated. At the last level, at node N_3, the decision is quite simple as we have only one descendent. Thus, after the first vector '001', we generate '101' as the second vector. The generation procedure continues for the lower subtree rooted at the node denoted by the vector '101' until the desired length $L = L_0/ratio$ (that is, 9 vectors) is achieved. Despite its locality, this decision strategy performs very well in practice; as we will see in the experimental part, the overall level of error is very small in all practical cases.

In the second scenario, illustrated in Figure 4.8, the *model_size* parameter is set by the user as being 30 therefore the tree in Scenario 1 cannot be grown as such because the limit of 30 nodes is reached before the whole sequence is scanned. As a consequence, once we reach this limit (this actually happens immediately after processing the subsequence v_1, v_2,..., v_9), we stop growing the tree and call *generate_seq* procedure with parameter *ratio* = 2 (Figure 4.8a). This will produce a subsequence of 4 vectors which best approximate the first 'segment' (v_1, v_2,..., v_9) of the original sequence. After that we flush the model (keeping only the very last processed vector v_9) and start a new Markov tree as shown in Figure 4.8b. When the whole sequence is exhausted, based on this new Markov tree, we generate a new subset of 4 vectors which best approximate the second 'segment' (v_9, v_{10}, v_{11},..., v_{17}) of the original sequence.

We also note that this strategy does note allow 'forbidden' vectors that is, those combinations that did not occur in the original sequence, will not appear in the final

compacted sequence either. This is an essential capability needed to avoid 'hang-up' states during simulation for power estimation.

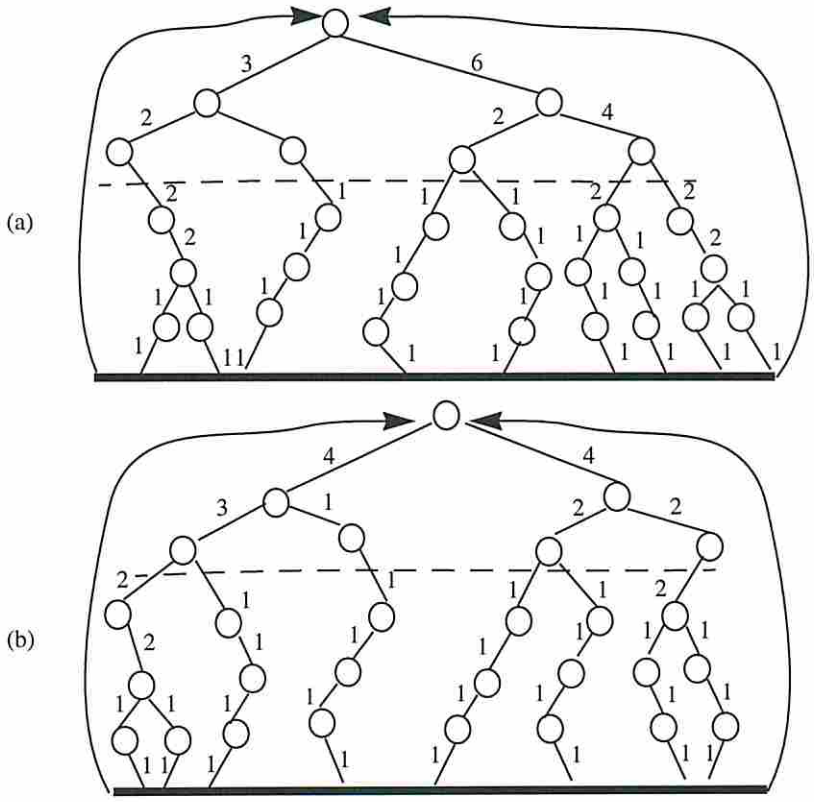


Figure 4.8 DMT_1 for sequence in Example 4.3 (*model_size* = 30)

In general, by alternating the generation and flushing phases in the DMC procedure, the complexity of the model can be effectively handled. The issue of accuracy in the context of these repeated flushes is discussed in the subsequent section.

4.4.3 Practical considerations

4.4.3.1 Complexity related issues

The DMC modeling approach offers the significant advantage of being a *one-pass adaptive technique*. As a one-pass technique, there is no requirement to save the whole sequence in the on-line computer memory. Starting with an initial empty tree DMT_1, while

the input sequence is scanned incrementally, both the set of states and the transition probabilities change dynamically making this technique highly adaptive.

Input sequences having a large number of bits b are very common in practice; the success of DMC models for sequence compaction when b is large is based on two key observations:

- The larger the value of b is, the sparser the structure of DMT_1 will be. The DMC modeling technique exploits this observation by starting with an initially empty model and dynamically growing ('on-demand') the Markov tree that characterizes the input sequence. By doing so, one can expect to build much smaller trees than the ones otherwise obtained by using a static model based on an initial full tree.
- Biased sequences which usually occurs in practice as candidates for power estimation, contain a relatively small number of distinct patterns which arise in many different contexts in the whole sequence. Therefore, a probabilistic model is ideally suited for modeling them.

A natural question still remains: when should the growing process be halted? If it is not halted, there is no bound on the amount of memory needed. On the other side, if it is completely halted we lose the ability to adapt if some characteristics of the source message change. A practical solution is to set a limit on the number of states in the DMC [9], [29]. When this limit is reached, the Markov model is flushed and a new model is started. Although this solution may appear too drastic, in practice it performs very well. The intuition behind this property is the capability of DMC model to adapt very fast to changes that occur while the input is scanned. A less extreme solution to limit model

growing is also possible; we can keep a backup buffer that retains the last p vectors emitted by the source and whenever the model should be discarded, we may reuse this information to avoid starting the new model from the scratch.

4.4.3.2 Accuracy related issues

Let us consider first the case when the number of nodes allowed in the model is sufficiently large to process the whole sequence. In this case, we may use two compaction strategies. The first is to monitor the current values of the transition probabilities and compare them with the transition probabilities of the original sequence. When the difference between the two sets of probabilities becomes sufficiently small, the generation process is halted. This way, we are able to satisfy any user specified error level for the transition probabilities. The second strategy is to set the compaction ratio upfront, perform compaction, and then compute the error induced by compaction a posteriori. In this second scenario, the user may define the largest value of the compaction ratio based on the stationarity hypothesis which should be satisfied on any segment of the original sequence that is compacted. More precisely, the shortest subsequence where the stationarity hypothesis must be satisfied limits the highest compaction ratio that can be achieved. No matter what strategy is used, if the initial sequence has the length L_0 and the newly generated sequence has the length $L < L_0$, then the outcome of this process is a compacted sequence, equivalent to the initial one as far as total power consumption is concerned; we say that a *compaction ratio* of $r = L_0/L$ was achieved.

If the number of cells allowed in the model (*model_size* parameter in DMC procedure) is too small to allow the processing of the whole sequence, then we must resort to flushing. We should note that in this case the same compaction ratio has to be used for all subsequences and thus its value has to be set upfront. To see how the flushing technique affects the accuracy, suppose that during the building of the Markov model, flushing occurs after the first n_1 vectors, then after the next n_2 vectors, and so on. If the number of flushes is f, then $n_1 + n_2 + ... + n_f = n$. We have the following result:

Theorem 4.3 If the i-th subsequence is approximated with an error less than ε_i, then the accuracy for the whole sequence is:

$$\varepsilon = (1/n) \cdot \sum_{i=1}^{f} n_i \cdot \varepsilon_i \leq max(\varepsilon_i) \tag{4.5}$$

Proof: We have that:

$$p(x_n x_{n-1}) = \frac{\sum_{i=1}^{f} n_i \cdot p_i(x_n x_{n-1})}{n} \text{ and } p^*(x_n x_{n-1}) = \frac{\sum_{i=1}^{f} \frac{n_i}{r} \cdot p_i^*(x_n x_{n-1})}{\frac{n}{r}},$$

where r is the compaction ratio and p_i (p_i^*) is the probability distribution in subsequence i that corresponds to the original (compacted) sequence. If the i-th subsequence is approximated with an error less than ε_i, then the accuracy for the whole sequence is

$$\varepsilon = \frac{\sum_{i=1}^{f} n_i \cdot \varepsilon_i}{n} \leq max(\varepsilon_i) \ . \ \blacksquare$$

Therefore, as long as the models for partial DMCs accurately capture the transition probabilities for the initial subsequences, the transition probabilities for the entire sequence are preserved up to some ε. However, the non-homogeneous sequences that may arise in practice (e.g. sequences with bi-modal distribution) can have very different transition probabilities for each subsequence. In such cases, if flushing is done properly so as to distinguish between subsequences with different transition behavior, then the overall accuracy can be significantly improved. This is a very interesting problem and it will be investigated later in this chapter.

4.4.4 Experimental results

The overall strategy is depicted in Figure 4.9. Starting with an k-bit input sequence of length n, we perform a one-pass traversal of the original sequence and simultaneously build the basic tree DMT_1; during this process, the frequency counts on edges of DMT_1 are dynamically updated.

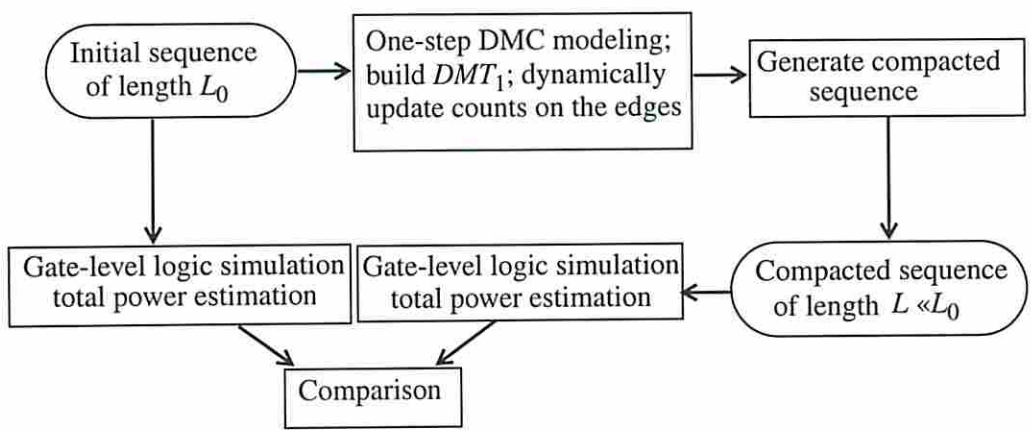


Figure 4.9 Experimental setup

The next step in Figure 4.9 does the actual generation of the output sequence. If the initial sequence has the length L_0 and the new generated sequence has the length L, then we say that a *compaction ratio* of $r = L_0/L$ was achieved.

Finally, a validation step is included in the strategy; for short sequences we used the commercial tool PowerMill [18] whilst for long sequences we resorted to an in-house gate-level logic simulator developed under SIS.

In Table 4.1 and Table 4.2, we provide the real-delay results for two types of initial sequences. Sequences of type 1 are large input streams having the same initial length L_0 =100,000 and being then prime candidates for compaction; type 1 refers to biased sequences obtained by doing bit-level logical operations on ordinary pseudorandom sequences. The sequences of type 2 (having the length of 4,000 vectors) are highly biased sequences obtained from real industry applications.

As shown in Table 4.1, sequences of type 1 were compacted with two different compaction ratios (namely r = 50 and 100); we give in this table the total power dissipation measured for the initial sequence (column 3) and for the compacted sequence (columns 4, 5). In the last column, we give the time in seconds (on a Sparc 20 workstation with 64 Mbytes of memory) necessary to read and compress data with DMC modeling. Since the compaction with DMC modeling is linear in the number of nodes in the structure DMT_1, the values reported in the last column are far less than the actual time needed to simulate the whole sequence. During these experiments, the number of states allowed in the Markov model was 20,000 (about 560 Kbytes).

Table 4.1. Total Power (μW@20MHz) for sequences of type 1

Circuit	No.of Inputs	Power for initial seq.	Estimated power for $r = 50$	Estimated power for $r = 100$	Time for DMC (sec)
C432	36	1816.32	1838.89	1779.60	42
C499	41	3697.84	3546.65	3622.26	48
C880	60	3314.07	3229.85	3329.31	75
C1355	41	3205.27	3044.20	3109.18	48
C3540	50	10876.22	9910.08	10687.32	61
C6288	32	110038.69	114199.50	109077.42	37
s344	9	751.58	748.54	719.53	10
s386	7	818.11	844.58	848.80	8
s838	34	1052.05	1061.73	1091.14	41
s1196	14	3687.47	3702.32	3580.63	16
s9234	36	9192.75	9157.31	9209.75	43
		Avg.% error	2.80	2.93	

As we can see, the quality of results is very good even when the length of the initial sequence is reduced by 2 orders of magnitude. Thus, for C432 in Table 4.1, instead of simulating 100,000 vectors with an exact power of 1816.32 μW, one can use only 2000 vectors with an estimate of 1838.89 μW or just 1000 vectors with a power consumption estimated as 1779.60 μW. This reduction in the sequence length has a significant impact on speeding-up the simulative approaches where the running time is proportional to the length of the sequence which must be simulated.

The sequences of type 2 were compacted for two compaction ratios ($r = 5$ and $r = 10$) using PowerMill; to asses the potential of efficiency of the approach, for both original and compacted sequences, we report also the actual running time required by PowerMill to provide power estimates. The number of nodes allowed for the Markov model

construction, was 5,000 (about 140 Kbytes); the CPU time for DMC modeling was below 3 seconds in all cases.

As it can be seen in Table 4.2, the average relative error is below 5% while the speed-up in power estimation is about one order of magnitude on average. For example, using the original sequence of 4000 vectors, PowerMill takes about 1186 seconds to estimate a total current of 0.4135 mA for C432. On the other side, using the sequence generated with DMC of only 400 vectors ($r = 10$), PowerMill can estimate a total current of 0.4404 mA in only 120 seconds.

Table 4.2. Total Current (mA) for sequences of type 2

		Initial sequence		Compacted sequence		
Circuit	No. of Inputs	Current (mA)	Time to simulate (sec)	Current (mA) $r = 5$	Current (mA) $r = 10$	Time to simulate (sec) $r = 10$
C432	36	0.4135	1186	0.4352	0.4404	120
C499	41	0.8188	2675	0.8337	0.8290	235
C880	60	0.7907	2289	0.8324	0.8023	274
C1355	41	1.1375	2993	1.1549	1.1461	284
C1908	33	1.2976	4034	1.2821	1.2833	367
C3540	50	3.4490	9467	4.0500	3.8580	1082
C6288	32	14.5749	88032	14.8020	15.9315	5005
			Avg. % err	4.85	4.80	

Finally, we compare our results with simple random sampling of vector pairs from the original sequences. In simple random sampling, we performed 1,000 simulation runs with 0.99 confidence level and 5% error level on each circuit[1]. We report in Table 4.3 the maximum and average number of vector pairs needed for total power values to converge

1. This means that the probability of having a relative error larger than 5% is only 1%.

[37]. We also indicate the percentage of error violations for total power values, using the thresholds of 5%, 6% and 10%. Using different seeds for the random number generator (and therefore choosing different initial states in the sequence generation phase), we run a set of 1,000 experiments for the DMC technique. In Table 4.4, we give the DMC results for the same thresholds as those used in simple random sampling.

Table 4.3. Results for Simple Random Sampling

	Number of vector pairs		Error violations		
Circ.	Max.	Avg.	> 5%	> 6%	>10%
C432	3300	2176	1.1	0.7	0.4
C499	1500	862	1.4	1.3	0.2
C880	3990	2705	1.8	0.4	0.7
C1355	1380	814	1.7	1.0	0.2
C1908	1620	846	1.9	1.3	0.2
C3540	2340	1446	2.0	1.3	0.4
C6288	7470	5422	1.4	1.4	0.3

Table 4.4. Results for DMC Approach

		Error violations		
Circ.	No. of vect.	> 5%	> 6%	>10%
C432	2000	6.7	1.9	0.0
C499	800	0.3	0.0	0.0
C880	2000	1.4	0.1	0.0
C1355	800	0.2	0.0	0.0
C1908	800	1.9	1.2	0.0
C3540	1000	0.9	0.0	0.0
C6288	2000	0.0	0.0	0.0

Once again, the results obtained with DMC modeling technique score very well and prove the robustness of the present approach. As we can see, using fewer vectors, the accuracy of DMC is higher than the one of simple random sampling in most of the cases.

4.5 Hierarchical models

In this section, we introduce the hierarchical modeling of Markov chains as a flexible framework for capturing not only complex spatiotemporal correlations, but also the dynamic changes in the sequence characteristics such as different circuit operating modes or varying power distributions.

4.5.1 Nonhomogeneous sequences

As the experimental results show, for homogeneous input sequences, the model presented so far performs very well. Large compaction ratios of few orders of magnitude can be obtained without significant loss (less than 5% on average) in the accuracy of power estimates. However, for input sequences that exhibit widely different transition behaviors over time, the overall accuracy can suffer because the probabilistic model used to represent the input sequence is a *flat model*; that is, it models the *average* behavior of the input sequence but does not adapt very well to changes in the input characteristics. For real sequences which may contain a mixture of stimuli with very different switching activities, a compaction technique with higher adaptability is obviously needed. To illustrate the significance of this issue, we consider an example.

Example 4.4 Let S_1 be a 5-bit sequence as shown in Figure 4.10a; next to it, we represent the word-level transition graph that corresponds to this sequence. Each state in this graph corresponds to a distinct pattern in the sequence and each edge represents a valid transition between any two patterns that occur in the sequence; the label of each edge captures the conditional probability of transition from the source node to the destination

node. This particular set of inputs behaves like a pseudorandom sequence because any vector is equally likely to be followed by any other remaining pattern. The total number of bit flippings in the whole sequence is 36; then, dividing this value by the sequence length, we get an average value of 3 transitions per time step.

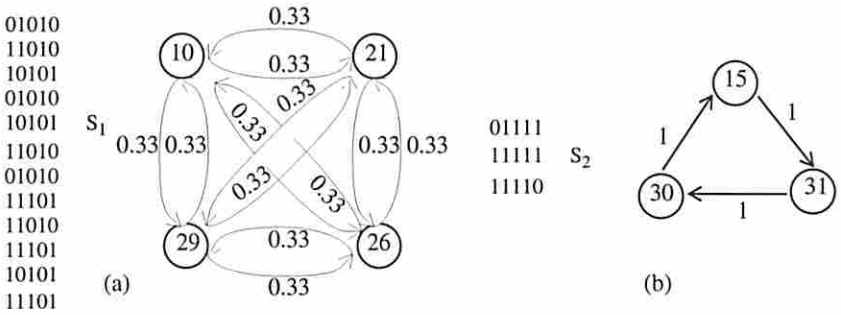


Figure 4.10 Two sequences and their corresponding transition graphs

In Figure 4.10b we consider another sequence S_2, which is completely deterministic and highly correlated. It has an average value of 1.33 transitions per step, thus producing less activity compared to S_1.

Suppose that we duplicate 25 times the original sequence S_1 and 100 times the sequence S_2, getting two new sequences S_1^* and S_2^*, respectively. Based on S_1^* and S_2^*, we construct now a new sequence S^* which is formed by concatenating S_1^* and S_2^* in the order S_1^* @ S_2^* @ S_1^*; the transition graph representation of this new 'macrosequence' is given in Figure 4.11.

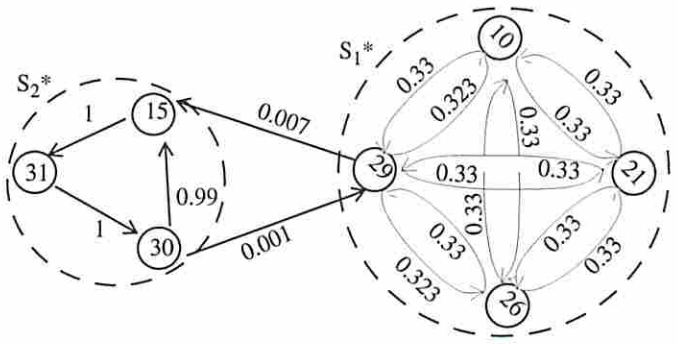


Figure 4.11 The transition graph for the composite sequence S^*

The question now becomes, how will the average power consumption look like as a function of time, when S^* is applied to any circuit? Obviously, the circuit now has two very different modes of operation: one where much activity is generated at the primary inputs, and a second one where about one single input bit toggles at every time step. In Figure 4.12 we can see the effect of these two different regimes on average power consumption for benchmark C17. Starting initially with S_1^*, after 300 time steps the value of average power stabilizes around 110μW; then, when the characteristics of the input sequence change, the power value goes down toward 70μW and finally, due to the increase of the switching activity at the primary inputs, it comes up towards 90μW.

This type of behavior is very common in practice. More precisely, only homogenous input sequences (which contain statistically similar vectors) will exercise the circuit such that the value of average power will converge rapidly. A typical example is a set of pseudorandom vectors where the average power value stabilizes after applying only tens

of vectors. However, in practical applications, the stimuli may contain a mixture of vectors, each one very different as far as average switching activity per bit is concerned.

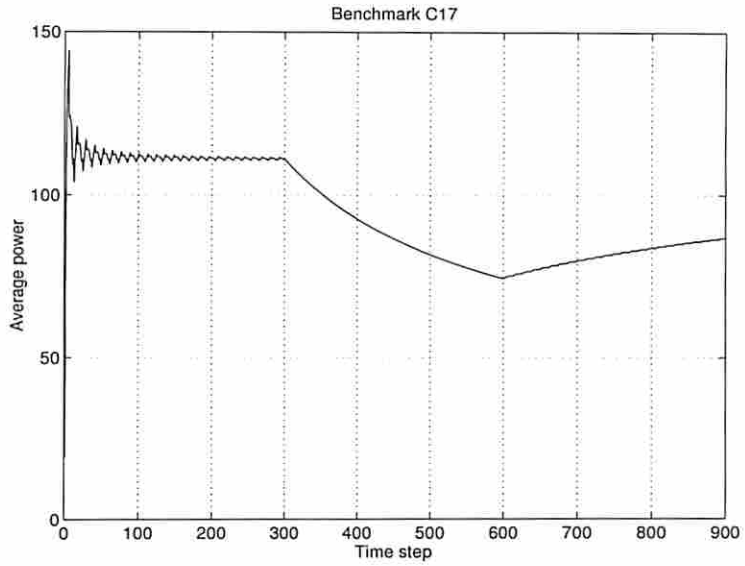


Figure 4.12 Average power dissipation for C17

A compaction procedure based on random walks in graphs where some pairs of vectors have very small transition probabilities, has the potential drawback of 'hanging' into a small subset of states. As a consequence, an erroneous power value can be obtained depending on which component (with low or high activity) is visited more often. The same type of phenomenon is observed for statistical methods when the distribution is very different from a normal one (e.g. bi-modal distributions [11]), or when one selects from the initial sequence only the first few hundred vectors. Thus, in practice, to compact large sequences that contain non-homogeneous power behaviors, a technique with high adaptability is needed.

We use hierarchical Markov models to structure the input space into a *hierarchy* of *macro-* and *micro-states*: at the first (high) level in the hierarchy we have a Markov chain

of macrostates; at the second (low) level, each macrostate is in turn characterized by a Markov chain for all its constituent microstates. Our primary motivation for this hierarchical structure is to enable a better modeling of the different stochastic levels that are present in sequences that arise in practice. Another important property of such models is to capture the different operating modes of a circuit using the first level in the hierarchical Markov model, thus providing high adaptability to different operating modes of the circuit.

After constructing the hierarchy of the input sequence, starting with some macrostate, a compaction procedure with a specified compaction ratio is applied to compact the set of microstates within that macrostate. Next, the control returns to the higher-level in the hierarchy and, based on the conditional probabilities that characterize the Markov chain at this level, a new macrostate is entered and the process repeats until the end of the sequence (last macrostate) is reached. By doing so, we combine the advantages offered by the hierarchical model with the flexibility of DMC modeling.

4.5.2 Micro/macro-state modeling

Having the transition graph associated to a vector sequence, our task now is to partition the transition graph associated with a vector sequence into subgraphs that correspond to different behaviors (in particular, different power consumptions). To do this, we provide first some useful definitions and results.

Definition 4.1 (Weighted transition graph). A *weighted transition graph* is a directed graph where any edge from state s_i to state s_j is labelled with a conditional probability $p_{ij} = p(s_j|s_i)$ *and* a weight w_{ij}[1] associated with the transition $s_i \rightarrow s_j$.

Definition 4.2 (Weight of a random walk). The *weight of a random walk* in a weighted transition graph is given by:

$$W = \sum_{s_i, s_j} p(s_i) \cdot p(s_j|s_i) \cdot w_{ij} \tag{4.6}$$

where $p(s_i)$ is the i'th component of the state probability vector and w_{ij} is the weight associated to the transition $s_i \rightarrow s_j$.

Definition 4.3 ((ε, δ)-property). A weighted transition graph is said to have the (ε, δ)-property if there exists a grouping $\{S_1, S_2,...,S_p\}$ on the set of states $\{s_1, s_2,..., s_n\}$ of the transition graph satisfying:

- (ε-criterion): $\forall\ s_i \in S_k$, $s_j \in S_l$ then $p(s_i|s_j) < \varepsilon$ and $p(s_j|s_i) < \varepsilon$;
- (δ-criterion): $\forall\ S_k$, for any two states s_i, $s_j \in S_k$, s_i, s_j connected by an edge, $\exists\ W_k$ such that $| W_k - w_{ij} | < \delta$. Also, if $k < l$, W_k's are such that $W_k < W_l$. S_k's are called *macrostates* whereas $s_i \in S_k$ are called *microstates* within the macrostate S_k.

The intuitive reason for the above definition is that the conditional probabilities from any microstate in S_k to another microstate in S_l ($S_k \neq S_l$) are negligible (ε-criterion), and all transitions among microstates belonging to the same macrostate have similar weights (δ-

1. We shall see later in this section the meaning of these weights for our particular application.

criterion). For instance, in Figure 4.11 microstates '10', '21', '26', '29' form the macrostate S_1^* (with high activity), while '15', '30', '31' form S_2^* (with low activity).

In general, a particular microstate may appear in more than one macrostate since not only the vector itself, but also the context in which it appears is important (as in Definition 4.3, the weight value for a transition determines whether the microstates belong to the macrostate or not). Therefore, the *grouping* of states is done such that the transitions are *clustered* according to their associated weights.

From what has been defined so far, it is possible to hierarchically structure the input space. Specifically, instead of considering the input sequence as a flat sequence of vectors we can see it as a structured multi-level discrete stochastic process called *Hierarchical Markov Model* (HMM). The HMM generalizes the familiar Markov chain concept by making each of its macrostates a stochastic model on its own, i.e. each macrostate is a Markov model as well. For instance, the graph in Figure 4.11 can be represented hierarchically as shown below, where the macrostate S_1^* identifies the high activity mode and S_2^* the low activity one.

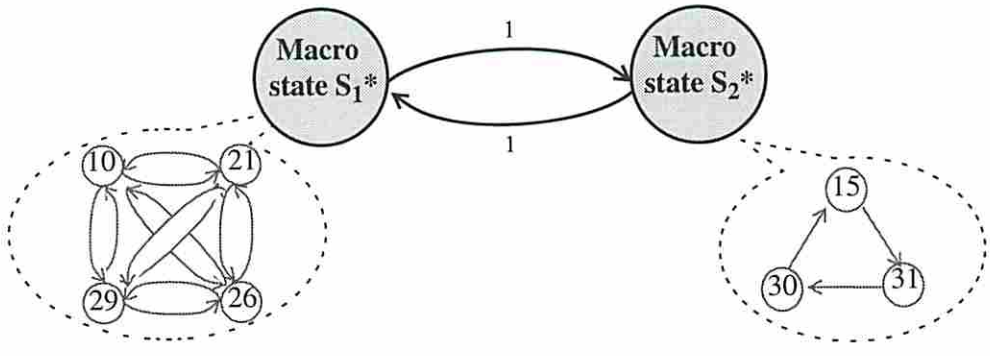


Figure 4.13 A two-level hierarchy for the sequence S*

Note: It should pointed out that, in general, the high-level Markov chain is not autonomous; that is, some conditional probabilities may be different from 1. For example, if the initial sequence can be structured as: S_1 @ S_2 @ S_3 @ S_2 @ S_1 (having thus three modes of operation), then in the high-level Markov chain we have $p(S_3|S_2) = p(S_1|S_2) = 0.5$ (because it is equally likely to go from S_2 to either S_3 or S_1).

The initial problem of compacting a flat input sequence becomes now equivalent to that of compacting a hierarchy of subsequences. Since the vectors from each macrostate are gathered using the same δ-criterion, the compaction is done now inside each macrostate. This way, the 'hang-up' problem mentioned in Section 4.5.1 is avoided, that is all macrostates are guaranteed to be visited (as their transition probabilities 'scale-up' after hierarchization). For instance, in Figure 4.11, the transition probabilities between S_1^* and S_2^* equal 0.001 and 0.007 respectively; in the hierarchical organization shown in Figure 4.13, these probabilities become 1.

We now present some useful results for HMM characterization.

Theorem 4.4 If the state probability of each macrostate and the state probabilities for all microstates within a macrostate are preserved, then the *state probability distribution* for the initial (flat) sequence is completely captured. ❑

Proof: Let s_i be any state from the original sequence. Then, its probability is given by:

$$p(s_i) = \sum_{S_k} p(s_i S_k) = \sum_{S_k} p(s_i|S_k) \cdot p(S_k) = \sum_{S_k} p_k(s_i) \cdot p(S_k) \tag{4.7}$$

where $p_k(s_i)$ denotes the state probability of s_i in macrostate S_k. ■

Thus, the state probabilities are the same if they are preserved inside each macrostate and also the probability distribution for macrostates is correctly captured.

Theorem 4.5 Having a hierarchical model satisfying the ε–criterion, if the transition probabilities of the microstates are preserved within each macrostate and if the state probabilities of the macrostates are correctly captured, then the *transition probabilities* of the initial sequence are reproduced with an error less than or equal to ε. ❑

Proof: The transition probability between two states s_i and s_j can be expressed as:

$$p(s_i s_j) = \sum_{\substack{S_k \\ s_i, s_j \in S_k}} p(s_i s_j S_k) + \sum_{\substack{S_k, S_l, k \neq l \\ s_i \in S_k, s_j \in S_l}} p(s_i s_j S_k S_l) = \sum_{S_k} p_k(s_i s_j) \cdot p(S_k) + \sum_{S_k, S_l} p_{kl}(s_i s_j) \cdot p(S_k S_l)$$

where $p_k(s_i s_j)$ is the transition probability between microstates s_i and s_j inside macrostate S_k and $p_{kl}(s_i s_j)$ denotes the transition probability between those states if they belong to macrostates S_k, S_l, respectively. Since our assumption is that the hierarchy satisfies the ε-criterion, we have that $p_{kl}(s_i s_j) < \varepsilon$ and hence:

$$p(s_i s_j) \leq \sum_{S_k} p_k(s_i s_j) \cdot p(S_k) + \varepsilon \cdot \sum_{S_k, S_l} p(S_k S_l) = \sum_{S_k} p_k(s_i s_j) \cdot p(S_k) + \varepsilon \,. \blacksquare$$

Thus, if the macrostate probability distribution and the transition probabilities inside each macrostate are preserved, then the actual transition probabilities are preserved up to some error ε.

Theorem 4.6 If the hierarchy satisfies the (ε, δ)-property (as in Definition 4.3), then the weight of a random walk in the flat model satisfies:

$$W \leq \sum_{S_k} p(S_k) \cdot W_k + \varepsilon \cdot \sum_{S_k, S_l, k \neq l} \sum_{\substack{s_i, s_j \\ s_i \in S_k, s_j \in S_l}} p(S_k S_l) \cdot w_{ij} + \delta \tag{4.8}$$

where $p(S_k)$ is the probability of being in macrostate S_k, W_k is the weight associated to macrostate S_k as in Definition 4.3 and $p(S_k S_l) = p(S_k)\, p(S_l|S_k)$ is the transition probability from macrostate S_k to macrostate S_l. ❑

Proof: Let W be the weight of the initial sequence. Then, using Definition 4.2 and Theorem 4.5, we have:

$$W \leq \sum_{s_i, s_j} \sum_{S_k} p_k(s_i s_j) \cdot p(S_k) \cdot w_{ij} + \varepsilon \cdot \sum_{\substack{s_i, s_j \\ s_i \in S_k, s_j \in S_l, k \neq l}} p(S_k S_l) \cdot w_{ij}$$

On the other hand, if s_i, s_j are in the same macrostate S_k and the hierarchy satisfies the δ–criterion, then there is some W_k such that $|w_{ij} - W_k| < \delta$. Hence, we also have:

$$W \leq \sum_{s_i, s_j} \sum_{S_k} p_k(s_i s_j) \cdot p(S_k) \cdot (W_k + \delta) + \varepsilon \cdot \sum_{\substack{s_i, s_j \\ s_i \in S_k, s_j \in S_l, k \neq l}} p(S_k S_l) \cdot w_{ij} \text{ or equivalently}$$

$$W \leq \sum_{S_k} p(S_k) \cdot W_k + \varepsilon \cdot \sum_{\substack{s_i, s_j \\ s_i \in S_k, s_j \in S_l, k \neq l}} p(S_k S_l) \cdot w_{ij} + \delta \text{ which is exactly the above claim.} \blacksquare$$

In other words, a random walk on the HMM *preserves* up to some error the average weight of the original sequence. The first term in the above sum represents the average weight per macrostate, whereas the second accounts for the weight of transitions among them.

This general formulation applies immediately to our problem defined in Section 4.3.1. In fact, if the input sequence is hierarchically structured, Theorem 4.5 guarantees that

equation (4.1) is still satisfied. Moreover, Theorem 4.6 guarantees that the average power value is maintained. Practically, this is very important because the hierarchical model has the advantage of being *highly adaptive* as opposed to a flat processing of the input which does well 'on average'.

4.5.3 A Hamming distance-based criterion for microstate grouping

In practice, it is hard to determine the weight w_{ij} for each individual transition. Specifically, an exact procedure would require detailed information about the circuit (e.g. capacitive loads, internal structure) and employ a fast simulation procedure to derive the exact power consumption for each pair of vectors from the original sequence. In practice, such an attempt may be unsatisfactory due to the simulative overhead and the requirement to have detailed circuit information. Therefore, we adopt a different, *circuit-independent* criterion to structure the input space. As suggested in Example 4.4, what we need is an indicator of the level of activity at the *primary inputs* of the circuit. To this end, we must correctly identify the different stochastic levels (subsequences) in the initial input sequence and then apply the (ε,δ)-grouping of microstates based on their associated weights. For instance, in Example 4.4, we need to correctly distinguish the two power modes which are present in the initial sequence S^*. Once we identify these different subsequences in the initial sequence S^*, our compaction procedure works fine regardless of the power consumption values that would arise from the application of these two subsequences to the target circuit. To structure the input space, we propose to use the *average Hamming distance* between two consecutive vectors because, from our

investigations, it is a reliable indicator of the level of activity. However, the framework we provide is open to other, more elaborate, weighting functions.

In the particular situation described in Example 4.4 (see Figure 4.13), based on the Hamming distance criterion, the input sequence can be roughly classified into "high activity" and "low activity" macrostates[1]. While this kind of partitioning into high and low activity modes can always be used, in practice it is better to have a more refined model. For instance, if the set of all possible values for the Hamming distance is divided in three equally-sized regions that correspond to low, medium and high activity, then we can identify more than two modes of operation. A more refined model might be required in some applications where a large number of operational modes exist (e.g. an initialization mode, an active mode, a standby mode, and a sleep mode).

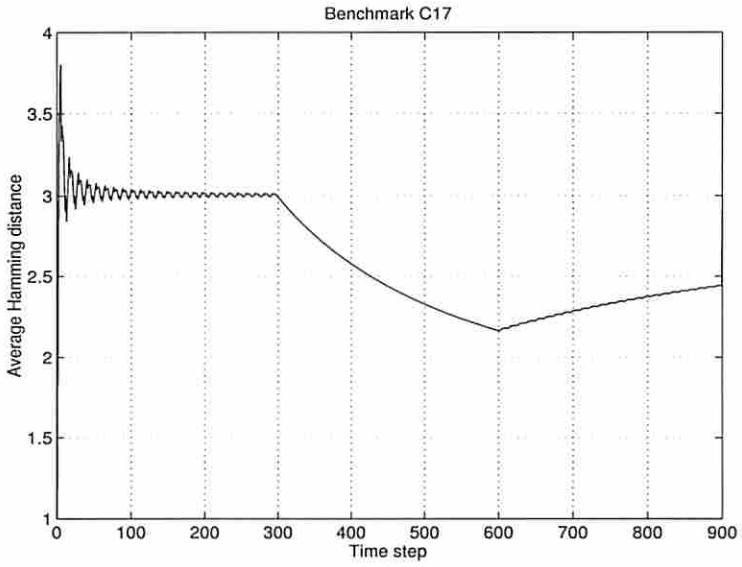


Figure 4.14 Average Hamming distance variation

We also note that the average Hamming distance may not always capture the specific behavior of different groups of bits in the input sequence. For instance, a criterion based

1. If more than 3 out of 5 bits change, we are in the high activity mode; otherwise we are in the low activity one.

on average Hamming distance may not distinguish between the two cases where either the most significant bits (MSB) are more active than the least significant bits (LSB) or vice versa. The two cases, however, induce different power modes (even if the average Hamming distance is the same). To handle this situation, we refine the high-level Markov model as follows. Instead of having only low and high activity macrostates, we define four macrostates:

- low activity LSB & low activity MSB
- low activity LSB & high activity MSB
- high activity LSB & low activity MSB
- high activity LSB & high activity MSB

and, in this way, we structure the input space to have a stronger correlation between the behavior of the input sequence and the power modes induced by this sequence in the target circuit. In the current implementation, the user has the freedom to specify the number of groups of bits to be considered, as well as the number of macrostates for each group. Thus, at the expense of a more complex model, we are able to find a better hierarchization of the input sequence which closely follows the power modes of the target circuit.

To detect the changes in the input sequence, a *variable-size sliding window* is used to compute the average value of the predictor function (in our case, the Hamming distance). At every time step, the average Hamming distance for all vectors starting with the first and ending with the current vector is computed. Next, we decide whether the behavior of the sequence has changed (that is, we are in the same macrostate or not) by comparing the average Hamming distances at steps $p \cdot k$ and $(p + 1) \cdot k$, where k is a fixed parameter

(window size) and p is an integer. If the difference between these average Hamming distances is larger than the δ parameter set by the user, then we start with a new macrostate. We note that the *size* of the chosen window should not be too small (due to the fragmentation, the high-level Markov chain becomes similar to the flat model) or too large (the low-level Markov chain becomes similar to the flat model). However, our experience shows that a window size k of 50-100 vectors works very well in practice. We note that the ε-criterion (if satisfied) is already accounted for by this procedure since all macrostates are guaranteed to be visited (due to the scaling of conditional probabilities in the high-level model). This does not result in an incorrect value for the total power consumption, as is the case for the flat model.

4.5.4 A practical procedure

We describe now a practical procedure for constructing DMT_1 and generating the compacted sequence. First, based on average Hamming distance criterion explained in Section 4.5.3, the vectors of the original sequence are assigned to macrostates. During the second traversal of the original sequence (when we extract the bit-level statistics of each individual vector and also those statistics that correspond to pairs of consecutive vectors (v_1v_2), (v_2v_3),...,$(v_{n-2}v_{n-1})$,$(v_{n-1}v_n)$), we grow simultaneously the trees DMT_1 inside each macrostate (the low-level of the hierarchy) and also the DMT_1 tree for the sequence of macrostates (the high-level of the hierarchy). Vectors within each macrostate are sequenced together in the same DMT_1. If the input sequence satisfies the (ε, δ)-property, the transitions introduced this way do not significantly change the characteristics (average

weight and transition probabilities) of the model. We continue to grow the trees at both levels of hierarchy as long as the Markov model is smaller than a user-specified threshold, otherwise we just generate the new sequence up to that point and discard (flush) the model. A new Markov model is started again and the process is continued until the original sequence is completely processed.

For the generation phase itself, we still use a modified version of the dynamic weighted selection algorithm as described in Section 4.4.2. In general, by alternating the generation and flush phases in the DMC procedure, the complexity of the model can be effectively handled. The only remaining issue is to determine how many vectors must be generated inside each macrostate before a transition to another macrostate is performed. In general, if a subsequence of length L_i is assigned to macrostate S_i, after compaction with ratio r, it has to be reduced to L_i/r. We note that inside all macrostates the *same compaction ratio* should be used, otherwise the composition of the sequence (as far as power consumption is concerned) may be totally different than that of the initial one. On average, each macrostate S_i should be visited $(p(S_i) \cdot M)$ times where M is the length of the 'macrosequence' (i.e. the length of the initial sequence of macrostates). Thus, each time a macrostate S_i is visited, a number of $L_i/(r \cdot p(S_i) \cdot M)$ vectors needs to be generated. Since the compaction is done only at the microstate level, the length of the macrosequence is preserved (the generation procedure stops when M macrostates are obtained). It is also noted that this strategy does *not* allow "forbidden" vectors which means that those

combinations that did not occur in the original sequence, will not appear in the final compacted sequence either.

4.5.5 Experimental results

The overall strategy (which is very similar to that in Figure 4.9) is depicted in Figure 4.15.

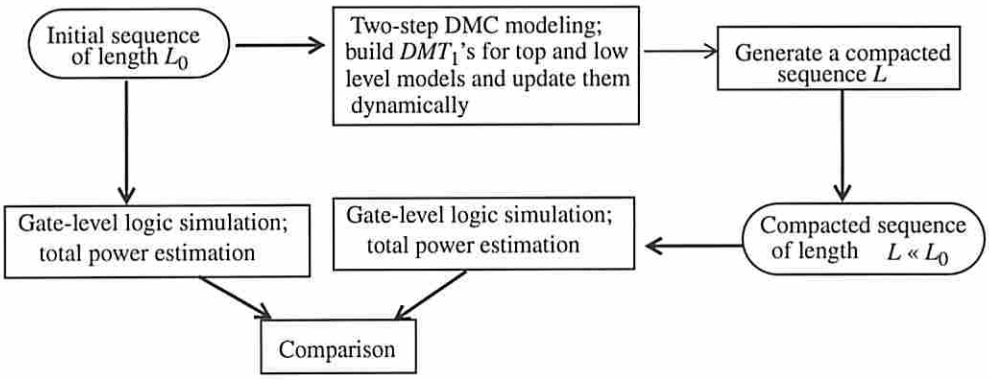


Figure 4.15 The experimental setup for combinational circuits

Starting with an input sequence of length L_0, we perform a one-pass traversal of the original sequence to assign microstates to macrostates. Next, we simultaneously build the trees DMT_1 for the entire hierarchy (macro- and microstates). During this process, the frequency counts on DMT_1's edges are dynamically updated.

Finally, a validation step is included in the strategy; we resorted to an in-house gate-level logic simulator developed under SIS. The total power consumption of some ISCAS'85 benchmarks has been measured for the initial and the compacted sequences, making it possible to assess the effectiveness of the compaction procedure.

In Table 4.5, we provide the real-delay results for a set of highly biased sequences (of length 4,000) which contains three types of sequences: a high activity sequence, followed by a low activity sequence, and finally by a pseudorandom sequence. As shown in

Table 4.5, the initial sequences were compacted with two different compaction ratios (namely $r = 5$ and 10); we give in this table the total power dissipation measured for the initial sequence (column 2) and for the compacted sequence (columns 3-6). The time in seconds (on a Sparc 20 with 64 Mbytes of memory) necessary to read and compress data with DMC modeling was below 5 seconds in all cases, for both the flat and the hierarchical models. Since compaction with DMC modeling is linear in the number of nodes in the DMT_1 structure, this value is far less than the actual time needed to simulate the whole sequence. During these experiments, the number of nodes allowed in the Markov model was 20,000 on average (around 500 Kbytes of memory). The sequences satisfied the ε-criterion for $\varepsilon = 0.001$, while the parameter δ in Definition 4.3, was set to $0.05 \cdot$(# of input bits). (This corresponds to having up to 20 macrostates in the hierarchical model.)

Table 4.5. Total power consumption (μW@20 MHz)

		Flat model		Hierarchical model	
Circ.	Exact power	Estimated power for $r = 5$	Estimated power for $r = 10$	Estimated power for $r = 5$	Estimated power for $r = 10$
C432	1810.02	1491.58	1230.77	1888.42	1906.84
C499	4390.79	5341.74	6126.58	4497.01	4591.46
C880	3788.22	4504.40	2803.14	3851.42	4006.19
C1355	3783.35	3065.71	4333.81	3910.45	3933.25
C1908	6352.07	4565.87	7094.39	6145.49	6493.44
C3540	14471.46	9005.19	3527.65	15056.43	15021.08
C6288	104158.25	81100.59	82652.47	98112.01	96295.36
	Avg.% err.	23.64	31.45	3.55	4.74

As we can see in Table 4.5, the quality of results is very good even when the length of the initial sequence is reduced by one order of magnitude. Thus, for C432 in Table 4.5,

instead of simulating 4,000 vectors with an exact power of 1810.02μW, one can use only 800 vectors with an estimate of 1888.42μW or just 400 vectors with a power consumption estimated as 1906.84μW. This reduction in the sequence length has a significant impact on speeding-up the simulative approaches where the running time is proportional to the length of the sequence which must be simulated. For instance, the average speed-up obtained for the set of benchmarks and sequences in Table 4.5 with PowerMill was 11. By comparison, if one uses the flat model (i.e. a single DMT_1 is built for the whole sequence), for the same benchmarks the relative error in power prediction is more than 20%, on average. The primary reason for this inaccuracy is the lack of adaptability which characterizes the flat model when it is applied to multi-modal sequences.

We present a typical case (Figure 4.16), that is the two-level hierarchy for benchmark C6288 in Table 4.5. In this diagram, for each macrostate, we indicate the number of microstates included in it; for instance, the largest macrostate contains 1560 microstates, while the smallest has only 80 microstates.

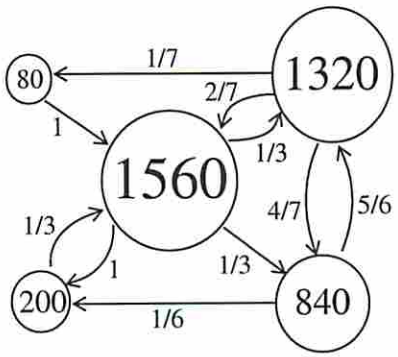


Figure 4.16 The two-level hierarchy for C6288

As we can see, for this input sequence and the particular values of parameters ε and δ, we have identified five power modes (macrostates), each one containing a very different

number of micro-states. We also note that the conditional probabilities at the highest level of the hierarchy scale-up after hierarchization.

Finally, we compare our results generated by HMM with simple random sampling of vector pairs from the original sequences. In simple random sampling, we performed 1,000 simulation runs with 0.99 confidence level and 5% error level on each circuit[1]. We report in Table 4.6 the maximum and average number of vector pairs needed for total power values to converge. We also indicate the percentage of error violations for total power values, using the thresholds of 5%, 6% and 10%. Using different seeds for the random number generator (and therefore choosing different initial states in the sequence generation phase), we run a set of 1,000 experiments for the HMM technique. In Table 4.7, we give the results obtained with the hierarchical model, for the same thresholds used in simple random sampling.

1. This means that the probability of having a relative error larger than 5% is only 1%.

Table 4.6. Results obtained for Simple Random Sampling

	Number of vector pairs		Error violations (%)		
Circuit	Max.	Avg.	> 5%	> 6%	>10%
C432	3300	2176	1.1	0.7	0.4
C499	1500	862	1.4	1.3	0.2
C880	3990	2705	1.8	0.4	0.7
C1355	1380	814	1.7	1.0	0.2
C1908	1620	846	1.9	1.3	0.2
C3540	2340	1446	2.0	1.3	0.4
C6288	7470	5422	1.4	1.4	0.3

Table 4.7. Results obtained for HMM approach

		Error violations (%)		
Circuit	Number of vectors	> 5%	> 6%	>10%
C432	800	2.6	0.6	0.0
C499	800	0.1	0.0	0.0
C880	800	2.8	0.9	0.0
C1355	800	0.1	0.0	0.0
C1908	800	0.1	0.0	0.0
C3540	1200	1.6	0.3	0.0
C6288	3600	1.5	0.0	0.0

Once again, the results obtained with the HMM modeling technique score very well and prove the robustness of the present approach. As we can see, using fewer vectors, the accuracy of HMM is higher than the one of simple random sampling in most of the cases. Noteworthy examples are benchmarks C499, C1908, C6288 where less than 60% of the maximal number of vector pairs needed in random sampling for convergence are sufficient for HMM to achieve higher accuracy and confidence levels.

4.6 Summary

The objective of this chapter was to address the vector compaction problem from a probabilistic point of view. Based on dynamic Markov Chain modeling, we proposed an original approach to compact an initial sequence into a much shorter equivalent one, which can be used after that with any available simulator to derive power estimates in the target circuit.

The mathematical foundation of this approach relies in Markov models; within this framework, a family of dynamic Markov trees is introduced and characterized as an effective and flexible way to model complex spatiotemporal correlations which occur during power estimation. A major improvement was obtained by introducing the hierarchical modeling of Markov chains as a flexible framework for capturing not only complex spatiotemporal correlations, but also the dynamic changes in the sequence characteristics such as different circuit operating modes or varying power distributions.

The experimental results show that large compaction ratios of few orders of magnitude can be obtained without significant loss in accuracy in total power estimates.

Chapter 5 Circuit Independent Techniques: The Sequential Case

5.1 Introduction

The approaches presented in Chapter 4 are applicable only to combinational circuits because they consider only first-order temporal effects (i.e. pairs of consecutive vectors) to perform sequence compaction. In the case of FSMs, this is not sufficient for accurate estimation of transition probabilities and power consumption. In this chapter, we present a solution for compacting an initial sequence of length L_0, assumed representative for the data applied to the target circuit, into a much shorter sequence L such that the steady-state and transition probabilities on the signal lines are almost the same.

Information about the steady-state and transition probabilities is very important because both of them completely characterize the behavior of the FSM. In particular, they have immediate application in power estimation area because, for sequential circuits, temporal correlations longer than one time step can affect the overall behavior of the FSM and therefore, result in very different power consumptions. This point is illustrated by a simple example.

Example 5.1 Let S_1 and S_2 be two 4-bit sequences of length 26, as shown in Figure 5.1a. These two sequences have exactly the same set of first-order temporal statistics as shown in Figure 5.1b. In this figure, we provide the wordwise transition graph for these two sequences where the topmost bit is the most significant bit (e.g. in S_1, $v_1 = v_2 =$ '1', $v_3 =$ '2',..., $v_{26} =$ '9'). Suppose now that S_1 and S_2 are fed to the benchmark *s8* from the *mcnc'91* sequential circuits suite. Looking at different internal nodes of the circuit, it can be noted that the total number of transitions made by each node is very different when the

circuit is simulated with S_1 or S_2. Moreover, the total power consumption at 20 MHz is 384μW and 476μW, respectively, showing a difference of more than 24% even for this short sequence. This raises the natural question, "why does this difference appear?", despite the fact that S_1 and S_2 have the same STG shown in Figure 5.1b.

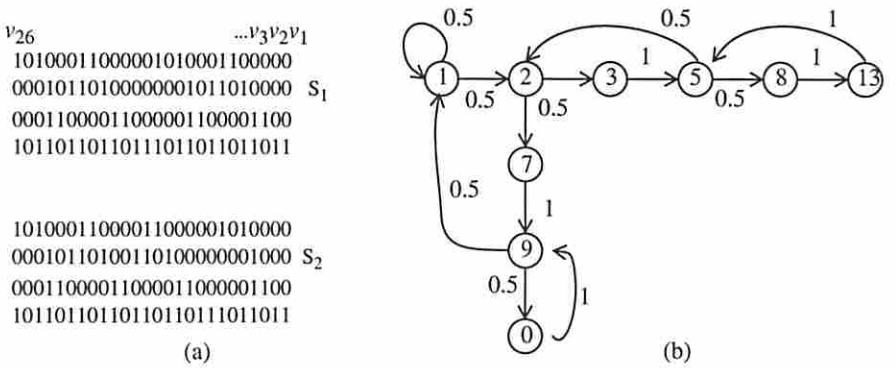


Figure 5.1 Two sequences with the same first-order statistics

The reason for this different power dissipation is that S_1 and S_2 have a different set of second-order statistics; that is, the sets of triplets (three consecutive patterns) are quite different. For instance, the triplet (1,2,7) in S_2 does not occur in S_1; the same observation applies to the triplet (5,2,3) in S_2. The conclusion is that having the same set of one-step transition probabilities *does not* imply that the sets of second-order or higher-order statistics are identical and, as was just illustrated by this small example, higher order statistics can make a significant difference in FSM's total power consumption. The initial problem of compacting an initial input sequence can be now cast in terms of power as follows: can we transform a given input sequence into a much shorter one, such that the new body of data is a good approximation of the initial sequence as far as total power consumption is concerned? In this chapter, we will answer this question by introducing a

family of variable-order dynamic Markov models that provide an effective way for accurate modeling of external input sequences that affect the behavior of the target FSM.

This chapter is organized as follows: First, we define the vector compaction problem for sequential circuits. After that, assuming the stationarity hypothesis, we present the main results on the effects of finite-order statistics on FSM behavior. Based on the concept of Markovian source of information, in Section 5.4, we present an efficient way to compact composite sequences that is, sequences generated with Markov sources of information having different orders. In Section 5.5, we give some practical considerations and present our experimental results on common sequential benchmarks. Finally, we conclude by summarizing our main contribution.

5.2 A high-order probabilistic model

The focus is now turned from the input sequence to the actual target circuit and to the investigation of the effect of input statistics on its transition probabilities (primary inputs and present state lines). As shown in Figure 5.2, we model the 'tuple' (*input_sequence, target_circuit*) by the 'tuple' (*Markov_chain, FSM*), where *Markov_chain* models the *input_sequence* and *FSM* is the sequential machine where the transition probabilities have to be determined. In what follows, x_n, s_n will denote the random variables associated to

the inputs and state lines of the target sequential machine; $p(x_n s_n)$ is the probability that, at time step n, the input is x_n *and* the state is s_n.

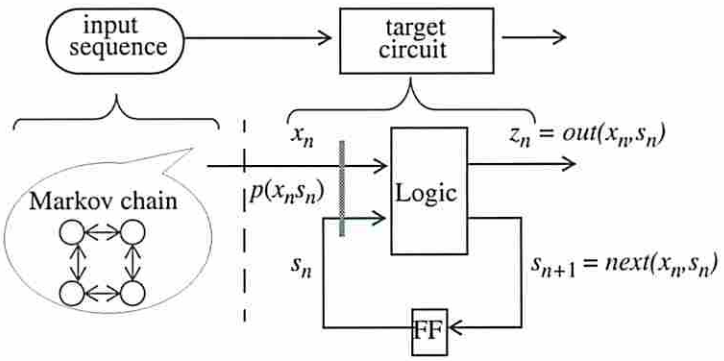


Figure 5.2 The tuple (*Markov-Chain, FSM*)

We are interested in defining the joint probabilities $p(x_n s_n)$ and $p(x_n s_n x_{n-1} s_{n-1})$ because, as shown in Figure 5.2., they completely capture the characteristics of the input (primary inputs and present state lines) that feeds the next state and the output logic of the target circuit.

5.2.1 Problem formulation

Having defined the joint random variables '$x_n s_n$' and '$x_n s_n x_{n-1} s_{n-1}$', the vector compaction problem for sequential circuits becomes essentially the vector compaction problem for combinational circuits as presented in Section 4.3.1: given a sequence of length L_0, find another sequence of length $L < L_0$ (consisting of a subset of the initial sequence), such that the *average joint transition probability* on the primary inputs *and* present state lines is preserved *wordwise,* for *two* consecutive time steps. More formally, the following condition should be satisfied:

$$\left| p(x_n s_n x_{n-1} s_{n-1}) - p^*(x_n s_n x_{n-1} s_{n-1}) \right| < \varepsilon \tag{5.1}$$

where p and p^* are the probabilities in the original and compacted sequences, respectively. ❑

This condition simply requires that the joint transition probability for inputs *and* states is preserved within a given level of error for two consecutive time steps. We note that, because the vector compaction problem is given in terms of joint probabilities $p(x_n s_n)$ and $p(x_n s_n x_{n-1} s_{n-1})$, the proof of correctness in Section 4.3.1 remains also valid for the sequential case. Once again, the stationarity condition an the primary inputs and state lines of the FSM is essential for this result.

Finally, we point out that in our approach, we do not make any attempt to compact the subsequence in the original sequence that is used for the initialization of the target FSM. More precisely, if the original sequence S consists of two components, one being the initialization sequence of the target FSM (S_{init}) (usually 50-100 vectors) and the other being a regular set of vectors that excite the FSM (S_{reg}), then S = S_{init} @ S_{reg}. We do consider for compaction only the S_{reg} component. (Usually, $S_{init} \ll S_{reg}$ and thus S_{reg} is the prime candidate for compaction). This is primarily because we do not want to alter the initializability properties of the FSM and also because we need stationary behavior on the state lines of the FSM to have our compaction procedure work.

5.3 The effect of finite-order statistics on FSM behavior

We will investigate now the conditions that can guarantee that equation (5.1) is satisfied. Under the general assumptions of *stationarity* and *ergodicity* [23], the following results can be proved.

Theorem 5.1 If the sequence feeding a target sequential circuit has order k, then a lag-k Markov chain which correctly models the input sequence, also correctly models the k-step conditional probabilities of the primary inputs *and* internal states, that is:

$$p(x_n s_n | x_{n-1} s_{n-1} x_{n-2} s_{n-2} \ldots x_{n-k} s_{n-k}) = p(x_n | x_{n-1} x_{n-2} \ldots x_{n-k}) \tag{5.2}$$

Proof: Since x_n is a lag-k Markov chain, we can first prove that, for any $n \geq k+1$, the following holds:

$$p(x_n s_{n-k} | x_{n-1} x_{n-2} \ldots x_{n-k}) = p(x_n | x_{n-1} x_{n-2} \ldots x_{n-k}) \cdot p(s_{n-k} | x_{n-1} x_{n-2} \ldots x_{n-k}) \tag{5.3}$$

Let $p(x_n s_n x_{n-1} s_{n-1} \ldots x_{n-k} s_{n-k})$ be the joint transition probability for inputs and states over k consecutive time steps. Then we have:

$$p(x_n s_n \ldots x_{n-k} s_{n-k}) = \begin{cases} p(x_n x_{n-1} \ldots x_{n-k} s_{n-k}) & \text{if } next(x_i, s_i) = s_{i+1} \quad i = n-k, \ldots, n-1 \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

For the first alternative we have:

$$p(x_n s_n \ldots x_{n-k} s_{n-k}) = p(x_n x_{n-1} \ldots x_{n-k} s_{n-k}) = p(x_n s_{n-k} | x_{n-1} \ldots x_{n-k}) \cdot p(x_{n-1} \ldots x_{n-k}) \tag{5.5}$$

which becomes using the first relation,

$$p(x_n s_{n-k} | x_{n-1} x_{n-2} \ldots x_{n-k}) = p(x_n | x_{n-1} x_{n-2} \ldots x_{n-k}) \cdot p(s_{n-k} | x_{n-1} x_{n-2} \ldots x_{n-k}),$$ or

equivalently: $p(x_n s_n \ldots x_{n-k} s_{n-k}) = p(x_n | x_{n-1} x_{n-2} \ldots x_{n-k}) \cdot p(x_{n-1} x_{n-2} \ldots x_{n-k} s_{n-k})$.

Dividing both sides by $p(x_{n-1}x_{n-2}\ldots x_{n-k}s_{n-k})$, and using the fact that $p(x_n s_n \ldots x_{n-k}s_{n-k}) = p(x_n x_{n-1}\ldots x_{n-k}s_{n-k}) = p(x_n s_{n-k} | x_{n-1}\ldots x_{n-k}) \cdot p(x_{n-1}\ldots x_{n-k})$, we obtain exactly $p(x_n s_n | x_{n-1}s_{n-1}x_{n-2}s_{n-2}\ldots x_{n-k}s_{n-k}) = p(x_n | x_{n-1}x_{n-2}\ldots x_{n-k})$.

For the second alternative, if $x_n s_n x_{n-1}s_{n-1}x_{n-2}s_{n-2}\ldots x_{n-k}s_{n-k}$ is not a valid sequence, then $p(x_n s_n | x_{n-1}s_{n-1}x_{n-2}s_{n-2}\ldots x_{n-k}s_{n-k}) = 0$ and this concludes our proof. ■

We note therefore that, preserving order-k statistics for the inputs implies also that order-k statistics will be captured for inputs and state lines. As long as the order k correctly models the input sequence, we cannot produce new transitions of the FSM and therefore 'forbidden' subsequences of order k. However, by using DMT_k, we may introduce new subsequences of order higher than k; this does not matter for the functionality of the FSM as long as we are guaranteed by Theorem 5.1 that the conditional probabilities (and thus steady state probabilities) for inputs and state lines are preserved.

In particular, the first-order statistics of the joint probabilities defined in equation (5.1) are implicitly preserved. However, modeling a k-order source with a lower order model may introduce accumulative inaccuracies. From a practical point of view, this means that underestimating a high-order source, it is possible not to correctly preserve even the first-order transition probabilities and thus violate the requirement in equation (5.1). In terms of power consumption, this will adversely affect the quality of the results as we can see from the following example.

Example 5.2 Once again the sequences S_1 and S_2 in Example 5.1 are considered and we assume that they feed the benchmark *dk17*. It will be illustrated that indeed, if the input

sequence has order two, then modeling it as a lag-one Markov Chain will *not* preserve the first-order joint transition probabilities (primary inputs and internal states) in the target circuit. We simulated the benchmark *dk17 (*starting with the same initial state *'19')* for both sequences and we present in Figure 5.3 (Figure 5.3a is for S_1 and Figure 5.3b is for S_2) the wordwise transition graphs obtained for the signal lines ($x_n s_n$).

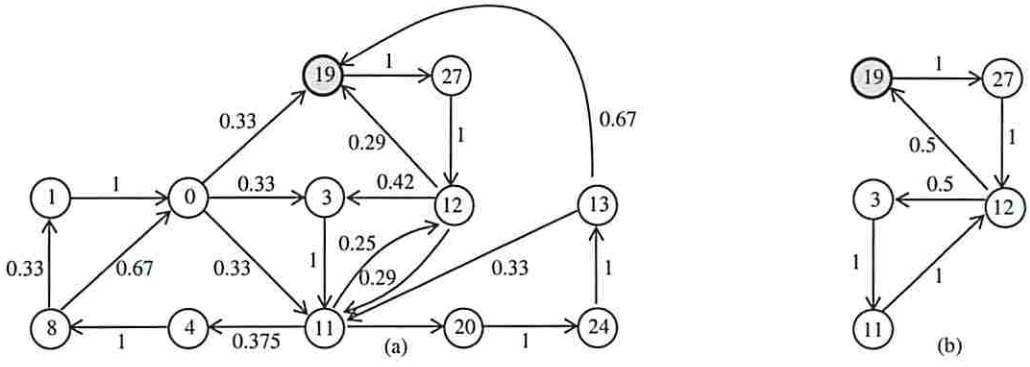


Figure 5.3 Two STGs obtained for the signal lines ($x_n s_n$) of benchmark *dk17*

The benchmark *dk17* has 2 primary inputs and 3 flip-flops, therefore, in Figure 5.3, any node is decimally encoded using 5 bits. For instance, the initial state *'19'* corresponds to the binary code '10011' that is, '10' for primary inputs and '011' for state lines. Starting from state '19' and applying '11' on the primary inputs, the present state becomes '011', therefore we enter the node '11011' = 27 in Figure 5.3. As we can see, because S_1 can be modeled as a first order Markov source, while S_2 must be modeled as a second order Markov source, the corresponding transition graphs are quite different. From a practical point of view, this means that if one high-order source is underestimated (for instance, assuming that second- or higher-order temporal correlations are not important), then even the first-order transition probabilities in the target circuit may not be preserved. In terms of

power consumption, this adversely affects the quality of the results as shown and discussed in Example 5.1.

Corollary 5.1 If the sequence feeding the target circuit has order one, then a lag-one MC will suffice to model correctly the joint transition probabilities of the primary inputs and internal states in the target circuit, that is:

$$p(x_n s_n | x_{n-1} s_{n-1}) = p(x_n | x_{n-1}) \tag{5.6}$$

Proof: Follows from Theorem 5.1 if $k = 1$. ■

Thus, if x_n is a lag-one Markov chain, then preserving the first-order statistics on the inputs is enough for preserving the joint transition probabilities for inputs and state lines. In other words, if the sequence feeding the target circuit can be accurately modeled as a first-order Markov chain, then a first-order power model can be successfully applied because the whole 'history' of the input is limited to only two consecutive time steps. In this particular case, the compaction problem for both FSMs and combinational circuits becomes essentially the same; that is, all that is needed is to efficiently model the input sequence as a first-order Markov model.

5.4 Fixed and variable-order Markov sources

5.4.1 The order of a Markov source

The next step is to determine the order of a Markov source, because, as proved in Theorem 5.1, knowing the correct value of k, is essential for FSM analysis. To this end, based on the

probability of finding a 'block' of vectors $<v_n,v_{n-1},...,v_1>$[1] in any sequence in S (denoted by $p(v_n v_{n-1} \ldots v_1)$), we introduce the following entropy-like quantities.

Definition 5.1 (Block entropy). The *block entropy* of length n is defined as:

$$H_n = \sum_{\langle v_n v_{n-1} \ldots v_1 \rangle} p(v_n v_{n-1} \ldots v_1) \cdot \log p(v_n v_{n-1} \ldots v_1) \tag{5.7}$$

where $n \geq 1$.

Definition 5.2 (Conditional entropy). The *conditional entropy* associated with the addition of a new vector v_{n+1} to the left of an already existing block $<v_n,v_{n-1},...,v_1>$ is defined as:

$$h_n = H_{n+1} - H_n \tag{5.8}$$

for $n \geq 1$ and $h_0 = H_1$.

Definition 5.3 (Source entropy). The *entropy of a source,* or the *uncertainty per step,* is defined as:

$$h = \lim_{n \to \infty} \frac{H_n}{n} = \lim_{n \to \infty} h_n \tag{5.9}$$

and often referred to as *metric entropy* or *entropy rate* [35].

For stationary and ergodic processes, Khinchin [23] has shown that H_n in equation (5.7) is monotonically increasing, H_n/n is monotonically decreasing and the limit in equation (5.9) exists. We can now justify the term 'conditional' used for the entropy in equation (5.8). Indeed, we have:

1. Indices 1, 2,... n designate the sequencing among vectors that occur within a block of length n.

$$h_n = H_{n+1} - H_n = \sum_{\langle v_{n+1} v_n \ldots v_1 \rangle} p(v_{n+1} v_n \ldots v_1) \cdot \log p(v_{n+1} v_n \ldots v_1) - \\ - \sum_{\langle v_n v_{n-1} \ldots v_1 \rangle} p(v_n v_{n-1} \ldots v_1) \cdot \log p(v_n v_{n-1} \ldots v_1) \tag{5.10}$$

The first term of the above summation can be written as:

$$\sum_{\langle v_{n+1} v_n \ldots v_1 \rangle} p(v_{n+1} | v_n v_{n-1} \ldots v_1) \cdot p(v_n v_{n-1} \ldots v_1) \cdot [\log p(v_{n+1} | v_n v_{n-1} \ldots v_1) + \quad = \\ + \log p(v_n v_{n-1} \ldots v_1)] = \\ = \sum_{\langle v_{n+1} v_n \ldots v_1 \rangle} p(v_{n+1} | v_n v_{n-1} \ldots v_1) \cdot p(v_n v_{n-1} \ldots v_1) \cdot \log p(v_{n+1} | v_n v_{n-1} \ldots v_1) + \\ + \sum_{\langle v_{n+1} v_n \ldots v_1 \rangle} p(v_{n+1} | v_n v_{n-1} \ldots v_1) \cdot p(v_n v_{n-1} \ldots v_1) \cdot \log p(v_n v_{n-1} \ldots v_1) \tag{5.11}$$

In turn, the second term in equation (5.11) can be written as:

$$\sum_{\langle v_n v_{n-1} \ldots v_1 \rangle} p(v_n v_{n-1} \ldots v_1) \cdot \log p(v_n v_{n-1} \ldots v_1) \cdot \sum_{\langle v_{n+1} \rangle} p(v_{n+1} | v_n v_{n-1} \ldots v_1) = \\ = \sum_{\langle v_n v_{n-1} \ldots v_1 \rangle} p(v_n v_{n-1} \ldots v_1) \cdot \log p(v_n v_{n-1} \ldots v_1)$$

which is exactly the second term in equation (5.10). As a consequence, equation (5.10) reduces to:

$$h_n = \sum_{\langle v_{n+1} v_n \ldots v_1 \rangle} p(v_{n+1} | v_n v_{n-1} \ldots v_1) \cdot p(v_n v_{n-1} \ldots v_1) \cdot \log p(v_{n+1} | v_n v_{n-1} \ldots v_1) \tag{5.12}$$

which illustrates the mean uncertainty of a new vector v_{n+1}, given the knowledge of vectors v_n, v_{n-1}, v_{n-2},..., v_1. Therefore the conditional entropy h_n is a measure of the predictability for the whole process.

Proposition 5.1 For any lag-k Markov chain, it holds that:

$$H_n = H_k + (n-k) \cdot (H_{k+1} - H_k), \ \forall n \geq k \tag{5.13}$$

Proof: We first show that $h_n = h_k \quad \forall n \geq k$. From equation (5.12) we can deduce that:

$$h_n = \sum_{\langle v_{n+1} v_n \cdots v_1 \rangle} p(v_{n+1} | v_n \cdots v_1) \cdot p(v_n v_{n-1} \cdots v_1) \cdot \log p(v_{n+1} | v_n v_{n-1} \cdots v_1) =$$

$$= \sum_{\langle v_{n+1} v_n \cdots v_{n-k+1} \rangle} p(v_{n+1} | v_n \cdots v_{n-k+1}) \cdot \log p(v_{n+1} | v_n v_{n-1} \cdots v_{n-k+1}) \cdot \sum_{\langle v_{n-k} \cdots v_1 \rangle} p(v_n v_{n-1} \cdots v_1)$$

which is exactly h_k because $\sum_{\langle v_{n-k} \cdots v_1 \rangle} p(v_n v_{n-1} \cdots v_1) = p(v_n v_{n-1} \cdots v_{n-k+1})$.

Now, we can also write:

$h_k = H_{k+1} - H_k$

$h_{k+1} = H_{k+2} - H_{k+1}$

:

$h_{k+n-1} = H_n - H_{n-1}$

From here by summation (after reducing the appropriate terms) we get: $(n-k) \cdot h_k = H_n - H_k$; using equation (5.8) we get the above claim. ■

Due to statistical correlations extending over a range of only k iterations, for Markov sources of order k, the conditional uncertainty h_n is decreasing [23] until it eventually reaches its limit value h for $n = k$. The memory effects of the source are reflected by this *saturation point* which represents that value of n when the limit h is reached exactly or with a good approximation.

Example 5.3 In Figure 5.4 we have the typical behavior for a lag-one and a lag-two Markov sequences, that have been generated by using a first and a second order recursive

relations, respectively. The two sources have the same order-0 and order-1 statistics, but different order-2 statistics.

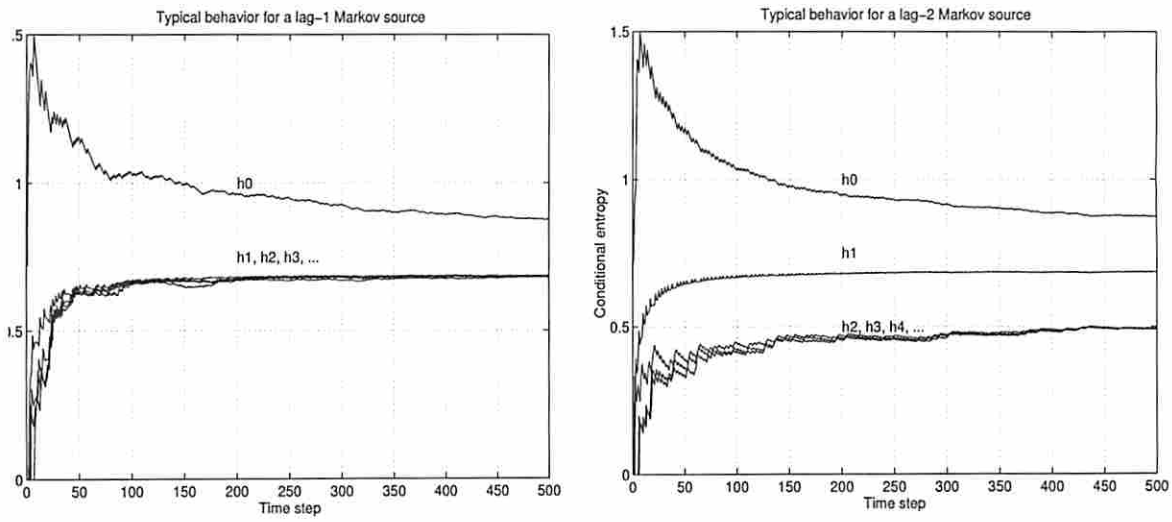


Figure 5.4 Conditional entropies for a lag-one and a lag-two Markov sequences

In both cases, the n-step conditional entropy h_n of the source reaches its limit h after few tens of vectors. In the first case, the limit is $h = h_1$, while in the case of the lag-two Markov source, the limit is given by h_2 as was expected according to the above discussion.

Finally, we note that there may exist different subsequences of order k within a given initial sequence. For instance, we may have an initial sequence S which consists of two very different second order subsequences, S_1 and S_2, generated with a second order Markov source. In Figure 5.5, we plot the variation of the conditional entropies as function of time for such a situation. In this case, the block entropy in Definition 5.1 (and implicitly the source entropy in Definition 5.3) is different for these subsequences (despite the fact that both of them exhibit the same type of temporal correlations, that is second order temporal correlations) and this is enough for our approach to make the distinction.

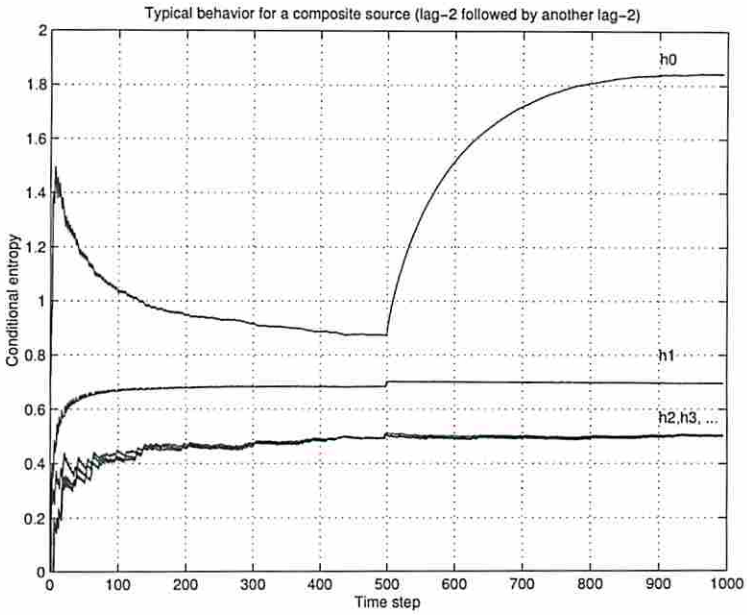


Figure 5.5 Conditional entropies for two lag-two Markov subsequences

We observe that, indeed, after time step 500, the conditional entropies h_0, h_1 stabilize to different values for S_1 and S_2.

5.4.2 Composite sequences

In practice, lag-k Markov chains (with fixed k) may not be a good enough approximation for a given sequence. For example, a real sequence can be a mixture of sequences, each generated from a different order Markov source. We call such a sequence a *composite sequence* to emphasize its non-homogeneous characteristics. When the sequence changes its behavior due to the change in order, the stationarity and convergence hypotheses from equation (5.9) no longer hold. For instance, if we consider two mixed sequences, the first containing a lag-one followed by a lag-two Markov subsequence, and the second a lag-two

followed by a lag-one Markov subsequence, the behavior of conditional entropies is the one depicted in Figure 5.6.

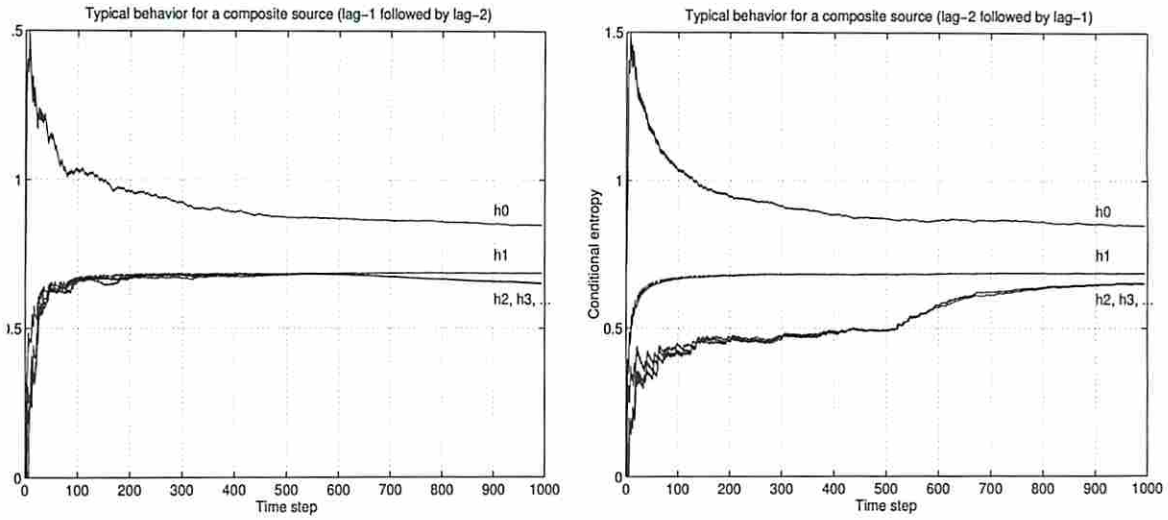


Figure 5.6 Typical behavior of conditional entropies for composite sequences

In the first case, after a sufficiently large number of steps, h_1 and h_2 do not remain the same. Actually, in the long run, the limit in equation (5.9) becomes h_2, thus detecting an order 2 for the source. In this case, the convergence to h_1 is violated and thus the sequence changes its order from 1 to 2. In the second case, after already being stabilized to the stationary values, the conditional entropies h_2, h_3, h_4,... tend to increase so that the order of the second half of the sequence can no longer be considered 2.

This discussion provides a starting point for determining the order of subsequences in a dynamic fashion. The only requirement that has to be satisfied is the stationarity of each subsequence and, in this case, the entire sequence is called *piecewise stationary.* This is the basic hypothesis of all our subsequent experiments. To test the condition of piecewise stationarity, in practice we can use again the metric entropy in conjunction with support from the simulation of the STG of the machine. We also note that, for sequential circuits,

we consider piecewise stationary sequences that exhibit only a single power mode. However, for sequences that have multiple power modes, the hierarchization procedure in Chapter 4 can be still applied but, in this case, we have to consider the average Hamming distance on the primary inputs *and* state lines of the FSM to build the hierarchical model. This implies that knowledge about the behavior and state encoding of the FSM is available to the user.

5.4.3 Variable-order dynamic Markov models

From results shown in the previous sections, we need an efficient way to model lag-k Markov chains that could characterize the input sequences that feed the target circuit. The structure DMT_1 (introduced in Chapter 4) is general enough to completely capture the correlations among all bits of the same input vector and also between successive input patterns. Indeed, the recursive construction of DMT_1 by considering successive bits in the upper and lower subtrees completely capture the word-level (spatial) correlations for each individual input vector in the original sequence. Furthermore, cascading lower subtrees for each path in the upper subtree, gives the actual sequencing (first-order temporal correlation) between successive input patterns. However, conceptually it has no inherent limitation to be further extended to capture temporal dependencies of higher orders. For instance, if we continue to define recursively DMT_2 (starting with DMT_1), we can

basically capture second-order temporal correlations. For any sequence where v_i, v_j, v_k are three consecutive vectors (that is, $v_i \rightarrow v_j \rightarrow v_k$), the tree DMT_2 looks like in Figure 5.7.

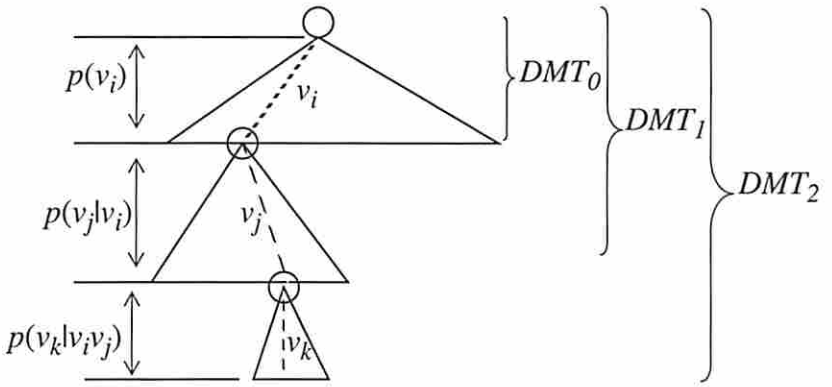


Figure 5.7 A second-order dynamic Markov tree

The following result, gives the theoretical basis for using the dynamic Markov trees to capture high-order temporal correlations.

Theorem 5.2 The general structure DMT_k and its parameters completely capture spatial and temporal correlations of order k.

Proof: Let $v = v_0\ v_1 \ldots\ v_k$ be a string in DMT_k (the substring v_i belongs to the i-level tree, DMT_i). We have that $p\ (v_k \mid v_{k-1}\ v_{k-2} \ldots\ v_0) = p\ (v_0\ v_1 \ldots\ v_k) \,/\, p\ (v_0\ v_1 \ldots\ v_{k-1})$ and thus the lag-k Markov chain characterizing the input can be fully modeled by the DMT_k structure. ■

5.5 Practical considerations and experimental results

The DMC modeling approach offers the significant advantage of being a *one-pass adaptive technique*. As a one-pass technique, there is no requirement to save the whole sequence in the on-line computer memory. Starting with an initial empty tree DMT_K[1],

1. K is the maximum possible order of the Markov model of the input sequence.

while the input sequence is incrementally scanned, both the set of states and the transition probabilities dynamically change making this technique highly adaptive. Also, by using this data structure, we can easily account for the conditional entropies and detect the order of the Markov source. Under stationarity conditions, the order is detected as the minimum k such that $|h_k - h_n| < \varepsilon$, for some $\varepsilon > 0$ and any $n = k + 1,..., K$ where K is the maximum order of the source to be detected. After that, if either this condition becomes violated or the stationarity hypothesis does not hold, the model is flushed and restarted. As in the combinational case, for each grown tree, the generation phase is driven by the user-specified compaction parameter ratio r. We also note that this strategy does note allow 'forbidden' vectors that is, those combinations that did not occur in the original sequence, will not appear in the final compacted sequence either. This is an essential capability needed to avoid 'hang-up' ('forbidden') states of the sequential circuit during simulation process for power estimation.

The overall strategy is shown in Figure 5.8.

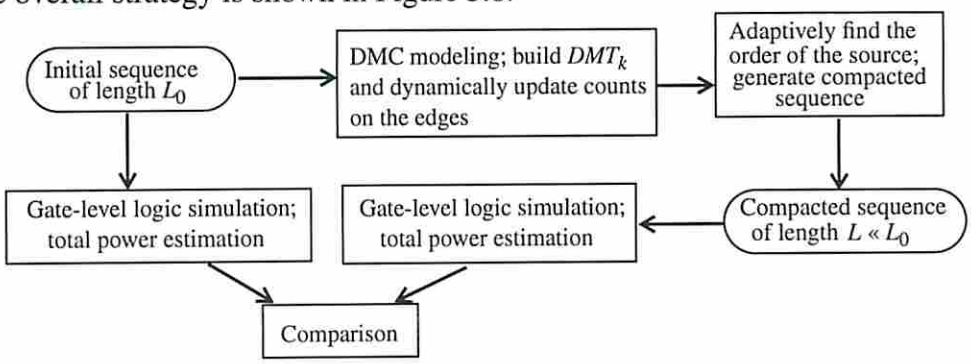


Figure 5.8 The experimental setup for sequential circuits

We assume that the input data is given in the form of a sequence of binary vectors. Starting with an input sequence of length L_0, we perform a one-pass traversal of the

original sequence and simultaneously build the basic tree DMT_K; during this process, the frequency counts on DMT_K's edges are dynamically updated. During this process, the order of the source is also determined. The generation step is done using a modified version of the weighted selection algorithm described in Chapter 4. Finally, a validation step is included in the strategy; for this purpose, we have used an in-house gate-level logic simulator developed under SIS. The total power consumption of some sequential benchmarks has been measured for the initial and the compacted sequences, making it possible to assess the effectiveness of the compaction procedure.

In Table 5.1, we provide the gate-level power simulation results for different initial sequences having the total length of 10,000 vectors; these sequences were produced using a second order information source based on Fibbonacci series. As shown in Table 5.1, the sequences were compacted with two different compaction ratios (namely r = 10 and 20) using three Markov models: one of order zero (that is, assuming temporal independence on the primary inputs), one of order one based on DMT_1 and another matching the actual characteristics of the sequence (that is, based on DMT_2). This table shows the total power dissipation measured for the initial sequence (column 3) and for the compacted sequence using all models (columns 4-9). Using a Sparc 20 workstation with 64 Mbytes of memory, the time necessary to read and compress data was less than 10 seconds for all models. Since the compaction with DMC modeling is linear in the number of levels in the DMT_k structure, these time values are far less than the actual time needed to simulate the whole

sequence. During these experiments, the maximum number of nodes allowed in the Markov model was 200,000.

Table 5.1. Total Power (μW@20MHz) for input sequences of order 2

			Estimated power for $r = 10$			Estimated power for $r = 20$		
Circuit	No of inp./ FFs	Power for initial seq.	Order 0	Order 1	Adaptive	Order 0	Order 1	Adaptive
s1196	14/18	7025.31	6740.16	6668.31	7027.58	6667.81	6414.11	7006.11
s1423	17/74	5624.64	5197.43	5203.94	5577.11	5068.73	5084.82	5458.20
s510	19/6	2073.55	2210.85	1940.11	1980.47	2277.32	1877.72	1974.88
s5378	35/164	13826.55	11955.22	13304.01	13666.80	11885.03	12838.12	13684.05
s820	18/5	4120.95	3864.65	3668.70	4077.06	3703.84	3561.86	4047.21
s9234	36/211	12531.45	13004.71	13037.10	12247.65	13239.04	13070.32	12187.86
s953	16/29	1158.04	1199.03	1122.11	1156.47	1207.44	1102.16	1149.19
		Avg.% err.	6.47	5.84	1.43	8.41	8.24	2.04
bbara	4/4	747.25	689.54	821.74	745.57	685.09	823.67	745.67
bbtas	2/3	337.74	284.40	362.00	337.43	284.15	323.36	337.39
dk17	2/3	1439.65	1203.75	1306.32	1437.21	1203.56	1286.46	1436.34
donfile	2/5	3020.63	2242.27	2874.66	3016.54	2207.00	2852.06	3010.74
mc	3/2	295.99	234.90	238.07	293.32	219.47	246.30	292.79
planet	7/6	8552.95	7314.99	6505.96	8442.30	7105.18	6565.44	8151.24
shiftreg	1/3	144.60	91.43	124.27	144.49	91.27	122.56	144.63
		Avg.% err.	19.65	12.69	0.41	21.03	12.29	0.95

As we can see, for the model having the order 2, the quality of results is very good even when the length of the initial sequence is reduced by more than one order of magnitude. Thus, for *s1196* in Table 5.1, instead of simulating 10,000 vectors with an exact power of 7025.31 μW, one can use only 1,000 vectors ($r = 10$) with an estimate of 7027.58 μW or just 500 vectors ($r = 20$) with power consumption of 7006.11 μW. This reduction in the sequence length has a significant impact on speeding-up the simulative approaches where the running time is proportional to the length of the sequence which must be simulated. On the other hand, using a zeroth- or first-order model, the quality of the results can be seriously impaired. For instance, in the case of benchmark *planet*, for r

= 20, we can erroneously predict a total power of 6505.96μW and, for r = 10, a value of 6565.44μW for a first-order model (more than 23% error). The value of the error can be even worse for a zero-order model (e.g. benchmark *donfile,* where the error is more than 25%). This is because for a sequence generated with a second-order source, a model that ignores temporal correlations or considers only pairs of consecutive vectors cannot correctly preserve even the first-order transition probabilities for the primary inputs and state lines (that is, the probabilities $p(x_n s_n x_{n-1} s_{n-1})$ in our notation).

Table 5.2 shows the results obtained for composite sequences of length 10,000. These sequences have been generated using different generators and exhibit temporal correlation of various orders (order 2, followed by order 1 and finally, once again order 2). This hybrid character of the sequences makes a significant difference in terms of total power consumption for the analyzed benchmarks. As we can see, compared to Table 5.1, the most dramatic increase in the level of error occurs for the zero-order model; in this case, the temporal independence assumption is seriously impairing the estimation accuracy. The error for the first-order model is, on average, around 10%, while the adaptive modeling technique provide accurate results, even for a compaction ratio of r = 20.

Table 5.2. Total Power (μW@20MHz) for composite input sequences

			Estimated power for $r = 10$			Estimated power for $r = 20$		
Circuit	No. of inp./ FFs	Power for initial seq.	Order 0	Order 1	Adaptive	Order 0	Order 1	Adaptive
s1196	14/18	5272.36	7212.58	5395.73	5237.65	7245.41	5501.05	4972.52
s1423	17/74	3964.48	5771.69	4173.21	4048.88	5836.40	4046.63	3834.35
s510	19/6	1520.46	2232.08	1814.65	1527.91	2215.03	1953.39	1501.10
s5378	35/164	11566.16	14251.17	11711.97	11282.86	14325.13	11871.03	10990.86
s820	18/5	3131.51	4158.82	3368.83	3096.87	4178.75	3431.04	3196.87
s9234	36/211	10046.05	12797.24	9688.30	9573.48	12951.02	9672.07	9288.65
s953	16/29	764.92	1188.94	796.02	755.66	1186.26	805.89	731.35
		Avg.% err.	38.29	6.20	1.82	38.80	8.04	4.18
bbara	4/4	771.16	790.23	842.87	769.97	806.44	749.96	766.41
bbtas	2/3	405.29	333.21	427.52	403.45	339.82	443.14	402.84
dk17	2/3	1392.67	1165.08	1258.88	1390.21	1135.61	1248.20	1388.09
donfile	2/5	3203.97	2455.90	3100.94	3197.36	2463.37	2997.86	3190.14
mc	3/2	335.97	289.37	298.76	332.92	295.64	285.17	328.27
planet	7/6	8619.93	7753.93	7234.10	8401.28	7605.67	7097.61	8030.61
shiftreg	1/3	149.59	100.62	140.30	149.29	97.36	139.68	148.97
		Avg.% err.	16.65	8.71	0.66	17.28	9.76	1.65

As for comparing our results with simple random sampling technique for FSM circuits, we are unable to do so for the following reason. Our compaction technique starts with a finite input sequence and a user-specified compaction ratio or error level, and performs compaction by DMC modeling and sequence generation. We must therefore compare our results with statistical sampling techniques which work on finite populations (i.e. an input sequence with fixed length). Although sampling techniques for combinational circuits under a given input sequence have been developed and published in the literature (hence, we could produce results presented in Chapter 4), such techniques for FSM circuits are not known. Note that Monte Carlo Simulation (which is based on a simple random sampling strategy) assumes an infinite population and it synthesizes the

input sequence used for sampling based on a Markov model of bitwise activities. Hence, it cannot be used for our comparison purpose.

Finally, we give in Figure 5.9 the node-by-node analysis of the switching activity for benchmark *planet* for a compaction ratio of 5.

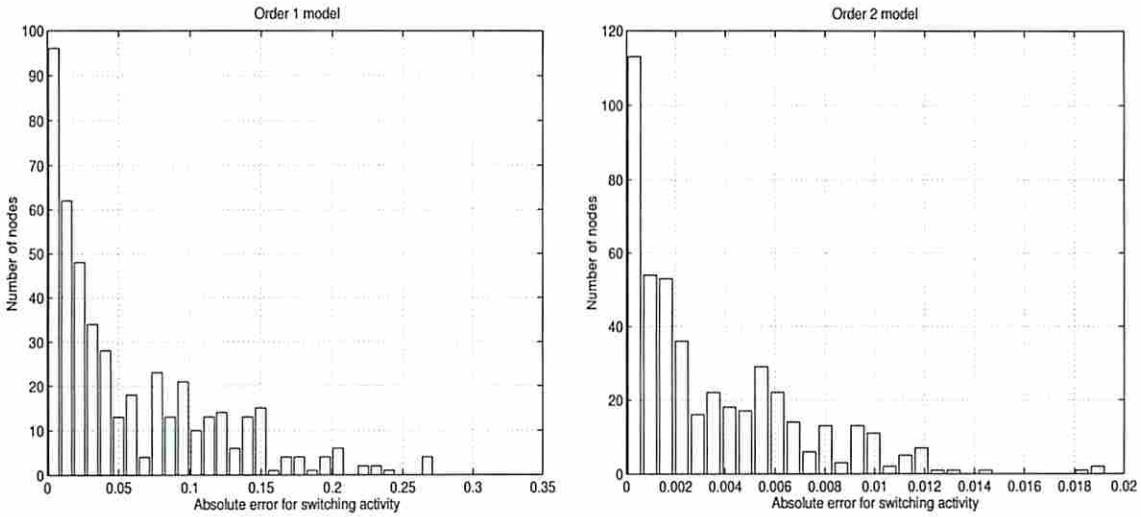


Figure 5.9 Node-by-node analysis for benchmark *planet*

Using a lower order model than the actual order of the input sequence can significantly impair the ability of correctly estimating the switching activity on a node-by-node basis. While for the first order model the absolute error[1] achieves a maximum value of 0.272 and a mean value of 0.057, it decreases to 0.019 and 0.003 respectively if a second order model is used. These results were typical for the whole set of benchmarks that we have analyzed. Based on these results, we can therefore conclude that for FSMs the adaptive technique is the only one appropriate for correctly modeling the input sequence.

In Table 5.3, we provide the gate-level power simulation results for a set of different

1. The absolute error is defined as $|sw_{comp}\text{-}sw_{exact}|$, where sw_{comp} is the switching activity obtained using the compacted sequence.

initial sequences having a length of 200,000 vectors. To generate these results, we use two different strategies. First, we compact the input sequences with three fixed compaction ratios and indicate the estimated values of total power consumption (columns 4-5). In the second scenario (columns 6-7), we consider a fixed threshold of 5% for the mean error of the transition probabilities. We monitor the current values of the transition probabilities and compare them with the transition probabilities in the original sequence. When the difference between the two sets of probabilities becomes sufficiently small, the generation procedure is halted. In this way we are able to satisfy any user specified error level for the transition probabilities. Using a Sparc 20 workstation with 64 Mbytes of memory, the time necessary to read and compact data was less than a few tens of seconds in all cases.

Table 5.3. Total Power (μW@20MHz) for long sequences

			Estimated power for fixed compaction ratio		Fixed error level (5% mean error for transition probabilities)	
Circuit	No of inp./ FFs	Power for initial seq.	$r = 10$	$r = 100$	Estimated power	Number of vectors required
s1196	14/18	5724.28	5723.42	5702.51	5714.14	2242
s1423	17/74	3544.75	3548.86	3548.02	3563.84	8984
s510	19/6	1685.12	1692.89	1307.42	1690.10	8002
s5378	35/164	9161.58	9139.20	9355.99	9169.60	11884
s820	18/5	3609.23	3608.02	3604.97	3616.75	5962
s9234	36/211	9570.30	9569.93	9568.40	9577.91	23922
s953	16/29	817.91	817.60	816.81	817.42	4922
		Avg. % err.	0.12	3.62	0.20	
bbara	4/4	777.10	777.10	776.29	776.71	1404
bbtas	2/3	422.18	422.19	422.26	423.23	84
dk17	2/3	1381.27	1381.16	1380.02	1359.41	84
donfile	2/5	3250.47	3250.43	3249.96	3184.50	84
mc	3/2	346.03	346.00	345.69	335.66	162
planet	7/6	7857.02	7859.57	7880.29	7881.93	2524
shiftreg	1/3	150.86	150.85	150.76	145.93	144
		Avg. % err.	0.01	0.10	1.54	

As we can see, the average error in total power prediction is below 2% for all

benchmarks while the achieved compaction ratio varies between 8 ($s9234$) and 90 ($s1196$) for large circuits and 80 (*planet*) and 2380 (*bbtas*) for small ones. This reduction in the sequence length has a significant impact on speeding-up the simulative power estimation approaches where the running time is proportional to the length of the sequence which must be simulated.

5.6 Summary

In this chapter, we addressed the issue of sequence compaction for power estimation in sequential circuits. More precisely, we proposed an original approach to compact an initial sequence into a much shorter equivalent sequence, which can be used with any available simulator to derive power estimates in the target circuit.

Our contribution is important in two ways: first, it shows the effects of finite-order statistics of the input sequence on FSMs behavior; second, based on the vector compaction paradigm, it provides an original solution for power estimation problem in FSMs. From the original results provided here under stationarity conditions, two are noteworthy for probabilistic FSM analysis:

- If the sequence feeding the target circuit has order k, then a lag-k Markov chain model of the sequence will suffice to model correctly the joint transition probabilities of the primary inputs and internal states in the target circuit.
- If the input sequence has order two or higher then modeling it as a lag-one Markov Chain cannot exactly preserve the first-order joint transition probabilities (primary inputs and internal states) in the target circuit.

In addition to this, we introduced and characterized a family of variable-order dynamic Markov models which provide an effective way for accurate modeling of external input sequences that affect the behavior of FSMs. Results obtained on standard sequential benchmarks, show that using this framework, large compaction ratios can be achieved without significant loss in the accuracy of total and node-by-node power estimates.

Chapter 6 Conclusions

6.1 Thesis summary

Power dissipation is widely recognized as one of the main concerns in the design process of digital CMOS circuits. In this thesis, we basically focused on average power analysis and produced a set of techniques that can be used to measure the dynamic power consumption in combinational and sequential circuits, at different levels of abstraction. Because we use extensively concepts from the theory of finite-order Markov chains, the overall flavor of our investigations is probabilistic in nature. While Chapter 1 and Chapter 2 provide a general introduction and background information, Chapter 3 to Chapter 5 contain the original contribution of our work. We note that some of the original results presented in Chapter 3 to Chapter 5 may have also application beyond the area of low-power systems. In what follows, we briefly summarize their content.

In Chapter 3, we introduced the spatio-temporal model for switching activity analysis in combinational logic modules. Assuming the zero-delay hypothesis, we extended the previous work done on switching activity estimation to explicitly account for complex spatiotemporal correlations which occur at the primary inputs when the target circuit receives data from real application. More precisely, using lag-one Markov Chains and two new concepts - conditional independence and signal isotropy - we provide sufficient conditions for exact analysis of complex dependencies. From a practical point of view, we have shown that the relative error in calculating the switching activity of a logic gate using only pairwise probabilities can be upper-bounded. We also proved that the conditional independence problem is **NP**-complete and thus, relying on the concept of signal isotropy,

we proposed approximate techniques with bounded error for estimating the switching activity.

Based on the spatio-temporal model, we developed a set of efficient techniques for power estimation in large combinational modules. The whole idea behind efficiency, is to calculate and propagate transition correlation coefficients from the circuit inputs toward the circuit outputs, and hence, eliminate the need for building the global function of the intermediate nodes. As quantitative illustration, these techniques can analyze combinational modules with 10,000 gates in less than 200 CPU sec., yet producing power estimates with less than 10-15% error, on average, compared to circuit-level simulation.

The model based on spatio-temporal correlations clearly demonstrates, qualitatively and quantitatively, the importance of being accurate node-by-node (not only for the total power consumption) and identifies potential drawbacks in previous approaches when patterns feeding the inputs become highly correlated.

In Chapter 4, we first introduced the paradigm of vector compaction for power estimation and provided different alternatives to solve the vector compaction problem for combinational circuits. The vector compaction paradigm is a very attractive solution for power estimation not only because it reduces the gap between static and dynamic techniques used in power estimation but also because it is practically independent of the actual implementation of the target circuit and it can therefore be used early in the design cycle when the structure of the target circuit has not been determined yet.

As the main contribution, assuming stationarity conditions, we formulated and proved the correctness of vector compaction problem. We also introduced and characterized a

family of dynamic Markov trees which can be used as an effective and flexible tool in modeling complex spatiotemporal correlations which occur during power estimation.

In the second part of Chapter 4, we introduced the hierarchical modeling of Markov chains as a flexible framework for capturing not only complex spatiotemporal correlations, but also the dynamic changes in the sequence characteristics such as different circuit operating modes or varying power distributions. To this end, we structure the input space into a hierarchy of *macro-* and *micro-states*: at the first (high) level in the hierarchy we have a Markov chain of macrostates; at the second (low) level, each macrostate is in turn characterized by a Markov chain for all its constituent microstates. Our primary motivation for constructing this hierarchical structure is to enable a better modeling of the different stochastic levels that are present in sequences that arise in practice.

As the experimental results show large compaction ratios of few orders of magnitude can be obtained without significant loss in accuracy in total power estimates.

Finally, in Chapter 5, we extended the techniques presented in Chapter 4 to deal with sequential circuits. We first analyzed the effects of finite-order statistics of the input sequence on FSM's behavior. Second, based on the vector compaction paradigm, we proposed an original solution for power estimation in FSMs. To this end, we introduce and characterize a family of variable-order dynamic Markov models which provide an effective way for accurate modeling of external input sequences that affect the behavior of FSMs. Relying on the concept of block entropy, we also presented a technique for identifying the actual order of variable-order Markov sources of information that is,

sources of information that can be *piecewise* modeled by Markov chains of different orders.

Results obtained on standard sequential benchmarks, show that using this framework, large compaction ratios can be obtained without significant loss in the accuracy of total and node-by-node power estimates.

In summary, in this thesis we have presented a set of techniques which offer a certain degree of flexibility and can be used in practice to estimate the power consumption. However, it is the designer's responsibility to choose the technique which is more suitable for his application and to interpret the results.

6.2 Future Work

Although the body of this thesis was devoted only to power estimation in digital CMOS circuits, there is room for new investigations and extensions of the present research.

Regarding the analytic model proposed in Chapter 3, the two most important extensions would be to consider sequential circuits and the real-delay hypothesis. Generally speaking, for the next generation of power estimation technique, it would be beneficial to raise the level of abstraction as much as possible. However, accurate techniques for power estimation at gate-level (like the one proposed in Chapter 3) will always have a place in the design flow so the two extensions mentioned above are very important.

Regarding the set of techniques proposed in Chapter 4 and Chapter 5, there are a number research directions too. First, the restriction of input space only to binary symbols

should be relaxed to also accommodate sequences that allow multiple symbols (e.g. Z) because, normally, such symbols are likely to occur in designers' specifications. In addition to this, the hierarchical model proposed for combinational circuits can be extended to be applicable to sequential circuits too. This will make the approach much more sensitive to the dynamic changes in the sequence characteristics which may occur in real hardware. Finally, the high-order modeling of the input data presented in Chapter 5 can be applied to steady-state analysis of FSMs.

References

[1] A. Ghosh, S. Devadas, K. Keutzer, and J. White, 'Estimation of Average Switching Activity in Combinational and Sequential Circuits,' in *Proc. ACM/IEEE Design Automation Conference,* pp. 253-259, June 1992.

[2] P. H. Bardell, W. H. McAnney, and J. Savir, 'Built-in Test for VLSI: Pseudorandom Techniques,' John Wiley & Sons, 1987.

[3] T. Bell, J. Cleary and I. Witten, 'Text Compression,' Prentice-Hall, 1990.

[4] L. Benini and G. De Micheli, 'Automatic Synthesis of Low-Power Gated-Clock Finite-State Machines,' in *IEEE Trans. on CAD*, vol. 15(6), pp. 630-643, June 1996.

[5] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, 'Algorithms for VLSI Logic Synthesis,' Kluwer Academic Publishers, 1984.

[6] R. E. Bryant, 'Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams,' in *ACM Computing Surveys*, Vol. 24, No. 3, pp. 293-318, Sept. 1992.

[7] R. Burch, F. N. Najm, P. Yang, T. Trick, 'A Monte Carlo Approach for Power Estimation,' *IEEE Transactions on VLSI Systems*, Vol.1(1), pp. 63-71, Mar.1993.

[8] T. L. Chou and K. Roy , 'Statistical Estimation of Sequential Circuit Activity,' in *Proc. IEEE/ACM Intl. Conference on Computer-Aided Design,* pp. 34-37, Nov. 1995.

[9] G. V. Cormack and R. N. Horspool, 'Data Compression Using Dynamic Markov Modeling,' in *Computer Journal*, Vol. 30, No. 6, pp. 541-550, 1987.

[10] S. Devadas, A. Ghosh, and K. Keutzer, 'Logic Synthesis,' McGraw-Hill, New York, 1994.

[11] C.-S. Ding, C.-T. Hsieh, Q. Wu and M. Pedram, 'Stratified random sampling for power estimation,' in Proc. *IEEE/ACM Intl. Conf. on Computer-Aided Design*, pp. 577-582, Nov. 1996.

[12] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco, 'Testability Measures in Pseudorandom Testing,' in *IEEE Trans. on CAD*, vol. 11(6), pp. 794-800, June 1992.

[13] M. R. Garey and D. S. Johnson, 'Computers and Intractability. A Guide to the Theory of NP-Completness,' Freeman, New York, 1979.

[14] J. W. Green and K. J. Supowit, 'Simulated Annealing without Rejected Moves,' in *Digest. of Intl. Conference on Computer Design,* pp. 658-663, Oct. 1984.

[15] L. H. Goldstein, 'Controllability/Observability Analysis of Digital Circuits,' in *IEEE Trans. Circuits and Systems*, vol. CAS-26, pp. 685-693, Sept.1979.

[16] G. Hachtel and F. Somenzi, 'Logic Synthesis and Verification Algorithms,' Kluwer Academic Publishers, Boston. 1996.

[17] J. Hartmanis and H. Stearns, 'Algebraic Structure Theory of Sequential Machines,' Prentice-Hall, 1966.

[18] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski, 'The Design and Implementation of PowerMill,' in *Proc. Intl. Symp. on Low-Power Design,* pp. 105-110, April 1995.

[19] C. M. Huizer, 'Power Dissipation Analysis of CMOS VLSI Circuits by means of Switch-level Simulation,' in *IEEE European Solid State Circuits Conference*, pp. 61-64, 1990.

[20] M. Iosifescu, 'Finite Markov Processes and Their Applications,' John Wiley & Sons, 1980.

[21] B. Kapoor, 'Improving the Accuracy of Circuit Activity Measurement,' in *Proc. ACM/IEEE Design Automation Conference*, pp. 734-739, June 1994.

[22] G. Kemeny, J. L. Snell, and A. W. Knapp, 'Denumerable Markov Chains,' D. Van Nostrand Company, Inc., 1966.

[23] A. I. Khinchin, 'Mathematical Foundations of Information Theory,' Dover Publisher, New York, 1957.

[24] Z. Kohavi, 'Switching and Finite Automata Theory,' McGraw-Hill, New York, second edition, 1978.

[25] E. Macii, M. Pedram, and F. Somenzi, 'High-Level Power Modeling, Estimation and Optimization,' in *Proc. ACM/IEEE Design Automation Conference,* pp. 504-511, June 1997.

[26] D. Marculescu, R. Marculescu, and M. Pedram, 'Stochastic Sequential Machine Synthesis Targeting Constrained Sequence Generation,' in *Proc. ACM/IEEE Design Automation Conference*, pp. 696-701, June 1996.

[27] R. Marculescu, D. Marculescu, and M. Pedram, 'Switching Activity Analysis Considering Spatiotemporal Correlations,' in Proc. *IEEE/ACM Intl. Conference on Computer-Aided Design*, pp. 294-299, Nov.1994.

[28] R. Marculescu, D. Marculescu, and M. Pedram, 'Efficient Power Estimation for Highly Correlated Input Streams,' in *Proc. ACM/IEEE Design Automation Conference*, pp. 628-634, June 1995.

[29] R. Marculescu, D. Marculescu, and M. Pedram, 'Vector Compaction Using Dynamic Markov Models,' in *IEICE Trans. on Fundamentals*, Vol. E80-A, No.10, pp. 1924-1933, Oct. 1997, Japan.

[30] G. De Micheli, 'Synthesis and Optimization of Digital Circuits,' McGraw-Hill, New York, 1994.

[31] J. Monteiro and S. Devadas, 'Techniques for Power Estimation of Sequential Logic Circuits Under User-Specified Input Sequences and Programs,' in *Proc. Intl. Workshop on Low-Power Design,* pp. 33-38, April 1994.

[32] F. N. Najm, 'Transition Density: A New Measure of Activity in Digital Circuits,' *IEEE Transactions on CAD*, Vol. 12(2), pp. 310-323, Feb.1993.

[33] F. N. Najm, R. Burch, P. Yang, and I. Hajj, 'Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits,' *IEEE Transactions on CAD*, Vol. 9(4), pp. 439-450, April 1990.

[34] F. N. Najm, S. Goel, and I. Hajj, ''Power Estimation in Sequential Circuits,' in *Proc. ACM/IEEE Design Automation Conference*, pp. 635-640, June 1995.

[35] A. Papoulis, 'Probability, Random Variables, and Stochastic Processes,' McGraw-Hill, 1984.

[36] K. Parker and E. J. McCluskey, 'Probabilistic Treatment of General Combinational Networks,' in *IEEE Trans. on Computers*, vol. C-24, pp. 668-670, June 1975.

[37] M. Pedram, 'Power Minimization in IC Design: Principles and Applications,' in *ACM Transactions on Design Automation of Electronic Systems,* vol.1(1), pp.1-54, Jan.1996.

[38] J. Rabaey and M. Pedram, 'Low-Power Design Methodologies,' Kluwer Academic Publishers, 1996.

[39] J. Savir, G. S. Ditlow, and P. H. Bardell, 'Random Pattern Testability,' in *IEEE Trans. on Computers*, vol. C-33(1), pp. 79-90, Jan.1984.

[40] P. Schneider and U. Schlichtmann, 'Decomposition of Boolean Functions for Low-Power Based on a New Power Estimation Technique,' in *Proc. 1994 Int'l Workshop on Low-Power Design*, pp. 123-128, April 1994.

[41] E. Seneta, 'Non-Negative Matrices,' John Wiley & Sons, New York, 1973.

[42] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. Sangiovanni-Vincentelli, 'Sequential Circuit Design Using Synthesis and Optimization,' in *Proc. Intl Conference on Computer Design*, pp. 328-333, Oct. 1992.

[43] J. Storer, 'Data Compression: Methods and Theory,' Computer Science Press, 1988.

[44] K. Trivedi, 'Probability and Statistics with Reliability, Queueing, and Computer Science Applications,' Englewood Cliffs, NJ: Prentice-Hall, 1982.

[45] C.-Y. Tsui, M. Pedram, and A. M. Despain, 'Efficient Estimation of Dynamic Power Dissipation Under a Real-Delay Model,' in *Proc. IEEE/ACM Intl. Conference on Computer-Aided Design,* pp. 224-228, Nov. 1993.

[46] C.-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, and A. M. Despain, 'Power Estimation in Sequential Logic Circuits,' in *IEEE Transactions on VLSI Systems*, Vol. 3(3), pp. 404-416, Sept.1995.

[47] N. Weste, K. Eshraghian, 'Principles of CMOS VLSI Design: A Systems Perspective,' Addison Wesley Publishing Company, 1988.

[48] J. H. Wilkinson, 'The Algebraic Eigenvalue Problem,' Clarendon Press, 1988.