September 1997

UILU-ENG-97-2229
DAC 61

University of Illinois at Urbana-Champaign

Power Macromodeling for High Level Power Estimation

Subodh Gupta

Coordinated Science Laboratory
1308 West Main Street, Urbana, IL 61801

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-97-2229 (DAC 61)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) 1308 W Main St Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Power Macromodeling for High Level Power Estimation

12. PERSONAL AUTHOR(S)
Gupta, Subodh

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	97 Sep 23	82

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Power macromodel, average input signal probability, average input transition density, average input spatial correlation coefficient, average output transition density

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In this thesis, we propose a modeling approach that captures the d ependence of the power dissipation of a combinational logic circuit on its input/output signal switching statistics. The resulting power macromodel, consisting of a single four-dimensional table, can be used to estimate the power consumed in the circuit for any given input/output signal statistics. Given a low-level (typically gate-level) description of the circuit, we describe a characterization process by which such a table model can be automatically built. In contrast to other proposed techniques, this can be done for any given logic circuit without any user intervention, and applies to all possible input/output signal statistics; it does not require one to construct specialized analytical equations for the power dissipation. The macromodel can then be used during high-level power estimation in order to estimate the power of a combinational logic block once its input/output statistics have been computed in the context of its environment. High-level power estimation is important in order to provide the designer early warning of any power problems before much design effort/cost has been expended. The four dimensions of our table-based model are the average input signal probability, average input transition density, average spatial correlation coefficient and average output zero-delay transition density. This approach has been implemented and models have been built for many benchmark circuits. Over a wide range of input signal statistics, we show that this model gives very good accuracy, with an RMS error of about 4% and average error of about 6%. Except for one out of about 10,000 cases, the largest error observed was under 20%. If one ignores the glitching activity, then the RMS error becomes under 1%, the average error becomes under 5% and the largest error observed in all cases is under 18%.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

POWER MACROMODELING FOR HIGH LEVEL POWER ESTIMATION

BY

SUBODH GUPTA

B.Tech., Indian Institute of Technology, Kharagpur, 1995

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1997

Urbana, Illinois

ABSTRACT

In this thesis, we propose a modeling approach that captures the dependence of the power dissipation of a combinational logic circuit on its input/output signal switching statistics. The resulting *power macromodel*, consisting of a single four-dimensional table, can be used to estimate the power consumed in the circuit for any given input/output signal statistics. Given a low-level (typically gate-level) description of the circuit, we describe a characterization process by which such a table model can be automatically built. In contrast to other proposed techniques, this can be done for *any* given logic circuit without any user intervention, and applies to all possible input/output signal statistics; it does not require one to construct specialized analytical equations for the power dissipation. The macromodel can then be used during high-level power estimation in order to estimate the power of a combinational logic block once its input/output statistics have been computed in the context of its environment. High-level power estimation is important in order to provide the designer early warning of any power problems before much design effort/cost has been expended. The four dimensions of our table-based model are the *average input signal probability*, *average input transition density*, *average spatial correlation coefficient* and *average output zero-delay transition density*. This approach has been implemented and models have been built for many benchmark circuits. Over a wide range of input signal statistics, we show that this model gives very good accuracy, with an RMS error of about 4% and average error of about 6%. Except for one out of about 10,000 cases, the largest error observed was under 20%. If one ignores the glitching activity, then the RMS error becomes under 1%, the average error becomes under 5% and the largest error observed in all cases is under 18%.

ACKNOWLEDGMENTS

I am deeply indebted to my advisor, Prof. Farid Najm, for the immense help, guidance, support and encouragement he provided to me during the course of this research. He has been a tremendous mentor, and I consider my association with him one of the best things that has happened to me. I have learned from him more than the subject matter of this research.

Thanks to office-mates Mahadev Nemani, Sudhakar Bobba, Manish Goel, Nikos Bellas and Raj Hegde, for many 'interesting' technical and nontechnical discussions we had. Especially, I would like to thank Mahadev for his invaluable help. And thanks to all my other friends for all the good times we shared.

Finally, I thank my parents, brothers and sister for their love and support throughout my life and during the period of this thesis.

To my parents

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

As the minimum feature size of VLSI circuits is continuously shrinking, and the device densities are increasing several folds, their increased power dissipation has begun to pose serious concerns with regard to their reliability, and marketability.

In the case of microprocessor circuits, the computing power has been increased tremendously, primarily by increasing the device count and the clock frequency on the chip. On the other hand, efforts to push the computing throughput to its extreme are causing the architecture of the processors to become deeply pipelined. Heavy pipelining of the circuit means that the devices on the circuit participate more *actively* in the computations. All of the above situations, viz. the higher device count, the higher clock frequency and the higher switching rate lead to the increased power consumption of integrated circuits to undesirable levels.

The high levels of power dissipation result in high chip temperature, which if unchecked can lead to reliability problems. Also, it degrades the performance of the chip as some of the device parameters are affected by the temperature. Thus the chip might be forced to operate at speeds lower than what it was originally designed for. High power dissipation also implies high currents which may result in high current densities due to shrinking feature size. Thus the chip becomes more vulnerable to electromigration failures. In order to overcome these problems affecting *reliability* and performance, one has to resort to expensive packaging. This in turn affects the cost of the chip and hence *profitability.*

In case of digital signal processing chips and microprocessors used in portable applications, smaller and longer-life battery is desirable. This situation, again requires that

the circuits consume very low power. Also, in case of desktop applications there is a need for low power chips for keeping the cooling cost low.

Thus from both a performance and cost point of view it is desirable to design low-power VLSI circuits. This has resulted in an increasing interest in designing low-power VLSI circuits that are power efficient without compromising their performance while keeping their size small, i.e., there is an interest and a need to design VLSI circuits keeping the three objectives of area, delay and power in mind. This is in sharp contrast to the past design styles where designers often were satisfied by trading off delay for area or vice versa with no concern for power. The added dimensionality to the design space (now spanned by area, delay and power) is much more challenging as these three objectives often conflict with each other. This makes the problem of optimizing one parameter, while keeping the other parameters at the desired value, very challenging, and can be restricted with respect to other parameters. Particularly, the task of power optimization is felt to be significantly more complex than that of optimizing the area or speed of a circuit. This is so, not without some very substantial reasons. For one, the power dissipation of a circuit depends on a much larger set of circuit parameters than its area and delay do. Some of these parameters, for instance the input pattern, render the problem of power estimation inherently difficult, compounding the difficulties with the problem of power optimization.

Although the above challenges have attracted enough interest in the area of low-power circuit design, nonavailability of fast and accurate power estimation tools has impeded the research in this area. Designers have, hitherto, resorted to several approaches, such as power-down methods, voltage scaling and feature size reductions, besides *one-time* decisions with regard to power-efficient design alternatives, design styles, etc., based on experience. The need for power estimation tools exists at all levels of the design. At the circuit level, circuit simulators, like SPICE, can be used for estimating the power. However, these simulators are too slow and power estimation at this level can often be used only to verify that the design actually meets the specification. This is because it is too late in the design cycle to alter the design. It was only recently that the problem of

power estimation has been addressed satisfactorily, at the gate level (see [1] for survey). But there are still some issues that need to be resolved at the gate level. However, by the time the design has been specified down to the gate level, it may be too late or too expensive to go back and fix high power problems. Hence in order to avoid costly redesign steps, power estimation tools are required that can estimate the power consumption at a high level of abstraction, such as when the circuit is represented only by the Boolean equations. This will provide the designer with more flexibility to explore design trade-offs early in design process, reducing the design cost and time.

In response to this need, a number of power estimation techniques have been recently proposed (see [2] for survey) at higher levels of abstraction–namely, RT level, behavioral, algorithm and system levels. In the following discussion, we present a summary of the research in the area of high-level power estimation, targeting all levels of abstraction from the register transfer level (RTL) to the system level. The style of presentation here closely resembles [2].

1.1 Register Transfer Level Power Estimation

This level is the lowest level of abstraction in the domain of high-level power estimation. As a result, power estimation at this level has received a lot of attention lately. At this level of abstraction, the primitives are functional blocks such as adders, multipliers, controllers, register files and memories. The difficulty in estimating power at this level stems from the fact that the gate, circuit and layout level details of the design may not be available, thus making analysis of interconnect and clock distribution networks difficult. The power estimation techniques at this level can be broadly classified into two categories: top-down methods and bottom-up methods. These two approaches are discussed next.

1.1.1 Top-down methods

The top-down methods attempt to relate power consumption of a particular RTL description to quantities that describe the physical capacitance and activity of a design. However, these methods assume that no lower-level description (i.e., gate, circuit or layout levels) of the components of the design is available. Thus these techniques attempt to capture the complexity of design from RTL description. One such strategy was proposed in [3]. Here the complexity of the chip architecture was described in terms of "gate-equivalents," by expressing the functional blocks in terms of reference gates. The power required by each functional block can then be obtained by multiplying the approximate number of gate equivalents by the average power consumed by the a gate in the functional block under uniform white noise (UWN). One disadvantage of this technique is that all power estimates are based on the energy consumption of a single reference gate. This does not take into account different circuit styles and clocking strategies. This technique is particularly inaccurate for highly specialized blocks like memories.

Liu and Svennson [4] improved on this method [3] by applying different estimation techniques to different design entities: logic, memory, interconnect and clock. The logic component of power was estimated in a manner similar to [3]. The interconnect length and capacitance was modeled using Rent's rule. The clock capacitance was calculated by assuming an H-tree clock distribution network.

Both the above techniques are very easy to employ and they require very little information. However, they do not model circuit activity accurately. An overall (fixed) activity factor is assumed. In reality, however, activity factors vary with block functionality and with the data bring processed. So even if the user provides an activity factor that results in a good estimate of the total chip power, the predicted power breakdown is likely to be incorrect, making it hard to make meaningful architectural trade-offs.

The above-proposed techniques can be described as performing "rough synthesis" of the functional block in terms of reference gates. In [5] an attempt is made to solve the problem of power estimation without resorting to synthesis. However, in [5], only the

problem of estimating the activity of the functional block is addressed. These methods utilize information theoretic metrics like entropy to estimate the switching activity of the functional blocks.

In [6], the authors observe that the power is proportional to the product of the total physical capacitance and the average switching activity. In other words,

$$P \propto Capacitance \times Activity \tag{1.1}$$

Area and entropy of the circuit are employed as measures of its capacitance and activity respectively. It was shown that the average activity in a functional block can be estimated as a function of the input and output activities and is given by:

$$Activity \approx \frac{2/3}{n+m}(D_i + 2D_o) \tag{1.2}$$

In [7], the authors propose an area model for single-output Boolean function by employing a new measure called *average area cube complexity*. The above area model was further modified in [8], using two proposed measures called the *linear measure* and the *exponential measure*. In [9], the area model was extended to multioutput Boolean functions. This was achieved by transforming the given multioutput Boolean function into an equivalent single-output function. Then the area measure of the single output function was employed to determine the area of the multioutput function. Also, for the first time, activity and capacitance models were combined to give the power estimate at RTL.

1.1.2 Bottom-up methods

The bottom-up techniques, unlike the top-down methods, try to relate the power consumption of RTL components to fundamental parameters. The strategy here is to "measure" the power consumption of existing implementations and produce a model based on those measurements. In other words these techniques attempt to build power macromodels for power estimation. The power macromodel of a block can be used during high-level power estimation, in order to estimate the power dissipation of this block without performing a more expensive gate-level power estimation on it.

Clearly, this approach is best suited for designs that will be built using a library-based approach. This is true for digital signal processing applications. In [10], [11] a method called the Power Factor Approximation (PFA) was proposed. Here all the circuit input bits are treated as digital "white noise" and because of this assumption can give errors of up to 80% in comparison to gate-level tools. Another power macromodeling approach was proposed by Landman and Rabaey [12], [13]. This is based on the dual bit-type (DBT) model. The dual bit-type model is based on the observation that fixed-point and twos complement data streams are characterized by two distinct activity regions. The region corresponding to LSB exhibit activity similar to uniformly distributed white noise. The region corresponding to sign bits (MSB) exhibits an activity dependent on the temporal data stream correlation. Different empirically derived coefficients are used for these two regions. This results in greater accuracy, however, its main disadvantage is that it treats different modules differently, requiring specialized analytical expressions for the power, to be provided by the user. Thus depending upon the functionality of the module, a different type of macromodel (analytical equation) may have to be used.

In [14], Mehta et al. characterize the power dissipation of circuits based on input transitions rather than input statistics. Because the number of possible input transitions for an n-input combinational circuit is 2^{2n}, they present a clustering algorithm to compress the input transitions into clusters of input transitions that have approximately the same power values. They use heuristics to implement the clustering algorithm, but it is not clear how efficient the method would be on large circuits. In [15], Raghunathan et al. present a technique to estimate switching activity and power consumption at the RTL for data path and control circuits, in the presence of glitching activity. To construct a power macromodel, they use both analytical equations and look-up tables. The method is quite good and uses 9 or more variables in the power macromodel. In [16], Gupta and Najm also propose a look-up table-based approach to high-level power macromodeling with tables that have fewer variables (up to 3). The models of [15], [16] have the advantage of having a fixed template irrespective of the functionality of the block and hence are simple to generate and easy to use. Recently, in [17], the authors presented a

macromodel for estimating the cycle-by-cycle power at RTL. The proposed methodology consists of three steps: module equation form generation and variable selection, variable reduction, and population stratifications. The generated macromodel has 15 variables. They show good accuracy in estimating average and cycle-by-cycle power. However, the macromodels are dependent on a training set of vectors, which may result in erroneous power estimation if the training set is not similar to the vector set to be applied.

In this thesis we will present an extension of the approach discussed in [16].

1.2 Behavioral Power Estimation

At this level of abstraction, power estimation becomes even more difficult, as very little is known about the design at this level. The typical approach is to assume some architectural style or template and produce power estimates based on it. Some of the numerous unknowns that must be predicted include the foreground/background memory configuration, the number of memory accesses, the bus architecture and average wire length, the number of bus transactions, the control path complexity and the control line activity. It is apparent from the above that some of the parameters relate to the physical capacitance of the resources being accessed, while the others describe the activity of those resources. At this level of abstraction, activity associated with a resource type corresponds to the frequency of access of that resource type.

There are two ways of predicting activity for every resource type used in the behavior. One is called the static activity prediction [18], [19], as it performs a static analysis of the behavioral description to determine the access frequency. This approach has to resort to approximations in the presence of data dependent conditionals, branches and loops. The other approach, which goes by the name dynamic activity prediction [20], performs a simulation of the behavior from user defined input to determine the access frequencies of various modules. The disadvantage of this approach is that it is much slower than the static method and that it requires user specified input information.

To translate the activity measurements into power estimates, they have to be weighted with capacitance, switched per access of a given resource. For functional blocks like adders and multipliers, and for memories and registers, it is possible to get measured or estimated values of capacitance from design libraries. Predicting the dissipation of other components, such as I/O, busses, clocks and controllers is harder as their capacitance is strongly influenced by subsequent design phases. Ignoring their contribution can lead to erroneous power estimates. A popular approach adopted for these components is to generate an empirical model based on a large set of benchmark examples. This is the approach adopted in [18], [19]. A disadvantage with this approach is that it is design tool and design methodology dependent.

A more recent approach for estimating power at this level of abstraction has been presented in [21]. Here, an attempt was made at computing lower-bounds on the power dissipated by an algorithm. An equivalence between computation and communication was established by adopting an information-theoretic view of VLSI computation. More specifically, a DSP algorithm is viewed as a mechanism for information transfer with a requirement on the information transfer rate. An architecture implementing the algorithm is treated as a network with a certain capacity. A lower bound on the power dissipation is then the signal power that makes the channel capacity equal to the information transfer rate (reminiscent of Shannon's channel capacity theorem). However, more work has to be done in the proper characterization of "noise" affecting information transfer before this technique can be used to predict power.

1.3 Instruction Level Power Estimation

It is possible to realize behaviors in software on a programmable instruction set processor. In such a case the power dissipated by the behavior is given by the power dissipated by the software, when executing on the instruction set processor. Thus it makes sense to have an instruction-level power model. In [22], such a model was proposed for general purpose and DSP processors. Their model associates each instruction in the instruction

set with a base cost, which corresponds to the current drawn by the processor when executing this instruction placed in a loop. The model also accounts for interinstruction effects (corresponding to the change of state of circuit between two instructions) along with effects due to pipeline stalls and cache misses. It was noted in [22] that although the model is accurate for most instructions, it can produce significant errors on certain types of instructions due to the effect of operand activity. In order to account for this, a model that is activity sensitive is needed.

1.4 System Level Power Estimation

At the earliest stages of design specification we can consider performing system level power estimation. Here the objective is to come up with rough budgeting of power to all the components of the system which could include analog, digital, mixed signal portions. A power exploration tool at this level of abstraction would be useful for identifying power hungry parts of the design on which maximum optimization effort needs to be focussed. This helps reduce turn around time and minimizes redesign.

A tool called PowerPlay has recently been developed for design exploration at the system level [23] and can be found on the web at http://infopad.eecs.berkley.edu/PowerPlay. PowerPlay employs a spreadsheet-like interface to facilitate hierarchical design entry, rapid exploration of design partitionings, and parameter variations.

It is clear from the above discussion that there is an urgent need for high-level power estimation tools at all levels of abstraction, from the system level to the RT level. However, it is more important to solve the power estimation problem at the RT level before attempting to systematically solve the problem at higher levels of abstraction, as the solution to the power estimation at the RT level can be used to guide us to meaningful solutions at higher levels of abstraction.

1.5 Thesis Overview

In this thesis, we propose a modeling approach that captures the dependence of the power dissipation of a combinational logic circuit on its input/output signal switching statistics. The resulting *power macromodel*, consisting of a single four-dimensional table, can be used to estimate the power consumed in the circuit for any given input/output signal statistics. Given a low-level (typically gate-level) description of the circuit, we describe a characterization process by which such a table model can be automatically built. In contrast to other proposed techniques, this can be done for *any* given logic circuit without any user intervention, and applies to all possible input/output signal statistics; it does not require one to construct specialized analytical equations for the power dissipation. The macromodel can then be used during high-level power estimation in order to estimate the power of a combinational logic block once its input/output statistics have been computed in the context of its environment. High-level power estimation is important in order to provide the designer early warning of any power problems before much design effort/cost has been expended. The four dimensions of our table-based model are the *average input signal probability*, *average input transition density*, *average spatial correlation coefficient* and *average output zero-delay transition density*. This approach has been implemented and models have been built for many benchmark circuits. Over a wide range of input signal statistics, we show that this model gives very good accuracy, with an RMS error of about 4% and average error of about 6%. Except for one out of about 10,000 cases, the largest error observed was under 20%. If one ignores the glitching activity, then the RMS error becomes under 1%, the average error becomes under 5% and the largest error observed in all cases is under 18%.

The thesis is organized as follows. In Chapter 2 we will discuss the gate-level power estimation technique that will be used in the thesis for comparison purposes. Chapter 3 describes the power macromodeling approach. Chapter 4 describes the characterization process used for building the power macromodel. Chapter 5 gives the results showing the accuracy of our approach, and finally, some conclusions are presented in Chapter 6.

CHAPTER 2

GATE LEVEL POWER ESTIMATION

In the previous chapter we presented a motivation to the problem of high-level power estimation and also discussed current research trends in the area of high-level power estimation at various levels of abstraction. In this chapter we will provide a brief review of the issues involved in estimating power at the gate level of abstraction. This is because the power estimation techniques at the gate level of abstraction would be used to motivate a high-level power estimation methodology, to be discussed in Chapter 3.

2.1 Introduction

There are several important issues related to power estimation. These issues are concerned with the accuracy, speed and effectiveness of the solution, and have direct bearing on the power model, the estimation methodology, and the assumptions about the input information. In the following, we will bring out these different issues affecting the power estimation methodology.

The total average power dissipated by a static fully complementary CMOS implementation consists of two components: static and dynamic power. Thus,

$$P_{total} = \underbrace{P_{pn} + P_{sub}}_{static} + \underbrace{P_{sc} + P_{cap}}_{dynamic} \tag{2.1}$$

where P_{pn} is due to leakage in reverse-biased pn-junctions (source/drain and well regions), P_{sub} is due to subthreshold conduction current in MOSFETS, also called sub-threshold leakage, P_{sc} is due to the short-circuit current and P_{cap} is due to charging/discharging of circuit capacitors when a gate makes a logic transition.

The first two components in the above equation constitute the *static* power dissipation, which is independent of the switching of the transistors. Leakage power is due to the junction and substrate leakages, and is primarily dependent upon the device technology, threshold voltage and power supply voltage. Although this component is becoming more important in sub-micron technologies, its magnitude is negligible compare to other components of the power.

The short-circuit power P_{sc} is due to the rush-through currents that flow through the transistors when a direct path is established between V_{dd} and ground during switching. The input signal slope, the switching speed, the drive strength of the transistors, and the load driven determine the magnitude and duration of the short-circuit. The stringent design rules that are followed in most designs, especially in standard cell designs, ensure that this component is small by careful selection of signal slope, transistor size, and load values.

Finally, the capacitive component P_{cap}, a direct result of switching of the circuit nodes, is due to the effort required to charge and discharge the (capacitive) loads at the nodes, during their upward and downward transitions. The capacitances that are charged/discharged are primarily the diffusion capacitances of the driving gate, the input capacitances of the driven gates, and the capacitances of the interconnects. Thus, the capacitive power dissipated in a CMOS gate is directly dependent on the amount of switching of its output, and the capacitance that is switched, as given by the relation:

$$P_{cap} = \frac{1}{2} V_{dd} V C_{load} D \tag{2.2}$$

where V_{dd} is the supply voltage, V is the voltage swing at the node, C_{load} is the capacitance at the output of the node, and D is the *average* number of transitions per second at the node. The voltage swing, V, at the output of the CMOS gates equals V_{dd}, thus leading to a quadratic dependence of switching power on the supply voltage, as below.

$$P_{cap} = \frac{1}{2} V_{dd}^2 C_{load} D \tag{2.3}$$

At this juncture, it is pertinent to point out that the above expression is based on the notion that the transitions at the nodes are *complete*, and *not partial*. This is the result of

approximating the transition signals, which are really analog signals, to be digital signals. The resulting difficulty with such a notion is in counting the glitching transitions which do not rise/fall all the way to V_{dd} or ground potential. Although it might appear that this situation will result in either overcounting or undercounting of transitions, and hence overestimating or underestimating the switching power, very effective *filtering* techniques [24] have been proposed for adaptation in digital power estimation schemes.

It must be noted from the above expression that, except for D, every other term of the equation is known, given a gate level implementation of the circuit. The term D is dependent on the *input switching pattern.* Thus power estimation is pattern dependent. Moreover, there is wasteful power dissipation due to nodes making several transitions before evaluating to a final value, for a given input change. These spurious transitions have been referred to as *hazards* or *glitches.* These spurious transitions occur due to the difference in arrival times of inputs to a gate. Although the glitches do not contribute to the functionality of the circuit, they dissipate power. Once generated at a node, they can propagate through several gates in the node's fanout cone region. The propagation of glitches through a gate depends on its delay characteristics and the input combination itself. This makes it hard to capture the dependence of D on glitches. The above discussion points to two different types of transitions that can occur at the output of a gate, namely a zero-delay transition and a glitch, which are defined below.

Definition 1. (zero-delay transition) *A transition at the output of a gate in a circuit is said to be a zero-delay transition, provided that transition would have occurred if the primary input changes responsible for the transition were to occur simultaneously, and all the gates in the circuit were to have zero delays.*

In other words, the zero-delay transitions are the fewest transitions a node would undergo with a given sequence of primary inputs. These transitions are *legitimate* in the sense that they contribute toward the functionality of a circuit, assuming no redundancy in the circuit. The zero-delay transitions have the nice property of being independent of the physical parameters of the circuit, but dependent only on the input pattern and

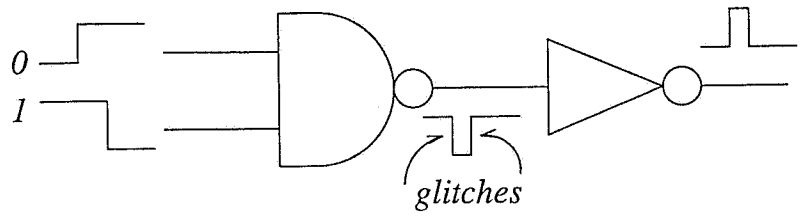

Figure 2.1 Illustration of glitch generation and propagation.

the Boolean functionality. This relation is given in the section on power models to be presented.

Definition 2. (glitch) *A transition is said to be a glitch if it is not a zero-delay transition.*

Figure 2.1 illustrates the generation of glitches at the output of a NAND gate and their propagation through another (INV) gate.

From the above classification of transitions that can occur at a node of a circuit, we can break down the power dissipated by a circuit into two parts, namely the zero-delay power and the glitch power. A power estimation at the gate level must estimate both the components in order to be considered a good power estimation scheme. In the next section we present a power model used in this thesis, to address the problems of pattern dependence and glitching.

2.2 Power Model

To get around the input pattern-dependence problem, one can use probabilities to describe the set of *all possible logic signals.* The work of [25] characterizes signals using two parameters, viz., *equilibrium probability* and *transition density*, based on a stochastic modeling of the logic signals. For convenience, we repeat the definition of these quantities below.

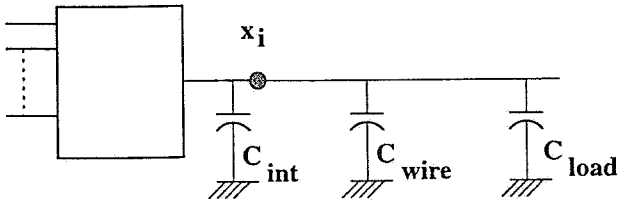

Figure 2.2 Capacitances at a node.

Definition 3. (equilibrium probability) *If $x(t)$ is a logic signal that can take values either 0 or 1, then its equilibrium probability is defined as:*

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

Definition 4. (transition density) *If a logic signal $x(t)$ makes $n_x(T)$ transitions in a time interval of length T, then the transition density of $x(t)$ is defined as:*

$$D(x) = \lim_{T\to\infty} \frac{n_x(T)}{T}$$

The transition density, which we shall refer to simply as density, provides an effective means to capture the switching activity in logic circuits. If the density $D(x_i)$ at every node x_i is made available, the overall average power dissipation in the circuit can be computed as:

$$P_{avg} = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{n} C_i D(x_i) \tag{2.4}$$

where C_i is the total capacitance at the node x_i, and is an aggregate of three components, C_{int}, C_{wire}, and C_{load}, as shown in Figure 2.2. C_{int} is the *internal* capacitance (also known as the *output* capacitance) of the driving gate connected at x_i, which is basically the diffusion capacitance of the drains. C_{wire} is the capacitance due to the interconnects between the driving and driven gates connected at x_i. C_{load} is the sum of the *gate* capacitances (also known as *input* capacitances) of the driven gates connected at x_i.

It is important to note that the transition density measure, used above, is general enough to account for every kind of transition at the output of a gate, whether it be a zero-delay transition or a glitching transition. In view of this, we choose to use (2.4)

as the power model for power estimation at the gate level of abstraction. Also we will propose a high-level power estimation model based on (2.4).

In spite of the apparent simplicity of the above power model, the problem of power estimation at the gate level is complicated by the fact that the estimation of probabilities and densities at the nodes of the circuit is sensitive to the stochastic dependencies (spatial and temporal) between the primary inputs. These stochastic dependencies are often referred to as spatial and temporal correlations. We now make the notion of spatial and temporal correlations more precise.

Definition 5. *(***spatial correlation***)* *When the occurrence of a certain logic state (0 or 1) is considered as an event, a signal x is said to be spatially correlated to another signal y if the occurrence of an event for one of the signals affects the probability of occurrence of an event on the other.*

Definition 6. *(***temporal correlation***)* *When the occurrence of a certain logic state (0 or 1) is considered as an event, a signal x is said to be temporally correlated if the probability of its event at a given time is affected by the occurrence of some event at some previous time.*

The aforementioned correlations arise because of the following situations:

- Two signals in a circuit become spatially correlated owing to their having a common ancestor node in the gate level structure.
- A signal 'remembering' its past behavior due to memory or feedback in the circuit leads to temporal correlation in that signal.

A number of techniques for VLSI power estimation at the gate level have been proposed [1] . These estimation techniques can be broadly classified into two categories, namely, *static methods* and *dynamic methods.* Both methods use probabilistic information such as probability and density.

The static methods attempt to propagate the input statistics to nodes in the circuit, calculating the corresponding statistics for these nodes. The statistics at the nodes

are used to obtain the required switching activity. The dynamic methods, on the other hand, use the input statistics to generate random input patterns, conforming to the given statistics, and apply these patterns in a simulation sequence. The node transitions are monitored and the simulation is stopped when a prescribed accuracy can be guaranteed with a certain degree of confidence.

The advantage of the static methods is that they can be made very fast (linear time in the number of nodes) under certain assumptions. However, as a result of these assumptions, and also due to not considering the delays of nodes and signals, they are less accurate than the dynamic methods. The dynamic techniques take longer but at the gain of much higher accuracy.

The correlations are not naturally accounted for in the static methods, thus resulting in less accurate power estimates. Several methods have been proposed to fix this problem with static methods [26] . However, this problem has not been addressed satisfactorily, and is a topic of ongoing research. On the other hand, correlations are naturally accounted for dynamic methods. In this research we use a simulation based power estimation tool [27] to estimate power at the gate level of abstraction, a dynamic method. This simulation technique is based on gate-level statistical simulation of the network under a user-specified delay model. The network is simulated with repeated application of random input vectors, generated from user specified input probabilities and densities, until some user-specified convergence criterion is satisfied. As signal correlations are automatically accounted for, this approach provides accurate power estimates.

The above mentioned simulation tool requires delay models for the gates in order to perform an accurate estimation of node probabilities and densities. In the following subsection we discuss this delay model.

2.3 Delay Model

The statistical simulation method uses libraries that have a precharacterized delay information. The delay of a gate is characterized using two parameters, an intrinsic delay,

t_{block}, which is independent of the load, and a load dependent delay factor, r_{load}, which is the delay per unit external capacitance load. If C_{ext} is the external load driven by the gate, then the gate delay is given as:

$$\text{gate delay} = t_{block} + r_{load} C_{ext} \tag{2.5}$$

Note that C_{ext} does not include the internal (output) capacitance of the gate, C_{int}, but is the aggregate of the wire capacitance, C_{wire}, and the input capacitances of the driven gates whose sum is C_{load}. The input capacitance of the gates are also specified, in addition to t_{block} and r_{load}. Moreover, separate values of t_{block} and r_{load} can be specified for rising and falling output conditions, and for each input to output pin combination.

CHAPTER 3

POWER MACROMODELING

In this chapter, we will discuss the parameters that are required to build the power maromodel for estimating the RTL power. Initially, we will discuss the parameters that can be chosen for making the power macromodel, assuming the primary inputs are independent. We then discuss the additional parameters that are required, when primary inputs are correlated.

What should a power macromodel look like? Which features are desirable and which are too expensive and infeasible? To begin with, it is clear that a macromodel should be simple to evaluate, otherwise there would be no advantage in using it and one might as well perform the analysis at the gate level. Furthermore, it must apply over the whole range of possible input signal statistics. Finally, it should consist of a fixed template, in which certain parameter values can determined by a well-defined and automatic process of *characterization*, without user intervention. We present a macromodel that has all these properties.

3.1 Power And Input Parameters Relationship

Before, building the macromodel we investigated the relationship between power and input parameters like average probability and average transition density (see Eq. (4.7) for definitions) of the primary inputs. Simulations were performed for different values of average input probability and average input density to determine the nature of their relationship with power. Figure 3.1 shows the plot of real-delay power for different values of average input probability and average input density for c6288 combinational

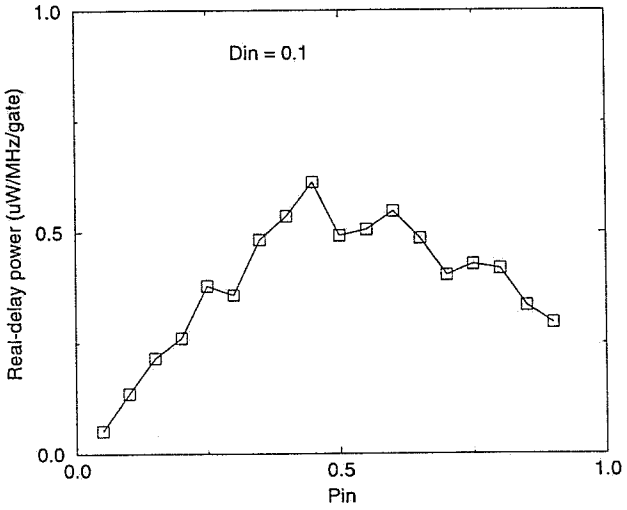

Figure 3.1 Plot of total power for $D_{in} = 0.1$ and different P_{in}.

benchmark circuit [28]. Figures 3.2 and 3.3 show the same plot for c3540 combinational benchmark circuit [28]. It can be seen that the relationship is nonlinear and the plots do not have a consistent shape. Similar results were obtained for other circuits. Hence an equation-based model was not used. In the next section the look-up table based approach for power macromodeling is described.

3.2 Power Macromodeling Assuming Independence

Because the power depends on the circuit input switching activity, it is clear that a power macromodel should take the input activity into account. The question is, however, exactly what information about the inputs should be taken into account and included in the macromodel. When the circuit being modeled is small (one or a few gates), then a simple modeling strategy is to create a table that gives the power for every possible input vector pair. In this case, there is no loss of accuracy. However, this strategy cannot be applied to large circuits. A circuit with 32 inputs will have 2^{64} possible input vector pairs, which would be prohibitively expensive to store in a table.

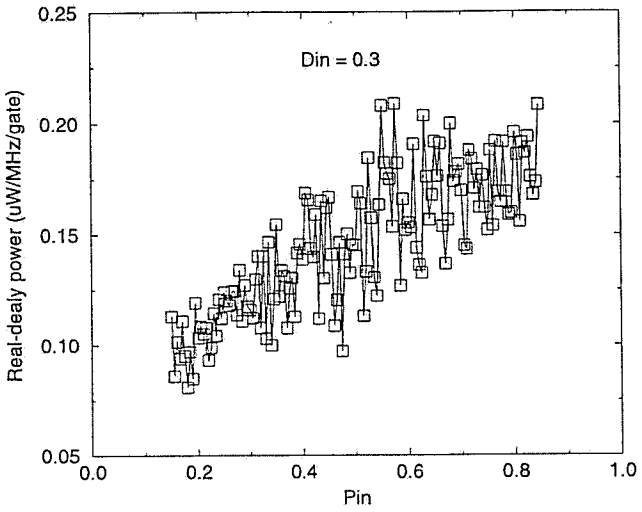

Figure 3.2 Plot of total power for $D_{in} = 0.3$ and different P_{in}.

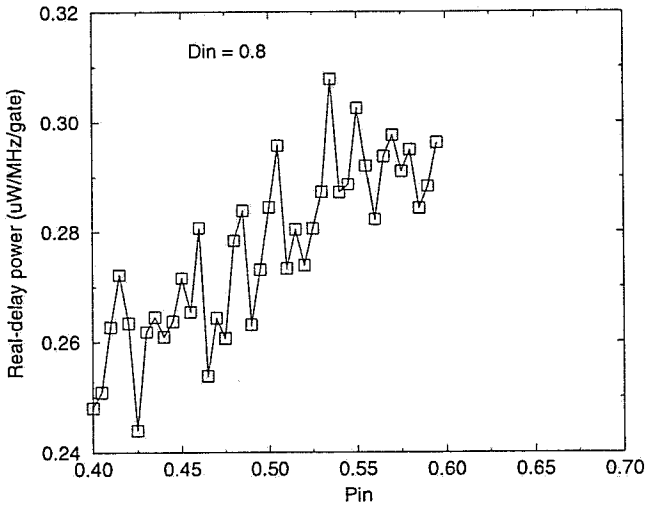

Figure 3.3 Plot of total power for $D_{in} = 0.8$ and different P_{in}.

Table 3.1 Details of ISCAS85 circuits

Circuit	Function	#inputs	#outputs	#gates
c432	Interrupt control	36	7	160
c880	ALU	60	26	383
c1908	Error correction	33	25	880
c2670	ALU and control	233	140	1193
c3540	ALU	50	22	1669
c5315	ALU	178	123	2307
c6288	Multiplication	32	32	2406
c7552	ALU	207	108	3512
c499	Error detection	41	32	202
c1355	Error detection	41	32	546

This leads to a trade-off between the amount of detail that one includes about the inputs and the accuracy resulting from the model. One possibility is to consider the signal probability $P(x_i)$ and transition density $D(x_i)$ at every input node x_i, and to build a model that depends only on these two variables. Notice that any information about correlations between the input nodes is lost when this is done. Thus, for instance, one could consider building a table which gives the power for every given assignment of input $P(x_i)$ and $D(x_i)$ values. Even in this case, however, such a table-based model would be too expensive, because a circuit with 32 inputs would require a 64-dimensional table.

Given the above observations, we have considered what aggregate compact descriptions of the $P(x_i)$ and $D(x_i)$ values would be sufficient to model the circuit power. For instance, one could consider building a two-dimensional table whose axes would be the average input $P(x_i)$, which we will denote by P_{in}, and the average input $D(x_i)$, to be denoted D_{in}. In this case, two different input assignments of $P(x_i)$ and $D(x_i)$ values, which may lead to different power values, may have the same P_{in} and D_{in} averages, and the table would predict the same power for both assignments, obviously with some error.

Table 3.2 Error in 2-d approach, when total power is estimated.

Circuit	P_{in}	D_{in}	RMS.Error	Max.Error
c432	0.4	0.4	1.61%	34.88%
c880	0.4	0.4	1.77%	40.46%
c1908	0.4	0.4	1.74%	16.80%
c2670	0.4	0.4	2.43%	-31.61%
c3540	0.4	0.4	2.96%	35.77%
c5315	0.4	0.4	1.76%	20.94%
c6288	0.4	0.4	16.6%	-40.04%
c7552	0.4	0.4	3.37%	19.02%

We have studied how big this error can be, as follows. Given a gate-level circuit and for a certain fixed P_{in} and D_{in}, we generate a large number (80 or more) of P and D assignments at the circuit inputs that each have averages equal to the specified P_{in} and D_{in}. We then perform an accurate power estimation for each assignment using a Monte Carlo gate-level (with full-delay model) simulation technique [27]. The average of the resulting power values is a good candidate value to store in the table. For each of the estimated power values, any deviation from this average value is considered to be an "error" relative to this table. The root-mean-square (RMS) and maximum errors for ISCAS85 circuits [28] (see Table 3.1 for details of these circuits) are reported in Table 3.2, for $P_{in} = 0.4$ and $D_{in} = 0.4$. A density of 0.4 means that the node makes an average of 4 transitions in 10 consecutive clock cycles. The largest RMS error is about 17% and the largest maximum error is -40%.

The power estimator (simulator) used to generate this table uses a scalable-delay timing model that depends on fanout and gate output capacitance. Thus, it captures the glitching power accurately (multiple transitions per cycle due to unequal delay from the inputs to an internal node). The glitching power is hard to account for in a high-level model. This is why such a high RMS error is seen for c6288, in which some internal nodes make up to 20 transitions per cycle. The errors improve considerably if the power estimates are based on a zero-delay timing model, in which the glitches are excluded, as

Table 3.3 Error in the 2-d approach, while estimating zero-delay power.

Circuit	P_{in}	D_{in}	RMS.Error	Max.Error
c432	0.4	0.4	0.59%	16.02%
c880	0.4	0.4	0.85%	27.5%
c1908	0.4	0.4	0.46%	-7.28%
c2670	0.4	0.4	0.92%	-18.82%
c3540	0.4	0.4	0.83%	-19.07%
c5315	0.4	0.4	0.47%	10.88%
c6288	0.4	0.4	0.72%	-16.82%
c7552	0.4	0.4	1.01%	-15.54%

shown in Table 3.3. The largest RMS error is now 1% and the largest maximum error is 27%.

In any case, with such a high RMS error in the general case, the total power estimation using Table 3.2 is too inaccurate. The simple 2-dimensional table approach is too simplistic. Another parameter is needed by which we can accurately model the variation of the power due to various input P and D assignments. We have found that if one more dimension is added to the table, reasonably good accuracy can be obtained. The third axis is the average output transition density over all the circuit output nodes, measured from a zero-delay (functional) simulation of the circuit, and which we will denote by D_{out}. The stipulation that D_{out} corresponds to zero-delay is not optional, but rather required for the following reason. We envision that during high-level, say RTL, power estimation, one would perform an initial step of estimating the signal statistics at the visible RTL nodes from a high-level functional simulation. These (zero-delay) statistics would then be applied to the power macromodel in order to estimate the power. Thus, the power model will be given by:

$$P_{avg} = f(P_{in}, D_{in}, D_{out}) \tag{3.1}$$

Table 3.4 Error in the 3-d approach, while estimating total power.

Circuit	P_{in}	D_{in}	D_{out}	RMS.Error	Max.Error
c432	0.4	0.4	0.44	0.97%	16.48%
c880	0.4	0.4	0.32	1.58%	27.87%
c1908	0.4	0.4	0.44	1.18%	12.71%
c2670	0.4	0.4	0.37	1.78%	-18.82%
c3540	0.4	0.4	0.44	1.94%	-20.33%
c5315	0.4	0.4	0.42	1.76%	17.16%
c6288	0.4	0.4	0.44	6.05%	-33.54%
c7552	0.4	0.4	0.42	2.97%	-15.67%

In order to study the accuracy in this 3-d approach, and to perform a direct comparison with Tables 3.2 and 3.3, we will show the errors in the estimation for the same $P_{in} = 0.4$ and $D_{in} = 0.4$ specifications as before. The value of D_{out} will naturally be different in different runs. For each circuit, we selected the largest subset of cases that has the same (approximately) D_{out} value and examined the errors based on the results in that subset. It is clear from Table 3.4 that the errors are much less now, and the RMS error in c6288 is now reduced to an acceptable 6%. The largest error of -33% is somewhat undesirable, but we will see later on that the spread of the error values over a wide range of input/output statistics is quite acceptable. For comparison with Table 3.3, the errors in the zero-delay power are given in Table 3.5. The RMS error is now below 0.77% and the maximum error is under about 12%.

3.3 Power Macromodeling For Correlated Inputs

In the previous section we assumed that the primary inputs are independent, but in practice the primary inputs are correlated. For example the primary inputs could be the output of another circuit block, which can be very highly correlated. In these situations the 3-dimensional table-based macromodel will give erroneous power values. Figure 3.4 compares the correlated and 3-d table-based power values for all ISCAS-85 circuits, over

Table 3.5 Error in the 3-d approach, while estimating zero-delay power.

Circuit	P_{in}	D_{in}	D_{out}	RMS.Error	Max.Error
c432	0.4	0.4	0.44	0.33%	4.90%
c880	0.4	0.4	0.32	0.55%	9.87%
c1908	0.4	0.4	0.44	0.19%	-3.23%
c2670	0.4	0.4	0.37	0.65%	-9.70%
c3540	0.4	0.4	0.44	0.47%	-12.37%
c5315	0.4	0.4	0.42	0.45%	6.32%
c6288	0.4	0.4	0.44	0.45%	-10.18%
c7552	0.4	0.4	0.42	0.77%	-8.82%

the wide range of P_{in}, D_{in}, and D_{out} values. An enlarged view of the lower section of the Figure 3.4 is shown in Figure 3.5. This comparison is also shown for different ISCAS-85 circuits in Figures 3.6 to 3.14. It can be seen from the figures that the 3-dimensional table-based macromodel gives erroneous estimate of the power when primary inputs are correlated. Table 3.6 gives the RMS, average and maximum error, when the inputs are correlated and the total power is estimated using the 3-d table-based macromodel, over a wide range of P_{in}, D_{in}, and D_{out} values, for all ISCAS-85 circuits. It can be seen from the table that RMS error is quite high. For c6288, it's around 42%. The average error is more than 15% for most of the circuits and in the case of c6288 it reaches 90.17%. The maximum error is more than 100% for most of the circuits, showing the inaccuracy of the 3-d table-based macromodel when total power of correlated input vectors is estimated. This led us to consider other parameters in our macromodel.

The primary inputs can be either temporally or spatially correlated. A signal x is said to be *temporally correlated* if its event (occurrence of certain logic state) at a given time is correlated to an event at some past time and is said to be *spatially correlated* to another signal y if their events are correlated. The question now arises as to what other parameters we should include in our 3-dimensional table-based macromodel, such that the temporal and spatial correlations at the primary inputs are accounted for while estimating the power at RTL.

Table 3.6 Error when total power of correlated inputs is estimated.

Circuit	RMS.Error	Average Error	Max.Error
c432	3.84%	35.5%	122.16%
c880	2.00%	16.26%	73.9%
c1908	3.73%	25.75%	114.78%
c2670	4.46%	27.08%	116.4415%
c3540	2.936%	20.59%	120.0089%
c5315	3.72%	21.72%	121.75%
c6288	41.4%	90.17%	226.64%
c7552	4.56%	28.73%	124.34%
c499	3.36%	43.15%	160.79%
c1355	2.846%	29.66%	134.7045%

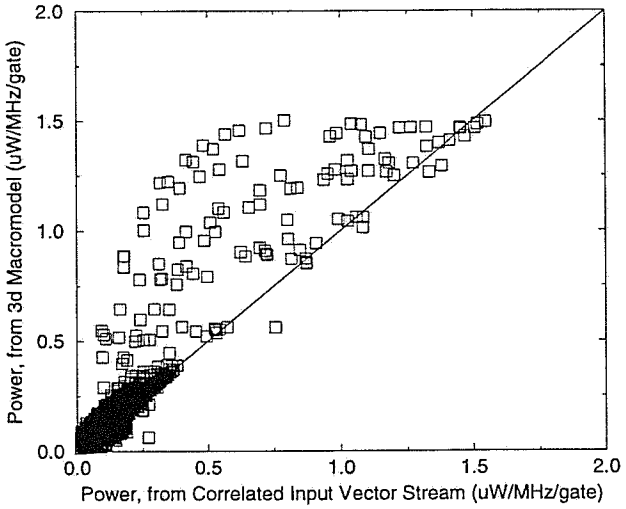

Figure 3.4 Power comparison while estimating total power.

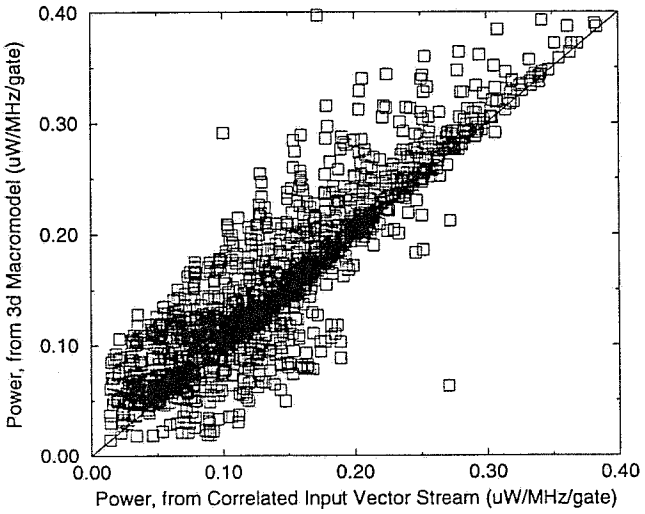

Figure 3.5 Power comparison while estimating total power.

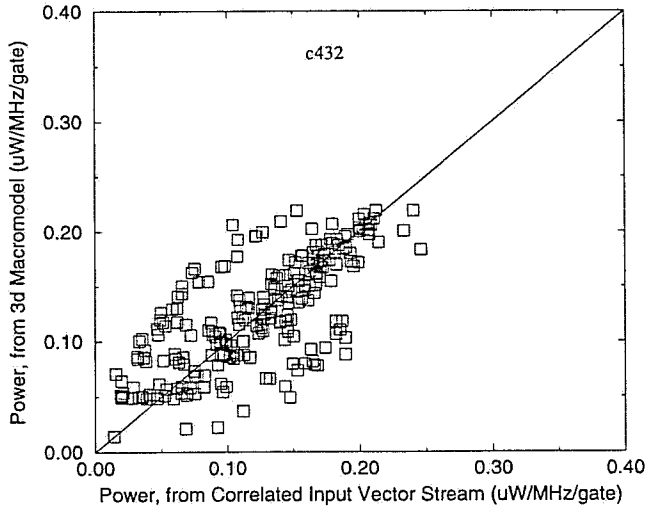

Figure 3.6 Power comparison for c432, while estimating total power.

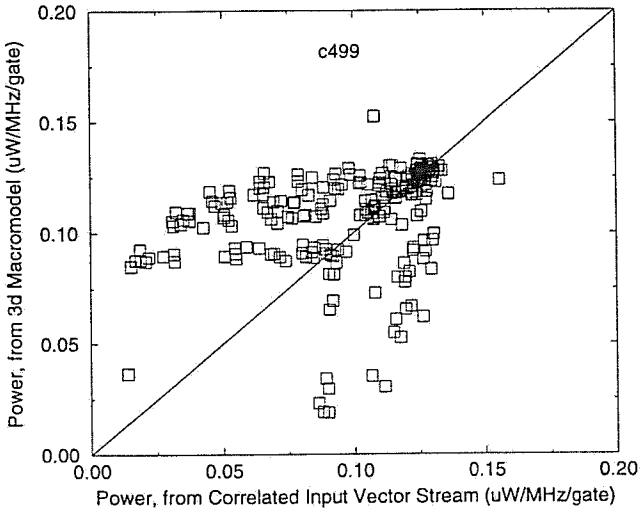

Figure 3.7 Power comparison for c499, while estimating total power.

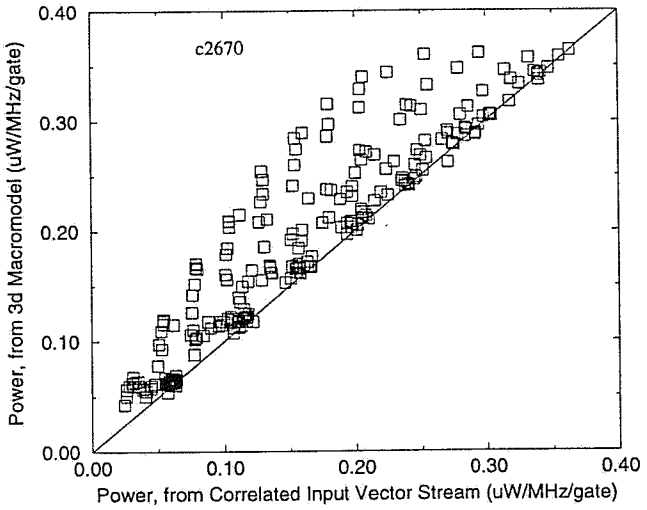

Figure 3.8 Power comparison for c2670, while estimating total power.

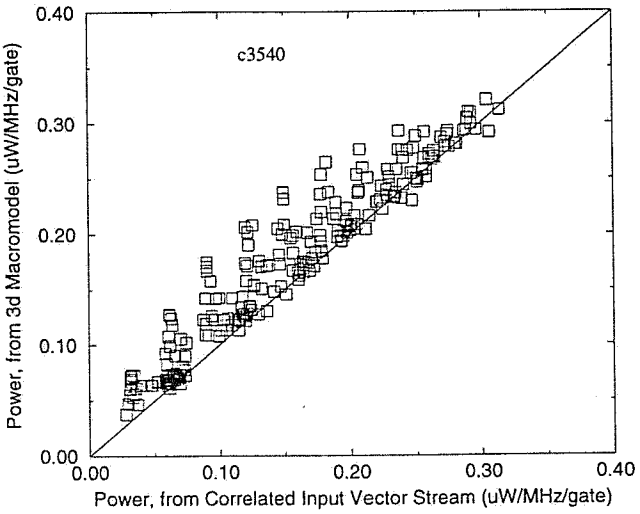

Figure 3.9 Power comparison for c3540, while estimating total power.

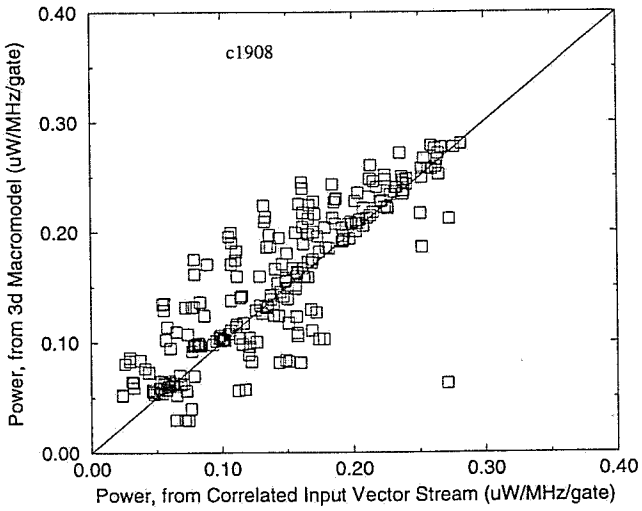

Figure 3.10 Power comparison for c1908, while estimating total power.

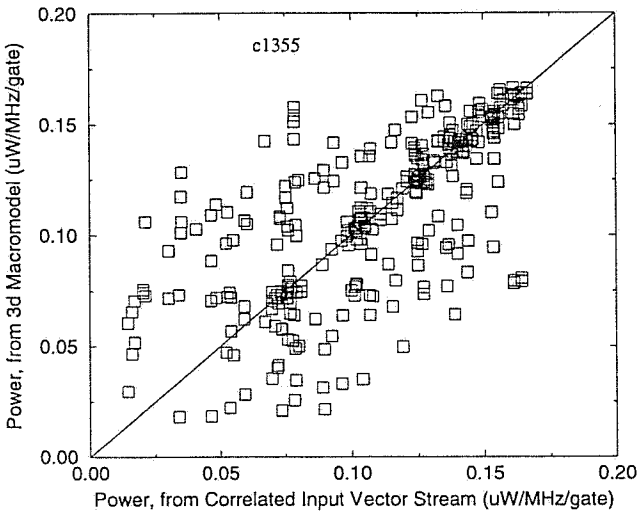

Figure 3.11 Power comparison for c1355, while estimating total power.

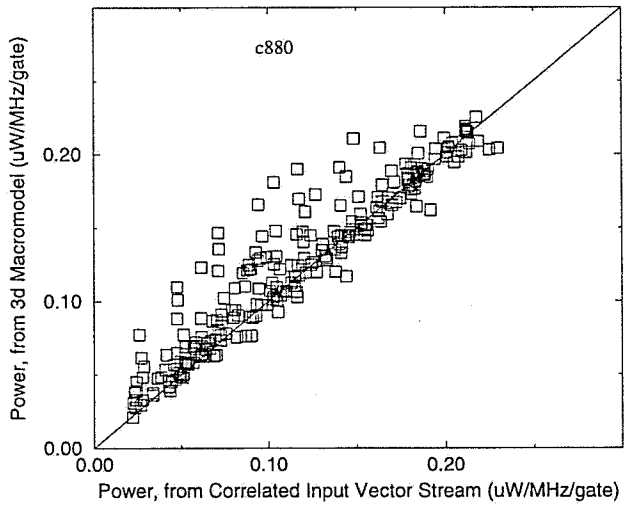

Figure 3.12 Power comparison for c880, while estimating total power.

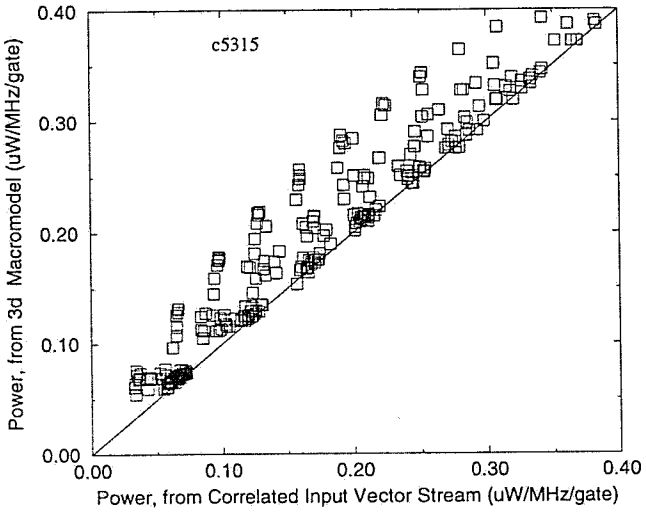

Figure 3.13 Power comparison for c5315, while estimating total power.

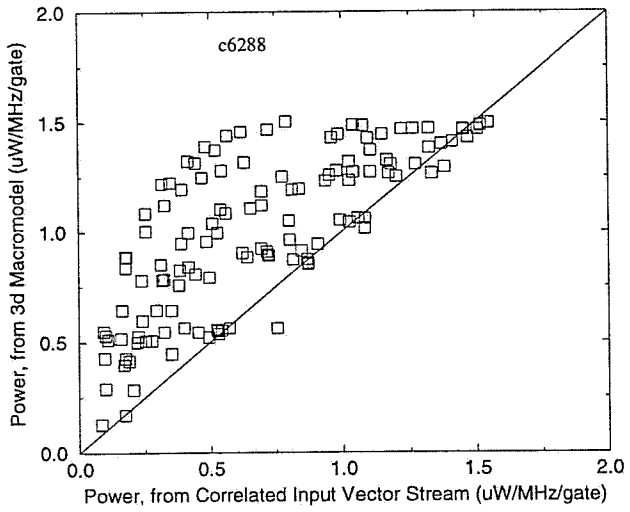

Figure 3.14 Power comparison for c6288, while estimating total power.

We will start our discussion by considering temporal correlation at the primary inputs. Here we will consider only lag-one temporal correlation. For the temporally correlated primary inputs, define TC_i for the ith input, as

$$TC_i = \mathcal{P}\left\{x_i^t \wedge x_i^{t-1} = 1\right\} \tag{3.2}$$

where $t-1$ and t are consecutive clock periods and where $\mathcal{P}\{\cdot\}$ denotes probability. Temporal correlation coefficient (γ_i) for ith input is defined as [29]:

$$\gamma_i = \frac{\mathcal{P}\left\{x_i^t \wedge x_i^{t-1} = 1\right\} - P(x_i)^2}{P(x_i)(1 - P(x_i))} \tag{3.3}$$

In Eq. (3.3) $P(x_i)$ is the probability at an input node x_i, which is known, as individual input probabilities are required to determine P_{in} for the 3-dimensional table based power macromodel and the only quantity which is unknown is $\mathcal{P}\left\{x_i^t \wedge x_i^{t-1} = 1\right\}$. Therefore, γ_i can be estimated accurately, if we can determine TC_i. But, we will show now that TC_i can be uniquely determined from the knowledge of P_i and D_i.

Proposition 1. *For any primary input node:*

$$TC_i = P_i - \frac{D_i}{2} \tag{3.4}$$

where TC_i, P_i and D_i are temporal correlation, signal probability and transition density, as described earlier, respectively.

PROOF. Let us denote probability of a high-to-high transition by P_{hh}, probability of a low-to-high transition by P_{lh} and probability of a high-to-low transition by P_{hl}.

$$P_i(t) = P_{lh} + P_{hh} \tag{3.5}$$

$$P_i(t-1) = P_{hl} + P_{hh} \tag{3.6}$$

But $P_i(t) = P_i(t-1) = P_i$.

$$\Rightarrow P_{lh} = P_{hl} \tag{3.7}$$

Transition density D_i can be expressed as:

$$\begin{aligned} D_i &= P_{lh} + P_{hl} \\ &= 2P_{lh} \\ &= 2\left(P_i - P_{hh}\right) \end{aligned} \tag{3.8}$$

$$\Rightarrow P_{hh} = P_i - \frac{D_i}{2} \tag{3.9}$$

where P_{hh} is nothing but TC_i. □

Therefore, temporal correlation at the primary inputs is taken care by P_i and D_i and we do not need an additional parameter to represent temporal correlation at the primary inputs.

Now we will discuss spatial correlation between the primary inputs. Spatial correlation is the pairwise correlation between every input. Here we will consider only first order correlation; higher order correlation are neglected. We define SC_{ij}, the spatial correlation between ith and jth inputs as:

$$SC_{ij} = \mathcal{P}\left\{x_i \wedge x_j = 1\right\}, \tag{3.10}$$

i.e., the probability of both inputs being high simultaneously.

The reason for considering SC_{ij} as the measure of spatial correlation coefficient follows from the definition of spatial correlation coefficient. Spatial correlation coefficient (ρ_{ij}) between two inputs is given by the following expression [29]:

$$\rho_{ij} = \frac{\mathcal{P}\left\{x_i \wedge x_j = 1\right\} - P\left(x_i\right)P\left(x_j\right)}{\sqrt{P\left(x_i\right)P\left(x_j\right)\left(1 - P\left(x_i\right)\right)\left(1 - P\left(x_j\right)\right)}} \tag{3.11}$$

The only unknown quantity in Eq. (3.11) is $\mathcal{P}\left\{x_i \wedge x_j = 1\right\}$, as we know $P\left(x_i\right)$ and $P\left(x_j\right)$. Therefore, it is evident from the Eq. (3.11) that if we can determine $\mathcal{P}\left\{x_i \wedge x_j = 1\right\}$, ρ_{ij} can be accurately estimated. From the definition given in Eq. (3.10), it is clear that SC_{ij} is the sufficient measure to capture ρ_{ij}. In the rest of this thesis, the spatial correlation coefficient between ith and jth primary input will be represented by SC_{ij}.

Now one possibility is to consider spatial correlation between every primary input pair of nodes as the parameter in our macromodel. But as the number of primary inputs increases, the number of parameters will also increase quadratically. Hence we cannot consider SC_{ij} between every primary input as the parameter in our table based power macromodel. We have found empirically that if we consider SC_{in} (*average spatial correlation coefficient*, i.e, average of all SC_{ij} terms), as the fourth parameter in our table based power macromodel, sufficient accuracy can be obtained for estimating the power of highly correlated primary inputs. Now, our table based power macromodel in presence of the fourth parameter looks as follows:

$$P_{avg} = f(P_{in}, D_{in}, SC_{in}, D_{out}) \tag{3.12}$$

CHAPTER 4

CHARACTERIZATION PROCESS

In the previous chapter, we discussed the parameters that are sufficient to build a power macromodel. In this section we will discuss how to characterize the macromodel for a given combinational logic circuit. First we will discuss the bounds for D_{in} and SC_{in} given P_{in}. After that we will discuss the algorithms to generate input vectors having the required P_{in}, D_{in} and SC_{in} values.

4.1 Bounds For D_{in} And SC_{in}

We assume that the combinational circuit is embedded in a larger sequential circuit, so that its input nodes are the outputs of latches or flip-flops and that they make at most one transition per clock cycle. We assume that the sequential design is a single clock system and ignore clock skew, so that the combinational circuit inputs $x_1, x_2, \ldots, x_n$ switch only at time 0.

At this point it is helpful to recall some definitions. The signal probability $P(x_i)$ at an input node x_i is defined as the average fraction of clock cycles in which the final value of x_i is a logic high. The transition density $D(x_i)$ at an input node x_i is defined as the average fraction of cycles in which the node makes a logic transition (its final value is different from its initial value). For brevity, in this section we will write P_i and D_i to represent $P(x_i)$ and $D(x_i)$. Both P_i and D_i are real numbers between 0 and 1.

Because the input signals x_i make at most a single transition per cycle, there is a special relationship between probability and density, given by:

$$\frac{D_i}{2} \leq P_i \leq 1 - \frac{D_i}{2} \tag{4.1}$$

The derivation of this property is rather simple, as follows:

From [25], $P(x)$ and $D(x)$ at a node x are given by

$$P(x) = \frac{\mu_1}{\mu_0 + \mu_1} \tag{4.2}$$

$$D(x) = \frac{2}{\mu_0 + \mu_1} \tag{4.3}$$

where μ_1 (μ_0) is the average of high (low), intertransition times of signal $x(k)$.

Using Eqs. (4.2) and (4.3), it is easy to arrive at:

$$\mu_1 = \frac{2P(x)}{D(x)} \tag{4.4}$$

$$\mu_0 = \frac{2(1 - P(x))}{D(x)} \tag{4.5}$$

Because time is discrete at the primary inputs, then $\mu_1 \geq 1$ and $\mu_o \geq 1$. Combining this with Eqs. (4.4) and (4.5) leads to the required result.

Eq. (4.1) can be rewritten as:

$$D_i \leq 1 - 2|P_i - 0.5| \tag{4.6}$$

so that for a given $P(x)$, $D(x)$ is restricted to the shaded region shown in Figure 4.1.

We also recall the definitions of the average input probability, denoted P_{in}, and average input density, denoted D_{in}, as follows:

$$P_{in} = \frac{1}{n}\sum_{i=1}^{n} P_i \qquad D_{in} = \frac{1}{n}\sum_{i=1}^{n} D_i \tag{4.7}$$

where n is the number of input nodes. It is clear from Eq. (4.1) that similar bounds hold for P_{in} and D_{in}:

$$\frac{D_{in}}{2} \leq P_{in} \leq 1 - \frac{D_{in}}{2} \tag{4.8}$$

from which we also have:

$$D_{in} \leq 1 - 2|P_{in} - 0.5| \tag{4.9}$$

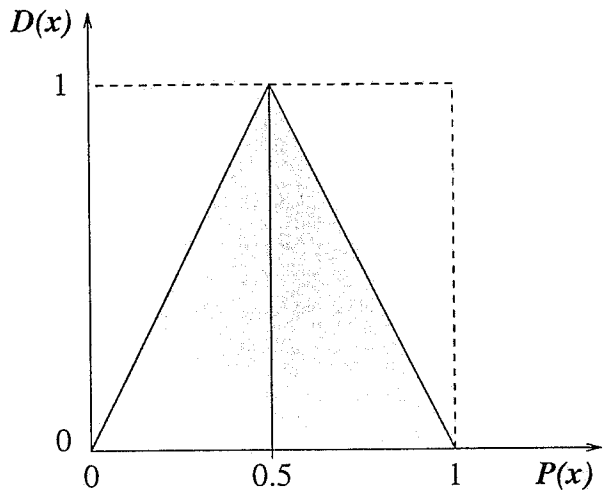

Figure 4.1 Relationship between density and probability

Similarly we have found a special relationship between SC_{in} and P_{in}, i.e., given P_{in} we can find lower and upper bounds for SC_{in}. Because SC_{in} is a probability it can take values only between 0 and 1. Before describing the bounds, we first recall the definition of SC_{in}:

$$SC_{in} = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathcal{P}\{x_i = 1, x_j = 1\} \tag{4.10}$$

where n is the number of primary inputs.

Let us consider that we have to generate a block of input vectors of size N, with each vector consisting of 1s and 0s, and let us represent the kth vector by V_k. SC_{in} can be written in terms of the input vectors as:

$$SC_{in} = \lim_{N \to \infty} SC_{in}^{N} \tag{4.11}$$

where:

$$\begin{aligned} SC_{in}^{N} &= \frac{1}{N} \sum_{k=1}^{N} \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_{i,k} x_{j,k} \\ &= \frac{2}{n(n-1)N} \sum_{k=1}^{N} \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_{i,k} x_{j,k} \end{aligned} \tag{4.12}$$

Notice that $\sum_{i=1}^{n}\sum_{j=i+1}^{n} x_{i,k}x_{j,k}$ = number of bit pairs, in kth vector, that are $(1,1)$. Therefore,

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n} x_{i,k}x_{j,k} = \frac{n_1(k)\left(n_1(k)-1\right)}{2} \tag{4.13}$$

where $n_1(k)$ = number of $1s$ in V_k.

By substituting Eq. (4.13) into Eq. (4.12), we get

$$SC_{in}^{N} = \frac{2}{Nn(n-1)}\sum_{k=1}^{N}\frac{n_1(k)\left(n_1(k)-1\right)}{2} \tag{4.14}$$

At this point, it will be helpful to define P_{in}^{N}. For a block of N vectors, P_{in} can be written as:

$$P_{in} = \lim_{N\to\infty} P_{in}^{N} \tag{4.15}$$

where:

$$P_{in}^{N} = \frac{1}{N}\sum_{k=1}^{N}\frac{n_1(k)}{n} \tag{4.16}$$

Notice that:

$$\frac{1}{N}\sum_{k=1}^{N} n_1(k) \approx nP_{in} \tag{4.17}$$

It can be shown that the minimum value of SC_{in}^{N} occurs when, for all k (see Appendix A for proof):

$$n_1(k) = nP_{in} \tag{4.18}$$

Therefore, a lower bound on SC_{in} is given by

$$SC_{in}^{N} \geq \frac{nP_{in}^{N}\left(nP_{in}^{N}-1\right)}{n(n-1)} \tag{4.19}$$

For large values of N, Eq. (4.19) becomes:

$$SC_{in} = \lim_{N\to\infty} SC_{in}^{N} \geq \frac{nP_{in}^{2}-P_{in}}{(n-1)} \tag{4.20}$$

Similarly, the upper bound on SC_{in}^{N} is given by the following expression:

$$SC_{in}^{N} \leq \frac{\sum_{i=1}^{m} n(n-1) + r(r-1)}{Nn(n-1)} \tag{4.21}$$

where, $r = \lfloor NnP_{in} - mn \rfloor$, and m is the largest integer $\leq N$ for which $0 \leq r \leq n$.

The proof is clear from the fact that maximum value of SC_{in} in Eq. (4.14) will occur when $n_1(k)$s are set to their maximum values, because of the quadratic term. Therefore, maximum value will be achieved by setting $n_1(k) = n$. Since not all $n_1(k)$s can be set to n due to (4.17), in Eq. (4.21) up to the mth vector, every vector contains $n_1(k) = n$, i.e., $i = 1 \ldots m, n_1(i) = n$. The $(m+1)$th vector contains the remaining ones and the remaining vectors contain all zeros.

Therefore,

$$0 \leq NnP_{in} - mn \leq n \tag{4.22}$$

$$\Longrightarrow 0 \leq P_{in} - \frac{m}{N} \leq \frac{1}{N} \tag{4.23}$$

$$\Longrightarrow \lim_{N\to\infty} \frac{m}{N} = P_{in} \tag{4.24}$$

For large value of N, and using Eq. (4.24), Eq. (4.21) becomes:

$$SC_{in} = \lim_{N\to\infty} SC_{in}^{N} \leq \lim_{N\to\infty} \frac{m}{N} + \lim_{N\to\infty} \frac{r(r-1)}{Nn(n-1)} = P_{in} \tag{4.25}$$

Combining the lower and upper bounds gives:

$$\frac{nP_{in}^{2} - P_{in}}{(n-1)} \leq SC_{in} \leq P_{in} \tag{4.26}$$

The relationship between P_{in} and D_{in} given by Eq. (4.8) will also hold for the block of input vectors.

The transition density at a primary input in terms of number of input vectors N is given by:

$$\begin{aligned} D(x_i) &= E\left[\frac{n_{x_i}(k)}{k}\right] \\ &= \lim_{k\to\infty} \frac{n_{x_i}(k)}{k} \\ &\approx \frac{n_{x_i}(N)}{N-1} \end{aligned} \tag{4.27}$$

From which D_{in} can be written as:

$$\begin{aligned} D_{in} &= \frac{1}{n}\sum_{i=1}^{n} D(x_i) \\ &\approx \frac{1}{n(N-1)}\sum_{i=1}^{n} n_{x_i}(N) \end{aligned} \tag{4.28}$$

where

$$n_{x_i}(N) = \sum_{k=1}^{N-1} x_i(k) \oplus x_i(k+1) \tag{4.29}$$

By substituting Eq. (4.29) into Eq. (4.28), D_{in} becomes

$$\begin{aligned} D_{in} &\approx \frac{1}{n(N-1)}\sum_{i=1}^{n}\sum_{k=1}^{N-1} x_i(k) \oplus x_i(k+1) \\ &= \frac{1}{n(N-1)}\sum_{k=1}^{N-1}\sum_{i=1}^{n} x_i(k) \oplus x_i(k+1) \end{aligned} \tag{4.30}$$

It can be shown that

$$|n_1(k) - n_1(k+1)| \leq \sum_{i=1}^{n} x_i(k) \oplus x_i(k+1) \leq n - |n - n_1(k) - n_1(k+1)| \tag{4.31}$$

from which bounds on D_{in} are given by:

$$\begin{aligned} \frac{1}{n(N-1)}\sum_{k=1}^{N-1} |n_1(k) - n_1(k+1)| \;\leq\; D_{in} \leq \\ \frac{1}{n(N-1)}\sum_{k=1}^{N-1} (n - |n - n_1(k) - n_1(k+1)|) \end{aligned} \tag{4.32}$$

Therefore, only those values of D_{in} which lie in the bound given by Eq. (4.32) can be realized.

Figure 4.2 shows the lower and upper bounds for SC_{in}, given P_{in}, i.e., given a value of P_{in}, there is a range of SC_{in} values which can be achieved, as given by Eq. (4.26). The shaded region in the figure shows the feasible region for P_{in} and SC_{in}. Shown in Figure 4.3 is the three-dimensional plot showing the relationship between P_{in}, D_{in}, and SC_{in}. The upper two shaded surfaces are the lower and upper bounds for SC_{in} for

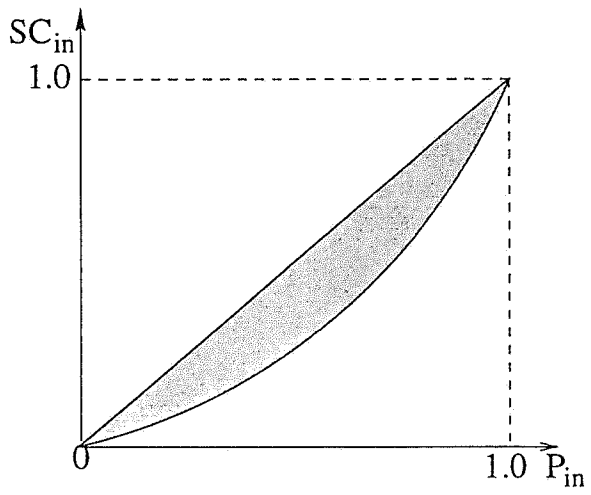

Figure 4.2 Relationship between probability and spatial correlation.

different values of P_{in} and D_{in}. It is evident from the figure that D_{in} does not have any effect on SC_{in}. The surface in the (P_{in} D_{in}) plane shows the relationship between P_{in} and D_{in} as given by the Eq. (4.8).

Thus, the 4-dimensional table with axes P_{in}, D_{in}, SC_{in} and D_{out} will not be completely full, and the choices of P_{in}, D_{in}, and SC_{in} during characterization will have to satisfy the above constraints (4.8) and (4.26). We subdivide the probability, density and spatial correlation axes between 0 and 1 into intervals of size 0.1, so that we form a $10 \times 10 \times 10$ grid in the $(P_{in}, D_{in}, SC_{in})$ plane. This choice is rather an arbitrary one, which we have found works well. Only about half of these points are valid, namely those that fall inside the shaded regions in Figures 4.2 and 4.3. Each valid grid point will correspond to a column of cells in the table along the D_{out} axis as shown in Figure 4.4.

For each valid grid point in the $(P_{in}, D_{in}, SC_{in})$ space, we generate a block of input vectors at the primary inputs such that the average probability, density and spatial correlation at the primary inputs equal to P_{in}, D_{in}, and SC_{in} respectively and that they satisfy the constraints (4.8) and (4.26). Using this block of input vector, the circuit power is computed using Monte Carlo power estimation [27], and the value of D_{out} is computed as the average of the individual (zero-delay) density values at the circuit outputs, also

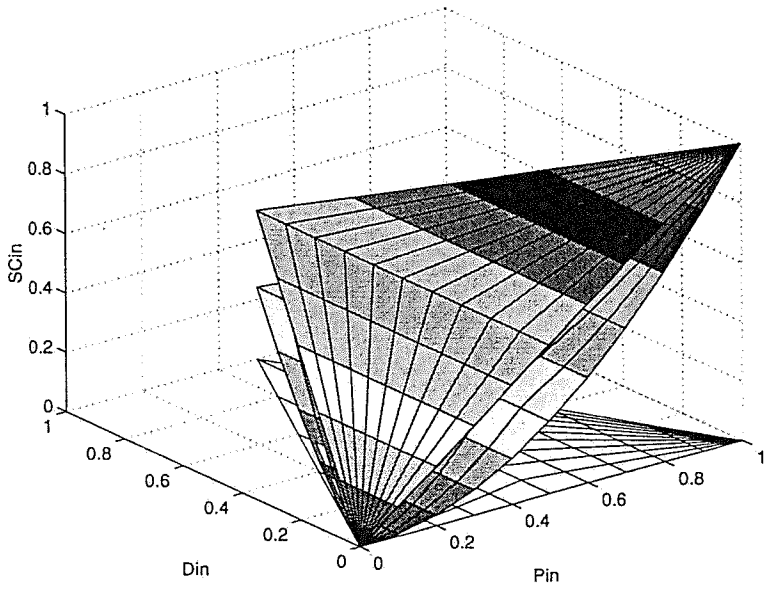

Figure 4.3 Relationship between probability, density and spatial correlation.

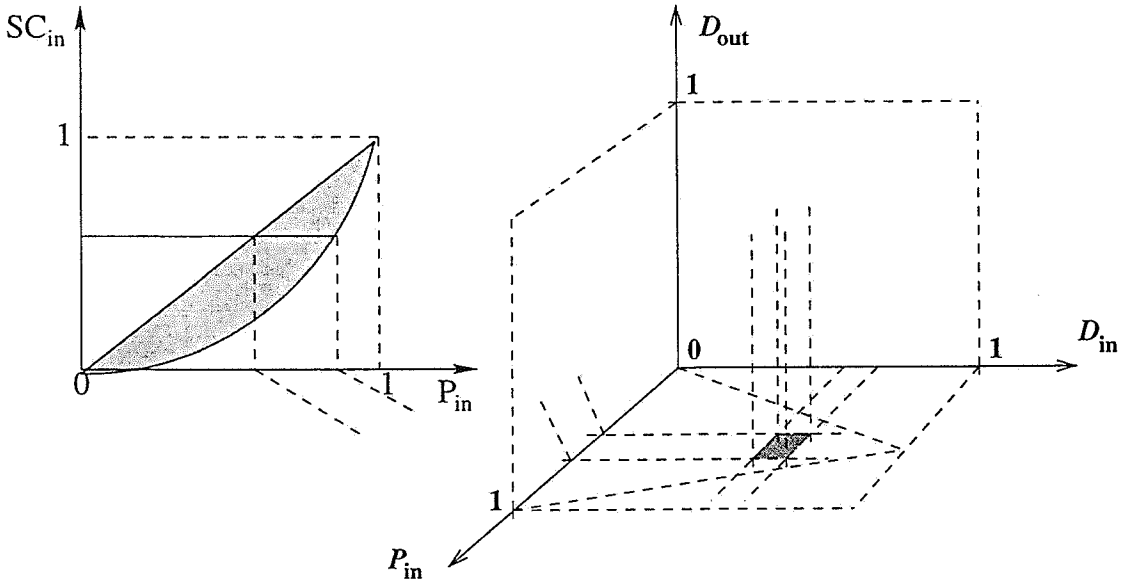

Figure 4.4 Relationship between probability, density and spatial correlation.

found during the Monte Carlo analysis. The value of D_{out} is rounded to the nearest grid point on the D_{out} axis, and the power value obtained is associated with the resulting cell location $(P_{in}, D_{in}, SC_{in}, D_{out})$ in the table. Eventually, a number of power values may be associated with a single cell in the table. At the end of the characterization, every cell is filled with the average of the power values associated with it. Some cells may have no power values associated with them, in which case their contents are left at zero. When it comes time to use the table, interpolation and extrapolation can be used to find the power for a $(P_{in}, D_{in}, SC_{in}, D_{out})$ combination which does not exist in the table. In the next section, we will show a number of results that demonstrate the accuracy of this approach over a wide range of input statistics, in which interpolation and extrapolation were used whenever required.

The above characterization process is straightforward, except for the generation of the block of input vectors at the primary inputs such that the average values of probability, density, and spatial correlation are equal to P_{in}, D_{in}, and SC_{in} respectively.

4.2 Generation of Input Vectors

Mathematically, the problem can be stated as to generate a block of N input vectors (as shown in Figure 4.5) such that they satisfy the following requirements:

$$\begin{aligned} P_{vec} &= P_{in} \\ D_{vec} &= D_{in} \\ SC_{vec} &= SC_{in} \end{aligned} \tag{4.33}$$

where P_{in}, D_{in}, and SC_{in} are the required average signal probability, average transition density and average spatial correlation coefficient, respectively, at the primary inputs which satisfy Eqs. (4.8) and (4.26). Similarly, P_{vec}, D_{vec}, and SC_{vec} are the averages obtained from the generated input vectors.

We have developed heuristic algorithms to generate the required block of input vectors. First we will present the algorithm for the generation of input vectors satisfying

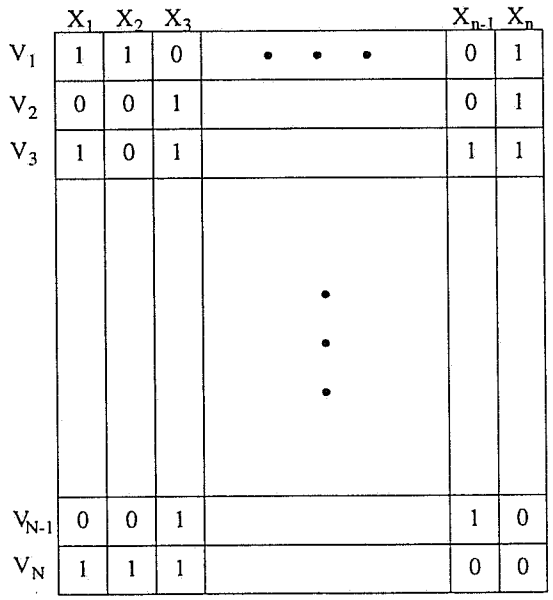

Figure 4.5 A block of N input vectors.

$P_{vec} = P_{in}$ and $SC_{vec} = SC_{in}$. Once we have these vectors we will describe an algorithm to modify these vectors in order to also satisfy the constraint $D_{vec} = D_{in}$.

4.2.1 Generation of vectors satisfying P_{in} constraint

First we will describe the algorithm for generating the vectors satisfying the P_{in} constraint. The algorithm (**Algorithm 1**) works as follows:

1. Find the required total number of ones by $T_{n_1} = nNPin$, based on Eq. (4.17).

2. Randomly choose the number of ones ($n_1(k)$) in each vector, based on a uniform distribution over $\{0, 1, \ldots, n\}$ for each $k = 1, 2, \ldots, N$.

3. Find $S_{n_1} = \sum_{k=1}^{N} n_1(k)$.

4. If $S_{n_1} < T_{n_1}$, increase the $n_1(k)$ in the following way:

   (a) Set $k = 1$.

   (b) If $S_{n_1} \neq T_{n_1}$, stop. Else, if $0 \leq n_1(k) \leq n$, do the following:

i. Set $n_1(k) = n_1(k) + 1$, $k = k + 1$ and $S_{n_1} = S_{n_1} + 1$

ii. If $k = N + 1$, set $k = 1$ and go to step (b).

(c) If $n_1(k) \geq n$, set $k = k + 1$ and go to step (b).

5. If $S_{n_1} > T_{n_1}$, decrease $n_1(k)$ in the following way:

(a) Set $k = 1$.

(b) If $S_{n_1} \neq T_{n_1}$, stop. Else, if $0 \leq n_1(k) \leq n$, do the following:

i. Set $n_1(k) = n_1(k) - 1$, $k = k + 1$ and $S_{n_1} = S_{n_1} - 1$

ii. If $k = N + 1$, set $k = 1$ and go to step (b).

(c) If $n_1(k) \leq 0$, set $k = k + 1$ and go to step (b).

6. Stop.

At the end of this process we will get a sequence of $n_1(k)$ values satisfying the P_{in} constraint.

4.2.2 Generation of vectors satisfying SC_{in} constraint

The main idea behind this algorithm is to increase the number of ones in some vectors and to decrease the number of ones in other vectors, while keeping the total number of ones constant, untill we achieve some level of accuracy.

The algorithm (**Algorithm 2**) works as follows:

1. Arrange the vectors in decreasing order of $n_1(k)$.

2. Find SC_{vec} using Eq. (4.14).

3. Set $i = 1$, $j = N$, $x = 1$ and $y = N$.

4. If $|x - y| < 1$, stop. Else set $i = x$, $j = y$ and continue.

5. If $SC_{vec} < SC_{in}$, do the following, untill $\frac{|SC_{vec} - SC_{in}|}{SC_{in}} < 0.01$:

(a) If $n_1(i) < n$, do the following:

i. If $n_1(j) > 0$, do the following:

A. $n_1(i) = n_1(i)+1$, $n_1(j) = n_1(j)-1$, $i = i+1$ and $j = j-1$. Calculate SC_{vec} using Eq. (4.14) and go to step 4.

ii. Else, do the following:

A. $j = j-1$, set $y = j$ and go to step 4.

(b) Else, do the following:

i. $i = i+1$, set $x = i$ and go to step 4.

6. Else, do the following, untill $\frac{|SC_{vec}-SC_{in}|}{SC_{in}} < 0.01$:

(a) If $n_1(i) > 0$, do the following:

i. If $n_1(j) < n$, do the following:

A. $n_1(i) = n_1(i)-1$, $n_1(j) = n_1(j)+1$, $i = i+1$ and $j = j-1$. Calculate SC_{vec} using Eq. (4.14) and go to step 4.

ii. Else, do the following:

A. $j = j-1$, set $y = j$ and go to step 4.

(b) Else, do the following:

i. $i = i+1$, set $x = i$ and go to step 4.

7. Stop.

From Eq. (4.14) it is clear that SC_{in} depends quadratically on $n_1(k)$. Therefore, for increasing SC_{vec}, increasing the number of ones of the vectors which have higher number of ones and decreasing the number of ones of the vectors which have the fewer number of ones will result in increase of SC_{vec}, as the increase in the square term will be more than the decrease. Similarly, for decreasing SC_{vec}, decreasing the number of ones of the vectors that have the higher number of ones and increasing the number of ones of the vectors which have the fewer number of ones will result in decrease of SC_{vec}. Hence the above algorithm.

At the end of this process we have an $n_1(k)$ sequence which satisfies the P_{in} and SC_{in} constraints. In the next section an actual vector sequence will be generated from the $n_1(k)$ sequence, while satisfying the third and final constraint.

4.2.3 Generation of vectors satisfying D_{in} constraint

This third algorithm is based on setting the number of transitions between the vectors, untill we reach the required number of transitions. The algorithm (**Algorithm 3**) works as follows:

1. Choose first vector randomly. Set $i = 2$ and $sum = 0$.
2. Do the following, until $i < N$:
   (a) Calculate the number of transitions required with the next vector, by using the following expression:

$$T_{vec} = n(i-1)D_{in} - sum \tag{4.34}$$

   (b) Calculate the minimum (t_{min}) and maximum (t_{max}) number of transitions of the $(i-1)$th vector with the remaining vectors by using the Eq. (4.31).
   (c) Choose the vector for which $t_{min} \leq T_{req} \leq t_{max}$, and set the number of transitions of the $(i-1)$th vector with ith vector as $nT(i-1) = T_{vec}$. Also set $sum = sum + T_{vec}$, and $i = i + 1$. Go to step 2.
   (d) If condition in step 4 is not satisfied, do the following steps:
      i. If $T_{vec} \geq nD_{in}$, choose the vector with the largest t_{max} and set $nT(i-1) = t_{max}$. Also set $sum = sum + t_{max}$ and $i = i + 1$. Go to step 2.
      ii. If $T_{vec} < nD_{in}$, choose the vector with the minimum t_{min} and set $nT(i-1) = t_{min}$. Also set $sum = sum + t_{min}$ and $i = i + 1$. Go to step 2.
   (e) Calculate $T_{tot} = n(N-1)D_{in}$. If $T_{tot} \neq \sum_{i=1}^{N-1} nT(i)$, increase or decrease the number of transitions between vectors in the following way:

i. If $T_{tot} > \sum_{i=1}^{N-1} nT(i)$, do the following:

A. Calculate the minimum and maximum vector of each vector with the immediate next vector using Eq. (4.31) and store it in $t_{min}(k)$ and $t_{max}(k)$ arrays.

B. Set $k = 1$.

C. If $k = N$, stop. Else continue.

D. Calculate $T_{diff} = T_{tot} - \sum_{i=1}^{N-1} nT(i)$.

E. If $t_{max}(k) - nT(k) \leq T_{diff}$, set $nT(k) = t_{max}(k)$ and go to step C.

F. Else, if $t_{min}(k) - nT(k) \leq T_{diff}$, set $nT(k) = nT(k) + T_{diff}$ and stop.

G. Else, set $k = k + 1$ and go to step C.

ii. If $T_{tot} < \sum_{i=1}^{N-1} nT(i)$, do the following:

A. Calculate the minimum and maximum vector of each vector with the immediate next vector using Eq. (4.31) and store it in $t_{min}(k)$ and $t_{max}(k)$ arrays.

B. Set $k = 1$.

C. If $k = N$, stop. Else, continue.

D. Calculate $T_{diff} = T_{tot} - \sum_{i=1}^{N-1} nT(i)$.

E. If $t_{min}(k) - nT(k) \geq T_{diff}$, set $nT(k) = t_{min}(k)$, $k = k + 1$ and go to step C.

F. Else, if $t_{max}(k) - nT(k) \geq T_{diff}$, set $nT(k) = nT(k) + T_{diff}$ and stop.

G. Else, set $k = k + 1$ and go to step C.

3. Stop.

In above algorithm, Eq. (4.34) is a general form of Eq. (4.30). The above algorithm does not work for all values of D_{in} and SC_{in}, especially when D_{in} and SC_{in} approach their maximum values. But, using this algorithm, we get a more random arrangement of vectors. For the cases in which this algorithm will not work, we have developed another

algorithm. The main disadvantage of this algorithm is that vector distribution is not random.

The second algorithm is based on first arranging the vectors for D_{min} and then moving the vectors to meet the D_{in} constraint.

The algorithm (**Algorithm 4**) works as follows:

1. For $i = 1, \cdots, N$, arrange $n_1(i)$ in descending order of number of ones. Calculate number of transitions from this vector arrangement by:

$$T_{vec} = \sum_{i=1}^{N-1} |n_1(i) - n_1(i+1)| \tag{4.35}$$

   This arrangement of vectors will have minimum number of transitions and hence D_{min}.

2. Calculate the required number of transitions (T_{req}) by using the following expression:

$$T_{req} = n(N-1)D_{in} \tag{4.36}$$

   and set $T_{diff} = T_{req} - T_{vec}$.

3. Set $i = 1$.

4. Until $T_{vec} < T_{req}$, repeat the following steps:

   (a) If $i > N$, go to step 5.

   (b) Calculate the maximum (T_{max}) and minimum (T_{min}) increase in transitions by moving Nth vector to $(i+1)$th position:

$$\begin{aligned} T_{max} &= [n - |n - n_1(i) - n_1(N)|] + [n - |n - n_1(i) - n_1(N)|] \\ &\quad - [|n_1(N) - n_1(N-1)|] - [|n_1(i) - n_1(i+1)|] \end{aligned} \tag{4.37}$$

$$\begin{aligned} T_{min} &= |n_1(i) - n_1(N)| - |n_1(i+1) - n_1(N)| \\ &\quad - |n_1(i) - n_1(i+1)| - |n_1(N) - n_1(N-1)| \end{aligned} \tag{4.38}$$

(c) If $T_{max} <= T_{diff}$, do the following steps:

i. Move the Nth vector to the $(i+1)$th position and move the remaining vectors one position down.

ii. Set the transitions between ith and $(i+1)$th by $nT(i) = n - |n - n_1(i) - n_1(i+1)|$. Similarly, set the transitions between $(i+1)$th and $(i+2)$th vectors.

iii. $i = i + 2$, and $T_{vec} = T_{vec} + T_{max}$.

(d) Else, if $T_{min} <= T_{diff}$, do the following steps:

i. Move the Nth vector to the $(i+1)$th position and move the remaining vectors one position down.

ii. Set the transitions between ith and $(i+1)$th to $nT(i) = \frac{T_{diff}}{2}$. Similarly, set the transitions between $(i+1)$th and $(i+2)$th vectors to $nT(i+1) = T_{diff} - \frac{T_{diff}}{2}$.

iii. $i = i + 2$, and $T_{vec} = T_{vec} + T_{diff}$.

iv. For $j = i, \cdots, N-1$, set $nT(j) = |n_1(j) - n_1(j+1)|$

(e) Else, do the following steps:

i. $nT(i) = |n_1(i) - n_1(i+1)|$.

ii. $i = i + 1$.

5. Stop.

Figures 4.6 and 4.7 show the vector distribution for Algorithms 3 and 4 respectively. The X-axis shows the number of vectors and the Y-axis shows the number of 1s in each vector. It can be seen from the figures that vector distribution in the case of Algorithm 3 is more random than that of Algorithm 4, showing the advantage of Algorithm 3.

After finding the number of transitions, 1s and 0s are arranged to get the desired number of transitions between every vector pair. This way, at the end, we will get a block of vectors that satisfy P_{in}, SC_{in}, and D_{in} constraints.

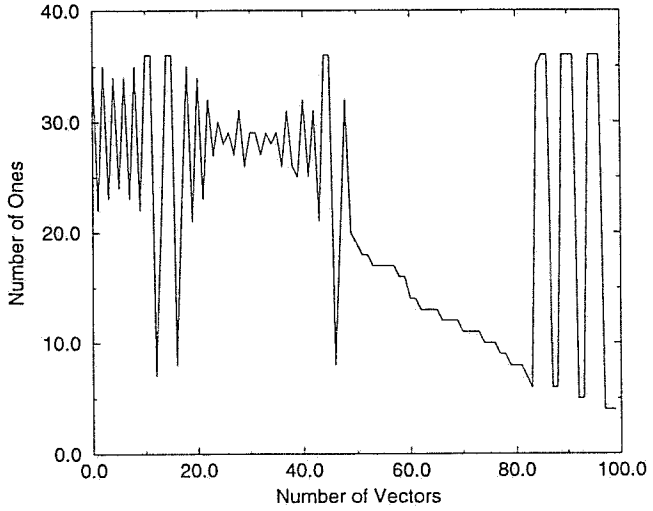

Figure 4.6 Distribution of number of ones for Algorithm 3.

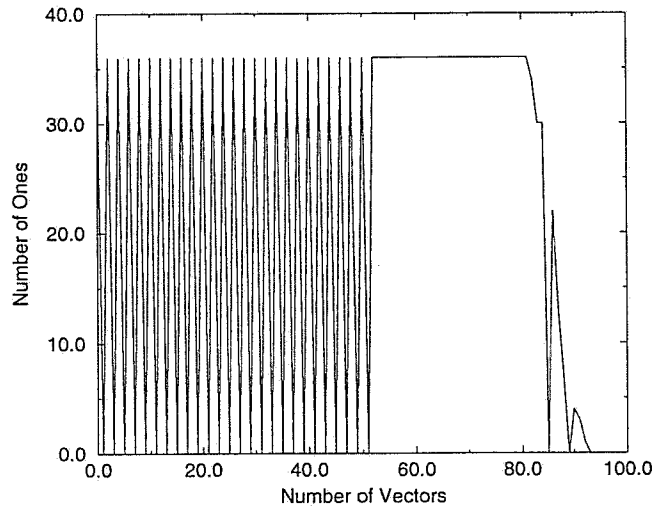

Figure 4.7 Distribution of number of ones for Algorithm 4.

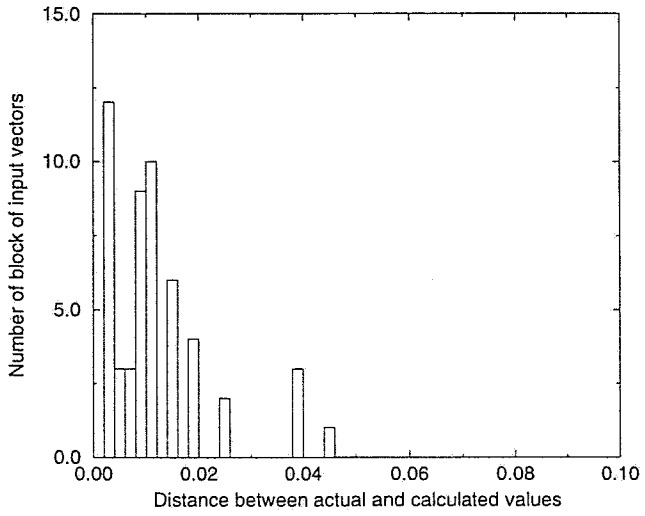

Figure 4.8 Distribution of distance.

Figure 4.8 shows the histogram of Euclidian distance between $(P_{in}, D_{in}, SC_{in})$ and $(P_{vec}, D_{vec}, SC_{vec})$, for blocks of input vector of size N =100, over a wide range of P_{in}, D_{in} and SC_{in} values. It is evident from the figure that for all of the cases the distance is near zero and maximum error is under 5%, thus demonstrating the accuracy of these algorithms.

CHAPTER 5

MODEL ACCURACY EVALUATION

In this chapter, we report the results of our 4-dimensional power macromodeling approach on the ISCAS-85 circuits. We have implemented this approach and built the power macromodels (4-dimensional look-up tables) for a number of combinational circuits. In order to study the accuracy over a wide range of signal statistics, we randomly generated block of input vectors at the circuit inputs while covering a wide range of P_{in}, D_{in}, and SC_{in} values. Because Eqs. (4.8) and (4.26) must be enforced, any case that violated these constraints was rejected. Approximately 1000 such valid blocks of input vectors were generated this way for every ISCAS-85 circuits, for which the power was estimated from gate-level Monte Carlo simulation; the Monte Carlo simulation also provides accurate estimation of D_{out}. The power values predicted by the look-up table were compared to those from simulation, and the RMS, absolute average and maximum errors were computed.

The results are summarized in Table 5.1 for the case when total power is estimated. The RMS, absolute average and maximum errors are computed over the 1000 different cases for every ISCAS-85 circuits, covering the wide range of P_{in}, D_{in}, and SC_{in} values. Over a wide range of statistics, it is seen that the RMS error is very good, under about 5%. The largest maximum error is at 22.56% for c432, because the estimated power value is very small and a slight difference in power value causes a lot of error. The average error in all cases is less than 6%, which shows the accuracy of our macromodeling approach. The combined scatter plot of all ISCAS-85 circuits showing the accuracy of 4-d approach while estimating the total power is shown in Figure 5.1. An enlarged view of the lower section of this plot is given in Figure 5.2. Both these plots report the normalized power

Table 5.1 Error in the 4-d approach, when total power is estimated.

Circuit	RMS.Error	Average Error	Max.Error
c432	0.868%	5.56%	22.56%
c880	0.647%	3.73%	14.64%
c1908	0.729%	3.85%	16.89%
c2670	0.738%	3.08%	11.52%
c3540	0.802%	3.61%	16.53%
c5315	0.612%	2.48%	-14.58%
c6288	4.14%	3.75%	18.23%
c7552	0.847%	3.03%	-16.58%
c499	0.497%	4.05%	16.4%
c1355	0.5167%	4.19%	15.6%

values, so that the results for all the circuits can be examined on the same plot. The scatter plots of all ISCAS-85 circuits while comparing the total power, obtained from 4-dimensional power macromodel and simulation, are shown in Figures 5.3-5.12.

For completeness, the accuracy of the macromodels when zero-delay power is estimated is shown in Table 5.2 and in the scatter plot in Figure 5.13. Over a wide range of signal statistics, the RMS error is below 0.60%, the average error is under 5% and the maximum error is under 18%. The scatter plot also shows excellent agreement. Similarly, the scatter plots of all ISCAS-85 circuits while comparing the zero-delay power, obtained from 4-dimensional power macromodel and simulation, are shown in Figures 5.14-5.23.

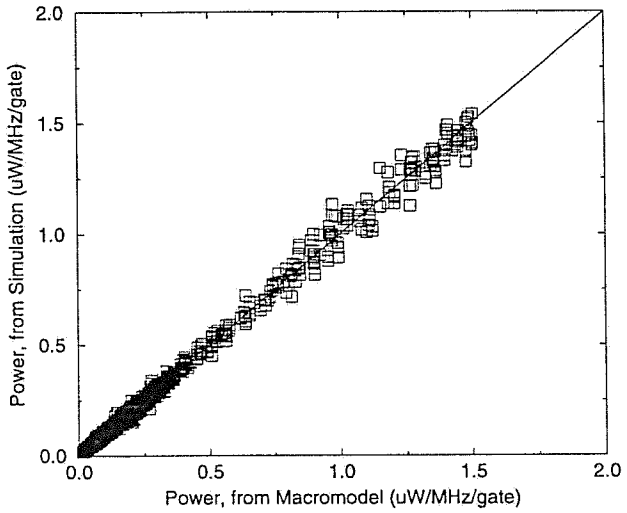

Figure 5.1 Power comparison for ISCAS-85 circuits while estimating total power.

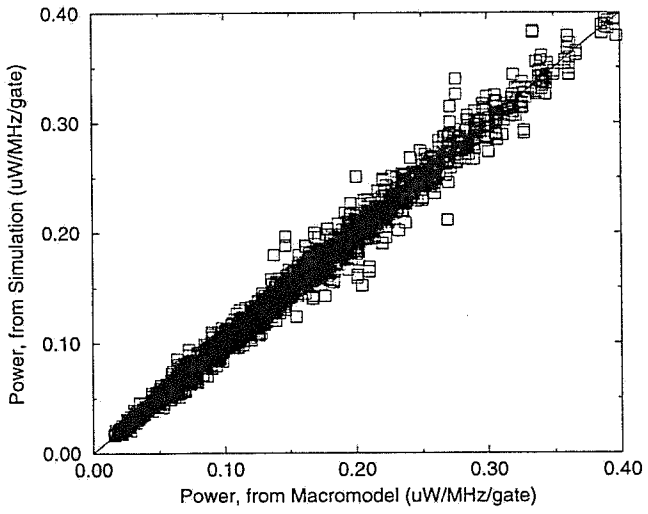

Figure 5.2 Power comparison for ISCAS-85 circuits when total power is estimated.

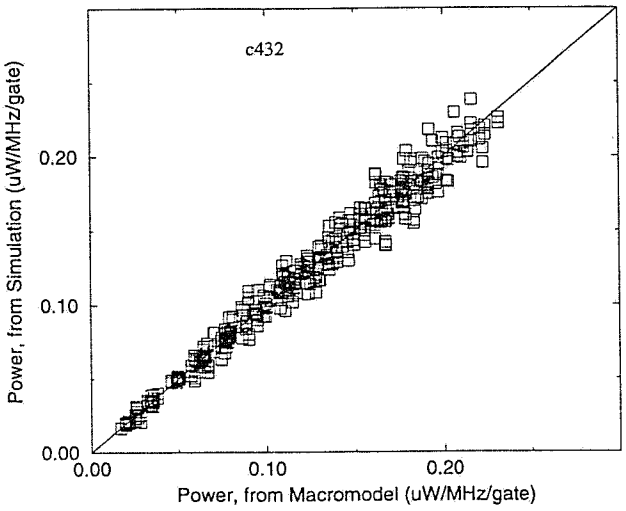

Figure 5.3 Power comparison for c432, while estimating total power.

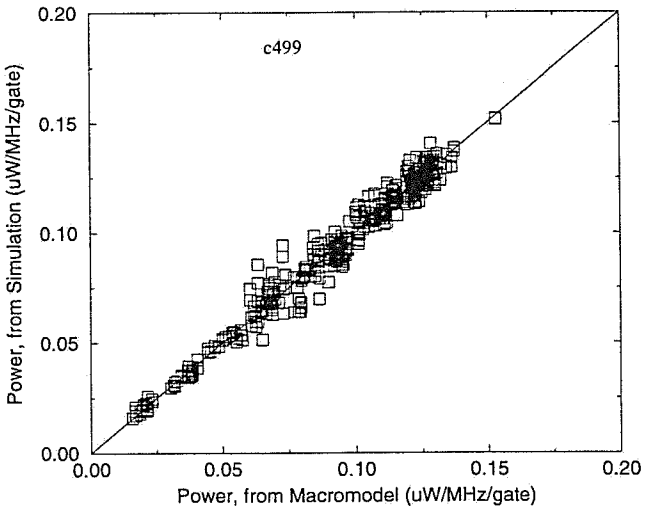

Figure 5.4 Power comparison for c499, while estimating total power.

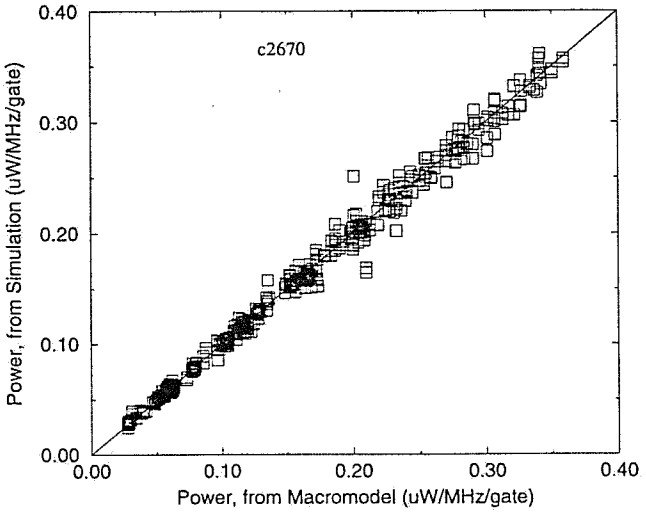

Figure 5.5 Power comparison for c2670, while estimating total power.

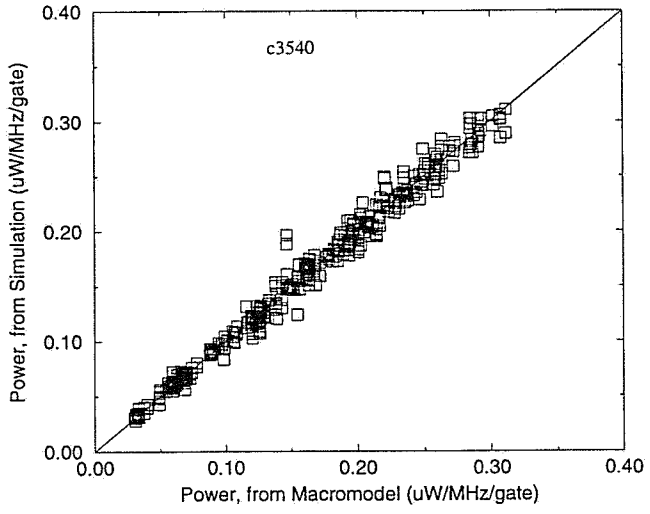

Figure 5.6 Power comparison for c3540, while estimating total power.

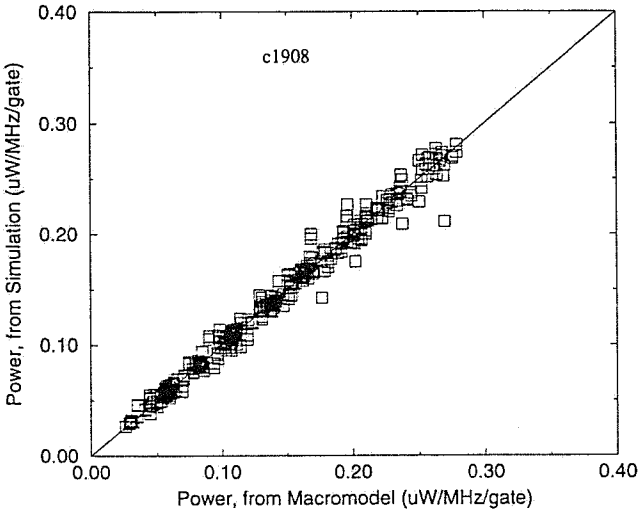

Figure 5.7 Power comparison for c1908, while estimating total power.

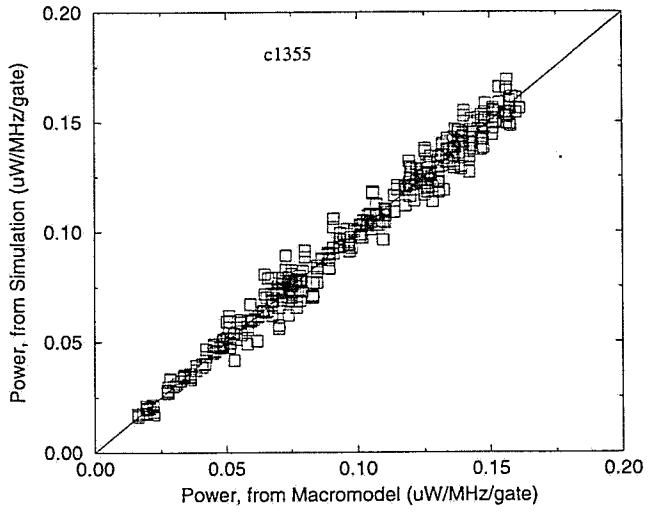

Figure 5.8 Power comparison for c1355, while estimating total power.

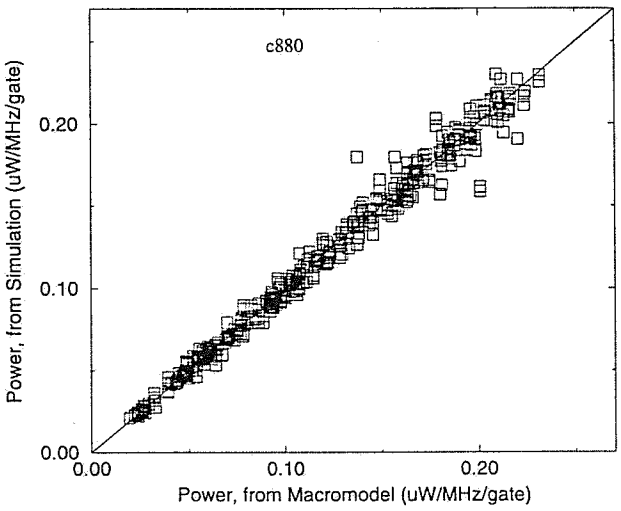

Figure 5.9 Power comparison for c880, while estimating total power.

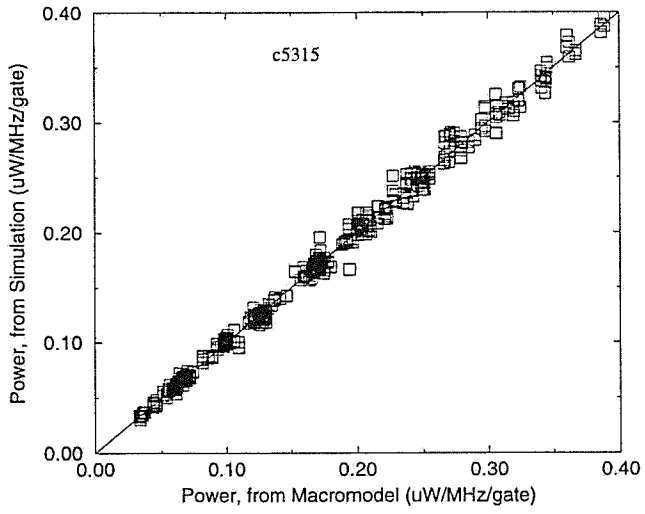

Figure 5.10 Power comparison for c5315, while estimating total power.

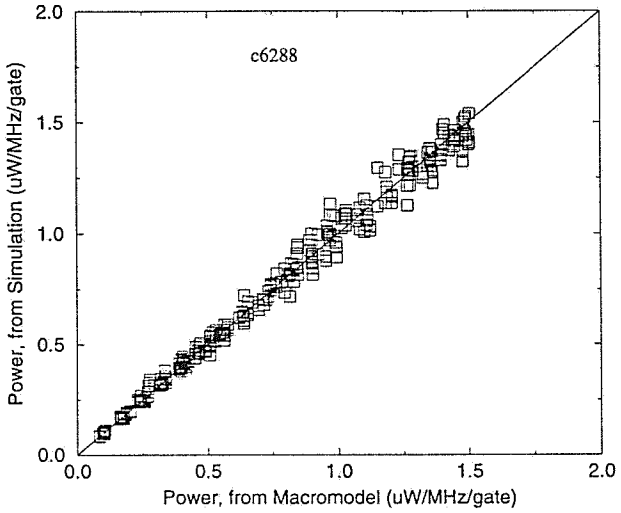

Figure 5.11 Power comparison for c6288, while estimating total power.

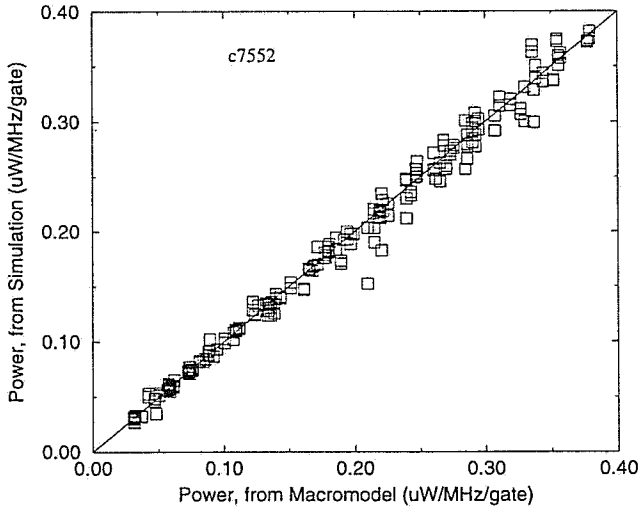

Figure 5.12 Power comparison for c7552, while estimating total power.

Table 5.2 Error in the 4-d approach, while estimating zero-delay power.

Circuit	RMS.Error	Average Error	Max.Error
c432	0.428%	4.409%	17.35%
c880	0.519%	3.62%	13.97%
c1908	0.461%	3.73%	15.69%
c2670	0.307%	2.18%	10.16%
c3540	0.413%	3.22%	15.55%
c5315	0.29%	2.08%	-12.20%
c6288	0.332%	2.218%	17.37%
c7552	0.23%	2.65%	-14.32%
c499	0.45%	3.95%	16.34%
c1355	0.383%	4.03%	15.04%

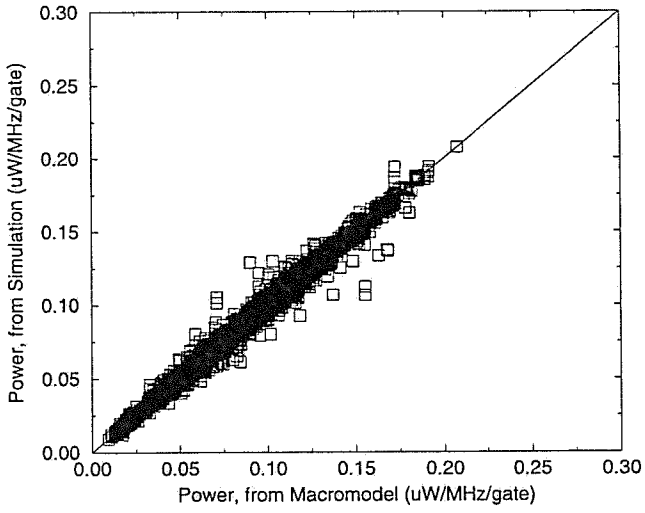

Figure 5.13 Power comparison for ISCAS-85 ckts while estimating zero-delay power.

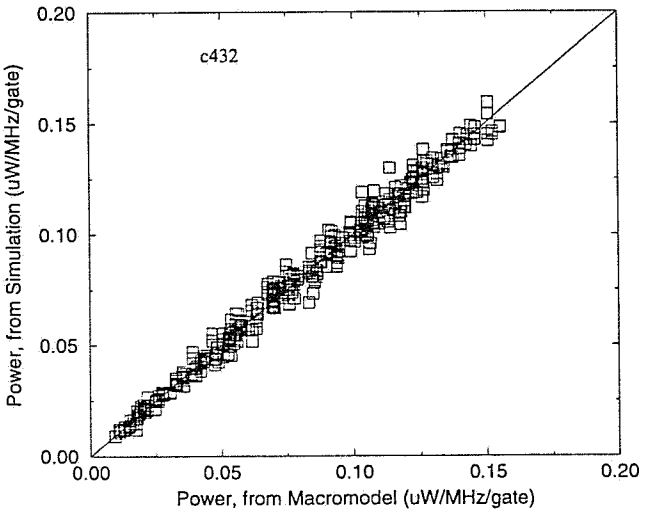

Figure 5.14 Power comparison for c432, while estimating zero-delay power.

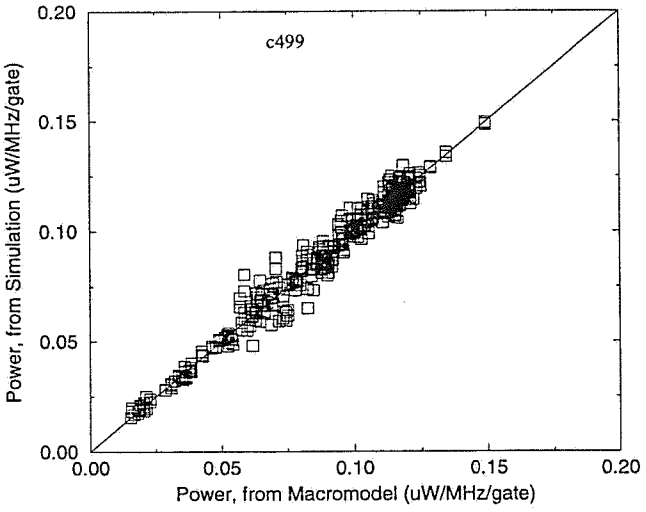

Figure 5.15 Power comparison for c499, while estimating zero-delay power.

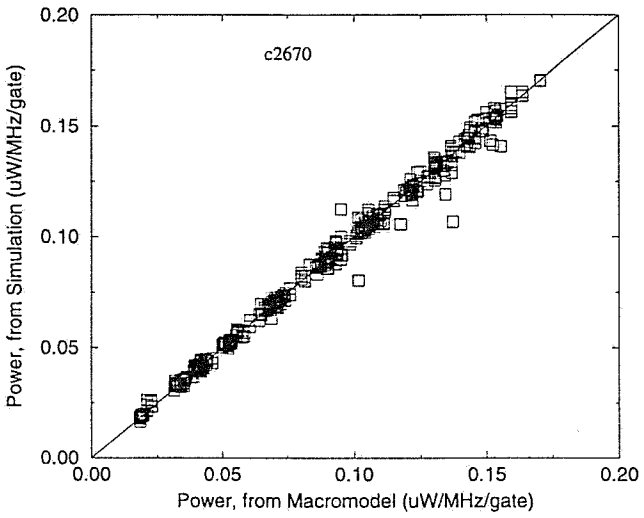

Figure 5.16 Power comparison for c2670, while estimating zero-delay power.

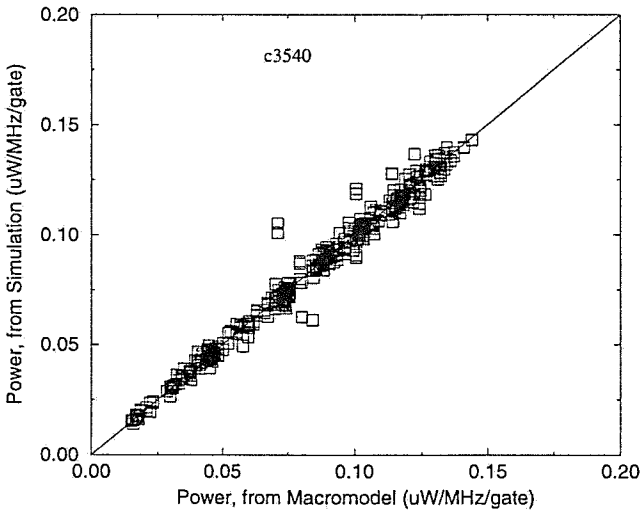

Figure 5.17 Power comparison for c3540, while estimating zero-delay power.

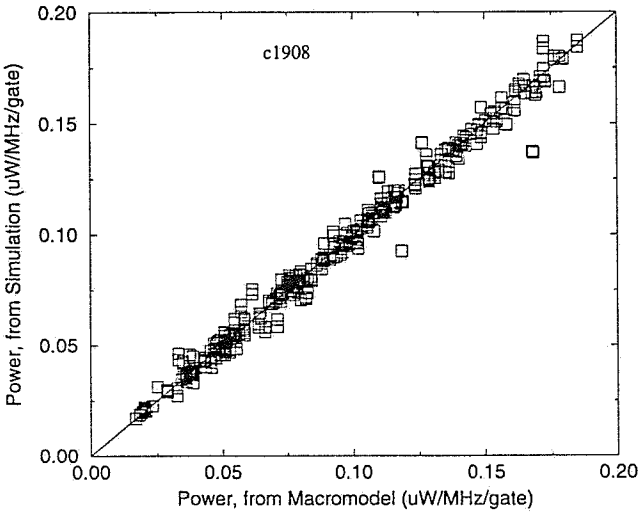

Figure 5.18 Power comparison for c1908, while estimating zero-delay power.

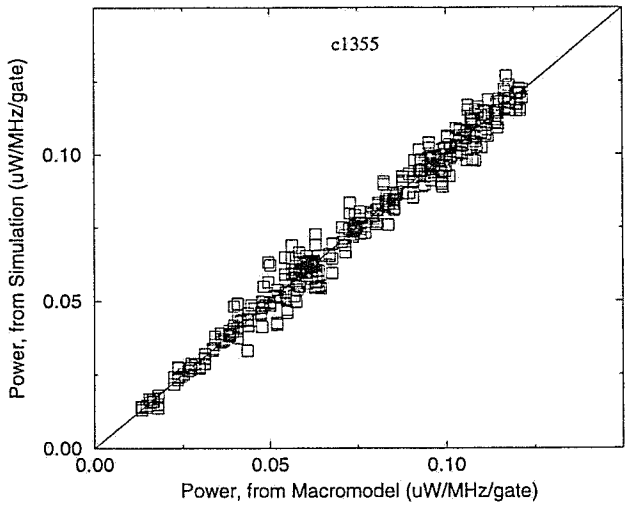

Figure 5.19 Power comparison for c1355, while estimating zero-delay power.

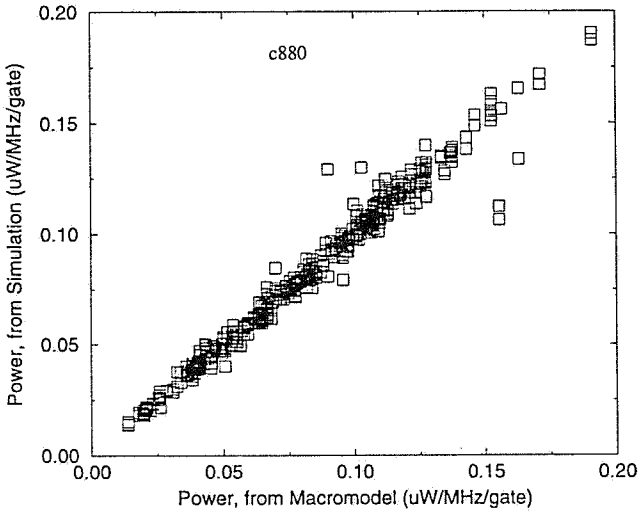

Figure 5.20 Power comparison for c880, while estimating zero-delay power.

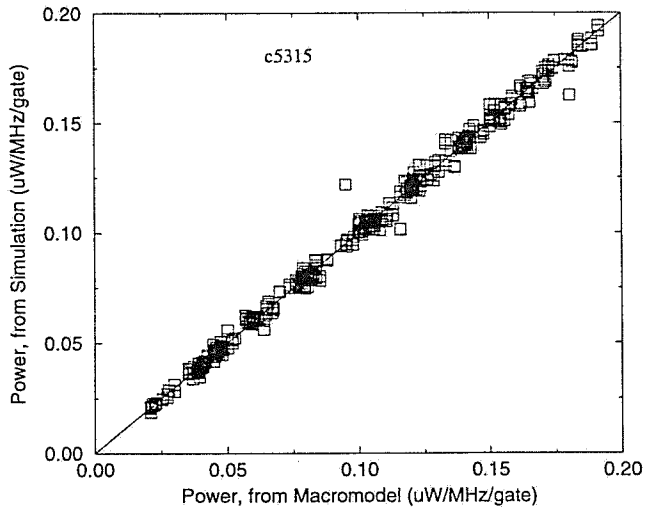

Figure 5.21 Power comparison for c5315, while estimating zero-delay power.

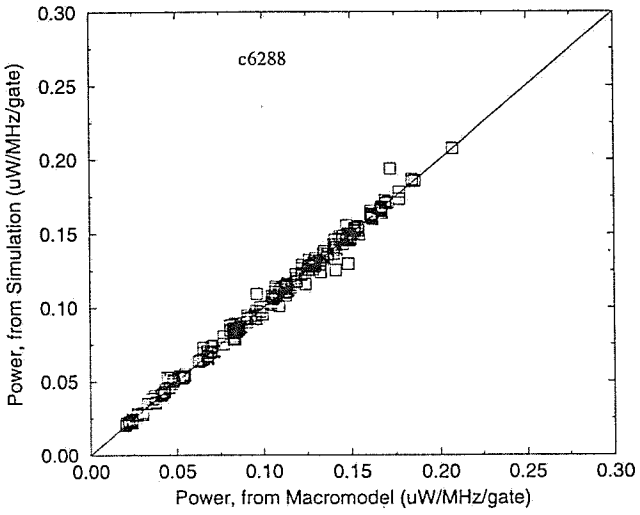

Figure 5.22 Power comparison for c6288, while estimating zero-delay power.

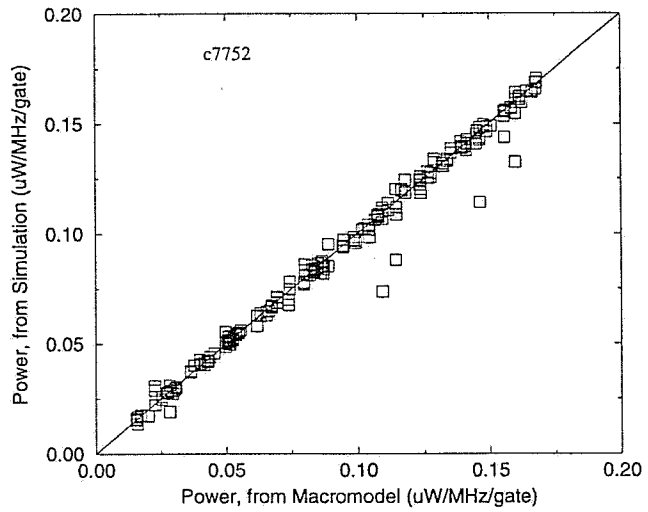

Figure 5.23 Power comparison for c7552, while estimating zero-delay power.

CHAPTER 6

CONCLUSION

Since gate-level power estimation can be time-consuming and because power estimation from a high level of abstraction is desirable so as to reduce design time and cost, we have proposed a power macromodeling approach for combinational circuits with synchronous inputs. Our macromodel consists of a 4-dimensional look-up table with axes for average input signal probability, average input transition density, average input spatial correlation coefficient and average output (zero-delay) transition density. A novel and significant aspect of this approach is that we use the same model template for all types of combinational circuits, and no specialized analytical expressions are required. Another important fact is that this model works for all possible signal switching statistics.

We have shown why it is advantageous to use a 4-d rather than 3-d table, and described an automatic procedure for building the 4-d macromodel, without the need for user intervention. Once the model for a combinational block has been built, it can be used to estimate power during high-level power estimation, based on signal statistics that are computed from a high-level functional simulation. Over a wide range of input/output signal statistics, we have shown that this model gives very good accuracy, with an RMS error of about 4%. Except for one out of about 10,000 cases, the largest error observed was under 20%. The average error was under 6%. If one ignores the glitching activity, then the RMS error becomes under 0.60%, the average error under 5% and the largest maximum error under 18%.

APPENDIX A

MINIMUM VALUE FOR SC_{in}

In this appendix we will derive the value of $n_1(k)$, for which SC_{in}^N and hence SC_{in} takes the minimum value.

Eq. (4.14) can be rewritten as:

$$\sum_{k=1}^{N} n_1^2(k) - \sum_{k=1}^{N} n_1(k) = n(n-1)NSC_{in} \tag{A.1}$$

From Eq. (4.17) we have:

$$\sum_{k=1}^{N} n_1(k) = nNP_{in} \tag{A.2}$$

Therefore, the minimization problem becomes:

$$\begin{aligned} &minimize && \sum_{k=1}^{N} n_1^2(k) \\ &s.t. && \sum_{k=1}^{N} n_1(k) = nNP_{in} \end{aligned} \tag{A.3}$$

Proposition 1. *Minimum value of* $\sum_{k=1}^{N} n_1^2(k)$ *occurs when for all* k*:*

$$n_1(k) = nP_{in} \tag{A.4}$$

PROOF. The problem given by Eq. (A.3) is a constrained minimization problem. Because, it is a convex problem it can be solved by converting (by introducing a Lagrangian) into an unconstrained problem [30]. Therefore, the unconstraint problem becomes:

$$minimize \quad \sum_{k=1}^{N} n_1^2(k) - \lambda\left(nNP_{in} - \sum_{k=1}^{N} n_1(k)\right) \tag{A.5}$$

where λ is a constant. Let us denote $\sum_{k=1}^{N} n_1^2(k) - \lambda\left(nNP_{in} - \sum_{k=1}^{N} n_1(k)\right)$ by J. Differentiating J with respect to $n_1(k)$ and putting it equal to 0 we get

$$2n_1(k) + \lambda = 0 \tag{A.6}$$

$$\Rightarrow n_1(k) = -\frac{\lambda}{2} \tag{A.7}$$

Putting this value of $n_1(k)$ in Eq. (A.2), we get

$$\lambda = -\frac{2nNP_{in}}{N} \tag{A.8}$$

$$\Rightarrow n_1(k) = nP_{in} \tag{A.9}$$

Hence proved. □

REFERENCES

[1] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on VLSI Systems*, pp. 446-455, Dec. 1994.

[2] P. Landman, "High-level power estimation," in *Proc. International Symposium on Low Power Electronics and Design*, 1996, pp. 29-35.

[3] K. Muller-Glaser, K. Kirsch, and K. Neusinger, "Estimating essential design characteristics to suuport project planning for asic design management," in *Proc. IEEE International Conference on Computer-Aided Design*, 1991, pp. 148-151.

[4] D. Liu and C. Svennson, "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid State Circuits*, pp. 663-670, 1994.

[5] D. Marculescu, R. Marculescu, and M. Pedram, "Information theoretic measures of energy consumption at register transfer level," in *Proc. International Symposium on Low Power Design*, 1995, pp. 81-86.

[6] M. Nemani and F. Najm, "Towards a high level power estimation capability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.15, pp. 588-598, 1996.

[7] M. Nemani and F. Najm, "High-level power estimation and area complexity of boolean functions," in *Proc. International Symposium on Low Power Electronics and Design*, 1996, pp. 329-334.

[8] M. Nemani and F. Najm, "High-level area prediction for power estimation," in *Proc. IEEE Customs Integrated Circuits Conference*, 1997, pp. 483-486.

[9] M. Nemani and F. Najm, "High-level area and power estimation for VLSI circuits," in *Proc. IEEE International Conference on Computer-Aided Design*, 1997.

[10] S. Powell and P. Chau, "Estimating power dissipation of VLSI signal processing chips: The PFA technique," in *Proc. VLSI Signal Processing IV*, 1990, pp. 250-259.

[11] S. Powell and P. Chau, "A model for estimating power dissipation in a class of DSP VLSI chips," *IEEE Transactions on Circuits and Systems*, pp. 646-650, 1991.

[12] P. Landman and J. Rabaey, "Architectural power analysis: The Dual Bit model," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, no. 2, pp. 646-650, 1995.

[13] P. Landman and J. Rabaey, "Activity-sensitive architectural power analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 6, pp. 571-587, 1996.

[14] H. Mehta, R. M. Owens, and M. J. Irwin, "Energy Characterization based on Clustering," in *Proc. 33rd ACM/IEEE Design Automation Conference*, June 1996, pp. 702-707.

[15] A. Raghunathan, S. Dey, and N. K. Jha, "Register-Transfer Level Estimation Techniques for Switching Activity and Power Consumption," in *Proc. IEEE International Conference on Computer-Aided Design*, Nov. 1996, pp. 158-165.

[16] S. Gupta and F. Najm, "Power Macromodeling for High Level Power Estimation," in *Proc. 34th ACM/IEEE Design Automation Conference*, June 1997, pp. 365-370.

[17] Q. Qiu, Q. Wu, Chih-S. Ding, and M. Pedram, "Cycle-accurate macro-models for RT-level power analysis," in *Proc. International Symposium on Low Power Electronics and Design*, 1997, pp. 125-130.

[18] R. Mehra and J. Rabaey, "High-level power estimation and exploration," in *Proc. International Workshop on Low Power Design*, 1994, pp. 197-202.

[19] A. Chandrakasan, M. Potkonajk, R. Mehra, and J. Rabaey, "Optimizing power using transformations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 1, pp. 12-31, 1995.

[20] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-driven behavioral synthesis for low-power VLSI systems," *IEEE Design and Test of Computers*, pp. 70-84, 1995.

[21] N. Shanbhag, "Lower bounds on power dissipation for DSP algorithms," in *Proc. International Symposium on Low Power Electronics and Design*, 1996, pp. 43-48.

[22] V. Tiwari, S. Malik, and A. Wolfe, "Power-analysis of embedded software," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 437-445, 1994.

[23] D. Lidsky and J. Rabaey, "Early power exploration: A world wide web applications," in *Proc. ACM/IEEE Design Automation Conference*, 1996, pp. 27-32.

[24] F. Najm, "Low-pass filter for computing the transition density in digital circuits," *IEEE Transactions on Computer-Aided Design*, vol. 13, no. 9, pp. 1123-1131, Sep. 1994.

[25] F. Najm, "Transition Density: A new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 446-445, Feb. 1993.

[26] R. Marculescu, D. Marculescu, and M. Pedram, "Logic level power estimation considering spatial-temporal correlation," in *Proc. IEEE International Conference on Computer-Aided Design*, 1994, pp. 294-299.

[27] M. Xakellis and F. Najm, "Statistical Estimation of the Switching Activity in Digital Circuits," in *Proc. 31st ACM/IEEE Design Automation Conference*, June 1994, pp. 728-733.

[28] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a traget translator in Fortran," in *Proc. IEEE International Symposium on Circuits and Systems*, June 1985, pp. 695-698.

[29] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd edition. New York: McGraw-Hill, 1991.

[30] D. Luneberger, *Linear and nonlinear programming*, 2nd edition. New York: McGraw-Hill, 1988.