A FRACTAL COMPACTION ALGORITHM FOR EFFICIENT POWER ESTIMATION OF CMOS DESIGNS

by

Radjakichenin Radjassamy

A Dissertation Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
DOCTOR OF PHILOSOPHY
In the Graduate College
THE UNIVERSITY OF ARIZONA

1998
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700  800/521-0600
A Fractal Compaction Algorithm for Efficient Power Estimation of CMOS Designs

by

Radjakichenin Radjassamy

A Dissertation Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
DOCTOR OF PHILOSOPHY
In the Graduate College
THE UNIVERSITY OF ARIZONA

1998
As members of the Final Examination Committee, we certify that we have read the dissertation prepared by RADJAKICHENIN RADJASSAMY entitled A FRACTAL COMPACTION ALGORITHM FOR EFFICIENT POWER ESTIMATION OF CMOS DESIGNS and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of DOCTOR OF PHILOSOPHY.

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copy of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Dissertation Director  DR. JO DALE CAROTHERS
STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under the rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: [Signature]

[Signature]
ACKNOWLEDGMENTS

I would like to thank my supervisor, Prof. Jo Dale Carothers for the valuable guidance and unflinching support that she offered me since my first day in her Computer System Design Laboratory. I would also like to express my sincere gratitude to my parents, wife, brothers and sisters for the affection and belief they have in me. My sincere appreciation goes to my fellow lab mates: Craig Beebe and Niraj Shah for their suggestions. Kusnadi for his assistance in setting up PowerMill, and Tingyang Liu for putting up with my Indian music. Special mention also goes to Prof. Jeffrey Rodriguez for his assistance in generating the correlated signals. I would like to acknowledge the cheerful assistance that Miss Tami Whelan, the Graduate Academic Advisor of the ECE Department, offered me in academic matters.

I would like to thank the NSF Center for Low Power Electronics at The University of Arizona for funding this research work in part. I would also like to thank EPIC/Synopsis for providing the evaluation copy of PowerMill.
To my parents Mr. B. Radjassamy and Mrs. R. Jayalakshmi.

whose love and encouragement

made this possible
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>9</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>10</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>12</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>13</td>
</tr>
<tr>
<td>1.1 Estimation of Power</td>
<td>13</td>
</tr>
<tr>
<td>2 BACKGROUND</td>
<td>18</td>
</tr>
<tr>
<td>2.1 CMOS Power Components</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Power Estimation</td>
<td>21</td>
</tr>
<tr>
<td>2.2.1 Dynamic Estimation</td>
<td>21</td>
</tr>
<tr>
<td>2.2.2 Static Estimation</td>
<td>22</td>
</tr>
<tr>
<td>2.2.3 Statistical or Monte Carlo Estimation</td>
<td>22</td>
</tr>
<tr>
<td>2.3 Vector Compaction for Dynamic Power Estimation</td>
<td>23</td>
</tr>
<tr>
<td>2.3.1 Compaction Techniques</td>
<td>24</td>
</tr>
<tr>
<td>3 PROBLEM DEFINITION AND MOTIVATION</td>
<td>27</td>
</tr>
<tr>
<td>3.1 Problem Definition</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Factors Influencing Switching Activity</td>
<td>28</td>
</tr>
<tr>
<td>3.2.1 Logic Function</td>
<td>28</td>
</tr>
<tr>
<td>3.2.2 Logic Style</td>
<td>29</td>
</tr>
<tr>
<td>3.2.3 Signal Statistics</td>
<td>30</td>
</tr>
<tr>
<td>3.2.4 Dynamic Activity</td>
<td>30</td>
</tr>
<tr>
<td>3.3 Motivation</td>
<td>31</td>
</tr>
<tr>
<td>3.3.1 Spatial and Temporal Correlation in Input</td>
<td>31</td>
</tr>
<tr>
<td>3.3.2 Fractal Behavior of Internal Node Toggle Distribution</td>
<td>32</td>
</tr>
<tr>
<td>3.3.3 Estimation of Correlation</td>
<td>32</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1.1 Dynamic Power Estimation with Vector Compaction .......................... 14
1.2 Fractal Compaction Technique .................................................. 15

2.1 Power Consuming Components in a CMOS Inverter .......................... 19

4.1 Fractal Koch Curve .............................................................. 35
4.2 R/S Plot for C880 .................................................................. 40
4.3 ACF Plot for C880 .................................................................. 41
4.4 R/S Plot for C432 .................................................................. 42
4.5 ACF Plot for C432 .................................................................. 43

5.1 Fractal Compaction Algorithm ...................................................... 47

6.1 Comparison of Compaction Ratio ................................................ 59
6.2 Comparison of Estimation Error ................................................ 60
6.3 Comparison of Compaction Ratio ................................................ 62
6.4 Comparison of Estimation Error ................................................ 63
6.5 Compaction Ratio for Various Correlation (Before Modification) ... 66
6.6 Gate Toggle Frequency for Various Correlation Values .................... 68
6.7 ACF Plot for C432 with Correlated Input ...................................... 69
6.8 ACF Plot for C880 with Correlated Input ...................................... 70
6.9 Range Parameter for Various Correlation Values (Before Modification) 71
6.10 Comparison of Range with Modified Range .................................. 73
6.11 Comparison of Compaction Ratio ................................................ 74
6.12 Comparison of Simulation Time Before Compaction ...................... 76
6.13 Comparison of Simulation Time After Compaction ....................... 76
LIST OF TABLES

4.1 Hurst Values for Combinational Circuits ........................................ 44
4.2 Hurst values for Sequential Circuits ............................................. 45

5.1 Compaction Time for Combinational Circuits .................................. 51
5.2 Compaction Time for Sequential Circuits ....................................... 51
5.3 ISCAS85 Combinational Circuits Tested ......................................... 52
5.4 ISCAS89 Sequential Circuits Tested ............................................. 53

6.1 Compaction Results for Combinational Circuits ............................... 57
6.2 Simulation Time Before and After Compaction ................................ 58
6.3 Comparison of Fractal Algorithm with Symbolic Vector Algorithm ....... 59
6.4 Compaction Ratio for Different Vector Set Sizes .............................. 60
6.5 Compaction Results of C432 for Different Vector Set Sizes ............... 61
6.6 Compaction Results for Sequential Circuits ................................... 62
6.7 Comparison of Fractal Algorithm with Back Propagation Algorithm ...... 63
6.8 Compaction Ratio for Correlated Input Vectors (Before Modification) .. 65
6.9 Compaction Ratio for Various Correlation (Before Modification) ....... 65
6.10 Compaction Ratio for C432 with Temporal, Spatial and Spatio-

Temporal Correlation (Before Modification) ........................................ 66
6.11 Toggle Counts and Gate Toggle Frequency of C17 for Various Cor-

relation ................................................................. 67
6.12 Toggle Counts and Gate Toggle Frequency of C432 for Various Cor-

relation ................................................................. 68
6.13 Range Parameter of C432 and C880 for Various Correlation (Before

Modification) ........................................................................ 70
6.14 Comparison of Compaction Ratio Achieved with R and R_{mod} ........... 72
6.15 Range Parameter of C432 and C880 for Various Correlation Values

(After Modification) .................................................................... 73
6.16 Compaction Results with R_{mod} for Correlated Input ...................... 74
6.17 Simulation Time Before and After Compaction ............................... 75
6.18 Compaction Ratio with $R$ and $R_{mod}$ for S298

6.19 Compaction Results of Sequential Circuits for Correlated Input (After Modification)

6.20 Simulation Time of Sequential Circuits for Correlated Input
ABSTRACT

In this dissertation, a vector compaction technique called the Fractal Compaction Algorithm is presented. The fractal compaction algorithm significantly reduces the time needed for estimating the power consumed in CMOS circuits. CMOS IC design requires accurate power estimation at every level in the design hierarchy. Power estimation methods that are currently available are either dynamic or static. Dynamic methods simulate the design using specific input vector sets and estimate the power consumed. Though accurate, dynamic methods require prohibitively long simulation time for large designs and large vector sets. Static power estimation methods, on the other hand, use analytical tools such as statistics and probability to estimate power. The static methods are usually fast but less accurate. To achieve both speed and accuracy, one approach would be to simulate with a compact vector set that has similar switching behavior as the original vector set. The algorithm presented in this work generates such a compact set using fractal concepts. It exploits the correlation present in the toggle distribution of a circuit’s internal nodes for compacting the vector set. Instead of using correlation co-efficients, the fractal algorithm uses a simpler parameter, called the Hurst parameter to quantify correlation. The performance of fractal algorithm with combinational and sequential circuits showed very high compaction that can lead to a shorter design phase and quicker tape out. When compared to previously reported results, the fractal algorithm compacts much higher while keeping the estimation error low.
CHAPTER 1

INTRODUCTION

This dissertation presents a vector compaction technique called the Fractal Compaction Algorithm. The algorithm is used to significantly reduce the time needed to estimate the power consumed in CMOS circuits. It uses fractal concepts to compact a large vector set so that power can be estimated fast while keeping the error low. When compared to previously published results [40, 21], the fractal compaction algorithm performs much better for both combinational and sequential circuits.

1.1 Estimation of Power

The ever increasing need for smaller device size and higher clock speed has resulted in problems with heat dissipation. As a preventive measure, the current trend is to introduce designs that consume less power so that heat can be dissipated with smaller packaging overhead. A good low power design requires an accurate power estimation tool to ensure that the power constraints are satisfied at every design stage. However, with the present day complexity in VLSI designs, power estimation takes a significant amount of time. For example, it may take up to 3 months to estimate the power consumed by a large design with 50K gates and an input vector size of approximately 1 million [15].

Currently available power estimation methods are either dynamic or static. Dynamic power estimation methods simulate the design with a set of input vectors and determine the power consumed. Since the number of switching transitions at each node is found deterministically, dynamic power estimation is accurate. However, it requires prohibitively long simulation time for larger designs. Static power estimation methods, on the other hand, use statistical and probabilistic information
about node switching activity for estimating the power consumed. These methods are usually faster because they do not simulate the design but instead use only an analytical estimate of the node switching activity. The important limitation is that they are less accurate when compared to dynamic methods.

In order to estimate power quickly and accurately, one obvious solution would be to simulate the design with a smaller vector set. This requires compacting a large vector set into a smaller one so that the simulation time will be shorter. An obvious and simple method for compaction is to randomly select subsets of vectors from the original vector set and group them together [40]. However, such randomly chosen vectors do not retain the average switching behavior of the original vector set and could lead to erroneous and biased estimation. Thus the main objective of a vector compaction technique should be to generate a compact vector set which is representative of the original vector set such that it mimics the power-determining behavior of the latter. The compact vector set can then be simulated in a logic simulator to estimate the power consumed, as shown in Fig. 1.1.

In this dissertation, a vector compaction algorithm which makes power estimation fast and accurate is proposed. The algorithm uses fractal concepts to compact the input vector set. Fractals are defined by Mandelbrot [25] as shapes made of parts similar to the whole. Such shapes exhibit very high correlation in their data structures. In order to quantify the correlation inherent in these structures, the fractal parameter, called the Hurst parameter has been used. The Hurst parameter can be estimated using graphical methods available in both the time and frequency domains.

The fractal compaction algorithm is illustrated in Fig. 1.2. The algorithm first
divides the larger input vector set into smaller vector subsets. The size of the vector subsets is determined by the statistical Range parameter of the internal node toggle distribution. The Hurst value and the toggle counts for each of the vector subsets is then estimated. Subsequently, each fractal subset is tested for similarity with other subsets by comparing the Hurst value and toggle counts of each subset. The fractal subsets that are similar to a particular subset are grouped together and a representative subset from each group is chosen. The representative vector subsets are subsequently combined together to form the compact vector set. The compact vector set can then be used for simulation so that power estimation is faster and accurate.

As the fractal compaction algorithm evolved, certain new concepts were iden-
tified. The prominent one is the identification of fractal behavior in internal node toggle distribution. Fractals have so far been used in computer engineering for modeling computer network traffic [19, 17, 23, 24]. However, this is the first time that they have been used in VLSI design for generating compact vector sets to estimate power. In addition, correlation in the toggle distribution was quantified using the Hurst parameter instead of the correlation co-efficient which is usually computationally intensive. A simpler and faster analytical tool to estimate the Hurst parameter was devised and the estimation of correlation co-efficients has been obviated. Another interesting concept that has been identified relates to the effect of input correlation on the internal node toggle distribution. It is well known that as the input vector correlation increases, the toggle frequency of the internal nodes will decrease [10, 26]. However, it was also found that as correlation in the primary input vectors increases, the correlation in internal node toggle distribution also decreases due to the reduced toggling frequency.

The salient advantages of the fractal compaction algorithm are:

1. Significant savings in time for power estimation through simulation
2. Fast and accurate power estimation
3. Better compaction results when compared to other reported algorithms
4. Fully exploits the spatial and temporal correlation in input data
5. Uses simpler tool to quantify correlation
6. Smaller overhead time for compaction

The rest of the dissertation is organized as follows. Chapter 2 discusses the background information on power estimation and vector compaction. Two vector compaction techniques that have been published in the literature are also discussed. The vector compaction problem and the motivating factors for the fractal compaction algorithm are explained in Chapter 3. In Chapter 4, the basic concepts
of Fractals are reviewed. Some of the methods for estimating the Hurst parameter are also discussed. The fractal compaction algorithm is explained in detail in Chapter 5 and the results of the simulation before and after using the compaction algorithm is presented in Chapter 6. The final Chapter concludes by highlighting the important features of the fractal compaction algorithm.
CHAPTER 2

BACKGROUND

Low power VLSI design, seemingly a recent topic of research interest, however evolved with the invention of the transistor, MOSFET and IC’s. Of late, it has gained more importance due to the conflicting combination of shrinking device size and higher clock speed that has resulted in higher power consumption and heat dissipation. For example, the DEC Alpha 21064 microprocessor which operates at 200 MHz consumes 30W (peak power) and in its upgraded version, the DEC Alpha 21164 consumes 50W at 300MHz [3]. The major creative challenge in designing a low power ‘Green Computer’ lies in developing techniques to reduce power consumption without compromising the device performance and size.

Low power VLSI design basically includes two concomitant tasks – power estimation and power optimization. An effective low power optimization technique depends largely on an accurate power estimate since the power level considered is a few watts. Power estimation and optimization can be carried out at each of the levels in the design hierarchy namely, the behavioral, architectural, logic/circuit and physical design levels. The choice of the design level is based on the accuracy required and the computational complexity that can be tolerated. In this research, issues in power estimation at the circuit level were considered.

2.1 CMOS Power Components

Ideally, a CMOS gate consumes power when its output node makes a switching transition. However, there are short circuit and leakage currents through the gate that usually result in wasteful power dissipation. Refer Fig. 2.1 for various power consuming currents in a CMOS inverter.
The average power $P_{\text{avg}}$ consumed by a CMOS circuit can be represented mathematically [11] as:

$$P_{\text{avg}} = P_{\text{switch}} + P_{\text{ShortCkt}} + P_{\text{lkg}}$$

$$P_{\text{avg}} = (C_L \times V_{\text{swing}} \times V_{dd} \times f_{\text{clk}} \times E[0 \rightarrow 1]) + (I_{\text{shockt}} \times V_{dd}) + (I_{\text{lkg}} \times V_{dd})$$

![Figure 2.1: Power Consuming Components in a CMOS Inverter](image)

The first term on the right hand side of Eq. 2.2, i.e., switching power $P_{\text{switch}}$, represents the power consumed by the switching capacitance or load capacitance $C_L$ when a $0 \rightarrow 1$ transition is made. It represents approximately 60% - 70% of the total power consumed. In fact, only 50% of the switching power consumed is used for charging the load capacitance $C_L$ [3] while the rest is dissipated as heat in the PMOS device. In addition to $C_L$, the switching power is also a function of the supply voltage $V_{dd}$, the voltage difference between logic 1 and logic 0, $V_{\text{swing}}$, the clock frequency $f_{\text{clk}}$, and $E[0 \rightarrow 1]$, the node transition activity factor. $V_{\text{swing}}$ usually equals the supply voltage $V_{dd}$ but for internal nodes, it could be less than $V_{dd}$. The node transition activity factor $E[0 \rightarrow 1]$, also called the node switching
activity factor is the probability that a power consuming transition \((0 \rightarrow 1)\) will occur at the output node of the gate. It is defined as the expected (average) value of the number of transitions per clock cycle [31]. It can be represented as

\[
E[0 \rightarrow 1] = \lim_{c \to \infty} \frac{t(c)}{c}
\]  

(2.3)

where \(c\) is the number of clock periods and \(t(c)\) is the number of \(0 \rightarrow V_{dd}\) transitions made in \(c\) clock periods. \(E[0 \rightarrow 1]\) is dependent on static and dynamic properties of the circuit such as logic functions (NAND, NOR, XOR, etc.), logic style (Static CMOS, Dynamic CMOS), signal statistics and glitches [10]. For example, for an \(N\) input NAND/NOR static gate with independent and uniformly distributed signals, the node transition activity reduces as \(N\) increases while for a similar XOR logic, it is a constant. Also, the transition activity for dynamic logic design is higher than that for static design due to pre-charging.

The short circuit power, \(P_{\text{short Ckt}}\) refers to the power dissipated due to the direct path short circuit current, \(I_{\text{shckt}}\), through the PMOS and NMOS transistors of static logic circuits, during a switching transition. It accounts for nearly 20% of total power dissipated in static CMOS designs. The short circuit current is a function of the rise and fall time of the input and output signals, amount of capacitive load, size of the MOSFET devices and the gate capacitance, especially the equivalent gate to drain capacitance. The last term in Eq. 2.2 corresponds to power dissipated due to leakage current \(I_{\text{kg}}\). Usually, leakage current accounts for approximately 2% to 3% of the total power [31]. Though ideally no power is consumed when PMOS and NMOS transistors are OFF, power due to \(I_{\text{kg}}\) arises from inherent reverse biased diode currents \(I_{\text{rev kg}}\) and sub-threshold effects \(I_{\text{sub kg}}\) of the transistors. The leakage current is strongly a function of the fabrication technology. When the operating voltage is lowered as the device size is scaled down, the leakage current becomes predominant and can add noise into the device operation.
2.2 Power Estimation

Out of the various parameters influencing power consumption Eq. 2.2, the node transition activity parameter $E[0 \to 1]$ alone is stochastic and hence needs to be estimated. Hence power estimation is fundamentally estimation of $E[0 \to 1]$. Several researchers have reported various techniques for estimating power at the logic/circuit level [7, 8, 12, 41]. They can be broadly classified into Dynamic techniques that are simulation based and Static techniques that are analytical in nature. There is one another technique, called Statistical or Monte Carlo Estimation technique which uses both static and dynamic tools for estimating power. See [9, 31, 38] for a review of dynamic and static power estimation methods.

2.2.1 Dynamic Estimation

Dynamic power estimation is the most straightforward method because the circuit is simulated to monitor its power supply current for various input data vectors. The current values are then averaged to determine the average power consumed. An example of a dynamic power estimation tool is PowerMill [14, 20] which is a commercial transistor-level simulator. It uses event-driven timing simulation algorithms and is faster than SPICE by two or three orders of magnitude. PowerMill provides information about average, rms and instantaneous currents and the power consumed by switching transitions, glitches, short-circuit and leakage currents. The advantages of simulation based power estimation are its accuracy of estimation and applicability irrespective of the technology, design architecture, etc. The power estimated is accurate because the number of output switching transitions, and hence the power consumed is found deterministically and is not estimated. However, an important limitation of the dynamic approach is that, for larger designs the simulation time would be prohibitively long [15].
2.2.2 Static Estimation

Static power estimation, on the other hand, does not explicitly simulate the design. It uses statistical and probabilistic information (such as average switching activity and correlation coefficients) about the input vector and calculates the same for the internal nodes of the circuit in order to estimate the power consumed. Though the static power estimation methods are faster, they are inherently less accurate. Binary Decision Diagram (BDD), Markov chains are some of the analysis tools used in static power estimation [26, 31]. Binary Decision Diagrams (BDDs) attempt to handle both spatial and temporal correlation. A BDD is used to represent successive Boolean functions at every node in terms of the primary inputs. The circuit topology defines the Boolean function corresponding to every node in terms of the primary inputs. The intermediate values that the node takes before reaching steady state are not represented by this function. An important advantage of BDDs is better speed of computation when compared to statistical techniques. BDD was used in [26] to estimate power in combinational modules with highly correlated input streams. The activities at the primary outputs and all internal nodes were estimated using conditional independence and isotropy of signals. In the most general case, the conditional independence problem was shown to be NP-complete and appropriate heuristics was used to estimate switching activity. In [40], a new technique was presented for estimating the transition probabilities of internal signals in combinational circuits. It used Reduced Ordered Binary Decision Diagrams (ROBDDs) in order to estimate power faster when compared to BDDs. It also accounted for the temporal dependence, multiple concurrent transitions, and mutual dependence present in the signals.

2.2.3 Statistical or Monte Carlo Estimation

Another power estimation method which uses both dynamic and static tools is the Statistical power estimation method. It relies on the convergence of the circuit/node power dissipation to the average power through a series of simulations.
Normally, the inputs are randomly generated and statistical mean estimation, especially Monte Carlo methods [7] are used to decide the stopping criteria. Usually the stopping criterion is based on the probability distribution of the power consumed. Two statistical tools reported in the literature are McPower [8] and MED [41]. McPower used Monte Carlo simulation to estimate the total average power of the circuit. It applied randomly generated input patterns at the primary inputs and monitored the energy dissipated per clock cycle using a simulator. In order to stop the simulation, it used stopping criteria which were based on the desired error percentage and confidence level. MED, however, did not provide an estimate of the power consumed by individual gates or by small group of gates. MED [41] is a modification of McPower, in that it provides both total and individual gate power estimates with user specified accuracy. Power estimation in MED is equivalent to estimating transition density at every node. The stopping criteria used in MED is different from the one used in McPower. It is based on the central limit theorem that makes convergence faster. In general, the important advantage of any statistical power estimation tool is that they can be built around existing simulation tools and libraries. Besides, they are accurate but are slower due to the time needed for convergence.

2.3 Vector Compaction for Dynamic Power Estimation

In the previous section, it was mentioned that dynamic techniques are accurate but take too much time for simulating larger designs. On the other hand, static or statistical tools are fast but less accurate. In order to estimate power quickly and accurately, one obvious solution would be to simulate the design with a smaller input vector. A large input vector must be compacted into a smaller one so that the simulation time will be shorter. A simple technique for compaction is to randomly select subsets of vectors from the original vector set and group them together. However, such a compact vector set does not retain the average switching behavior of the original vector and hence could lead to erroneous and biased estimation.
Thus the main objective of a vector compaction algorithm is to generate a compact vector set which is a representative of the original vector set in that it mimics the power-determining behavior of the latter. Two different compaction techniques that have been reported in the literature [21, 40] will be discussed in the next section.

2.3.1 Compaction Techniques

**Backward Weight Propagation**

Huang et al. [21] proposed a compaction algorithm that makes use of a novel Backward Weight Propagation technique to generate a compact vector set. Their compaction algorithm can be summarized as follows. Given a large vector set of original simulation vectors, its transition profile (desired) for a given circuit is derived using a logic simulator and an appropriate delay model. A starting vector is then assigned and the transition profile (actual) due to the vector is derived. The difference between the desired and actual transition profile which is quantified as transition momentum is determined for each of the switching nodes. Another parameter called the signal momentum, which is a measure of the logic signal expected at a node, is also determined. Using these two parameters, the signals at each of the switching nodes are weighted. The signal weights are then propagated to the primary inputs so that the weights can be translated into input vectors based on certain rules. The input vector thus generated forms part of the compact vector set. The algorithm is repeated for another starting vector and is continued until either of the following criteria is satisfied: the model error is less than the user-specified threshold or the limit on the number of generated vectors is reached. The backward weight propagation algorithm was tested on some of the sequential circuits from the ISCAS89 benchmarks. The tests used uncorrelated vector sets of size and resulted in a maximum compaction of 5X with estimation error ranging between 6.38% and 0.53%. Compaction results for correlated vector sets were not mentioned in the literature.
Symbolic Vector Generation

Tsui et al. [40], proposed a compaction technique called Symbolic Vector Generation. The technique casts the problem of observing pairwise transition probabilities to that of observing pairwise signal probabilities and generating a vector set of symbols called symbolic vectors. The symbolic vectors are then converted back into a sequence of binary vectors, which forms the compact vector set. Symbolic vector generation can be either constrained or unconstrained depending on whether the consecutive symbolic vectors generated are temporally compatible or not. The paradigm proposed for power estimation using this compaction technique is envisaged as follows. The chip for which power is to be estimated is divided into several building blocks, each with a detailed structural model at the gate or circuit level. A behavioral model of the chip is then built using the building blocks as components. Simulation is then carried out at the behavioral level and switching statistics for the buses or nets connected to the blocks are determined. The vector compaction algorithm is then used to generate a compact vector set for each building block which can then be fed to a lower level simulator to estimate power for each block. The symbolic vector generation algorithm was tested on some of the combinational circuits from the ISCAS85 benchmark suite. The results showed a compaction of 100X and 20X for input vectors of size 100,000 and 4000 respectively, with power estimation errors below 5% (max) and 2% (avg). However, the paper does not mention the overhead time needed for generating the compact vector.

The vector compaction techniques discussed above suffer from the following limitations. They both estimate switching activity at each node through certain parameters such as transition momentum, transition probability, etc. Inherently, estimation of these statistical parameters are computationally intensive. The backward weight propagation technique is iterative in nature and is dependent on the starting vectors. If the starting vectors used are random, the back propagation technique would have to iterate many times before an acceptable input vector can be found. It is also possible that such an acceptable input vector could not be
found if the stopping criteria are stringent. The backward propagation technique also does not mention what the procedure is for generating the starting vectors.

The symbolic vector generation technique requires a behavioral level simulator to determine the switching activity at each of the buses and nodes interconnecting various building blocks. In addition behavioral level models for various building blocks are also required. Thus, the need for a behavioral level simulator is a serious overhead requirement for the symbolic vector generation technique. Another limitation of the technique arises from the need for generating the signal statistics at the behavioral level. Usually signal statistics at higher levels such as the behavioral level are highly error prone because of the limited design information. As a result, the compact vector set that is generated based on these statistics may have a higher estimation error. The other limitation of the symbolic vector generation technique is that it uses only pair-wise spatial and temporal correlation due to computational limitations. Even so, the determination of pair-wise correlations for any larger design could become an indomitable task.
CHAPTER 3

PROBLEM DEFINITION AND MOTIVATION

In this chapter, the vector compaction problem is discussed quantitatively and is followed by a discussion on the motivating factors that have been key to the vector compaction technique presented in this dissertation.

3.1 Problem Definition

The two important objectives of vector compaction used for power estimation are:

1. Maximize compaction and

Based on these objectives, the vector compaction problem can be quantitatively defined as follows.

Given a circuit design with \( m \) primary inputs and a binary input vector set \( I^{(n\times m)} \) with expected number of switching transitions \( s \) per clock cycle, \( E_i(s) \), generate a compact vector set \( C^{(p\times m)} \) with \( E_c(s) \), such that:

\[
p \ll n
\]

and \( |E_c(s) - E_i(s)| \approx 0. \) \hspace{1cm} (3.2)

The ratio \( n/p \) is the Compaction Ratio \( X \) achieved for the input vector set. Since the average power consumed is proportional to switching activity \( E(s) \), error in \( E(s) \) is used in condition 3.2 instead of the error in power estimated.

Any vector compaction algorithm, irrespective of the technique it uses, should satisfy both the above mentioned conditions. The first condition assures a smaller
compact vector set so that power can be estimated faster. In addition, by maximizing compaction, it also justifies the overhead on power estimation in terms of the time and resources needed for generating the compact set. Since maximizing compaction alone would not suffice, the second condition limits the error in switching activity so that the error in the power estimate is within acceptable limits. It should be noted that both the compaction ratio and estimation error tend to have an inverse relationship. A smaller compact vector set may have a higher error in estimation since there may not be enough representative vectors in the compact set. Thus it is onus of the compaction algorithm to optimize both size and error by generating a compact vector set that has the same average switching activity as the original vector set.

3.2 Factors Influencing Switching Activity

There are several design related factors, such as the choice of logic gates (NAND, NOR), logic style (static, dynamic) etc., that will influence the switching activity of a circuit. Since switching activity, in turn, can affect the results of a compaction algorithm, it is imperative to carefully analyze these factors in designing the compaction algorithm.

3.2.1 Logic Function

Consider a two input static NAND gate, the input signals being independent and uniformly distributed and only one input signal that makes a transition in a clock cycle. The probability that the output makes a transition from logic 0 to 1 [10] is

\[ P_{0 \rightarrow 1} = P_0 \cdot P_1 \]  

(3.3)

where \( P_0 \) is the probability that the output is logic 0 in a particular clock cycle and \( P_1 \) is the probability that the output will be logic 1 in the next clock cycle. The probability for a 0 to 1 transition then works out to be 3/16. For a 2 input
NOR gate and XOR under similar conditions, the probability is $3/16$ and $1/4$, respectively. Extending the analysis to a general case, the transition probability for an $N$ input static gate with independent and uniformly distributed signals is [10]:

$$P_{0\rightarrow 1} = \left(N_0/2^N\right) \times \left(N_1/2^N\right)$$  \hspace{1cm} (3.4)

where $N_0$ and $N_1$ are the logic 0 and logic 1 entries, respectively, in the truth table of the logic function. For NAND (NOR) logic, the output will have exactly one logic 0(1). Eq. 3.4 can be then modified as

$$P_{0\rightarrow 1} = (2^N - 1)/2^{2N}.$$  \hspace{1cm} (3.5)

Thus, as the number of input signals increases, the node transition activity for NAND/NOR logic reduces. However, for XOR logic, the number of times the output is logic 0 is exactly the same as that for logic 1. Hence $P_{0\rightarrow 1} = 1/4$, a constant.

### 3.2.2 Logic Style

CMOS logic design can be either static or dynamic. Static CMOS is simple but requires more components for a particular circuit design. The switching probability $P_{0\rightarrow 1}$ depends on the state i.e., history of the input. For a two input static NAND/NOR gate with independent and uniformly distributed input signals [11],

$$P_{0\rightarrow 1} = P_0 \times P_1 = 3/16$$  \hspace{1cm} (3.6)

Dynamic design, on the other hand, requires fewer components but needs pre-charging circuitry. An important limitation with pre-charging is charge sharing during the evaluation phase of the circuit. Due to charge sharing, the output voltage drops and the circuit consumes additional power proportional to this drop. Besides this limitation, the node switching activity $E(s)$ for a dynamic design is more than that for an equivalent static design [11]. For an $N$-tree structure in which the P network is replaced by a single PMOS pre-charge transistor, the
output will make a $0 \rightarrow 1$ transition during the pre-charge phase, only if the output had been discharged by the N tree during the evaluation phase. Hence, the switching probability $P_{0\rightarrow1}$ depends only on the signal probabilities present currently at the input. The switching probability for dynamic two input NAND logic is then $P_{0\rightarrow1} = P_0 = 1/4$. For NOR and XOR dynamic logic, the switching probability is $3/4$ and $1/2$, respectively. Thus dynamic logic has a higher switching activity factor than the static design.

### 3.2.3 Signal Statistics

In the above analysis, it was assumed that the signals are independent and uniformly distributed. But if the signals have distinct probabilities, then the transition factor will be different. Consider two signals $a$ and $b$ that are uncorrelated and with probabilities for logic 0 and logic 1 being $P_a$ and $P_b$, respectively. The probability that the output node is logic 1 for a two input NOR gate is given by $P_1 = P_a \times P_b$ [10]. The switching probability is then given by

$$P_{0\rightarrow1} = P_0 \times P_1 = (1 - P_1) \times P_1 = (1 - P_a \times P_b) \times (P_a \times P_b)$$

Depending on the values of $P_a$ and $P_b$ values, $P_{0\rightarrow1}$ can be more or less than the value which was calculated assuming random inputs.

### 3.2.4 Dynamic Activity

Dynamic activity refers to glitches which represent spurious switching at a node caused by internal delay within the logic device. It is especially notorious in combinational logic circuits and can cause the switching transition activity $E(s)$ at a node to be greater than one. A node in a circuit can have multiple transitions within a clock cycle before settling to the correct logic value. The number of extra transitions or glitching transitions is a function of the logic depth, the signal skew due to different arrival times of the input and signal patterns. In the worst case,
the number of extra transitions can grow as $O(N^2)$ where $N$ is the logic depth [3]. Power dissipated by the glitches and short circuit current constitutes 20% of the total average power with a maximum of 70% [31].

3.3 Motivation

The main motivating factors for developing the fractal based compaction algorithm are as follows: existence of spatial and temporal correlation in input vectors, the fractal behavior of the internal node toggle distribution, and the need for a simpler method to quantify correlation. Each of the factors will now be explained in detail.

3.3.1 Spatial and Temporal Correlation in Input

In order to speed up power estimation, most power estimation techniques assume that the inputs are uncorrelated. Such an assumption, in reality, is not always true since there are applications such as with microprocessors, where the signal patterns generated from architectural level traces of instruction opcodes and instruction mixes for typical applications are highly correlated, i.e., redundant. In general, input correlation is due to two sources: spatial and temporal correlation. The spatial or structural correlation is induced by the re-convergent fan-out of gate signals while the temporal or stochastic correlation is by virtue of correlation in primary input vectors.

For better compaction, any compaction algorithm should ensure that there is little redundancy in the vectors of the compact set. The fractal compaction algorithm exploits the spatial and temporal correlation in input vectors to identify redundant vectors. In contrast, the previously reported compaction algorithms that were discussed earlier, do not exploit the correlation in the input vectors. The backward weight propagation algorithm [21] does not use any correlation measures at all for generating compact vector sets. The symbolic vector algorithm [40], on
the other hand, uses some measure of correlation not for identifying redundant vectors but for generating new vectors for the compact set.

### 3.3.2 Fractal Behavior of Internal Node Toggle Distribution

One of the important findings of the fractal compaction algorithm is the correlated nature of the internal node toggle distribution. The correlated toggle distribution, in fact, manifested a fractal or self-similar property that lends itself to allow for vector compaction. The self-similar property of the toggle distribution was effectively utilized in designing the subset mapping step of the fractal compaction algorithm. The previously reported algorithms have used conventional techniques such as backward propagation of signal weights or generation of symbolic vectors for generating the compact set. However, the fractal compaction algorithm is special in that it uses an unconventional tool such as Fractals for compaction. It should also be noted that the author believes this is the first time that Fractals have been used in VLSI design.

### 3.3.3 Estimation of Correlation

In order to estimate spatial and temporal correlation, two probabilistic measures of the signals namely, the signal probability $P_n$ at node $n$, and the transition probability $P_{0 \to 1}$, are usually used [12]. The signal probability $P_n$ is defined as the probability that the signal at a switching node is logic 0. The transition probability of a switching node can be estimated from the knowledge of the input signal probabilities. For example, for a 2 input NOR gate with input signal probabilities $P_a$ and $P_b$, the output transition probability [11] is

$$P_{0 \to 1} = (1 - (1 - P_a)(1 - P_b))((1 - P_a)(1 - P_b))$$  \hspace{1cm} (3.7)

Incidentally, accounting for all possible signal correlations is practically impossible even for small circuits, since the estimation of correlation co-efficients is
computationally intensive. For a gate with \( N \) input signals, even if only pair-wise correlation is considered, the number of correlation co-efficients grows exponentially \((2^N)\). That is the reason why the symbolic vector algorithm [40] limits to just pair-wise correlation while generating the compact vector set. An important disadvantage of limiting to pair-wise correlation is the error introduced in the estimate of the correlation measure. Thus an alternative and simpler tool is needed to quantify correlation in the gate signals for \textit{effective} compaction of the input vector so that dynamic power estimation is faster and accurate. In this research work, a fractal parameter called the Hurst parameter, has been used to quantify signal correlations. The estimation of the Hurst parameter is relatively simpler when compared to analytical methods used for determining correlation co-efficients.

To better appreciate the fractal compaction algorithm presented in this dissertation, it is essential to understand the basics of Fractals. The following Chapter discusses some of the important details about Fractals.
In this chapter, a brief discussion on Fractals is presented. It is followed by an in-depth analysis of the methods available for estimating the Hurst parameter, which is used by the vector compaction algorithm. Finally, a new analytical method for estimating the Hurst parameter is explained.

4.1 Fractal Geometry

Fractal geometry was propounded by B. B. Mandelbrot at a time when it was felt that Euclidean geometry over simplified the representation of patterns and shapes in nature which are, in reality, very much irregular and fragmented. In reality, mountains are not cones and coast lines are not circles as are usually represented in Euclidean space. Mandelbrot [25] coined the term 'Fractals' to identify such fragmented and irregular shapes. He defined Fractals as shapes made of parts similar to the whole [See Fig. 4.1]. The ‘generator’ in Step 1 of Fig. 4.1 is replicated recursively on each of its vertical and horizontal segments as shown in Step 2 and Step 3 so as to result in a fractal Koch curve.

4.1.1 Fractal Types

Fractals exhibit the property of scaling which means the degree of irregularity in them tends to be identical at all scales. They can be classified based on the properties of the scaling parameter as follows: time or space fractals (domain of representation), self-similar or self-affine fractals (symmetry) and deterministic or stochastic fractals (statistics).

Data structures exhibiting the fractal property in the space domain are called
space fractals while those exhibiting the fractal property in time domain are time fractals. Some of the salient space fractals are galactic clusters, the earth’s surface, colloidal aggregates, ceramics, etc. Time fractals evolve due to the inherent random nature of a process or phenomenon such as stock market variations, fluctuating signals, toggles at a CMOS switching node, or for that matter any time series. In this work, the relevant fractal property is exhibited in time domain by the toggle distribution of internal nodes in CMOS circuits.

Self-similar fractals have symmetry across scale, implying recursion, i.e. a pattern inside another pattern. They not only have details at finer and finer scales, but also produce details with certain constant measurements. Fractal shapes like the Koch curve [Fig. 4.1] display self-similarity in space domain because they look exactly the same even under high magnification. In contrast to self-similar fractals, self-affinity fractals do not have symmetry in scaling and are anisotropic. The self-similar shapes repeat statistically or exactly under a magnification, the self-affine traces repeat statistically only when the co-ordinates are magnified by different degrees. For example, the self-affine trace of fractional Brownian motion (fBm) has a different scaling factor for time $t$ and for trace increments. Irrespective of the type, fractals can in general be classified as deterministic or stochastic depending on the number of recursions of the fractal pattern needed to arrive at a full-fledged
fractal structure.

4.1.2 Fractal Parameters

Fractals are usually quantified using two parameters: Fractal Dimension $D$ and Hurst parameter $H$. In general, the Fractal Dimension is used to quantify the dimension of a space fractal while the Hurst parameter is used to measure the correlation in a time fractal [25]. Mandelbrot discussed the Fractal Dimension by contrasting it with the topological and Euclidean dimension. He offered the mathematical definition of Fractals as: “A fractal is by definition a set for which the Hausdorff-Besicovitch dimension strictly exceeds the topological dimension”. The Hausdorff-Besicovitch dimension, also called the Fractal Dimension, is unitless and is represented as

$$N(\lambda) = \frac{1}{\lambda^D}$$

(4.1)

where $\lambda$ represents the smallest segment used in the fractal structure and $N(\lambda)$ is the number of such $\lambda$ segments in the structure. For example, the Fractal Dimension for the Koch curve shown in Fig. 4.1 is 1.2628 as $\lambda = 1/3$ and $N = 4$.

In the time domain, Fractals can be characterized by the Hurst parameter and the Auto-Correlation Function. A stationary stochastic process is long-range dependent (LRD) if its Auto-Correlation Function (ACF) for lag $k$ is non-summable [4] i.e.,

$$\sum_k ACF(k) = \infty.$$  

(4.2)

The lag $k$ refers to the number of observations of the stochastic process used for computing $ACF(k)$. A simple model with LRD are the self-similar processes, with a hyperbolically decaying auto-correlation function. Self-similar and asymptotic self-similar processes are particularly important because the LRD in them can be quantified by a single parameter called the Hurst parameter. A stochastic time series process $X = (X(t), t = 0, 1, 2..)$, is said to be self-similar with Hurst parameter $H$, if the process is covariance stationary and the corresponding aggregated process
has the same correlation structure as the original process (exactly self-similar) or agrees asymptotically with the correlation structure of the original process over long intervals (asymptotically self-similar) \[2, 16\]. More specifically, the stochastic process is asymptotically self-similar if \(ACF(k)\) is defined as
\[ACF(k) \propto k^{2H_0}. \tag{4.3}\]
\(k \to \infty\). The Hurst parameter \(0 < H < 1\) is a measure of the correlation or long range dependence in the data which is responsible for the fractal behavior of the data set. In this dissertation, the correlation in the internal node toggle distribution is quantified using the Hurst parameter.

4.1.3 Conditions for Existence of Fractals

In general, a stochastic process can be said to exhibit fractal behavior if it satisfies the following conditions - its Hurst value is \(0.5 < H < 1.0\) and the its ACF rolls off with a square law dependence on lag \(k\) as in Eq. 4.3. The first condition ensures the existence of correlation in the process while the second implies that the correlation extends over larger scales.

4.2 Estimation of Hurst Parameter

The Hurst parameter can be estimated in the time and frequency domain. Some of the salient time domain estimation methods are R/S plot, Variance plot, Correlogram plot, etc., while in the frequency domain, Whittle Estimator is a very popular method \[4\]. In this section, the R/S plot and the Variance plot methods for estimating the Hurst parameter are discussed.

4.2.1 R/S plot method

The R/S plot method is a graphical method that has been widely used to model a variety of geological phenomena \[17\]. It is based on the rescaled adjusted range
statistics $R/S$. Originally introduced by H. E. Hurst [22] to capture the fluctuations in a given time series in order to size, for example a dam so that it neither underflows nor overflows for a given set of data. For a set $G$ containing $n$ observations of a stochastic process $X_y, y = 1, 2, \ldots, n$, a sequence of adjusted partial sums, $P_k$, can be defined such that

$$P_k = (X_1 + X_2 + \ldots + X_k) - k\overline{X(n)}.$$  \hspace{1cm} (4.4)

where lag $k = 1, 2, 3, \ldots, n$ and $\overline{X(n)}$ is the statistical mean of first $n$ observations. The adjusted range $R(n)$ of the fluctuations can be determined from

$$R(n) = [\max(0, P_1, P_2, \ldots, P_n) - \min(0, P_1, P_2, \ldots, P_n)].$$  \hspace{1cm} (4.5)

The adjusted range is subsequently normalized by the standard deviation $S(n)$ of the observations in set $G$ to obtain $R(n)/S(n)$. Hurst showed that for many naturally occurring time series, the expected value of $R(n)/S(n)$ has a power law dependence on lag $n$ as is

$$E[R(n)/S(n)] \approx n^H.$$  \hspace{1cm} (4.6)

as $n \to \infty$ with $H$ values typically around 0.7. This analysis is repeated for different lag values in observing $X_y$ and $\log_{10}(R/S)$ versus $\log_{10}(k)$ is plotted. Such a plot is usually called the Rescaled Adjusted Range plot or the POX diagram. The R/S plot is fitted with simple least squares fit and the slope of the fitted line is used as an estimate of Hurst parameter.

### 4.2.2 Variance Plot Method

One of the important results in statistics is that the variance of the sample mean is equal to the variance of one observation divided by the sample mean. In other words, for a stochastic process $X_y, y = 1, 2, \ldots, n$, with common mean $\mu$ and variance $\sigma^2$, the variance of

$$\overline{X} = n^{-1} \sum_{y=1}^{n} X_y$$  \hspace{1cm} (4.7)
is equal to

\[ \text{var}(\overline{X}) = n^{-1} \sigma^2. \] (4.8)

This result is valid only if the stochastic process \( X_p \) is an independent and identical process (i.i.d). However, for processes that are dependent (i.e. correlated), the variance relation [4. 17] is

\[ \text{var}(\overline{x}) = n^{-\beta} \sigma^2. \] (4.9)

where \( \beta = (2 - 2H) \). In order to estimate the Hurst parameter, the normalized variance of the aggregated series, i.e., \( \text{var}[\overline{X(k)}]/\sigma^2 \), is determined for different lag \( k \) values and the values are plotted on a log-log plot. The plot is fitted using simple least square fit and the slope of the fitted line, \( \beta \), is found. The estimate of the Hurst parameter can then be determined from \( \beta \) using the relation \( \beta = (2 - 2H) \).

Since in this dissertation, an analytical technique based on the R/S plot is used for estimating the Hurst parameter, further discussions will be based only on R/S plots.

### 4.3 Fractals in Internal Node Toggle Distribution

The vector compaction algorithm that is presented in this dissertation is based on the claim that the internal node toggle distribution exhibits fractal behavior by virtue of the correlation introduced or modified in the distribution as the input vectors are fed to the circuit under consideration. In order to prove the validity of the claim, it is mandatory to check whether the internal node toggle distribution satisfies both the conditions mentioned in Section 4.1.3. To this end, the combinational circuits from the ISCAS85 benchmark suite were tested with uncorrelated vector sets of size 4000. The results of the tests for the circuits C880 and C432 are presented below. The circuit C880 has 60 primary inputs, 26 primary outputs, 880 nets. The circuit C432, on the other hand, has 36 primary inputs, 7 primary outputs and 432 nets. Note that though the primary input vectors are uncorrelated, the structural dependency due to re-convergent nodes in the circuits introduces or
modifies the correlation in the internal signals.

The R/S plot for the circuit C880 generated using the R/S plot method is shown in Fig. 4.2. The lag \( k \) used in generating the plot refers to the number of internal nodes in the circuit used for computing \( R/S \). The slope of the least square fit for the \( \log_{10}(R/S) \) versus \( \log_{10}(k) \) is the Hurst parameter which in this case is \( H = 0.83 \). Since the Hurst value for the internal node toggle distribution is greater than 0.5, the first condition necessary for fractal behavior of the distribution is satisfied. In order to check the slow roll-off of the auto-correlation function, the ACF-Lag plot for the toggle distribution was generated as shown in Fig. 4.3. The slower roll of ACF with respect to lag is obvious from the plot. Since the H value is more then 0.5 and the ACF roll off is slower, it can be concluded that the internal node toggle distribution for C880 exhibits fractal behavior.
Figure 4.3: ACF Plot for C880
Figure 4.4: R/S Plot for C432
The R/S and ACF plots for the circuit C432 are shown in Fig. 4.4 and Fig. 4.5 respectively. The Hurst value for the circuit C432 from the R/S plot is 0.901, which is again more than the minimum value of 0.5 needed for exhibiting fractal behavior. The ACF plot shown in Fig. 4.5 depicts the slower roll-off of the autocorrelation function of the toggle distribution. Since the Hurst value is more than 0.5 and ACF has a slower roll off, it can be concluded that the toggle distribution of the internal nodes in C432 exhibits fractal behavior.

4.4 Analytical Method for Hurst Parameter Estimation

Since the R/S plot method is a graphical method, an analytical technique that can estimate the Hurst parameter within acceptable error tolerance was needed. To this end, an analytical method based on the $R/S$ values was devised for estimating
the Hurst parameter. Generally, the R/S plot (Fig. 4.2) starts with a transient zone or smaller lags representing the nature of short range dependence in the sample, but eventually settles down and fluctuates around the least square fit for larger lags. In other words, the Hurst parameter which is the slope of the least square fit of the R/S plot would be closer to its asymptotic value for larger lags than for smaller lags. Hence, the ratio of $\log_{10}(R/S)$ and $\log_{10}(k)$, where $k = (n - 1)$ is the longest lag possible for a data set of size $n$, would be a better approximation for the Hurst parameter, i.e.,

$$H = \frac{\log_{10}(R/S)}{\log_{10}k} \quad (4.10)$$

Based on this reasoning, the Hurst parameter for C880 was estimated analytically with the same set of input data that was used for generating the R/S plot. The analytical Hurst value was found to be 0.816, an error of only 2.0% when compared to the graphical value of 0.833. Since the vector compaction problem does not require a more accurate estimate of the Hurst parameter, the simpler analytical technique is quite adequate.

Based on the analytical method, the Hurst parameter for the toggle distribution of the internal nodes in various combinational and sequential benchmark circuits were estimated. The combinational circuits were estimated with uncorrelated input vector sets of size 4000, while for the sequential circuits, vector sets of size 1000 were used. The reason for choosing different vector set sizes will be explained in

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Hurst Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C17</td>
<td>0.681</td>
</tr>
<tr>
<td>C432</td>
<td>0.869</td>
</tr>
<tr>
<td>C499</td>
<td>0.878</td>
</tr>
<tr>
<td>C880</td>
<td>0.816</td>
</tr>
<tr>
<td>C1908</td>
<td>0.689</td>
</tr>
<tr>
<td>C3540</td>
<td>0.729</td>
</tr>
</tbody>
</table>

Table 4.1: Hurst Values for Combinational Circuits
Table 4.2: Hurst values for Sequential Circuits

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Hurst Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>S526</td>
<td>0.627</td>
</tr>
<tr>
<td>S832</td>
<td>0.656</td>
</tr>
<tr>
<td>S1196</td>
<td>0.723</td>
</tr>
<tr>
<td>S1238</td>
<td>0.711</td>
</tr>
<tr>
<td>S1423</td>
<td>0.758</td>
</tr>
<tr>
<td>S5378</td>
<td>0.807</td>
</tr>
</tbody>
</table>

The next chapter. The Hurst parameter estimated using the analytical method for the combinational and sequential circuits is presented in Table 4.4 and Table 4.4. The Hurst value for each of the circuits is more than 0.5, which implies that the circuits satisfy one of the necessary conditions for exhibiting fractal behavior. It can be also observed from the tables that the Hurst parameter for the sequential circuits is smaller when compared to combinational circuits. The reason for this can be attributed to the smaller size of the vector sets used for estimating the Hurst parameter in sequential circuits.
CHAPTER 5

FRACTAL COMPACTION ALGORITHM AND SIMULATION EXPERIMENTS

In this Chapter, the fractal compaction algorithm is explained in detail. It is followed by a discussion of the experiments conducted with various combinational and sequential benchmark circuits. Finally, the commercial power estimation tool, PowerMill, which was used to estimate power is explained briefly.

5.1 Fractal Compaction Algorithm

5.1.1 Definition

Given an input vector $I^{n \times m}$, create non-overlapping smaller vector subsets $I_i^{q \times m}$, where $i = 1, 2, 3, ... r$ and $q < n$, such that every subset $I_i$ is checked for possible mapping onto other available subsets $I_d, (d \neq i)$ based on toggle counts $t_i$ and $H_i$ of the subset. $I_i \equiv I_k$ iff.

\[ |t_i - t_k| < t_{\text{max-error}} \] (5.1)

\[ |H_i - H_k| < H_{\text{max-error}} \] (5.2)

5.1.2 Algorithm

The input data used by the compaction algorithm are the design netlist and the input vector set $I^{n \times m}$ to be compacted. The algorithm has three steps. The first step determines the rescaled range statistic $R/S$, also called in short as the Range parameter $R$. The Range parameter decides the size of vector subsets. The second step generates the vector subsets from the original vector set and also estimates the total toggle count and Hurst parameter for each subset. The final step uses
the vector subsets created in Step 2 to form a compact vector set. The algorithm is presented as a flow chart in Fig. 5.1.

In order to generate fractal vector subsets, the algorithm determines the number of toggles at each of the internal nodes of the design for the given vector set. The algorithm also keeps track of the total number of toggles in the circuit due to every vector in the set. Thus for the input vector set \( I^{(n \times m)} \) and a circuit with \( r \) internal nodes, \( \sum_r \sum_n t_{rn} = \sum_n t_n \), where \( t \) is the toggle count. Using toggle information \( t_{rn} \) and \( t_n \), the Range parameter is determined and the fractal vector subsets are formed. The Range parameter ensures that the fractal subsets \( V \) will, on the average, have an equal number of toggles. In the second part of the algo-

**Figure 5.1: Fractal Compaction Algorithm**
rithm, the Hurst parameter for each of the fractal subsets $V$ is found. Then the subsets are clustered based on the Hurst parameter and on the total number of toggles for each of the subsets. A representative subset from each of the clusters is selected and the compact vector $C^{(p \times m)}$ is formed resulting in the compaction ratio $n/p$.

Step 1:
Begin
$t_{rn} = 0, t_n = 0$
For each input vector $m_i, i = 0$ to $n$
Find $t_i$ and $t_{ri}$
$t_{rn} = t_{ri} + t_{rn}; t_n = t_i + t_n$
End For
Calculate the Range parameter $R$
End

Step 2:
Begin
Using $R$, generate fractal subsets $V$
End

Step 3:
Begin
$H_{max-err} = 0.001; t_{max-err} = 0.001.$
Compact Vector $C = NULL$
For each $V(i), i = 0$ to $q$
Find Hurst parameter $H_{V(i)}$ and
total toggle count $t_V(i)$
mapped($i$) = FALSE
End For

For each \( V(i) \), \( i = 0 \) to \( q \)

If \( (\text{mapped}(i) == \text{FALSE}) \)

For each \( V(r) \), \( r = (i+1) \) to \( q \)

If ((\( \text{mapped}(r) == \text{FALSE} \)) AND \\
\( (| H_v(r) - H_v(i) | \leq H_{\text{max-err}}) \) AND \\
\( (| t_v(r) - t_v(i) | \leq t_{\text{max-err}}) \))

\( \text{mapped}(i) = \text{TRUE} \)

End If

End For

Append \( V(i) \) to Compact Vector \( C \)

End If

End For

End

The fractal compaction algorithm was implemented in C and was run on Sun-SPARC stations. The code has two sections - the Toggle section and the Compact section. The toggle section has a library of standard logic gates and memory elements (NAND, NOR, AND, OR, NOT, XOR, XNOR, DFF) which get instantiated depending on the design netlist. The input vectors are then read in one after another and the output at each of the gate outputs is determined. The toggle counter for the gate is incremented if its output switches from 0 \( \rightarrow \) 1. Once all the input vectors have been read, the total number of toggles that each gate output went through is available. The Compact section uses the toggle information to create vector subsets which are then clustered to form the compact vector set.

5.1.3 Compaction Quality Parameters

Two parameters are used to judge the quality of the compaction algorithm: Compaction Ratio \( X \) and the Estimation Error Percentage (EEP). The compaction
ratio $X$ is the ratio of the size of the compact vector set $C$ to the size of the original vector set $I$. i.e.,

$$X = \frac{\text{Vector size after compaction}}{\text{Vector size before compaction}} \times 100$$  \hspace{1cm} (5.3)

The factors influencing the compaction ratio are the spatial and temporal correlation in the input vector set, input vector set size and allowable error in estimation. Since the time for power estimation is determined by the compaction ratio, the compaction algorithm should try to maximize compaction. However, it should be noted that for good power estimation, the compaction algorithm should also take into consideration another parameter, namely the estimation error percentage. The estimation error percentage is defined as the percentage error in average power estimation with the original vector set and compact vector set.

$$EEP = \left(\frac{P_{\text{orig}} - P_{\text{comp}}}{P_{\text{orig}}}\right) \times 100$$  \hspace{1cm} (5.4)

In general, as the compaction ratio increases, the estimation error percentage also increases since more and more of the input vectors will be left out in the compact vector set. Thus, a good compaction algorithm should not only aim for a higher compaction ratio, but also limit EEP to within smaller and acceptable ranges. Such an algorithm can significantly reduce the time for power estimation through simulation.

### 5.1.4 Compaction Time

Though the time for power estimation is influenced by the compaction ratio, there is always an overhead from compaction time which is the time for generating the compact vector set. It is imperative that this overhead be kept small so that power estimation is faster. The factors influencing the compaction time are the processor speed, input vector size, netlist parsing, logic gate/flipflop data structure and vector subset size. Since a large netlist or input vector set is expected to increase the compaction time, the contribution from other factors should be kept very
small. For example, the fractal compaction algorithm uses a simple data structure for logic gates and flipflops to keep track of toggle counts and present/previous output value. The netlist is parsed once and the gates/flipflops are instantiated using a simple data structure. The data structure keeps track of the toggle counts and the current/previous output values for the logic elements. Table 5.1 and Table 5.2 show the compaction time for some of the benchmark circuits used for testing the fractal algorithm. It can be noticed that the compaction time overhead is smaller for simple circuits such as C17, C432, S27, S298, etc., while it is relatively higher for larger circuits such as C3540, C1908, S1494, etc.
5.2 Benchmark circuits

The Fractal compaction algorithm was tested with circuits from the ISCAS benchmark suite. These circuits were used in order to compare the fractal algorithm with previously published results [40, 21].

5.2.1 Combinational Circuits

The ISCAS85 combinational benchmarks that were used for testing the fractal compaction algorithm are shown in Table 5.3. The table also includes the number of nets and primary input signals for each of the circuits tested. The logic circuits are multi-leveled and are comprised of the following combinational gate elements: NOT, AND, OR, XOR, NAND, and NOR. The netlists for these circuits were converted from the ISCAS format to the PowerMill format using a netlist converter in order to estimate power before and after compaction. These circuits were tested with input vector sets of size 4000 for comparison with [40]. However, tests with vector sets of size ranging from 100 to 100K were also carried out for C432, C880, C1355 and C1908.


<table>
<thead>
<tr>
<th>Circuit</th>
<th>Primary Inputs</th>
<th>DFF</th>
<th>Primary Outputs</th>
<th>Net Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>S27</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>S298</td>
<td>3</td>
<td>6</td>
<td>14</td>
<td>432</td>
</tr>
<tr>
<td>S526</td>
<td>3</td>
<td>6</td>
<td>21</td>
<td>526</td>
</tr>
<tr>
<td>S832</td>
<td>18</td>
<td>19</td>
<td>5</td>
<td>832</td>
</tr>
<tr>
<td>S1196</td>
<td>14</td>
<td>14</td>
<td>18</td>
<td>1196</td>
</tr>
<tr>
<td>S1238</td>
<td>14</td>
<td>14</td>
<td>18</td>
<td>1238</td>
</tr>
<tr>
<td>S1423</td>
<td>17</td>
<td>5</td>
<td>74</td>
<td>1423</td>
</tr>
<tr>
<td>S1488</td>
<td>7</td>
<td>19</td>
<td>6</td>
<td>1488</td>
</tr>
<tr>
<td>S1494</td>
<td>7</td>
<td>19</td>
<td>6</td>
<td>1494</td>
</tr>
</tbody>
</table>

Table 5.4: ISCAS89 Sequential Circuits Tested

5.2.2 Sequential Circuits

In order to test the fractal compaction algorithm on sequential circuits, some of the circuits from the ISCAS89 suite were chosen. The circuits that were tested are shown in Table 5.4 along with the number of primary input and output lines. Since the only sequential logic element used in all these circuits was the D Flipflop (DFF), the logic library in the compaction code was augmented with a template for DFF. The DFF was designed with NAND gates and its Clear input is assumed to be kept Disabled always. The Clock input is assumed to be implicit and synchronized with the start of each input vector.

The compaction tests were carried for both uncorrelated and correlated (0.5) input vectors of size 1000 each. The vector set size of 1000 was chosen in order to compare the results from [21]. The comparison of compaction results with [21] was done only for uncorrelated inputs since results for correlated inputs were not reported in the literature.
5.3 Input Vector Generation

The input data used for testing the fractal algorithm were both uncorrelated and correlated. The uncorrelated signals were generated using the Linear Congruential Generator (LCG) available in the C library functions `lrand48` and `srand48`. In order to ensure that the signals are uncorrelated, the middle order bit of the original random number was masked and the binary random number was found. In general, the vector set size used for the experiments was 4000, in order to compare the compaction results with the previously published results for combinational circuits [40].

In order to generate input vectors with spatial and temporal correlation, signal processing tools available in Matlab were used. Temporal (intra-signal) correlation was introduced into the uniformly distributed uncorrelated signals \( W^{n \times m} \sim \text{unif}[-\sqrt{3}, \sqrt{3}] \) using the First Order Regressive Function (FORF).

\[
T^n = (a \times T^{(n-1)}) + \sqrt{(1 - a^2)} \times W^n
\]  

(5.5)

where \( T^{n \times m} \) is the temporally correlated input vector, \( W^{-1} = 0 \) and \( 0 \leq a < 1 \) is the parameter controlling the intra-signal correlation. \( T^{n \times m} \) thus generated will have mean \( \mu = 0 \) and variance \( \sigma = 1 \) similar to Gaussian distribution.

The temporally correlated input vector is then converted into spatially and temporally correlated vector \( Y^{n \times m} \) using a special matrix \( C^{m \times m} \) that controls spatial correlation \( 0 \leq b < 1 \). The square matrix \( C^{m \times m} \) has unity along the diagonal while the other array elements are filled with \( b \). The matrix \( Y^{n \times m} \) is then

\[
Y^{n \times m} = \text{transpose}(V^{m \times m} \times \sqrt{D^{m \times m}} \times \text{transpose}(T^{n \times m}))
\]  

(5.6)

where \( D \) is the eigenvalue matrix of \( C \) and \( V \) is the eigenvector matrix of \( C \). The real valued matrix \( Y^{n \times m} \) is then converted to binary matrix \( S^{n \times m} \) such that \( s_m(n) = 1 \) if \( y_m(n) \geq 1 \) or 0, otherwise.

The spatial and temporal correlation values introduced in the random data was 0.5. This was achieved by setting the temporal correlation parameter \( a \) to 0.5 and
the spatial correlation parameter $b$ to 0.5. However, to test the effect of spatial and temporal correlation on the compaction ratio and estimation error, tests were repeated for some of the circuits with correlation values of 0.3, 0.5, 0.7 and 0.9. Further, to understand the influence of vector sizes on the Compaction Ratio $X$ and the Estimation Error Percentage, tests were carried out on C432, C880, C1355 and C1908 with vector sizes ranging from 100 to 100,000.

5.4 PowerMill

In order to test the effectiveness of the fractal compaction algorithm, the benchmark circuits listed in Tables 5.3 and 5.4 were simulated using PowerMill. PowerMill is an event-driven, transistor-level power simulator/analyzer used for estimating power consumption, hot-spot detection, leakage current estimation, etc. [21, 15]. The event-driven algorithm senses events in terms of small voltage changes instead of logic transitions, making it possible to detect non-digital behaviors also. The salient difference between SPICE and PowerMill is that, in SPICE, the model equations and partial derivatives are expressed in terms of complicated analytical equations, and must be evaluated whenever the operating point changes. PowerMill, on the other hand, the models are pre-computed and stored in tables which can be looked up fast.

The input files that have to be specified to PowerMill for simulating a circuit include: the design netlist in the format unique to PowerMill, input vector file, technology file and the command file. PowerMill provides various output information such as average power, peak power, number of toggles at an internal node etc. For evaluating the fractal compaction algorithm, several benchmark circuits were simulated with both the original vector set and compact vector set. The average power at $V_{dd}$, $P_{avg}$ has been used as a measure of the power consumed by the simulated circuit. From the $P_{avg}$ values for the original and compact vectors, the estimation error percentage is determined.
CHAPTER 6

EXPERIMENTAL RESULTS AND ANALYSIS

This chapter discusses the compaction and simulation results from the fractal compaction algorithm for various benchmark circuits. The results are presented with the twin objectives of highlighting the salient features and advantages of the fractal algorithm and of comparing the results with the previously published algorithms [40, 21]. The comparison is based on the two previously mentioned compaction parameters - compaction ratio $X$ and estimation error percentage. The first section discusses the results for uncorrelated input vectors while the second for spatially and temporally correlated input vectors. In each of the sections, the results are presented separately for combinational and sequential circuits.

6.1 Uncorrelated Input Vectors

The uncorrelated input vector sets were generated using C library functions as explained in Section 5.3. For combinational circuits, a vector set size of 4000 was used in order to compare the results with [40]. In addition, vector sets of size ranging from 100 to 100K was also used with circuit C432 to check the effect of vector size on compaction ratio. However, for sequential circuits, a vector set size of 1000 was used since the results reported in [21] were for that vector set size. The compaction code and the power estimation tool PowerMill was run on a Sun SPARCstation 10 for uncorrelated input vectors for the combinational circuits. However, the uncorrelated input vector sets for the sequential circuits were tested on a Sun Ultra1 workstation.
Table 6.1 shows the compaction ratio $X$, and the estimation error percentage for different benchmark circuits simulated with a vector set of size 4000. The compaction ratio $X$ obtained for the circuits varied from 65.57X to 18.95X, with estimation error below 2.4%. An important observation is that the compaction ratio increases as the size of the circuit becomes bigger. The reasoning behind this feature is that a bigger circuit will tend to have more internal nodes and hence a higher correlation in toggle distribution among the nodes which in turn results in larger compaction. The estimation error percentage for the circuits ranges from 1.7% to 2.8%. The smaller error percentage indicates that the compact vector set has retained most of the toggle information in the original larger vector set. Thus the performance of fractal compaction algorithm is significant not only in achieving high compaction but also in limiting the estimation error.

**Reduced Simulation Time**

It was mentioned in Section 1.1 that the main objective of a compaction algorithm is to reduce the time for estimating power by simulation. Table 6.2 shows that considerable simulation time can be saved while accurately estimating the
Table 6.2: Simulation Time Before and After Compaction

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Before Compaction</th>
<th>After Compaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>C17</td>
<td>85 sec</td>
<td>5 sec</td>
</tr>
<tr>
<td>C432</td>
<td>3240 sec</td>
<td>124 sec</td>
</tr>
<tr>
<td>C499</td>
<td>11403 sec</td>
<td>191 sec</td>
</tr>
<tr>
<td>C880</td>
<td>6751 sec</td>
<td>299 sec</td>
</tr>
<tr>
<td>C1355</td>
<td>8917 sec</td>
<td>173 sec</td>
</tr>
<tr>
<td>C1908</td>
<td>17338 sec</td>
<td>293 sec</td>
</tr>
<tr>
<td>C3540</td>
<td>45831 sec</td>
<td>1366 sec</td>
</tr>
</tbody>
</table>

power by using the compact vector set generated by the fractal compaction algorithm. Besides, the savings in simulation time is realized while still limiting the estimation error to values well within acceptable limits. The reduction in simulation time due to the fractal algorithm will in turn reflect in better economics such as shorter design time and lower design cost. However, there is a small overhead to the simulation time in terms of the time for generating the compact vector set, as was mentioned in Section 5.1.4.

Comparison of Fractal Algorithm with Symbolic Vector Algorithm [40]

Table 6.3 shows the comparison of the results of the fractal compaction algorithm with results from the symbolic vector algorithm proposed by Tsui et al. [40]. The comparison is based on the same set of circuits and vector set size as was used in [40]. The symbolic vector algorithm achieved a 20.00X (max) compaction with an estimation error of 3.48% (max) and 1.93% (avg). The fractal compaction algorithm, on the other hand, has a compaction of 65.57X (max) and 39.20X (avg) with the estimation error of 2.40% (max) and 1.96% (avg). It is evident from Fig. 6.1 and Fig. 6.2 that the fractal algorithm compacts two times more on the average while limiting the estimation error to within the same range as [40]. Since the fractal algorithm exploited the spatial and the temporal correlation in the input vector,
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Compaction Ratio X</th>
<th>Estimation Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fractal Symbolic</td>
<td>Fractal Symbolic</td>
</tr>
<tr>
<td>C432</td>
<td>25.47X 20X</td>
<td>2.07% 3.48%</td>
</tr>
<tr>
<td>C880</td>
<td>21.97X 20X</td>
<td>2.22% 3.34%</td>
</tr>
<tr>
<td>C1355</td>
<td>50.63X 20X</td>
<td>1.29% 0.24%</td>
</tr>
<tr>
<td>C1908</td>
<td>65.57X 20X</td>
<td>1.83% 0.67%</td>
</tr>
<tr>
<td>C3540</td>
<td>32.52X 20X</td>
<td>2.40% 1.81%</td>
</tr>
</tbody>
</table>

Table 6.3: Comparison of Fractal Algorithm with Symbolic Vector Algorithm

Figure 6.1: Comparison of Compaction Ratio

it was able to compact much more than the other method. An added advantage of the fractal algorithm is that it obviates the need for calculating the correlation coefficients which is computationally intensive and instead uses the simpler fractal equivalent, the Hurst parameter.

Effect of Input Vector Set Size on Compaction Ratio

As discussed earlier, the compaction ratio increases when the circuit size (i.e. netlist size) becomes bigger. Interestingly enough, the effect of input vector size on compaction ratio is also the same as for the circuit size. A large vector set increases the correlation in the toggle distribution among the internal nodes since the propensity for the internal nodes to toggle increases with the vector set size. Thus as the correlation in toggle distribution increases, the compaction ratio also
Increases. Table 6.4 shows the compaction ratio for circuits C880, C1355 and C1908 for different vector set sizes. A steady increase in compaction ratio of the circuits, as the vector set size increases is evident from the table. Table 6.5 shows the compaction results for the benchmark circuit C432, with different input vector sizes ranging from 100 to 100K. Again, the compaction ratio increases dramatically for larger vector set sizes. For example, the compaction ratio for vector sizes of 50K and 100K are 111X and 295X respectively with estimation errors of only 3.43% and 1.04% respectively. Since compaction ratio is strongly influenced by the input vector set size, merely stating that a compaction algorithm has a compaction ratio of 100X without specifying the input vector size would convey incomplete information.
Table 6.5: Compaction Results of C432 for Different Vector Set Sizes

<table>
<thead>
<tr>
<th>Vector Size</th>
<th>Compaction Ratio X</th>
<th>Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.81X</td>
<td>1.42%</td>
</tr>
<tr>
<td>500</td>
<td>5.61X</td>
<td>6.92%</td>
</tr>
<tr>
<td>2000</td>
<td>16.52X</td>
<td>5.42%</td>
</tr>
<tr>
<td>4000</td>
<td>25.47X</td>
<td>2.07%</td>
</tr>
<tr>
<td>5000</td>
<td>25.25X</td>
<td>2.12%</td>
</tr>
<tr>
<td>10000</td>
<td>48.78X</td>
<td>5.57%</td>
</tr>
<tr>
<td>50000</td>
<td>111.11X</td>
<td>3.43%</td>
</tr>
<tr>
<td>100000</td>
<td>295.85X</td>
<td>1.04%</td>
</tr>
</tbody>
</table>

6.1.2 Sequential Circuits

Compaction Results for Sequential Circuits

Table 6.6 shows the compaction ratio X and the estimation error percentage for various sequential circuits with uncorrelated input vector sizes of 1000. The maximum compaction achieved was 25.64X while the average compaction was 5X. The estimation error percentage was 6.00% (max) and 2.17% (avg). Just as it was in the case of combinational circuits, the compaction ratio for sequential circuits also improves as the circuit size increases. This can be attributed to the higher correlation in the toggle distribution of the internal nodes of the circuits. Since the compaction results for the sequential circuits are significant, it is obvious that the inclusion of memory elements such as D flipflops does not drastically affect the fractal compaction procedure. Thus, the fractal compaction algorithm can make power estimation of sequential circuits faster while producing accurate results.

Comparison of Fractal Algorithm with Back Propagation Algorithm [21]

Huang et al. [21] published a compaction algorithm that generated compact vector sets by back propagating the toggle information from the output end of the circuit back towards the primary input. Table 6.7 shows the comparison of...
Table 6.6: Compaction Results for Sequential Circuits

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Compaction Ratio X</th>
<th>Estimation Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>S27</td>
<td>8.47X</td>
<td>4.00%</td>
</tr>
<tr>
<td>S298</td>
<td>5.15X</td>
<td>1.35%</td>
</tr>
<tr>
<td>S832</td>
<td>8.84X</td>
<td>1.84%</td>
</tr>
<tr>
<td>S1196</td>
<td>15.15X</td>
<td>2.37%</td>
</tr>
<tr>
<td>S1238</td>
<td>22.22X</td>
<td>0.30%</td>
</tr>
<tr>
<td>S1423</td>
<td>13.15X</td>
<td>6.00%</td>
</tr>
<tr>
<td>S1488</td>
<td>22.21X</td>
<td>5.40%</td>
</tr>
<tr>
<td>S1494</td>
<td>25.64X</td>
<td>4.31%</td>
</tr>
<tr>
<td>S5378</td>
<td>25.00X</td>
<td>3.95%</td>
</tr>
</tbody>
</table>

Figure 6.3: Comparison of Compaction Ratio

compaction results from the fractal compaction algorithm with that from the back propagation algorithm. The back propagation algorithm had a 15X (max) compaction with an estimation error of 6.38% (max) and 1.93% (avg). The fractal compaction algorithm, on the other hand has a compaction of 25.64X (max) and 39.20X (avg) with the estimation error of 6.00% (max) and 1.96% (avg). Thus the fractal algorithm compacts 4 times more on the average with similar errors in estimation as [21] [See Fig. 6.3 and Fig. 6.4].
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Compaction Ratio X</th>
<th>Estimation Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fractal</td>
<td>Back Propagation</td>
</tr>
<tr>
<td>S1196</td>
<td>15.15X</td>
<td>5X</td>
</tr>
<tr>
<td>S1238</td>
<td>22.22X</td>
<td>5X</td>
</tr>
<tr>
<td>S1423</td>
<td>13.15X</td>
<td>5X</td>
</tr>
<tr>
<td>S1488</td>
<td>22.21X</td>
<td>5X</td>
</tr>
<tr>
<td>S1494</td>
<td>25.64X</td>
<td>5X</td>
</tr>
<tr>
<td>S5378</td>
<td>25.00X</td>
<td>5X</td>
</tr>
</tbody>
</table>

Table 6.7: Comparison of Fractal Algorithm with Back Propagation Algorithm

Figure 6.4: Comparison of Estimation Error
6.2 Correlated Input

Since in typical real-world applications, the input vectors are spatially and temporally correlated, the fractal compaction algorithm was tested with correlated vectors also. The spatial and temporal correlation was changed over a spectrum of 0.3 to 0.9 in order to assess the effect of correlation on compaction ratio. Correlated vector sets of size 4000 were used for all the tests carried out on combinational circuits. However, for sequential circuits vector sets of size 1000 were used. The various compaction tests and simulation runs for both combinational and sequential circuits were carried out on Sun Ultral stations.

6.2.1 Combinational Circuits

Table 6.8 shows the compaction ratio for combinational circuits with spatial and temporal correlation of 0.5 and 0.9. When comparing the compaction ratio achieved for these circuits with uncorrelated vectors, it is obvious that the compaction ratio has decreased significantly with correlated input vectors. In order to ascertain the negative influence of correlation on compaction, the spatial and temporal correlation was varied from 0.3 to 0.9 and the compaction tests were carried out for circuits C17 and C432 and the results are presented in Table 6.9 and Fig. 6.5. Again, as the spatial and temporal correlation in the input vectors increased, the compaction ratio decreased consistently.

Compaction tests were extended further to find out the influence of temporal correlation alone on the compaction ratio. To this end, the tests were carried out on the circuit C432 for input vectors of size 4000 with temporal correlation varying from 0.3 to 0.9. The spatial correlation in the input vectors was set to 0.0. The tests were also repeated for vectors with only spatial correlation. The results from these tests are presented in Table 6.10. It can be observed that the compaction ratio decreases irrespective of whether the correlation is only spatial or only temporal or both spatial and temporal.
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Compaction Ratio X</th>
<th>Correlation 0.5</th>
<th>Correlation 0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>C17</td>
<td>18.95X</td>
<td>6.39X</td>
<td>2.51X</td>
</tr>
<tr>
<td>C432</td>
<td>25.47X</td>
<td>9.54X</td>
<td>2.39X</td>
</tr>
<tr>
<td>C880</td>
<td>21.97X</td>
<td>11.49X</td>
<td>3.62X</td>
</tr>
<tr>
<td>C1355</td>
<td>57.14X</td>
<td>15.87X</td>
<td>7.59X</td>
</tr>
<tr>
<td>C3540</td>
<td>32.78X</td>
<td>8.26X</td>
<td>3.66X</td>
</tr>
</tbody>
</table>

Table 6.8: Compaction Ratio for Correlated Input Vectors (Before Modification)

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Compaction Ratio X</th>
<th>C17</th>
<th>C432</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncorrelated</td>
<td>18.95X</td>
<td>25.47X</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>9.43X</td>
<td>12.53X</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>6.39X</td>
<td>9.54X</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>3.96X</td>
<td>6.29X</td>
<td></td>
</tr>
<tr>
<td>0.9</td>
<td>2.51X</td>
<td>2.39X</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.9: Compaction Ratio for Various Correlation (Before Modification)

In order to understand the reason for the drop in compaction as correlation increases, it is necessary to fully investigate the effect of correlation on the toggle distribution of the switching nodes. This is because, any change in the toggle distribution can directly influence the various parameters, such as the total toggle counts, range parameter etc., used by the compaction algorithm.

6.2.2 Effects of Spatial and Temporal Correlation

Reduction in Toggle Counts

In general, spatial and temporal correlation in input vectors reduces the toggling
Figure 6.5: Compaction Ratio for Various Correlation (Before Modification)

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Compaction Ratio X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Temporal Only</td>
</tr>
<tr>
<td>Uncorrelated</td>
<td>25.47X</td>
</tr>
<tr>
<td>0.3</td>
<td>17.54X</td>
</tr>
<tr>
<td>0.5</td>
<td>16.80X</td>
</tr>
<tr>
<td>0.7</td>
<td>11.36X</td>
</tr>
<tr>
<td>0.9</td>
<td>3.92X</td>
</tr>
</tbody>
</table>

Table 6.10: Compaction Ratio for C432 with Temporal, Spatial and Spatio-Temporal Correlation (Before Modification)
tendency of the switching nodes [11]. The reason is that, for a correlated input vector set, the input does not change quite often so that toggling becomes less frequent. This fact is reiterated in Table 6.11 and Table 6.12 that presents the toggle counts of circuits C17 and C432 for input vectors with different correlation values. A new parameter called the Gate Toggle Frequency (GTF) was used to effectively highlight the effect of correlation on toggle counts. GTF is defined as the toggle count measure normalized with the vector set size and circuit size. It can be represented quantitatively as:

\[
GTF = \frac{\text{total toggle count}}{(\text{vector set size} \times \text{number of switching nodes})}
\]  

(6.1)

It is clear from both the tables and Fig. 6.6 that GTF decreases as the correlation increases.

**Reduction in Correlation of Toggle Distribution**

The correlation in internal node toggle distribution is an important factor influencing the fractal compaction algorithm since the algorithm exploits it for generating the compact vector set. Interestingly enough, it was identified from the ACF analysis for the toggle distribution that the correlation in internal node toggle distribution reduces as the input vector correlation increases. This fact becomes evident from the ACF plots shown in Fig. 6.7 and Fig. 6.8 for the circuits C432
<table>
<thead>
<tr>
<th>Correlation</th>
<th>Toggle Count</th>
<th>Gate Toggle Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncorrelated</td>
<td>235937</td>
<td>0.37</td>
</tr>
<tr>
<td>0.1</td>
<td>225846</td>
<td>0.35</td>
</tr>
<tr>
<td>0.3</td>
<td>204014</td>
<td>0.32</td>
</tr>
<tr>
<td>0.5</td>
<td>173777</td>
<td>0.27</td>
</tr>
<tr>
<td>0.7</td>
<td>139194</td>
<td>0.22</td>
</tr>
<tr>
<td>0.9</td>
<td>81436</td>
<td>0.13</td>
</tr>
</tbody>
</table>

Table 6.12: Toggle Counts and Gate Toggle Frequency of C432 for Various Correlation

Figure 6.6: Gate Toggle Frequency for Various Correlation Values
and C880 generated with correlated input vectors. However, the drop in ACF can be attributed to the reduced toggle frequency of the internal nodes as input vector correlation increases.

![ACF Plot for C432 with Correlated Input](image)

Figure 6.7: ACF Plot for C432 with Correlated Input

**Increase in Range Parameter Value**

A direct result of the drop in ACF of internal node toggle distribution is the increase in the Range parameter which is a measure of the spread in toggle count values of the internal nodes. This is obvious from Table 6.13 and Fig. 6.9 which show the Range value for circuits C432 and C880 for various input correlation. Since the Range value decides the size of the vector subset, any change in its value can strongly influence the compaction results that can be achieved by the fractal algorithm.

Having investigated the effect of correlation on the three important parameters
Figure 6.8: ACF Plot for C880 with Correlated Input

<table>
<thead>
<tr>
<th>Correlation</th>
<th>Range R</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>C880</td>
</tr>
<tr>
<td>Uncorrelated</td>
<td>833</td>
</tr>
<tr>
<td>0.3</td>
<td>1453</td>
</tr>
<tr>
<td>0.5</td>
<td>1658</td>
</tr>
<tr>
<td>0.7</td>
<td>2518</td>
</tr>
<tr>
<td>0.9</td>
<td>4936</td>
</tr>
</tbody>
</table>

Table 6.13: Range Parameter of C432 and C880 for Various Correlation (Before Modification)
of the internal node toggles, the reason for the drop in compaction ratio as the input correlation increased had to be analyzed. It was previously mentioned that the fractal compaction algorithm generates vector subsets based on the Range parameter value. Since the Range parameter value increases as the input correlation increases, the vector subsets that are generated are bigger in size. These bigger subsets are then mapped against each other so as to result in fewer subsets that are concatenated together to form the compact vector set. However, it should be noted that since the vector subsets are themselves bigger in size, the compact vector set, in turn, is bigger resulting in a smaller compaction ratio. Thus, the main reason for the drop in compaction as input correlation increases can be attributed to the increase in the Range parameter value.

6.2.3 Modified Range Parameter $R_{mod}$

An increase in vector subset size will decrease compaction and hence the vector subset size should be kept small. However, smaller vector subsets will increase the number of subsets so much that the time needed for generating the compact set could be prohibitive. Thus, the vector subset size should be a compromise between the compaction ratio and the time for compaction. In order to arrive at such a size, the Range parameter $R$ which decides the subset size was modified to $R_{mod}$. 
Table 6.14: Comparison of Compaction Ratio Achieved with R and $R_{mod}$

<table>
<thead>
<tr>
<th>Circuit</th>
<th>With R</th>
<th>With $R_{mod}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>$X_{total}$</td>
</tr>
<tr>
<td>C17</td>
<td>149</td>
<td>6.38X</td>
</tr>
<tr>
<td>C432</td>
<td>1658</td>
<td>9.54X</td>
</tr>
<tr>
<td>C880</td>
<td>9541</td>
<td>11.49X</td>
</tr>
<tr>
<td>C1355</td>
<td>3620</td>
<td>15.87X</td>
</tr>
<tr>
<td>C1908</td>
<td>1555</td>
<td>100.00X</td>
</tr>
<tr>
<td>C3540</td>
<td>6697</td>
<td>8.26X</td>
</tr>
</tbody>
</table>

$R_{mod}$ was arrived at heuristically and can be defined as

$$R_{mod} = R \times \frac{\text{total toggle count}}{(\text{input vector size} \times \text{gate count})}.$$  \hspace{1cm} (6.2)\)

It can be viewed as the original Range R normalized by two parameters, namely vector size and gate count that influence both compaction ratio and compaction time.

The modified Range parameter was subsequently used in the fractal compaction algorithm and various compaction tests were carried out. Table 6.14 shows the $R_{mod}$ and compaction ratio values for various circuits tested with input vector of size 4000 and correlation (spatial and temporal) 0.5. The $R_{mod}$ values for the circuits are smaller than R and the compaction ratio achieved by using $R_{mod}$ are also comparatively higher. Besides, the use of $R_{mod}$ for uncorrelated input vectors still resulted in a higher compaction. Similar results can be also observed with the compaction ratio of the circuits C432 and C880 shown in Table 6.15 for various input correlation values. When these compaction results are compared with those achieved by using R (see Fig. 6.10), it can be concluded that the effect of input vector correlation on compaction ratio has been effectively countered by using $R_{mod}$.\)
Figure 6.10: Comparison of Range with Modified Range

<table>
<thead>
<tr>
<th>Correlation</th>
<th>( R_{\text{mod}} )</th>
<th>( X )</th>
<th>( R_{\text{mod}} )</th>
<th>( X )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncorrelated</td>
<td>250</td>
<td>44.94X</td>
<td>574</td>
<td>54.05X</td>
</tr>
<tr>
<td>0.3</td>
<td>378</td>
<td>33.05X</td>
<td>1307</td>
<td>39.21X</td>
</tr>
<tr>
<td>0.5</td>
<td>367</td>
<td>26.84X</td>
<td>1362</td>
<td>24.53X</td>
</tr>
<tr>
<td>0.7</td>
<td>447</td>
<td>19.41X</td>
<td>1864</td>
<td>14.54X</td>
</tr>
<tr>
<td>0.9</td>
<td>512</td>
<td>8.81X</td>
<td>1752</td>
<td>10.00X</td>
</tr>
</tbody>
</table>

Table 6.15: Range Parameter of C432 and C880 for Various Correlation Values (After Modification)
6.2.4 Compaction Results of Combinational Circuits with $R_{mod}$

Though the previous section highlighted the improvement in compaction ratio when the modified range $R_{mod}$ is used, the error in the power estimate still needs to be evaluated. To this end, the compaction test and the subsequent PowerMill simulations were done for the various combinational circuits with input vectors of size 4000. The spatial and temporal correlation values used were 0.5 and 0.9. Table 6.16 shows the compaction results of the simulation tests. For a correlation of 0.5, the estimation errors varied between 0.29% and 4.86% while for correlation of 0.9 it was between 3.74% and 6.89%.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Correlation 0.5</th>
<th>Correlation 0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compaction X</td>
<td>Error %</td>
</tr>
<tr>
<td>C17</td>
<td>15.81X</td>
<td>0.29%</td>
</tr>
<tr>
<td>C432</td>
<td>27.21X</td>
<td>0.83%</td>
</tr>
<tr>
<td>C880</td>
<td>24.53X</td>
<td>3.25%</td>
</tr>
<tr>
<td>C1355</td>
<td>33.05X</td>
<td>3.95%</td>
</tr>
<tr>
<td>C1908</td>
<td>125.00X</td>
<td>4.04%</td>
</tr>
<tr>
<td>C3540</td>
<td>44.44X</td>
<td>4.86%</td>
</tr>
</tbody>
</table>

Table 6.16: Compaction Results with $R_{mod}$ for Correlated Input

Figure 6.11: Comparison of Compaction Ratio
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Before Compaction</th>
<th>After Compaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>C17</td>
<td>14 sec</td>
<td>1 sec</td>
</tr>
<tr>
<td>C432</td>
<td>694 sec</td>
<td>18 sec</td>
</tr>
<tr>
<td>C880</td>
<td>1252 sec</td>
<td>52 sec</td>
</tr>
<tr>
<td>C1355</td>
<td>1663 sec</td>
<td>48 sec</td>
</tr>
<tr>
<td>C1908</td>
<td>3192 sec</td>
<td>24 sec</td>
</tr>
<tr>
<td>C3540</td>
<td>7411 sec</td>
<td>109 sec</td>
</tr>
</tbody>
</table>

Table 6.17: Simulation Time Before and After Compaction

Table 6.17 shows the simulation time with PowerMill for estimating power with correlated (0.5) input vectors before and after compaction. The significant reduction in simulation time achieved by the fractal compaction algorithm is obvious from the table. Another interesting observation can be made by comparing the simulation time needed for uncorrelated and correlated input vector sets. From Fig. 6.12 and Fig. 6.13, it can be noticed that the simulation time for correlated vectors is much lesser than that for uncorrelated vectors. This is because PowerMill is an event-driven simulator and the reduced toggle frequency for correlated inputs causes lesser number of events to be simulated. Thus, the event-driven nature of the simulation tool compounded by the smaller size of the compact vector set results in a significant speed up in the time for power estimation.

### 6.2.5 Compaction Results of Sequential Circuits with $R_{mod}$

Having analyzed the effects of correlated input vectors on combinational circuits, similar tests were done on sequential circuits. The sequential circuits from the ISCAS89 benchmark suite were used. The vectors are of size 1000 and the spatial and temporal correlation used was 0.5. In order to check the influence of correlation on compaction ratio, vector sets with correlation varying from 0.3 to 0.9 were also used. The compaction tests and the simulation runs with PowerMill
Figure 6.12: Comparison of Simulation Time Before Compaction

Figure 6.13: Comparison of Simulation Time After Compaction
were executed on Sun Ultra1 stations.

Table 6.18 shows the effect of correlation on the Range parameter and compaction ratio of the circuit S298. Just as it was for combinational circuit, the Range parameter R of circuit S298 for correlated vectors was higher and resulted in smaller compaction. However, with \( R_{mod} \), the fractal algorithm generated smaller vector subsets and compacted better. The compaction results for various sequential circuits when simulated with correlated (0.5) input vectors are presented in Table 6.19. The maximum and average compaction achieved are 25X and 12X with estimation error ranging from 0.90% to 4.18%. The compaction ratio for large circuits such as S1238, S1488, etc., is high since the increased number of internal nodes in large circuits can increase the correlation in their toggle distribution. Finally, the significant savings in power estimation time by simulation is shown in Table 6.20. The fewer internal node toggles due to the correlation in input vectors, in addition to the smaller compact vector set, resulted in a significant reduction in the simulation time needed for power estimation.
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Compaction Ratio X</th>
<th>Estimation Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>S27</td>
<td>7.81X</td>
<td>1.49%</td>
</tr>
<tr>
<td>S298</td>
<td>4.54X</td>
<td>1.70%</td>
</tr>
<tr>
<td>S832</td>
<td>7.87X</td>
<td>4.18%</td>
</tr>
<tr>
<td>S1196</td>
<td>12.04X</td>
<td>0.90%</td>
</tr>
<tr>
<td>S1238</td>
<td>18.18X</td>
<td>2.77%</td>
</tr>
<tr>
<td>S1488</td>
<td>25.64X</td>
<td>1.69%</td>
</tr>
</tbody>
</table>

Table 6.19: Compaction Results of Sequential Circuits for Correlated Input (After Modification)

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Simulation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Before Compaction</td>
</tr>
<tr>
<td>S27</td>
<td>7 sec</td>
</tr>
<tr>
<td>S298</td>
<td>267 sec</td>
</tr>
<tr>
<td>S832</td>
<td>230 sec</td>
</tr>
<tr>
<td>S1196</td>
<td>336 sec</td>
</tr>
<tr>
<td>S1238</td>
<td>357 sec</td>
</tr>
<tr>
<td>S1488</td>
<td>369 sec</td>
</tr>
</tbody>
</table>

Table 6.20: Simulation Time of Sequential Circuits for Correlated Input
CHAPTER 7

CONCLUSION

In this dissertation, an algorithm for compacting the input vector has been presented. The algorithm helps to reduce the simulation time for estimating power while still limiting the error in estimation to low levels. Fractal concepts have been used to identify correlation in the toggle distribution of the internal nodes. The Hurst parameter was used to quantify correlation in the toggle distribution. Since estimation of the Hurst parameter is simpler, estimation of correlation coefficients has been obviated. The fractal algorithm was tested with combinational and sequential circuits from the ISCAS benchmark suite. Both uncorrelated and correlated primary inputs were used in the tests. The performance of the fractal algorithm in those tests were quite impressive. When compared to previously reported results for combinational circuits, the fractal algorithm compacts two times more with comparable errors in estimation. For sequential circuits the fractal algorithm achieved a compaction that was four times higher than previous results.

The fractal compaction algorithm can be extended, with suitable modifications, for compacting the input vectors used in other applications such as $I_{ddq}$ testing of logic circuits.
REFERENCES


