國 立 交 通 大 學

電子工程學系電子研究所

博 士 論 文

矽智產設計的功率估測方法之研究

On Power Estimation Methods for Silicon Intellectual Properties

研 究 生 ：許 智 揚
指導教授 ：周 景 揚 博士
劉 建 男 博士

中華民國 九十四 年 六 月

矽智產設計的功率估測方法之研究

學生:許智揚　　　　　　　　　　　　　　　　指導教授:周景揚 博士

劉建男 博士

國立交通大學

電子工程學系　電子研究所

摘　要

在這篇論文中，我們針對組合邏輯電路的矽智產單元提出了一系列的功率消耗估測方法及功率消耗模型。因為矽智產供應商為了保護他們的設計概念可能只提供矽智產使用者有限的設計資訊，因此我們對應電晶體層級、閘層級以及功能層級之電路設計資訊提出不同的功率消耗估測方法及功率消耗模型。

針對提供電晶體層級設計資訊的複雜數位電路，矽智產使用者可以採用電晶體層級的模擬估測功率消耗，而圖樣壓縮法已經被採用來加速電晶體層級的功率消耗之估測。在此，我們提出單一序列取樣法減少了傳統圖樣壓縮方法採用的隨機取樣法中無用的圖樣轉換，達到改進傳統圖樣壓縮的效果。並更進一步提出多序列取樣法改善單一序列取樣法過度取樣的缺點。

針對只提供閘層級設計資訊的電路，我們提出一個較小的功率消耗模型。這個功率消耗模型只須要使用輸入信號轉換時電路的零延遲充電及放電電容值當索引就可對照

出實際功率消耗的估測值。因此矽智產使用者只要使用閘層級模擬時得到的零延遲充電及放電電容值就可以查表得到實際功率消耗的估測值。在這個方法中我們採用了分群的方法減小對照表，並利用蒙地卡羅模擬法縮短了建立對照表的時間。根據實驗結果顯示，這個功率模型針對不同的輸入信號序列依然具有高度的準確性。

如果矽智產供應商只願意提供功能層級的設計資訊，則矽智產使用者只能獲得電路輸入及輸出的對應關係。我們針對這種應用提出一個採用類神經網路建立的全新功率消耗模型。假如矽智產供應商提供這樣的功率消耗模型，則矽智產使用者只須要使用功能層級模擬得到的電路輸入及輸出資訊就可以推估電路之功率消耗值。如同實驗結果所顯示，這個功率消耗模型同時具有低複雜度及高準確度的優點。

On Power Estimation Methods for Silicon Intellectual Properties

Student: Chih-Yang Hsu **Advisor: Dr. Jing-Yang Jou**
Dr. Chien-Nan Liu

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

Abstract

In this dissertation, we develop several power estimation and power modeling methods for combinational IPs. Because IP vendors may release only limited design information to protect their knowledge, we propose corresponding methods for the designs with only transistor-level, gate-level and function-level design information.

For complex digital circuits with transistor-level design information, users can estimation the power consumption of designs using transistor-level simulation. In order to reduce the simulation time on transistor-level simulation, we propose a single-sequence sampling approach to improve the performance of vector compaction techniques by reducing the useless transitions in random sampling techniques. A multi-sequence sampling approach is also proposed to improve the over sampling problem in the single-sequence sampling approach.

For the designs with only gate-level design information, we propose a smaller power model approach that only needs a 1-diemension lookup table for each design to map the zero-delay charging and discharging capacitance (CDC) during an input pattern transition to an estimative value of real power consumption. Therefore, IP users can estimate the power consumption according to the CDC values obtained from gate-level simulation. The dynamic grouping method is applied to reduce the size of lookup tables for circuits and Monte-Carlo simulation strategy is applied to reduce the characterization time. The experimental results show that our power model still has high accuracy for different input sequences.

If IP vendors only provide the function-level design information, we propose a novel power model based on neural network that only requires the input and output information of each IP. If IP vendors provide such a power model, IP users can estimation the power consumption of IPs with only input and output information under a function-level simulation. As shown in the experimental results, our power model can have much smaller size with better accuracy.

誌謝

首先我要誠摯地感謝周景揚教授以及劉建男教授。沒有他們的鼓勵、督促及耐心的指導，我不可能完成這篇論文。周老師不只在研究上給予我許多的指導，他也在日常的學習及生活態度上給予我最好的典範。劉教授對於我研究上的指導不遺餘力，對於這篇論文最後的撰寫也提供了他寶貴的意見。

我也要感謝我的另外一位指導教授沈文仁博士(1952~2002)， 他在 1995 年到 2002 年期間一直擔任我的指導教授，雖然他無法親眼看到我的畢業論文，但是在我完成這篇論文的同時，我也希望他能同感榮耀。

我也要感謝我的父母親及家人，沒有他們的支持，我不可能走過這條漫長的道路。由於父母親全力的支持，我才能在攻讀學位期間沒有後顧之憂。由於兄姊對家中的照顧，我才能在沒有家庭經濟壓力的情況下完成這項工作。我也要感謝我的女朋友馮麗卿，在我攻讀博士學位期間漫長的等待與支持。

我同時也感謝實驗室許多學長姊及學弟妹們的幫助。感謝許文俊學長、林景源學長、黃俊達學長、黃恆亮學長對於我研究及實驗室工作大力的幫忙，感謝中央大學電機系薛文燦學弟在類神經功率模型上長期的投入並忍受長時間的奔波，更感謝許多曾經共同與我在同一實驗室度過無數歲月的學弟妹們，他們都是我完成這篇論文的支持者。

謹將這篇論文獻給所有曾經對我的博士論文有過幫助或關心的人。

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the advance of semiconductor technology, the size of devices and the minimal width of metal lines are decreasing rapidly. Therefore, IC designers could integrate many functions, even a whole system, into a single chip. In fact, System-on-a-chip (SoC) is a trend of system integration in recent years. Meanwhile, the operating frequency of designs also rapidly grows up when the semiconductor technology getting advances. Unfortunately, after more devices are integrated into a chip and the operation frequency is increased, the power consumption of a chip is also increasing rapidly.

For SoC designs, most design teams will not design all circuit blocks in the system by themselves. Instead, they integrate many well-designed circuit blocks called intellectual properties (IPs) and some self-designed circuit blocks to build up the complex system in a short time. While designing such complex systems, power consumption is also a very important design issue because of the increasing requirement of portable devices. Traditionally, power estimation is often performed at transistor-level by SPICE-liked simulation. However, this approach is unpractical for SoC designs because the transistor-level description of the whole designs is often too large to be simulated and the IP venders may not provide such low-level description for an IP to protect their knowledge.

For this application, the power estimation method should consider the design information of IPs that IP vendors want to explore to IP users. For example, if the transistor-level descriptions of IPs are provided to IP users, IP users can estimate the power consumption with a transistor-level simulation. The only problem is the efficiency of the estimation. If the gate-level descriptions of IPs are the only design information provided to IP users, IP vendors should provide some information such that IP users can estimate the power consumption of IPs from some power characteristics, which can be calculated from a gate-level simulation. If IP vendors only want to provide the design information for function checking such that IP users can only get the primary input and primary output signals, IP vendors should provide a high-level power model in which IP users can estimate the power consumption of their designs using only the primary input and output information.

In this dissertation, we develop several power estimation and power modeling methods according to the design information that IP vendors will provide to IP users. We divide the design information into three levels. The first level is transistor-level information. The second level is gate-level information. The third level is behavior-level information. For those three levels, we develop three corresponding power estimation methods.

For the complex digital circuits with transistor-level and gate-level design information, we propose two consecutive sampling techniques to improve the losing of performance in those pattern compaction methods with random sampling techniques. IP users can apply our proposed pattern compaction method to reduce the simulation time on a transistor-level simulator with reasonable accuracy.

For the IPs with only gate-level design information, we propose a power model in which a lookup table is built for each IP that maps the zero-delay charging and discharging capacitance during a input pattern transition to an estimative value of real power consumption.

Therefore, IP users can estimate the power consumption according to the zero-delay charging and discharging capacitance from gate-level simulation. The grouping method is applied to reduce the size of lookup tables for circuits and Monte-Carlo simulation strategy is applied to reduce the characterization time.

If IP vendors only provide the behavior-level design information, we propose a power model based on neural network in which only the input and output information of an IP is required to obtain the estimated power. Therefore, IP users can estimation power consumption of IPs with the input and output information obtained from a function-level simulation. Compared to the state-of-the-art 4-D lookup table, our approach requires much less memory but still has competitive accuracy.

Before discussing how the power estimation can be done, we would like to introduce the power consumption of CMOS digital circuits and the incurred problems by the increasing power consumption.

1.1 Power Consumption

The power consumption of a CMOS digital circuit comes from four major types of current, which are dynamic transition current, short-circuit current, leakage current and static current. We will briefly explain the four kinds of current resources in the following descriptions.

The dynamic transition current is resulted from the charge and discharge of the node capacitance. In Figure 1-1, an inverter circuit is used to explain the dynamic transition current. The power consumption of dynamic transition current on output node is often formulated as

Equation (1-1), when logical value of output node is changed from 0 to 1. The same power consumption will be consumed during the logical value of output node is discharged from 1 to 0. In Equation (1-1), *f* is the frequency of signal transition on output node. In general, the dynamic power is the largest part of the total power consumption in a circuit.

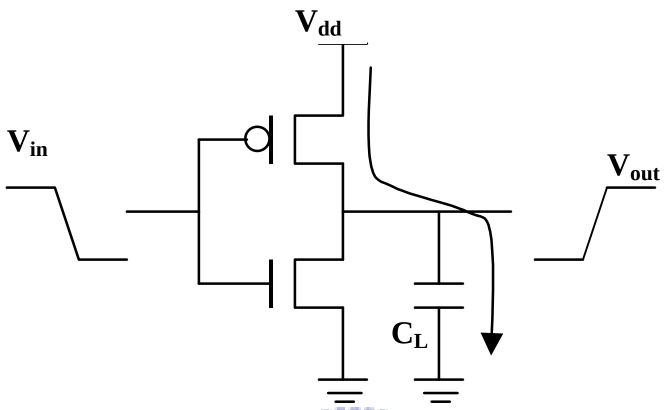


Figure 1-1. Dynamic transition current

$$P_{dynamic} = \frac{1}{2} * C_L * V_{dd}^{2} * f \qquad (1\text{-}1)$$

The short-circuit current is the current from V_{dd} to ground at the period that both PMOS and NMOS transistors turn on together during the input signal transitions. As illustrated in Figure 1-2, both PMOS and NMOS transistors are turned on together during the gray circle period of input signal V_{in}, and the short-circuit current occurs at that moment. The power consumption of short-circuit current can be formulated as Equation (1-2), in which I_{sc} is the mean value of short-circuit current. The short-circuit current could be minimized by matching the rise/fall times of the input and output signals.

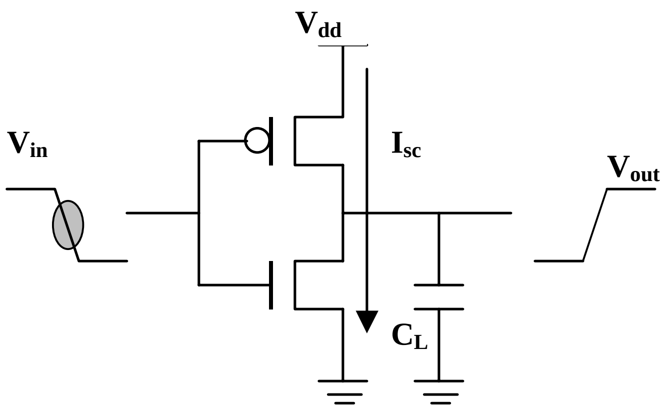


Figure 1-2. Short-circuit current

$$P_{short-circuit} = I_{sc} * V_{dd} \tag{1-2}$$

The leakage current is the current from V_{dd} to ground even when the PMOS path or the NMOS path is “OFF”. It is often consisted of the leakage current in the reverse biased P-N junction diode between n-wall and substrate and the sub-threshold current from sub-threshold conduction, as illustrated in Figure 1-3. The leakage power consumption can be formulated as Equation (1-3), in which $I_{leakage}$ is the mean value of leakage current.

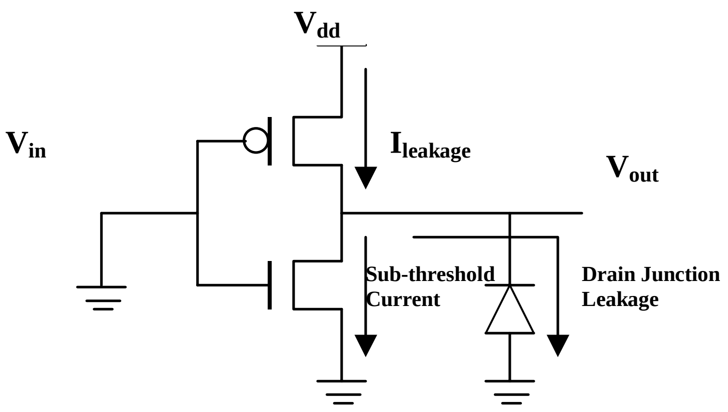


Figure 1-3. Leakage current

$$P_{leakage} = I_{leakage} * V_{dd} \tag{1-3}$$

Another power source is the static current that occurs in some special logic family, such as the pseudo-NMOS logic family. In such kind of circuits, there is always a constant current from the V_{dd} to ground, as illustrated in Figure 1-4. The static power consumption can be formulated as Equation (1-4), in which I_{static} is the mean value of static current.

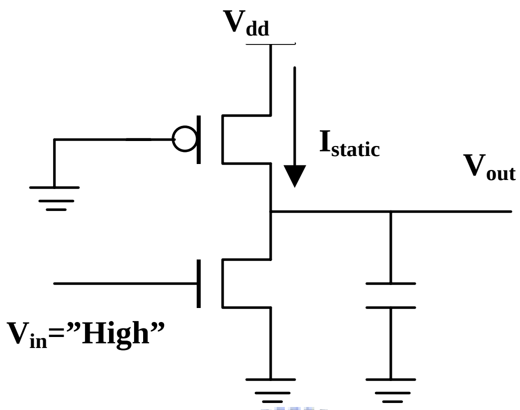


Figure 1-4. Static current

$$P_{static} = I_{static} * V_{dd} \tag{1-4}$$

1.2 Power Estimation

The demand of portable devices and the functions integrated in those devices are both dramatically increasing day-by-day. However, those portable devices are often battery-operated. Increased power consumption means reduced operating time. In addition, increased power consumption also increases the heat generation of chip. In order to remove the extra heat to protect inner circuits, special packaging, cooling and fans are required, which lead to higher cost. Furthermore, larger power consumption may increase the current density of the metal lines in a chip and the temperature of the chip such that several silicon failures, such as electromigration, junction fatigue, and gate dielectric breakdown [1,2], may become more likely to happen. Therefore, the power consumption has also become one of the most important design constraints beside timing and area constraints in present designs.

At the design stage, various low-power design techniques have been proposed at each different level of abstraction, such as system-level, architecture-level, register-transfer level, gate-level and transistor-level. Many surveys on those low-power design techniques could be found in [3~9]. However, before using those low-power design techniques, we have to know the power consumption of current designs. Based on the power estimation results, designers can understand whether any low-power techniques are required and where to apply those low-power techniques to reduce the power consumption of the circuit.

In the literature, many power estimation techniques have been proposed. They can be roughly categorized as simulation-based methods and statistics-based methods, according to the used information while calculating the final power consumption of circuits. In this section, we will introduce those two techniques briefly. Earlier surveys for those power estimation techniques could be found in [9~12].

1.2.1 Simulation-Based Methods

Basically, simulation-based techniques will use simulators to perform power estimation. In those techniques, the most straightforward method of power estimation is to perform circuit simulation to obtain the current information and the resulted power consumption. SPICE is the transistor-level simulator that can provide the most accurate estimation to date. It solves the combination of KVL and KCL equations for the voltages of all nodes and the currents of some certain branches. Although it can provide more accurate results, it will suffer from severe memory and execution time constrains, especially for large VLSI circuits.

In order to alleviate the memory and speed issues for power estimation, PowerMill [13,14], which is a transistor-level power simulator for CMOS and BiCMOS VLSI designs, applies an event-driven simulation algorithm to increase the speed by two or three orders of magnitude over SPICE. Moreover, it uses a nonlinear device model instead of resistor model to model the transistors such that it can maintain the SPICE accuracy while increasing the simulation speed. Estimating power consumption at switch level is another idea to have faster simulation speed, such as the well-known tool IRSIM [15]. Because it only counts the switching power dissipation, which is the transition counts of the circuit nodes weighted by physical node capacitances, the simulation speed is often much faster than that of a transistor-level simulation. However, the estimation results of such tools may not as accurate as those of transistor-level tools.

If the simulation can be done at higher level of abstraction, such as gate-level, the simulation speed could be improved much more. Therefore, many gate-level simulation techniques are proposed to estimate the power consumption of cell-based designs based on the characterization results of library cells. In general, the characterization results include the power model and delay model of basic cells. The power model could be a lookup table or an equation such that the power consumption of a circuit could be calculated by summarizing the power consumption of each basic gate in the circuit. In [16~20], the characterization flow for the cells in a cell library is proposed and the power consumption of circuits is calculated using a gate-level simulator according to the characterization result. The differences between those approaches are the accuracy of the delay model and power model of basic cells, or the accuracy on estimating glitch power. For example, in [18], the authors include a glitch filter to reduce the over estimation on glitch power. In [21, 22], the basic unit of power characterization is extended to complex logic blocks or some functional units on datapath such that their performance is much more improved. Although the gate-level simulation speed is much faster, most of those techniques have to make a tradeoff between the accuracy and the storage complexity.

Besides increasing the simulation speed, another approach is to reduce the number of input patterns for power estimation such that the transistor-level simulation time can also be reduced. Such kind of power estimation methods can be roughly divided into two directions: regeneration and sampling. Regeneration approach [23~30] is to generate a new input sequence that is shorter than the original input sequence but has the similar average power as that of the original one. Some characteristics of the original sequence, such as the pattern transition probabilities [23], the pattern transition probabilities of those input pins with higher power sensitivity values and the average transition probability of rest input pins [24], and the ratio of state transition number of each state [25,26], are preserved during the generation process. In order to reduce the complexity of the generation process, some approaches [27,28] only preserve the input characteristics between clustered inputs. There are also some methods trying to preserve the toggling behavior of the internal nodes [29~30] between the original input sequence and the regenerated input sequence.

Instead of generating new input patterns, the sampling approach chooses some input patterns from the original sequence to estimate the average power. The *Monte Carlo* approach for power estimation is proposed in [31,32]. These methods estimate the average power by sampling some input vectors with certain length *l* from the original sequence and feeding them into the simulator to derive a sample value of the average power. The average power consumption can be estimated with the average of several sample values. From *Central Limit Theorem* [33~36], the sample values can be assumed as a normal distribution when *l* approaches infinity. The probability that the estimated mean value is within a certain error range of the real mean value can also be derived under this assumption. In [31], only combinational circuits are considered and sequential circuits are considered in [32].

Sampling techniques can be further optimized through stratification of the population. Proper stratification of the population can reduce the sample variance such that the number of sampled input vectors can be further reduced. In order to stratify the input vectors, various indicator functions are proposed to provide a rough estimation of the power consumption of each input vector. According to the results of indicator functions, we can put the input vectors with similar power consumption into the same group and sample only a few patterns from each group. Therefore, the key point of this approach is to choose a good indicator function. In [37], the power characteristic is chosen as the zero-delay switching activity multiplied by the loading capacitance of each node. Those power characteristics of input pattern pairs are used to divide input pattern pairs into clusters such that the *Monte Carlo* simulation [36] could randomly sample the same number of pattern pairs from each cluster. This stratified random sampling could improve the convergence speed of the *Monte Carlo* simulation.

In [38], the transition numbers of primary inputs, primary outputs, latches and selective internal nodes are used as the indicator function. Only when the indicator function has enough value changes, this input pattern pair will be used in the transistor-level simulation to plot the power waveform of the input sequence. In [39], the zero delay switching conditions at gate level is used as the indicator function and the modules in the design is also clustered to improved the performance again. In [40], a cycle-based characteristic, the real-delay charging and discharging capacitance, is used as the indicator. They divide all pattern pairs into groups and select only one pattern pair from each group. This approach may not suitable for the designs with very deep logic level because the small delay time error will cause large power variance due to glitch.

1.2.2 Statistics-Based Methods

Statistics-based techniques can be categorized into several different levels. Some of them use the entropy [41~45] for high-level power estimation and modeling because the required information in this approach is independent of the wire loading, transistor sizes, gate types or even circuit structure. Using the analytical results between entropy and real power, we can obtain rough power estimation using only high-level or behavior-level information. Since the simulation at such a high level can be completed very fast, we can have a quick indicator about power consumption in early design stages. However, the power estimation results of this approach will not be very accurate due to the lack of real circuit information. In [46], the entropy is also used to estimate gate counts or area of Boolean functions, and the power consumption of the design could be calculated from switching activity and loading capacitance of each node in the circuit. The basic entropy calculation of logic circuit could refer to [47].

If gate-level design information is available, many researches will calculate the signal transition density [48] or switching activity of each node in the circuit to estimate the power consumption. The estimation results can be more accurate because the power consumption of CMOS digital circuits are dominated by the currents that charge and discharge load capacitances and the short circuit currents. In early researches, gate-level power estimation focuses on combination circuits [49~51]. They assume that the primary inputs are spatially and temporally independent and the propagation delay of each gate is zero. In order to improve the accuracy of power estimation results, the spatial and temporal correlations are included for switching activity estimation in [52~54]. However, the computational complexity to consider spatial correlation may become too high to be adopted.

In [55~59], they start to extend the gate-level power estimation to sequential circuits. In, [56, 57], the BDD (Binary Decision Diagram) is used for the switching activity estimation. In [58, 59], the OBDD (Ordered Binary Decision Diagram) is used for the switching activity estimation. However, the complexity of building BDD will grow exponentially when the number of primary inputs is increased. The work proposed in [60], which uses an ordered BDD method, can alleviate the complexity problem of those BDD-based approaches a little bit while estimating the switching activity. Another technique that uses the Bayesian networks is also proposed to calculate the switching activity of circuit [61].

In order to improve the accuracy of power estimation results, the idea of using transition density is proposed in [62]. It models the lag-1 temporal correlation about the density that an input makes a 0-to-1 or 1-to-0 transition. Based on the input probability and input transition density, a series of researches are conducted to study the sensitivity of the power consumption to the input probabilities and the input transition densities [63,64]. For most combinational circuits, this approach can have more accurate estimation results.

Since there are a lot of signal statistics values of inputs and outputs proposed to be the indicators of the power consumption at gate level, such as the input transition probability, input transition density, output transition density, and spatial correlation coefficient, some approaches use parts or all of the four mentioned indexes to form a look-up table of power consumption, which is often called the table-based power model [65~70]. In [68], the power model also considers the parameter of data width for arithmetic block and the process parameter of the technology. In [69,70], the power model is built based on the power sensitivity values of primary inputs. For each combination of those power indicators, there is a corresponding power value that is estimated in advance using many input sequences with similar indicator values. Therefore, once the values of those indicators for an input sequence are calculated, we can obtain a power estimation result directly from the table. In general, the accuracy of estimation results will increase when the dimensions of the lookup table are increased. However, the computation efforts for building the table will grow very fast when its dimensions are increased. The table size of the table-based power model is another important issue. A technique is proposed to reduce the table size of table-based power model in [71]. Another technique [72] can also reduce the table complexity by using neural networks to recognize the input pattern such that the power consumption of pattern pair could be estimated according to its class.

Besides the table-based power models, some researches use equations to represent their power model [73~75]. In equation-based power models, the major work is to decide the format, variables and factors of the equation. For example, if the equation of power is a quadratic function of 4 variables as in [73], there are 15 factors to be found out during the characterization process. Although the estimation results might be more accurate, the required characterization time is also much longer to find out those parameters.

1.3 Approaches for Different Design Information

In this section, we will discuss the useful power estimation techniques and relative power models at each level of abstraction for an IP. According to the commonly provided design information, the following discussion is made from three different levels, transistor level, gate level, and behavioral level. At each different design level, we also propose corresponding techniques to improve the traditional power estimation methods. Those improvements and their advantages will also be briefly introduced in this section.

1.3.1 Power Estimation at Transistor Level

If transistor-level design information of the IP is provided, the common approach is to perform a transistor-level simulation to estimate the power consumption of a circuit because of its high accuracy. Actually, in this case, almost all simulation-based techniques [13~32] [37~40] could be used for power estimation because it is very possible that gate-level and behavior-level design information is provided, too. Users can choose a technique from them that is most suitable to their applications.

In simulation-based approaches, especially transistor-level simulation, the simulation time is the most critical concern. If we cannot improve the simulation speed too much, we can use another approach to reduce the number of input patterns for power estimation such that the simulation time can also be reduced. In traditional vector compaction techniques, many useless transitions often exist in the compacted input sequence because they have to concatenate all selected input pattern pairs into a new input sequence. Therefore, we propose an improved vector compaction method with grouping and multiple-sequence consecutive sampling approach [76]. An algorithm to reduce the number of sequences is proposed such that the number of useless transitions and the length of the final compacted input sequence can be minimized together, thus greatly improving the efficiency and accuracy of traditional vector compaction approaches.

1.3.2 Power Estimation at Gate Level

If gate-level design information of the IP is provided, users can still use simulation-based approaches to estimate power consumption in gate-level simulation [16~20]. Although the simulation speed is much faster, those techniques often have worse accuracy because they cannot estimate leakage power and short-circuit power and cannot deal with glitch power accurately. Besides the simulation-based approaches, users can also use the statistics-based approaches that do not require transistor-level design information.

In order to provide more accurate power estimation results to users without transistor-level information, IP vendors can provide corresponding gate-level power models in which those power characteristics can be obtained from gate-level simulation such as [65~68]. In those approaches, different lookup tables with 2 dimensions, 3 dimensions, and 4-dimensions are proposed. In general, the accuracy of estimation results will increase when the dimensions of the lookup table are increased. However, the characterization time to fill up the lookup tables will grow very fast when its dimensions are increased. Therefore, we propose a one-dimensional table-based power modeling method for combinational circuits [77] in which the table size is very small and almost independent to the number of primary inputs. Using this approach, designers can build the required power model more efficiently with little accuracy loss.

1.3.3 Power Estimation at Behavior Level

If only behavior-level design information of the IP is provided, users can see only input and output values during the simulation. Therefore, most of those simulation-based techniques cannot be used to estimate power consumption because there are no implementation details in behavior-level design information. In the literature, a number of high-level power estimation techniques [41~45] have been proposed to estimate the power consumption at a high level of abstraction, such as when the circuit is represented only by Boolean equations. This will provide more flexibility to explore design tradeoffs early in the design process and reduce the redesign cost and time to fix power problems. Those high-level techniques can be roughly divided into two categories: top-down and bottom-up. In the top-down techniques [41,42], a combinational circuit was specified only as a Boolean function without any information on the circuit implementation. Therefore, top-down techniques are useful when users are designing a logic block that is not previously designed because they can provide a rough measurement about the trend of power consumption before implemented. However, they may not have very good accuracy due to lack of implementation details.

For SoC designs, bottom-up approaches [62~68] are more useful when one is reusing previously designed circuit blocks such as IPs. Since all internal structural details of the circuit are known, they can build a power model for this block to estimate its power consumption in the target system at function level. Building those power models often requires a power characterization process that uses low-level simulations of modules under their respective input sequences to record the relationship between high-level power characteristics and real power consumption. Because the power consumptions are measured in the low-level simulations with internal circuit information, the power models can provide more accurate estimations than those in the top-down approaches. After the characterization step, no more low-level simulations are required in the estimation step. Users can obtain the power consumption of the circuits by only providing the high-level power characteristics obtained in function-level simulations thus having a very fast estimation time.

Different to the gate-level power models, behavioral-level power models can only use input and output information. In this case, the proposed 1-dimension lookup table power model cannot be used because it requires gate-level structure and node capacitance information. Although those table-based power models in [62~68] can still be used in behavioral level, the table size and the characterization time to fill the lookup table are still the issues that can be further improved. Therefore, we propose a neural-network-based power model [78] using only the statistic information of primary inputs and outputs. The size of the selected neural network is quite small and is almost independent to the number of input/output pins and the size of the circuit. With such a simple structure, we can still have similar accuracy compared to the results of the most complete 4-dimensional table-based power model.

1.4 Organization

The remainder of this dissertation is organized as follows. First, the improved vector compaction method using sampling techniques is presented in Chapter 2. Both the single-sequence and multiple-sequence consecutive sampling techniques will be presented in this chapter. In Chapter 3, the gate-level power model using only 1-dimensional lookup-table will be presented. The proposed tableless power model using feed-forward neural network for behavioral-level simulation will be explained in Chapter 4. Finally, we will give our conclusions and make some discussions about the future works in Chapter 5 to complete this dissertation.

Chapter 2

Improved Vector Compaction Methods

2.1 Vector Compaction Techniques

Traditionally, power estimation is often performed at transistor-level by SPICE-liked simulation. However, it is impractical to simulate a complex design with a large number of test vectors by a transistor-level simulator because it may require too much simulation time. For efficiency consideration, many vector compaction techniques [23~32][37~40] have been proposed. The compacted input sequences are generated according to some characteristics of the original input sequences or the activity of the circuits while triggered by those input vectors. Therefore, the power characteristics can still be maintained because those statistics are carefully kept during the compaction process. Based on the vector compaction techniques, we can estimate the power consumption of a circuit with a much smaller input vector set thus reducing the power estimation time dramatically with little accuracy loss.

The vector compaction techniques can be roughly classified into two categories. The regeneration approaches [23~30] generate a new input sequence that is shorter than the original input sequence but has the similar average power as that of the original one. In [23], the pairwise transition probabilities of inputs are used to approximate the joint transition probabilities of the primary inputs. Those probabilities in the original input sequences will be the target to be kept in the compacted sequence.

In [24], the authors build an incomplete state transition graph, in which the primary inputs with higher power sensitivity are used as the state bits, to generate a smaller sequence after compacting the activity number of each edge in the state transition graph by the Eulerian walk algorithm. The average hamming distance of unselected primary inputs will also be considered when they regenerate the compacted sequence. Based on grouping and sampling techniques, the authors of [27] separate the primary inputs of the circuit into several groups according to their power sensitivity values. The input pattern pairs are also divided into several subsets such that they can generate a smaller input sequence by randomly sampling from each subset according to the size of each subset and the compaction ratio. In [24,27], the power sensitivity values of the inputs are obtained from a simulation. Those power sensitivity values may become inaccurate under different distribution of input signal probability and switching activity.

In [25], the authors build a transition graph of the original input sequence to model the transitions between vectors. With the transition graph, they can obtain the active numbers of all edges and keep their ratios in the compacted input sequence. In [26], the authors analyze the input sequence with the Markov chain model and generate a smaller input vector set that keeps the characteristics on the Markov chain model. In [28], the spatial correlation of input bits is used to cluster the input pins and the compacted sequence can be generated more easily compact input sequence because those bit clusters are treated as independent. In [29], the authors generate a compacted sequence that has the similar transition profile on the internal signals. In [30], the authors separate the input vectors into several vector sets based on the transition counts of internal nodes and generate a smaller sequence according to the fractal compaction algorithm. However, the backward weight propagation in [29] and the fractal algorithm in [30] have high computational overhead such that the speedup is limited.

Another category of the vector compaction techniques is the sampling approaches [31,32][37~40]. The sampling approach chooses some input patterns from the original sequence to estimate the average power. The Monte Carlo simulation method is proposed in [31] and [32] for combinational circuits and sequence circuits. In [37], the stratified random sampling technique is used to improve the convergence speed of the Monte Carlo simulation method. In [38], the gate-level simulation is used to draw the waveform of the indicator function. The transistor-level simulation is used to estimate the pattern pair only when its power variation is large enough. Finally, the power waveform could be used to estimate the power consumption of the original input sequence. In [39], the sampling process is done for module groups with similar power behavior. In this case, the sample size of Monte Carlo simulation could be reduced and the performance could be improved.

According to the cycle-based power information obtained from logic-level simulations, the authors of [40] separate input pattern pairs into to several groups and select the largest energy cycle per group to be simulated by a transistor-level simulator such as PowerMill. According to the power consumption of each sampled cycle and the size of each group, they can calculate the average power consumption of the circuit under the original input sequences. Because they select the largest energy cycle in each group as sampled cycle, those cycles might be randomly distributed in the original input sequence. Therefore, their method is called a random-liked sampling method.

2.2 Useless Transitions

For large circuits, vector compaction techniques could provide a faster solution for power estimation with reasonable accuracy. However, the random-liked sampling method may lose the compaction ratio and speedup as shown in Figure 2-1. Those group numbers in Figure 2-1 present different range of power characteristic values. After compaction, only 4 pattern pairs are randomly selected, one from each group. However, when those 4 pattern pairs are serially concatenated to be the compacted sequence, 7 pattern pairs will be found in the compacted sequence. That means the compaction ratio and speedup are lost because the compacted input sequence includes 3 useless transitions.

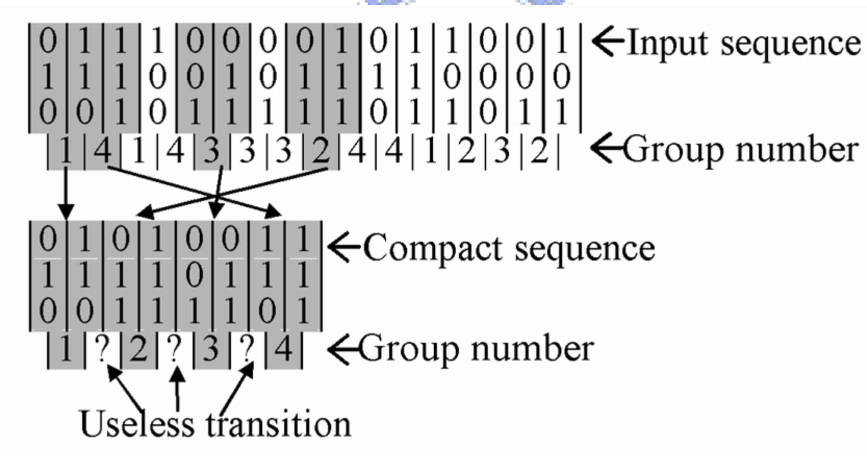


Figure 2-1. Random sampling with useless transition

Therefore, we propose a single-sequence consecutive sampling technique to reduce those useless transitions. Using the single-sequence consecutive sampling technique as shown in Figure 2-2, we can sample a single period of patterns instead of individual pattern pairs to reduce the loss of compaction ratio caused by the useless transitions. Compared to the example shown in Figure 2-1, there is no useless transition in the compacted sequence so that we can keep the compaction ratio as desired and shorten the length of the sequence.

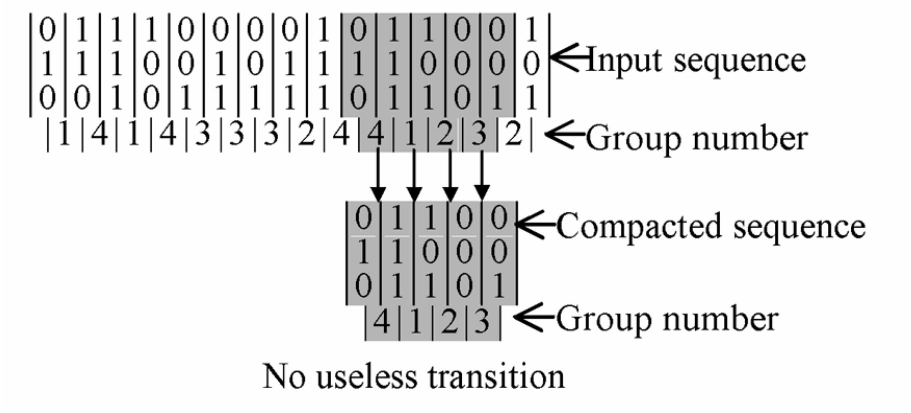


Figure 2-2. Consecutive sampling

However, due to non-uniform distribution of pattern pairs in some groups, it is very possible that we cannot find a perfect consecutive sequence without any undesired transitions as shown in Figure 2-2. Using single-sequence consecutive sampling technique, we will over-sample some groups in such cases to find an intact single sequence that have enough samples for all groups. Therefore, the compaction ratio of the sequence length may not be improved too much. In those cases, if we can relax the limitation a little bit such that multiple consecutive sequences are allowed, we may generate a shorter sequence that still has the desired distribution. For example, if the desired distribution is G1:G2:G3:G4 = 3:1:2:2 for the input vectors shown in Figure 2-3, the compacted sequence found in single-sequence approach will include at least 11 transitions as shown in Figure 2-3(a). However, as shown in Figure 2-3(b), we can find two subsequences that also satisfy the requirements but the number of transitions is only 9 after concatenated. It implies that we can find better solutions for vector compaction problem if we minimize the number of sequences instead of setting the number to be one. Of course, the number of sequences could be one as handled in the original single-sequence approach, but it is just a special case in the multi-sequence approach. Therefore, in this work, we focus on discussing this new extension and perform some experiments to show the improvements of this new approach.

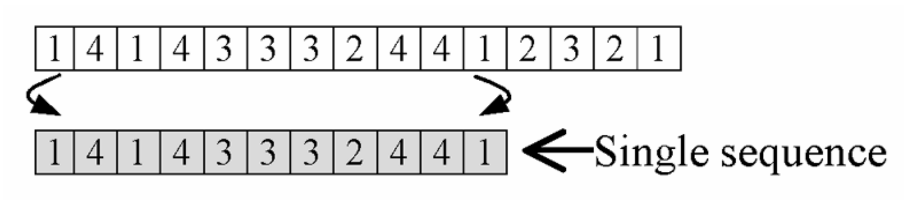


Figure 2-3(a). The result of single-sequence approach

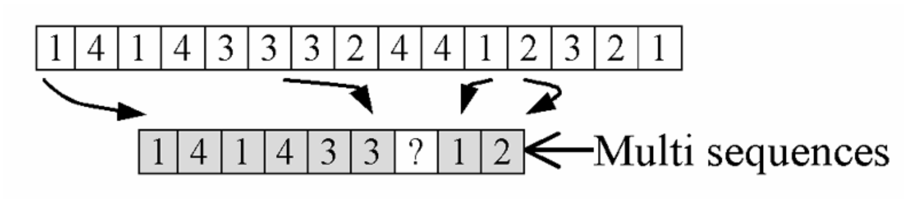


Figure 2-3(b). The result of multi-sequence approach

In this work, our focus is to reduce useless transitions in random-liked sampling method for vector compaction. Although those vector compaction methods including previous approaches and the proposed approach only focus on combinational circuits, they can still be applied to sequential circuits with full scan. The only difference is that we have to record the internal states of all flip-flops (FFs) when we estimate logic-level power characteristics of each input transition. This FF information will then be used in the transistor-level simulation for the compacted input sequence to set the internal states at the beginning of each composing subsequence. Therefore, if the compacted input sequence is composed of only one subsequence, we only have to set the initial condition once, which requires very little overhead.

2.3 Selection of Power Characteristics

The power consumption of a CMOS digital circuit is often formulated as Equation (2-1).

The static power (P_{static}) is often much smaller than the dynamic power ($P_{dynamic}$). The $P_{dynamic}$ is the summation of the functional transition power (P_{func_trans}), the glitch power (P_{glitch}) and the short-circuit power ($P_{short\text{-}circuit}$), which is represented as Equation (2-2). The $P_{short\text{-}circuit}$ is consumed when short-circuit current flows from V_{DD} to ground at the period that both PMOS and NMOS transistors turn on together during the signal transitions and is often smaller than the summation of the P_{func_trans} and the P_{glitch}. The proportion between P_{func_trans} and P_{glitch} depends on the circuit behavior and the design skill. Given a circuit with n nodes in its netlist, we could express the power consumptions of P_{func_trans} and P_{glitch} as Equation (2-3) and (2-4), where i denotes the index of each internal node, C_i is its load capacitance of node i, V_{dd} is supply voltage of the circuit, f_{i_func} is the frequency of functional transition at node i and f_{i_glitch} is the frequency of glitch at node i. Note that a node in the netlist is defined as the input or output of a logic gate in the circuit. Generally speaking, a functional transition only considers the signal transition from 0 to 1 or 1 to 0. On the contrary, a glitch is the signal transition from 0 to 1 to 0 or 1 to 0 to 1 such that it is not multiplied by a factor 1/2. τ_i is the factor of the width of glitch to the glitch power and should be between 1 and 0.

$$P = P_{static} + P_{dynamic} \qquad (2\text{-}1)$$

$$P_{dynamic} = P_{func\_trans} + P_{glitch} + P_{short-circuit} \qquad (2\text{-}2)$$

$$P_{func\_trans} = \frac{1}{2} \cdot V_{dd}^2 \cdot \sum_{i=1}^{n} C_i \cdot f_{i\_func} \qquad (2\text{-}3)$$

$$P_{glitch} = V_{dd}^2 \sum_{i=1}^{n} C_i \cdot f_{i\_glitch} \cdot \tau_i \qquad (2\text{-}4)$$

$$P = P_{func\_trans} + P_{glitch} + P_{short-circuit} + P_{static} \qquad (2\text{-}5)$$

In Equation (2-3), the term $\sum_{i=1}^{n} C_i \cdot f_{i\_func}$ is often defined as the charging and discharging capacitance (CDC) during an input transition, where f_{i_func} =1 if node i has signal transition and f_{i_func} =0 if node i has no signal transition. The C_i of node i is the summation of output capacitance for driving gate and the input capacitances of driven gates at node i. For commercial cell libraries, the vendors will provide the output loading capacitance and input loading capacitances of cells. If such loading information is not provided, users can easily characterize the loading capacitances by themselves using the characterization process proposed in [19]. Therefore, to calculate the CDC values of an input pattern pair only have to sum the loading capacitances of those nodes whose logic values are changed during the input transitions. Only a logic-level simulator is required to obtain the node transition information for calculating CDC values.

In the simulation-based vector compaction approaches that consider the circuit structures or behaviors, they often classify the input pattern pairs according to some power characteristics of each pattern pair. In the literatures, many power characteristics have been proposed [23,24][27] [30]. For example, Hamming distance (HD) of pattern pairs is adopted in [23][27] which use the number of transition bits of the primary inputs to approximate the average power consumption. Switching count (SC) is used in [30] to approximate the power consumption of a pattern pair using the summation of $\sum_{i=1}^{n} f_{i\_func}$. Charging and discharging capacitance (CDC) is adopted in [40] to approximate the power consumption of a pattern pair. Power sensitivity is used in [24,27] as an estimation on the influence of an input to the overall power consumption.

In the vector compaction approaches, the adopted power characteristics have large impacts on the accuracy of the estimated power and the extra computation overhead for the compacted input sequences. In order to determine which power characteristic is most suitable for different circuits, we define the average normalized error of a power characteristic as below to make a fair comparison between them.

An ***average normalized error*** (AVGNE) is the average error between the normalized power characteristics to the normalized real power of all combinations of input vectors. The normalized power characteristic is the power characteristic value divided by the average power characteristic value. The normalized real power is the power consumption divided by the average power.

For a combinational circuit with *n* inputs, we can formulate the AVGNE of any power characteristic *PC* as Equation (2-6). In Equation (2-6), $P_{j,k}$ is the power consumption of the transition from pattern *j* to pattern *k*, and $PC_{j,k}$ is the power characteristic value of that transition. P_{avg} is the average power consumption of all input pattern pairs and PC_{avg} is the average power characteristic value of all input pattern pairs.

$$AVGNE_{PC} = \frac{1}{2^{n+n}} \sum_{j=0}^{2^n-1} \sum_{k=0}^{2^n-1} \left| \left(\frac{PC_{j,k}}{PC_{avg}} - \frac{P_{j,k}}{P_{avg}} \right) \right| \qquad (2\text{-}6)$$

$$PC_{avg} = \frac{1}{2^{n+n}} \sum_{j=0}^{2^n-1} \sum_{k=0}^{2^n-1} PC_{j,k} \qquad (2\text{-}7)$$

$$P_{avg} = \frac{1}{2^{n+n}} \sum_{j=0}^{2^n-1} \sum_{k=0}^{2^n-1} P_{j,k} \qquad (2\text{-}8)$$

The AVGNE of a power characteristic can make a fair comparison between the power characteristic value and the real power consumption. The power characteristic with smaller AVGNE is considered as much closer to the real power. Therefore, we make some experiments to evaluate the AVGNE of some popular power characteristics.

In our previous work [79], we have compared the AVGNE of CDC and HD. In this work, we compare the AVGNE of three popular power characteristics, HD, zero-delay CDC and zero-delay SC. In our experiments, we evaluate the AVGNE by three input sequences with the same average input signal probability (P=0.5) but different average transition density (D) on several ISCAS'85 benchmark circuits. For the input sequence with high average transition density, D is set to 0.4. For the input sequence with middle average transition density, D is set to 0.25. For the input sequence with low average transition density, D is set to 0.1. For each test sequence, 500 pattern pairs are randomly generated according to the desired signal probability and transition density. Those patterns are then used in PowerMill to simulate their power consumptions. About the corresponding power characteristic values, the CDC and SC values are calculated by the Verilog-XL simulator, and the HD values is obtained by a simple self-developed C program. The comparison result is shown in Table 2-1. The AVGNEs of ISCAS'85 benchmark circuits are estimated by three test sequences and the overall average AVGNEs of CDC, SC and HD are 0.1278, 0.1399 and 0.2568 respectively. Therefore, we also choose CDC to be the power characteristic in this work.

Table 2-1. Average normalized errors for three power characteristics

	High D (P=0.5, D=0.4)			Mid D (P=0.5, D=0.25)			Low D (P=0.5, D=0.1)		
	CDC	**SC**	**HD**	**CDC**	**SC**	**HD**	**CDC**	**SC**	**HD**
C432	0.1592	0.1933	0.3093	0.2306	0.2501	0.4793	0.2091	0.2424	0.7749
C499	0.1044	0.1064	0.1334	0.1321	0.1308	0.1641	0.1491	0.1405	0.2662
C880	0.1044	0.1183	0.1729	0.0841	0.0991	0.2562	0.1016	0.1178	0.3153
C1355	0.1055	0.1099	0.1402	0.1026	0.1056	0.1579	0.1246	0.1268	0.2112
C1908	0.1441	0.1515	0.1809	0.1421	0.1597	0.2303	0.1490	0.1551	0.2851
C2670	0.1018	0.1213	0.1768	0.1067	0.1341	0.2158	0.1121	0.1363	0.3138
C3540	0.1445	0.1598	0.1750	0.1335	0.1477	0.2396	0.1417	0.1496	0.4756
C5315	0.0882	0.1003	0.1563	0.0836	0.0963	0.2400	0.1091	0.1190	0.3313
C6288	0.0982	0.1125	0.1262	0.1532	0.1684	0.1733	0.2096	0.2260	0.2391
C7552	0.0914	0.0933	0.1667	0.1014	0.1067	0.2699	0.1158	0.1177	0.3286
Average	0.1142	0.1267	0.1738	0.1270	0.1399	0.2426	0.1422	0.1531	0.3541

2.4 Grouping of Pattern Pairs

If we have a power characteristic that is almost proportional to the real power consumption for all pattern pairs, we can easily generate a compacted input sequence for estimating the average power consumption of a circuit by a simple random selection. For example, if the original input sequence is L and the compacted sequence is C, the power consumption of the original input sequence P_L can be calculated from $P_L=P_C*(PC_L/PC_C)$, where PC_L is the total power characteristic value of the original input sequence, PC_C is the total power characteristic value of the compacted sequence, and P_C is the power consumption of the compacted sequence. However, most of the power characteristics including CDC can only model the functional transition power. The glitch power is often not proportional to the functional transition power for all pattern pairs such that the power characteristic values may not always be proportional to the real power consumption.

Therefore, another solution is required instead of random selection to minimize the estimation errors. One popular approach is to separate the input pattern pairs into several groups according to their power characteristics and then sample pattern pairs from each group. This grouping method, which is also applied in this work, is widely used just like [40] with very low computation complexity. The variation caused by glitch power can be effectively reduced because the average value of each group is used to represent the power consumption of all pattern pairs that belong to this group such that the variation can be compensated.

In order to demonstrate the grouping effects, we perform a simple experiment on C1355 in ISCAS'85 benchmark circuits with 5,000 random input pattern pairs. The variance limitation of each group is set as ±2.5%. The experimental results are illustrated in Figure 2-4 to show the estimation error between normalized CDC values and normalized real power values of all pattern pairs. The estimation error of each pattern pair is defined as Equation (2-9), where NCDC$_i$ is the normalized CDC value of pattern pair i, and NP$_i$ is the normalized power consumption of pattern pair i. Without grouping, all pattern pairs are treated as a single group with group number 0 in Figure 2-4. We can see that there is a large error distributed from 20% to -60%. After divided those pattern pairs into 24 groups with group number 1 to 24 in Figure 2-4, we can see that the error distribution range of each group is significantly reduced if the average value is used to represent the real power value of each pattern pair in the same group.

$$Error_i = \frac{NCDC_i - NP_i}{NP_i} \times 100 \qquad (2\text{-}9)$$

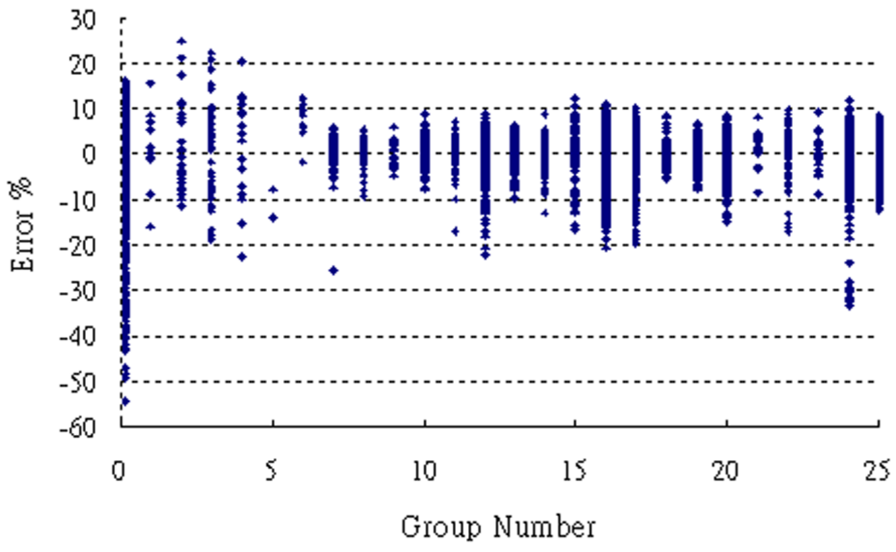
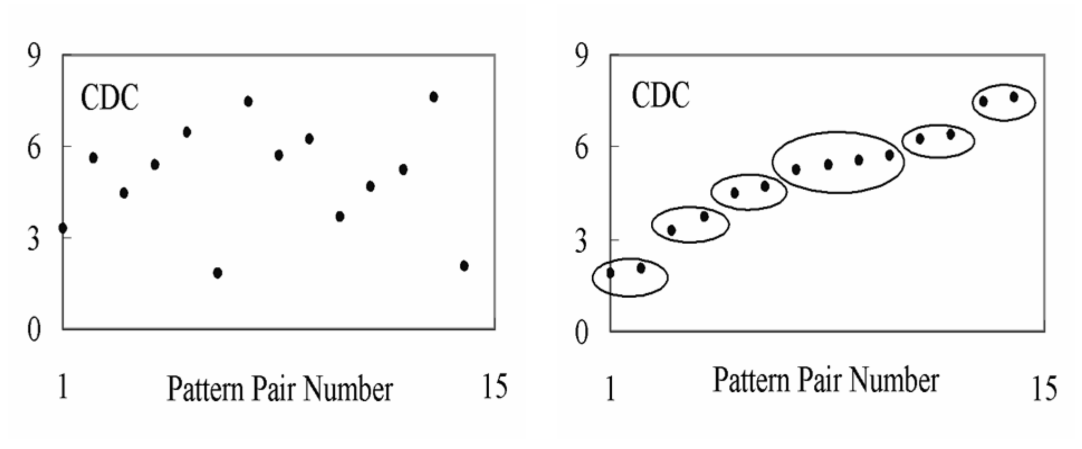


Figure 2-4. The effects of grouping

Figure 2-5 shows an example of the grouping process. Figure 2-5(a) is the distribution of the CDC values of 15 pattern pairs. After sorting and grouping, those pattern pairs with similar CDC values will be put together into a group as shown in Figure 2-5(b). In this example, there are six groups for the input pattern pairs. The group size and the number of groups is determined by a user-defined *variance limitation*, which is the range of CDC values in a group from the average CDC value to its maximum or minimum value.



(a). CDC distribution (b). Sorting and grouping

Figure 2-5. An example of grouping process

Our experience shows that the best variance limitation falls between ±2.5% to ±5%. If the variance limitation is smaller than ±2.5%, the number of groups is increased and the group sizes are decreased. In this case, it is hard to obtain a high compaction ratio because many groups are too small to provide enough samples. If the variance limitation is larger than ±5%, the grouping process will cause larger errors because the glitch power may be quite different between those pattern pairs in a group. Therefore, there is a trade-off between the compaction ratio, the estimation error, and the variance limitation, which can only be decided according to the characteristics of the circuits.

In order to demonstrate these effects, we use 50,000 pseudo random vectors to test C2670 in ISCAS’85 benchmark circuits with different variance limitation and compression ratio by using the two different vector compaction approaches that will be introduced in Section 2.5. The experimental results are shown in Table 2-2. In Table 2-2, DCR is the abbreviation of desired compaction ratio. From the results, we can see that the estimation errors will increase in both approaches when the compression ratio is increased. If we set the variance limitation to a smaller value (1%), we can see that the estimation errors are getting worst especially for high compression ratio because many groups are too small to provide enough samples. When the variance limitation is set to a larger value (5%), the estimation error will also increase because the variation in a group is increased.

Table 2-2. The relationship between compaction ratio, variance limitation and estimation error

C2670		**Variance Limitation**		
		±1%	±2.5%	±5%
Sampling Techniques	DCR	Error (%)	Error (%)	Error (%)
Single-Sequence	100	1.84	0.87	2.19
	250	4.04	3.83	3.32
	500	8.64	7.48	7.88
Multi-Sequence	100	2.16	2.82	2.23
	250	4.25	3.73	4.77
	500	9.56	7.32	7.69

The pseudo code of the grouping process is shown in Figure 2-6. The subroutine *QuickSort()* sorts all pattern pairs according to their CDC values. When we try to merge a pattern pair into the current group, the subroutine *cal_group_avg_cdc()* calculates the average CDC value of this group. And the subroutine *violate_var_limit()* will test whether the variance limitation is violated due to the added pattern pair. If this group will not violate the variance limitation with the added pattern pair, this pattern pair will be merged into the current group. Otherwise, a new group will be built in which this pattern pair is the first member. The grouping information will be recorded in a data structure *group[]* and the total number of groups will be returned after the whole process is finished.

```
Grouping(pattern_pair[],N, var_limit,group[])
{
    QuickSort(pattern_pair[],N);
    group_num = 1;
    g_h=g_t=N;
    while(g_h > 0)
    {
       group_avg_cdc=cal_group_avg_cdc(pattern_pair[],g_h,g_t);
       if (violate_var_limit(group_avg_cdc,pattern_pair[g_h].cdc,
          pattern_pair[g_t].cdc,var_limit))
       {
          group[group_num].size=g_t-g_h;
          group_num++;
          g_t=g_h;
          group_avg = pattern_pair[g_h].cdc;
          pattern_pair[g_h].group_num=group_num;
       }
       pattern_pair[g_h].group_num=group_num;
       g_h--;
    }
    return(group_num);
}
```

Figure 2-6. The pseudo code of the grouping algorithm

2.5 Consecutive Sampling techniques

2.5.1 Single-Sequence Approach

After grouping, we can sample a number of pattern pairs from each group according to the size of the group divided by a user-defined compaction ratio. It is often called the proportional sampling strategy [27]. Instead, we can sample a single pattern pair from each group, which is called single sampling strategy [40]. The single sampling strategy can only be used if the power characteristic is a very precise approximation of the real power. However, it can achieve a very high compaction ratio. The proportional sampling strategy can be used without a very precise power characteristic but the compaction ratio may not be as high as the single sampling strategy.

In traditional sampling methods, people sample some independent pattern pairs from each group and concatenate them into a continuous sequence for simulation. Therefore, the sequence will include about half useless transitions as shown in Figure 2-1. Using a state transition graph and selecting the Euler trails on it with enough samples could be an approach to reduce useless transitions. However, it can only be used when most of states and transitions have passed many times. In typical cases, not all parts of the state transition graphs will be visited many times such that we are hard to obtain enough samples. Therefore, in order to reduce the useless transitions in most cases, we propose a single-sequence algorithm to sample a sequence of consecutive pattern pairs from the original input sequence with the desired distribution and compaction ratio. The single sequence algorithm can be formulated as below.

Problem formulation: Given a sequence S of length N with entries in a set $G=\{g_1,g_2,\ldots,g_m\}$, where $g_i \in Z^+$ (1,2,…) for $1 \le i \le m$, and a set $T=\{t_1,t_2,\ldots,t_m\}$ with $t_i \in Z_0^+$ (0,1,2,…) for $1 \le i \le m$ and $t_i \le s(g_i)$ for $1 \le i \le m,$ where $s(g_i)$ represents how many times that g_i appears in S, find the shortest subsequence S' in S such that all $g_i \in G$ can be found in S' **at least** t_i times.

Solution: According to the problem formulation, we can see that the shortest subsequence that satisfies the requirements will also satisfy the following two conditions. The first condition is that the corresponding group of the start point in the shortest subsequence must exactly appear as the requirement in T. If the corresponding group of the start point is larger than the requirement in *T*, we can drop it to obtain a shorter sequence and the new subsequence will still satisfy the requirement in *T*. The second condition is that the corresponding group of the end point in the shortest subsequence must also exactly appear as the requirement in *T*. The reason of this condition is the same as the first condition. Based on these two conditions, we propose an algorithm that can find the shortest subsequence from the original input sequence to satisfy the requirement in *T*.

Step 1: Assume the subsequence starts from index *tail* and ends at index *head*. Find a subsequence, whose *tail* is located at the start point of *S*, to satisfy the requirements in *T* and the second condition.

Step 2: Trace this subsequence by moving *tail* forward until the subsequence satisfies the first condition. Then, this subsequence is one candidate of the shortest subsequence. We will keep tracking the shortest one of all candidates.

Step 3: Move *tail* one step forward. This subsequence is now violating the requirements in *T*.

Step 4: Find the next subsequence that satisfies the requirements and the second condition by moving *head* forward on *S*. If *head* is equal to N and the subsequence still does not satisfy the requirements, this procedure will be stopped. Else, go back to Step 2.

The global shortest subsequence will be the shortest one in those shortest subsequences found in Step 2. In our process, the shortest one will be found when the process stops because the process keeps tracking the shortest candidate. It is hard to give a formal proof for our algorithm, but we can explain it by simple descriptions as follows. If there is a shorter subsequence than the one we found, it means that some pattern pairs can be dropped from the subsequence we found. However, the numbers of pattern pairs of the groups for first pattern pair and last pattern pair in our subsequence are just satisfying the requirement in *T*. It implies that no pattern pairs could be dropped from our subsequences. Therefore, the subsequence is the shortest subsequence that satisfies requirements.

The pseudo code of this algorithm is shown in Figure 2-7. In Figure 2-7, the subroutine *Shortest_subsequence()* will find the shortest subsequence that satisfies the requirements in *T*. The *sub_seq_statisfied()* will test whether the subsequence from the index *tail* to the index *head* in *S* is satisfied the requirements in *T* and the first condition. The *trace_forward()* will increase the index *tail* one by one until the subsequence satisfies the second condition. The time complexity of this algorithm is *O(n)* because it only walks through the sequence *S* twice by *head* and *tail*.

```
Shortest_subsequence(S[],G[],T[],N,f_head,f_tail)
{
  head=tail=f_head=f_tail=1;
  length=N;
  while(head <= N)
  {
    if(sub_seq_satisfied(S[],G[],T[],head,tail))
    {
     trace_forward(S[],G[],T[],head,tail)
     if (length > head-tail+1)
     {
       length = head – tail + 1;
       f_head = head; f_tail = tail;
     }
     tail++;
    }
    head++;
  }
}
```

Figure 2-7. The pseudo code of the single-sequence algorithm

Figure 2-8 is a simple example for the shortest subsequence searching process that samples one pattern pair from each sampled group (*G*={1,2,3,4}, *T*={1,1,1,1}). The first step is *sub_seq_statisfied()*, which finds the first subsequence that stratifies the requirements in *T* and the second condition. Two indexes *tail* and *head* define the subsequence. The second step is *trace_forward()*, which moves the index *tail* forward until the subsequence satisfies the first condition. This subsequence is one candidate of the shortest subsequence. Therefore, we record it by index *f_tail* and index *f_head* as the temporal shortest subsequence. The third step moves the index *tail* one step forward and apply the *sub_seq_statisfied()* to find the next subsequence that satisfies the requirement in *T* and the second condition. The fourth step is *trace_forward()* again which moves the index *tail* forward until the subsequence satisfies the first condition. This subsequence is also one candidate of the shortest subsequence. Compared with the recorded temporal shortest subsequence, the existing shortest subsequence is shorter than the new one. Therefore, the indexes *f_tail* and *f_head* are not changed. The fifth step is the same as step 3 that moves the index *tail* one step forward and apply the *sub_seq_statisfied()* to find the next suitable subsequence. The sixth step is *trace_forward()* again that moves the index *tail* forward until the subsequence satisfies the first condition. This subsequence is also one candidate of the shortest subsequence. Compared with the recorded temporal shortest subsequence, this new sequence is shorter than the existing shortest subsequence. Therefore, the indexes *f_tail* and *f_head* are changed to define the new temporal shortest subsequence from *tail* to *head*. After the sixth step, we cannot find any new subsequence. The recorded temporal shortest subsequence is the final shortest subsequence.

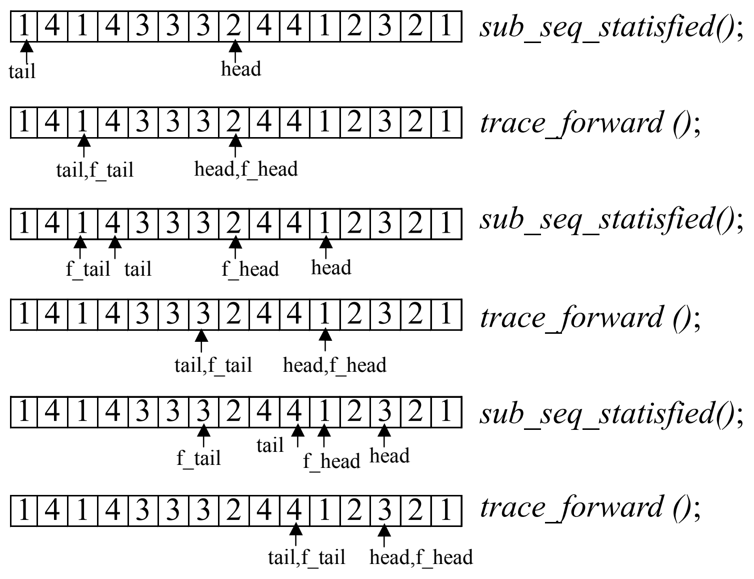


Figure 2-8. An example of the shortest subsequence searching

Ideally, we can find a compacted sequence without any useless transitions as shown in Figure 2-2. In real cases, however, the compacted sequence may still have some undesired or over-sampled transitions.

2.5.2 Multi-Sequence Approach

As shown in Figure 2-3, if we relax the limitation a little bit such that multiple consecutive sequences are allowed, we can find better solutions for vector compaction problem if we minimize the number of sequences instead of setting the number to be one. In this section, we will discuss this new extension and propose an algorithm to solve this more general problem.

Problem formulation: Given a sequence *S* of length *N* with entries in a set $G=\{g_1,g_2,\ldots,g_m\}$, where $g_i \in Z^+$ (1,2,…) for $1 \le i \le m$, and a set $T=\{t_1,t_2,\ldots,t_m\}$, $t_i \in Z_0^+$ (0,1,2,…) for $1 \le i \le m$ and $t_i \le s(g_i)$ for $1 \le i \le m$, where $s(g_i)$ represents how many times that g_i appears in *S*, find the_minimum number of disjoint subsequences in *S* such that all $g_i \in G$ can be found in those subsequences **exactly** t_i times, for $1 \leq i \leq m$.

This problem is very similar to the well-known EXACT COVER BY 3-SETS (X3C) problem [80]. The X3C problem is described as follows.

INSTANCE: Set X with $|X| = 3q$ and a collection C of 3-element subsets of X.

QUESTION: Does C contain an exact cover for X, i.e., a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'?

In fact, the X3C problem can be transformed into a special case of our problem in polynomial time. Instead the detailed deriving process, we briefly show the transforming process in Figure 2-9. If the minimum number of disjoint subsequences in our problem is equal to q in X3C problem, we can find that the subcollection C' will satisfy the requirement.

In X3C problem:

*X={1,2,3,4,5,6}, |X|=3*2, q=2 and*

C={{6,2,3}, {5,3,6}, {1,2,4}, {1,5,4}}

Transfer to multi sequences problem:

G={1,2,3,4,5,6,0}, s(1)=2, s(2)=2,s(3)=2, s(4)=2,

s(5)=2, s(6)=2, s(0)=3 and T={1,1,1,1,1,1,0}

$S \rightarrow$	6	2	3	0	5	3	6	0	1	2	4	0	1	5	4

Figure 2-9. Transforming a X3C problem into a multi-sequence problem

Because the X3C problem is a NP-complete problem, our multi-sequence problem is a NP-complete problem, too. Therefore, we propose a heuristic algorithm to solve it. First, we will find the longest subsequence in which the pattern pairs of each group do not appear more than the requirements in T. Of course, this longest subsequence will not include any useless transitions. After that, we modify the numbers in T by subtracting the required sample number in T with the number of pattern pairs that appear in the first subsequence for each group. Then we can find the next longest subsequence and modify the numbers in *T* again. This process will be iteratively executed until all numbers in *T* are equal to zero. In order to ensure that all subsequences found in this process are disjoint, the subsequence found in each iteration will be marked in *S*. Finally, the sequence that concatenates those subsequences is the solution of our algorithm. As a summary, we describe our algorithm step-by-step as follows and demonstrate the pseudo code of our algorithm in Figure 2-10.

Step 1: Find the longest subsequence in which each g_i in *G* appears $t_i^{'}$ times, where $t_i^{'} \leq t_i$ for all t_i in *T*.

Step 2: Mark the longest subsequence in *S* and set $t_i = t_i - t_i^{'}$ for all t_i in *T*.

Step 3: If all $t_i=0$, $1 \leq i \leq m$, STOP. Then concatenate all subsequences found in Step 1 to be the solution. Else go to Step 1.

In Figure 2-10, the subroutine *sub_seq()* is the function to find the longest subsequence in which the pattern pairs of each group do not appear more than the requirements in *T*. The subroutine *concatenate()* concatenates the sub-sequence found by the *sub_seq()* function. The subroutine *modify()* modifies the numbers in *T* according to the longest subsequence found by the subroutine *sub_seq()*. The subroutine *all_zero()* tests whether all entries in *T* equal to zero. The number of useless transitions in the final sequence will be the number of subsequences minus one.

```
Multi_sequence(S[],G[],T[],N)
{
    S'=NULL;
    S_tmp=NULL;
    flag=0;
    while(flag==0)
    {
        S_tmp=sub_seq(S[],G[],T[],N);
        S'=concatenate(S',S_tmp);
        modify(T[],S_tmp);
        S_tmp=NULL;
        If(all_zero(T[])) flag=1;
    }
    return(S');
}
```

Figure 2-10. The pseudo code of the multi-sequence algorithm

The time complexity of this algorithm is $O(n^2)$ that is dominated by the number of *sub_seq()* subroutine being executed. In the worst case, the number of *sub_seq()* subroutine being executed is *n* divided by the desired compaction ratio. Therefore, the time complexity is $O(n^2)$ for our proposed multi-sequence algorithm in the worst case because the operations in this algorithm are similar to those in the single-sequence algorithm, whose time complexity is $O(n)$.

Figure 2-11 is an example of the detailed searching process in the multi-sequence algorithm for *G*={1,2,3,4} and *T*={3,1,2,2}. The input sequence *S* in this example is the same as in Figure 2-3 for explaining the improvement of multi-sequence algorithm. The first step is *sub_seq()* that will find the longest subsequence in which the pattern pairs of each group do not appear more than the requirements in *T*. The second step is *concatenate()*, which concatenates the result of the first step into *S'*. The third step is *modify()*, which modifies the numbers in *T* according to the subsequence found in the first step and marks those transitions that appear in the subsequence on *S*. After modified, *T* becomes to *T*={1,1,0,0} and passes the examination of the *all_zero()* subroutine, which tests whether all entries in *T* equal to zero. The fourth step is *sub_seq()* again that finds the longest subsequence in which the pattern pairs of each group do not appear more than the requirements in *T*. The fifth step is *concatenate()*, which concatenates the result of the fourth step into *S'*. The sixth step is *modify()* again, which modifies *T* according to the subsequence found in the fourth step and makes proper marks on *S*. After modified, *T* becomes to *T*={0,0,0,0}. Because all entries in *T* equal to zero, the *all_zero()* subroutine will return 1 and the process is stopped. At this moment, the sequence *S'* is the final result.

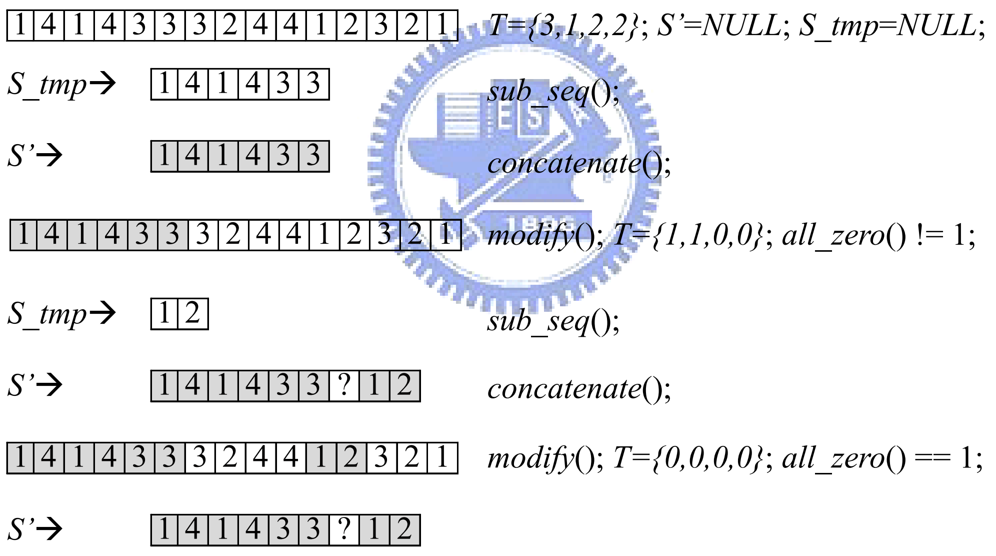


Figure 2-11. An example of the multi-sequence algorithm

In this example, the compacted sequence *S'* is consisted of two subsequences thus still having one useless transition. However, compared with the single-sequence approach, we still save 2 transitions with the multi-sequence approach. Using the single-sequence algorithm, we will find the shortest sequence with 11 transitions as shown in Figure 2-3(a). Using the multi-sequence algorithm, the final sequence has only 9 transitions including one useless transition as shown in Figure 2-3(b).

2.6 Average Power Calculation

For efficiency consideration, it is not necessary to sample pattern pairs from those groups with too few pattern pairs because those groups have only small contribution on the overall power consumption. If we put too much effort to sample those pattern pairs, the desired compaction ratio may be decreased. Instead, we can directly calculate their contribution to the overall power consumption for those non-sampled groups to provide a trade-off strategy between efficiency and accuracy. Therefore, we will set the sampling numbers to zero for those groups whose sizes are smaller than the desired compaction ratio. The detailed equations for deriving the average power consumption of a circuit are shown in Equations (2-10), (2-11) and (2-12).

$$P_{avg} = (P_{sam} + P_{non})/\mathrm{N} \qquad (2\text{-}10)$$

$$P_{sam} = \sum_{i=1}^{m}[(\sum_{j=1}^{h_i}\frac{CDC_j}{CDC_{avg\_i}} \times P_j) \times \frac{h_{total\_i}}{h_i}] \qquad (2\text{-}11)$$

$$P_{non} = P_{sam} \times \frac{CDC_{total\_non}}{CDC_{total\_sam}} \qquad (2\text{-}12)$$

In Equation (2-10), N is the number of pattern pairs in the original input sequence, P_{avg} is the average power consumption, P_{sam} is the total power consumption of sampled groups, and P_{non} is the total power consumption of non-sampled groups. In Equation (2-11), m is the number of sampled groups, h_i is the number of sampled pattern pairs in group i, CDC_{avg_i} is the average CDC value of group i, CDC_j is the CDC value of pattern pair j, P_j is the power consumption of pattern pair j, and h_{total_i} is the total number of pattern pairs in group i. In Equation (2-12), CDC_{total_non} is the total CDC value of non-sampled groups, and CDC_{total_sam} is the total CDC value of sampled groups.

2.7 Experimental Results

In this section, we will demonstrate the experimental results of our approaches with ISCAS'85 benchmark circuits. The estimation environment is a SUN UltraSPARC IIi workstation with 512MB memory. The original input sequence contains 50,000 pseudo random vectors for each circuit. The variance limitation in the grouping process is set to ±2.5%. The number of sampled groups is decided by a user-specified parameter "desired compaction ratio" (DCR). In our experiments, the desired compaction ratio is set to a high number (250) to demonstrate that those sampling techniques could achieve high compaction ratio and high speedup without losing too much accuracy. However, with the same DCR, the achieved compaction ratio may not as high as expected because the useless transitions may exist in the compacted input sequences. Therefore, the effective compaction ratio is also calculated to show the effects of reducing useless transitions in the proposed vector compaction technique.

The experimental results are shown in Table 2-3. The first row is the names of circuits. The second and third rows are the estimation results and the run time elapsed by PowerMill simulator with the original input sequence. The following six rows show the estimation results of the random sampling technique and the last twelve rows are the estimation results using the consecutive sampling techniques. The row L/U represents the length of the compacted sequence (L) and the useless transitions (U) in the compacted sequence. ECR is the abbreviation of effective compaction ratio, which is the number of pattern pairs in the original input sequence divided by the number of pattern pairs (including useful and useless transitions) in the compacted sequence. The speedup is the elapsed time of PowerMill with the original input sequence divided by the elapsed time of the power estimation with vector compaction technique that includes Verilog-XL simulation, grouping, sampling, PowerMill simulation and average power calculation.

Table 2-3. A comparison of random, single-sequence and multi-sequence techniques

		Circuit	**C432**	**C499**	**C880**	**C1355**	**C1908**	**C2670**	**C3540**	**C5315**	**C6288**	**C7552**	**Avg.**
PowerMill (L=50,000)		I (uA)	289.6	685.5	411.8	790.9	841.7	939.1	2058.7	2540.5	21936.6	3786.6	
		Time (s)	8542	16757	13774	22064	22925	18100	57450	49813	290751	63705	
Random Sampling		I (uA)	299.4	708.3	437.6	824.2	879.7	981.4	2134.3	2613.3	22603.6	3999.4	
		Error (%)	3.38	3.33	6.27	4.21	4.51	4.50	3.67	2.87	3.04	5.62	**4.14**
		L/U	377/187	385/190	390/194	388/192	380/188	388/192	393/195	391/194	388/193	390/193	
		ECR	132.63	129.87	128.21	128.87	131.58	128.87	127.23	127.88	128.87	128.21	**129.22**
		Time (s)	93.6	177.8	147.5	232.1	237.9	229.2	563.7	552.3	2640.0	703.4	
		Speedup	91.26	94.25	93.38	95.06	96.36	78.97	101.92	90.19	110.13	90.57	**94.21**
Consecutive Sampling	**Single Sequence**	I (uA)	291.6	694.6	422.0	824.4	891.2	978.4	2160.8	2620.3	22741.0	3887.7	
		Error (%)	0.69	1.33	2.48	4.24	5.88	4.18	4.96	3.14	3.67	2.67	**3.32**
		L/U	294/0	245/0	233/0	324/0	228/0	347/0	323/0	272/0	262/0	316/0	
		ECR	170.07	204.08	214.59	154.32	219.30	144.09	154.80	183.82	190.84	158.23	**179.41**
		Time (s)	79.6	129.3	103.2	201.4	169.0	214.3	482.9	433.1	1908.5	609.8	
		Speedup	107.31	129.60	133.47	109.55	135.65	84.46	118.97	115.02	152.35	104.47	**119.08**
	Multi Sequence	I (uA)	301.7	699.2	430.1	824.3	886.9	975.3	2143.6	2616.9	22829.2	4036.3	
		Error (%)	4.18	2.00	4.44	4.22	5.37	3.85	4.12	3.01	4.07	6.59	**4.19**
		L/U	195/5	201/6	202/6	200/4	196/4	198/2	202/4	200/3	197/2	201/4	
		ECR	256.41	248.76	247.52	250.00	255.10	252.53	247.52	250.00	253.81	248.76	**251.04**
		Time (s)	62.3	115.2	94.5	148.9	153.2	160.4	343.0	362.8	1530.4	463.1	
		Speedup	137.11	145.46	145.76	148.18	149.64	112.84	167.49	137.30	189.98	137.56	**147.13**

Table 2-4. Consecutive sampling techniques for LFSR input sequences

		Circuit	C432	C499	C880	C1355	C1908	C2670	C3540	C5315	C6288	C7552	Avg.
PowerMill (L=50,000)		I (uA)	293.3	668.1	383.9	772.4	847.1	963.9	1944.6	2341.7	19912.7	3867.8	
		Time (s)	8206	15296	13655	21123	21139	19741	52857	46819	273940	64931	
Random Sampling		I (uA)	307.3	683.4	408.2	794.6	883.4	1007.4	2033.9	2434.1	21290.4	3948.1	
		Error (%)	4.77	2.29	6.33	2.87	4.29	4.51	4.59	3.95	6.92	2.08	**4.26**
		L/U	380/188	386/192	395/195	386/191	382/189	387/192	383/189	391/193	388/192	390/193	
		ECR	131.58	129.53	126.58	129.53	130.89	129.20	130.55	127.88	128.87	128.21	**129.28**
		Time (s)	94.1	178.8	144.2	219.4	242.1	238.1	547.7	544.3	2458	701.4	
		Speedup	87.21	85.55	94.69	96.28	87.32	82.91	96.51	86.02	111.45	92.57	**92.05**
Consecutive Sampling	**Single Sequence**	I (uA)	311.1	697.4	398.6	795.3	878.7	1000.8	2047.3	2430.8	20973.8	3957.5	
		Error (%)	6.07	4.39	3.83	2.96	3.73	3.83	5.28	3.80	5.33	2.32	**4.15**
		L/U	260/0	357/0	240/0	306/0	285/0	263/0	266/0	254/0	473/0	244/0	
		ECR	192.31	140.06	208.33	163.40	175.44	190.11	187.97	196.85	105.71	204.92	**176.51**
		Time (s)	69.4	157.6	96.3	187.9	180.1	195.1	385.6	406.2	2971	528.4	
		Speedup	118.24	97.06	141.80	112.42	117.37	101.18	137.08	115.26	92.20	122.88	**115.55**
	Multi Sequence	I (uA)	315.2	704.5	403	781.3	891	999.9	2059.2	2460.4	21238.4	3956.4	
		Error (%)	1.57	3.83	6.41	0.17	6.7	3.73	5.89	5.07	6.66	2.29	**4.23**
		L/U	196/4	198/4	205/5	199/4	197/4	198/3	198/4	201/3	200/4	200/3	
		ECR	255.10	252.53	243.90	251.26	253.81	252.53	252.53	248.76	250.00	250.00	**251.04**
		Time (s)	56.5	106.1	85.4	138.9	140.8	164.3	319.4	358.7	1517.3	467.3	
		Speedup	145.24	144.17	159.89	152.07	150.13	120.15	165.49	130.52	180.54	138.95	**148.72**

According to the experimental results, the speedups of three methods are 94.21, 119.08 and 147.03 respectively. The multi-sequence approach improves 56% on speed compared to random sampling approach. The single-sequence approach only improves 26% on speed. It shows that we can obtain the highest speedup using the multi-sequence approach for all test cases in the benchmark. The average compaction ratio achieved in random sampling approach is 129.22 and the average error is 4.14%. The average compaction ratio achieved in single-sequence approach is 179.41 and the average error is 3.32%. The average compaction ratio achieved in multi-sequence approach is 251.04 and the average error is 4.19%. It shows that the multi-sequence approach can dramatically reduce the useless transitions in the random sampling method such that it can almost keep the desired compaction ratio exactly. In our experiments, the useless transitions in the multi-sequence approach for all cases are not larger than 6. In random sampling approach, the useless transition is at least 187. Compared to the single-sequence approach, the compaction ratio in the multi-sequence approach is still much higher, especially when the pattern pairs of some groups are not uniformly distributed in the original sequence, such as C2670. About the compaction error, the average errors of all three approaches are less than 5%. It shows that this multi-sequence approach can improve the speedup much more with reasonable accuracy.

In order to verify the effects of our approach on those input patterns that are not pure random, we perform another experiment that uses the input sequences generated from a linear feedback shifter register (LFSR) because a LFSR sequence is easier to generate and has highly spatial correlation that is quite different to pure random patterns. The experimental results are shown in Table 2-4. Compared with the results in Table 2-3, we can see that the proposed approach can still be effective with different input distribution.

2.8 Summary

In this work, we proposed a multi-sequence sampling technique to reduce the useless transitions in the compacted sequence and improve the over sampling problem of our previous single-sequence approach. By relaxing the limitation a little bit such that multiple consecutive sequences are allowed, we can find better solutions for vector compaction problem if we minimize the number of sequences instead of setting the number to be one. Of course, the number of sequences could be one as handled in the original single-sequence approach, but it is just a special case in the multi-sequence approach. As demonstrated in the experimental results, the multi-sequence approach improves 56% on speed compared to the random sampling approach. The single-sequence approach only improves 26% on speed. It shows that this multi-sequence approach can improve the speedup much more with reasonable accuracy.

Chapter 3

Gate-Level Power Modeling with 1-D LUT

3.1 Power Modeling with Lookup Table

In SoC designs, power models may provide an efficient solution to estimate the power consumption of IPs because the transistor-level simulation is only required at the characterization step. The power model of a design, which is built from a power characterization process, describes the relationship between power characteristics and real power consumption with specific input sequences or input signal statistics. Lookup tables are the most commonly used power models. Once the required power characteristics are obtained, the estimated power consumption can be easily found in the table.

Because the power dissipation of a combinational circuit depends on the previous and present input patterns, a fully characterized lookup table for an n-input combinational circuit will have 2^{2n} entries. For a sequential circuit with n-input and s internal state registers, a fully characterized lookup table will have 2^{2n+s} entries. It is obviously infeasible for complex circuits because the table size is too large to be stored and the characterization process will consume too much time. Efficient reduction methods are definitely required to make this approach become feasible.

In this approach, the chosen power characteristics have large impacts on the table size and accuracy of the estimated power consumption. Therefore, many power characteristics are proposed in the literature [19][65~69][71], such as the signal statistics (probability, transition density, correlation coefficient, etc.) of primary inputs and outputs, the active information of a design (switching count, charging and discharging capacitance, etc.), the power sensitivity of primary inputs, the Hamming distance of the pattern pairs at primary inputs, etc. The methods proposed in [19][71] are building the lookup tables according to the signal transitions at the primary inputs. In [71], the authors use a clustering algorithm to compress the input vectors with similar power consumption as a cluster such that the table size can be reduced. The experimental results show that the method is useful for the test circuits but there is no clear evidence about the effects on large circuits. The method in [19] operates on the state transition graphs (STGs) of macrocells with merging transition compatible nodes to reduce the sizes of lookup tables. However, it can only be used on the designs with small input numbers because the table size will increase rapidly when the number of primary inputs is increased.

The methods in [65~68] use the signal statistics of primary inputs and outputs to be the indexes of lookup tables. In [65], the lookup tables with 2 dimensions (average input signal probability, average input signal transition density), 3 dimensions (average output zero-delay transition density as the third dimension) and 4 dimensions (average spatial correlation coefficient as the fourth dimension) are compared. The results show that the estimation errors are decreased when the dimensions of tables are increased, but the sizes of tables are also increased. The increase of table size will require extra characterization time that may become a non-neglectable overhead. In [66], a power model based on the hamming distance of input pattern pairs is proposed. The model could be parameterized according to the input bit-width of the function unit. The method in [67] builds a five-dimension lookup table with average signal probability of high-switching inputs and low-switching inputs, average transition density of high-switching inputs and low-switching inputs and average output transition density. The estimation errors are close to the estimation errors of 4-D lookup table in [65]. In [68], they build a 3-D lookup table, which is similar to the 3-D table in [65], for soft macros with some different parameters such as the bit-width of inputs and the target manufacturing technologies. However, because the distribution of the average output transition density is hard to control, the characterization time to fill the lookup tables is hard to control in those approaches [65~68].

The method in [69] uses the power surface based on the power sensitivity of primary inputs to be the power models. They proposed an efficient method to approximate the surface with a finite number of points. They compared their results with logic simulation results under zero-delay and unit delay model but they did not show the comparison with transistor-level simulation results.

Based on above observations, we can recognize that the size of lookup table is a primary concern for the power models of complex designs such as commercial IPs. Therefore, we propose a table-based power modeling method in this work for combinational circuits in which the table size is very small and almost independent to the number of primary inputs. In order to reduce the table size, we build a one-dimension lookup table to map the zero-delay charging and discharging capacitance (CDC) to the real power consumption of input pattern pairs, which are obtained from gate-level simulation and transistor-level simulation individually. In order to simplify the description, we will use CDC to represents the zero-delay charging and discharging capacitance in the rest of this paper. The CDC of a pattern pair is the summation of charging and discharging capacitances of the nodes whose signals change from 0 to 1 or 1 to 0 during the transition of input patterns under zero-delay model. Using CDC as the index of the lookup tables is decided by our previous comparison results of the ***average normalized error*** between the three power characteristics, CDC, zero-delay switching count (SC) of internal nodes and Hamming distance (HD) of input pattern pairs in Section 2.3. Among those power characteristics, the CDC has the minimal average normalized error, which may provide more precise estimation results. Therefore, we choose CDC as the only one index of lookup tables to reduce the errors of power models. As shown in the experimental results, our CDC-based power models can still provide accurate results although the table size has been greatly reduced.

3.2 Proposed Power Modeling Methodology

In this work, the power model is represented by a lookup table, which maps the CDC values to real power consumption of input pattern pairs. According to Equation (2-5), it implies that we use the P_{func_trans} to indicate the trend of real power consumption, which is the dominated part of power consumption. The building flow of the lookup table is shown in Figure 3-1. We will first divide the input pattern pairs into several groups according to their CDC values that are calculated by a logic-level simulator. Those pattern pairs within an interval of CDC values will be grouped together and the average power of them, which is estimated by PowerMill, will be recorded in the corresponding entry of the lookup table. The input sequence for power characterization is randomly generated such that the pattern pairs in the same group can be viewed as randomly distributed. Therefore, we can use the Monte Carlo approach to reduce the characterization time for the average power consumption of those groups. In the following sections, the proposed approaches for grouping the pattern pairs and the power characterization of each group will be explained in detail.

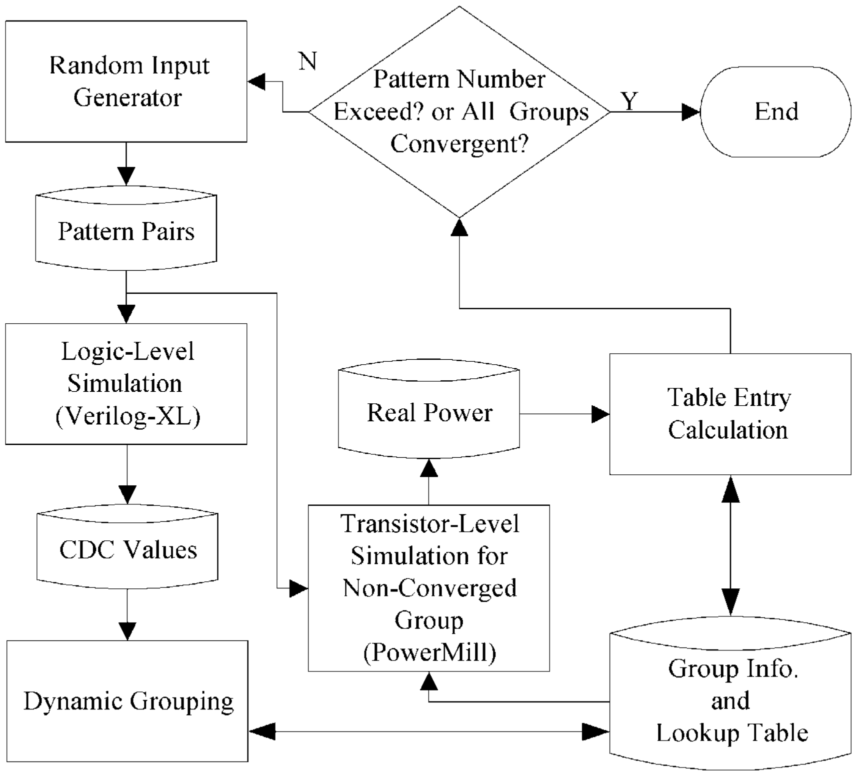

Random Input Generator
N
Pattern Number Exceed? or All Groups Convergent?
Y
End
Pattern Pairs
Logic-Level Simulation (Verilog-XL)
Real Power
Table Entry Calculation
CDC Values
Transistor-Level Simulation for Non-Converged Group (PowerMill)
Dynamic Grouping
Group Info. and Lookup Table

Figure 3-1. The block diagram for building the power model

3.3 Dynamic Grouping

Using the CDC values of pattern pairs to be the index of a lookup table may still require huge table size if we set a table entry for each different CDC value. Although the table size will be much smaller than 2^{n+n}, where n is the number of primary inputs of the circuit, it is still very huge. In order to reduce the table size, we can collect those pattern pairs with similar CDC values to be a group and only set one entry in the lookup table for each group. A similar grouping method was used in pattern compaction techniques for power estimation [39][44]. They calculate the CDC values of the pattern pairs in the input sequence by a logic-level simulator such that they can collect those pattern pairs with similar CDC values as a group. Then, the pattern compaction can be done by selecting some pattern pairs as representatives from each group because those pattern pairs in the same group have similar power consumptions. In [40][76], however, the compacted sequence is generated for a specific input sequence. In other words, the input sequence is deterministic and the distribution of CDC values is deterministic, too. Therefore, they can separate the pattern pairs into finite groups in their grouping algorithm. Unfortunately, when we build the lookup table for the proposed power model in our work, the CDC distribution is non-deterministic until we simulate all pattern pairs, which is almost impossible for large circuits even using a logic-level simulator.

In order to handle this situation without simulating all possible cases, we propose a method to dynamically increase the entries of the lookup tables to cover the current CDC distribution of designs when we characterize the average power of each entry in the table. As illustrated in Figure 3-2, the CDC values of pattern-pairs have been sorted before grouping. The X-coordinate is the number of pattern pairs and the Y-coordinate is the CDC value of each pattern pair. In the first iteration, we randomly generate several pattern pairs and the dynamic grouping in this step is similar to the grouping process in [40][76] as shown in Figure 3-2(a). Each group is defined with an interval of CDC values and the neighborhood groups have continuous CDC values. This is different to the grouping process [40][76] which defines a group with the CDC values of two boundary pattern pairs and the neighborhood groups may not have continuous CDC values. With continuous interval definition, we can easily find the corresponding groups for the pattern pairs with CDC values between existing ranges in the following iterations.

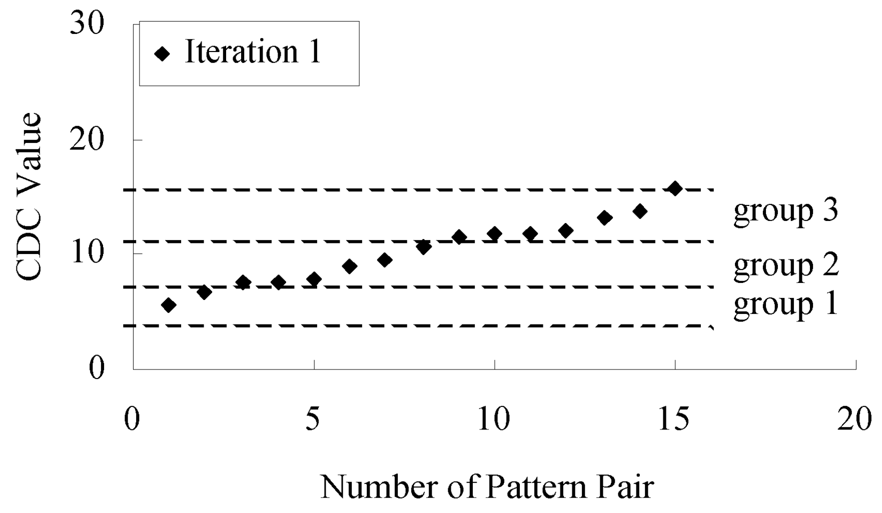


Figure 3-2(a). The dynamic grouping process after the first iteration

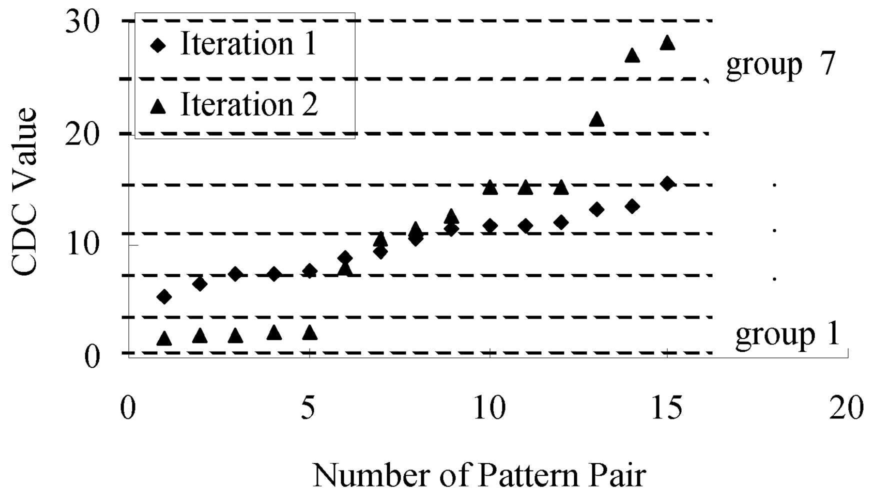


Figure 3-2(b). The dynamic grouping process after the second iteration

In the second iteration, we generate more random patterns and the number of group is spread because the CDC distribution area is increased as shown in Figure 3-2(b). The range of each group in Figure 3-2(a) is not changed but new groups are generated from the boundaries of the first and last groups in Figure 3-2(a). It implies that the size of the lookup table in our power model is determined by the number of groups in the dynamic grouping process, which can be controlled by the user-defined group interval. The larger group interval will lead to fewer table entries. This group interval is defined by a percentage of the range from the maximum CDC value to the minimum CDC value of each group and set as 5% of the maximum CDC value in this work. If the interval is smaller than the minimum load capacitance of the nodes, the interval will be set as the minimum load capacitance of nodes because it is impossible to have such CDC values. If the group interval is defined as 5% of the maximum CDC value in the group, the table will only increase 45 entries when the CDC distribution region is spread 10 times to the previous distribution region. Therefore, even the circuit size are increased, the table entries are only increased linearly.

In order to explain the dynamic grouping process more clearly, the pseudo code of the proposed algorithm is shown in Figure 3-3. In the first iteration, the grouping process is performed according to the CDC range of the first random sequence. Those groups are defined one by one from the minimum CDC value of the input sequence until the range of groups cover the maximum CDC value of the input sequence. Note that the interval of each group, which is defined by its minimum CDC value and maximum CDC value, cannot be smaller than the parameter MIN_NODE_CDC in the circuit. After all groups are defined, the ***allocate()*** function allocates each pattern pair in the input sequence into the corresponding group. In the following iterations, the number of groups may be increased by ***insert_group()*** or ***expand_group()*** functions if their CDC distribution is out of the range of current groups. The ***insert_group()*** and ***expand_group()*** functions will perform similar operations like the process in the first iteration to cover the CDC distribution of new input sequence. The only difference is that the two functions start with the boundary values of the previous iteration. The ***allocate()*** function will again allocate those pattern pairs in the new input sequence into corresponding groups as in the first iteration. After each iteration, the group information of each pattern pair can be obtained for the following power characterization process.

```
Dynamic_grouping(group[], seq[], iteration, p_n, group_num)
{
  if(iteration == 1)    /*Grouping for initial CDC distribution*/
  {
    max_cdc = Max(seq[]); min_cdc = Min(seq[]);
    /*max_cdc and min_cdc define the range of CDC distribution of current input sequence*/
    group_num=0;
    While( min_cdc < max_cdc)
    {
      group[group_num].min_cdc = min_cdc;
      if ((min_cdc*5/95) < MIN_NODE_CDC)
      {
        group[group_num].max_cdc = min_cdc + MIN_NODE_CDC;
      } else
      {
        group[group_num].max_cdc = min_cdc*100/95;
      }
      min_cdc = group[group_num].max_cdc;
      group_num++;
    }
    allocate(group[], group_num, seq[], p_n);
  } else        /*Group increasing if CDC distribution spread*/
  {
    max_cdc = Max(seq[]); min_cdc = Min(seq[]);
    if (max_cdc > group[group_num-1].max_cdc)
    {
      group_num=expand_group(group[], group_num, max_cdc);
    } elseif (min_cdc < group[0].min_cdc)
    {
      group_num=insert_group(group[], group_num, min_cdc);
    }
    allocate(group[], group_num, seq[], p_n);
  }
}
```

Figure 3-3. The pseudo code of dynamic grouping process

3.4 Power Characterization

In our power model, the corresponding power for each table entry is determined by the average power consumption of all pattern pairs located in the corresponding CDC interval. Therefore, we use a random input generator to generate a number of pattern pairs such that they can distribute over different groups. Figure 3-4 gives an illustration of this power characterization flow. The power characterization process will stop when one of the following two conditions is satisfied.

(a). The average power consumption of each group has reached the desired confidence level.

(b). The total pattern pairs have reached the constraint of maximum number of pattern pairs.

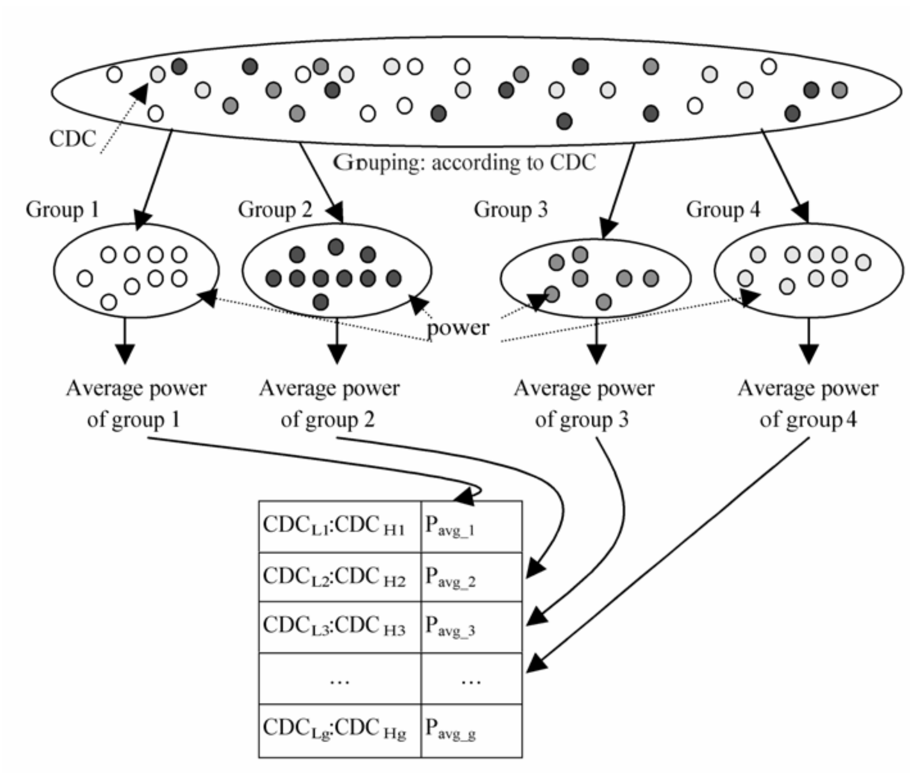


Figure 3-4. An illustration of the power characterization

The maximum number of characterized pattern pairs is used to control the characterization efforts. It can be decided by users to make a trade-off between accuracy and characterization efforts. If criterion (b) is used to stop the characterization process, the average power of those groups that do not have enough pattern pairs will be estimated with interpolation or extrapolation because the current samples may not have enough representatives.

In order to further improve the efficiency of the characterization process, we use the Monte Carlo approach to check the stop criteria (a) such that we can finish the characterization process as soon as possible. The Monte Carlo simulation [40] has been used to reduce the simulation efforts for estimating the average power of circuits in [31,32][37]. Under the assumption that the mean of any sample is normal distribution, the end of simulation can be decided according to the statistical stopping criterion shown in Equation (3-1). In Equation (3-1), ε is the maximum percentage of acceptable error, N is the number of sample, η_T is the sample mean and s_T is the sample standard deviation. For (1-α) confidence level, $t_{\alpha/2}$ is t-distribution coefficient with (N-1) degrees of freedom.

$$\frac{t_{\alpha/2} s_T}{\eta_T \sqrt{N}} < \varepsilon \qquad (3\text{-}1)$$

The proposed dynamic grouping process will categorize all pattern pairs of the input sequence into several groups. We can assume that the pattern pairs in a group are also randomly distributed because the input sequence is randomly generated. Therefore, the Monte Carlo approach can also be applied to speed up the estimation for the average power of each group. After the estimation of average power has converged according to the Monte Carlo stop criteria, we will not simulate the following pattern pairs for these groups in the transistor-level simulator because the current results already have the desired accuracy. More samples will not improve the accuracy too much. Therefore, we can skip those samples to save a considerable computation time.

Of course, this grouping-based power characterization process may induce some errors in the power model because of the variation in each group. However, if the adopted power characteristic represents the real power consumption very well, the induced errors can be very small. Therefore, we perform a simple experiment on ISCAS'85 benchmark circuits with 0.35*um* cell library to discuss the possible variation of various power characteristics. In this experiment, the group interval is set as 5% of the maximum CDC value. In characterization process, the random input generator generates a sequence with 5,000 pattern pairs in every iteration and the maximum number of characterized pattern pairs is set as 100,000. The confidence level in the Monte Carlo criteria is set as 0.99 ($\alpha = 0.01$) under the maximum acceptable error ε is set as 0.05. The sample size is set as 30.

In Table 3-1, we show the differences between the characterized power value of each group and the real power consumption of the pattern pairs belong to the corresponding group with three power characteristics, CDC, SC, and HD. The first column shows the percentage of the pattern pairs whose errors are smaller than 10%. The second and third columns show the percentage of the pattern pairs whose errors are large than 10% and 40% respectively. The same grouping method and characterization process are applied for each power characteristic. And the real power consumptions of those pattern pairs are obtained by transistor-level simulation in the characterization process. According to the experimental results, it is very clear that using CDC as the power characteristic is the best choice to minimize the errors of power models.

Table 3-1. The distribution of the variance in the groups

Circuits	<10%			10% ~ 40%			>40%		
	CDC	SC	HD	CDC	SC	HD	CDC	SC	HD
C432	65.67	58.54	28.46	27.39	34.13	53.73	6.94	7.33	17.81
C499	93.33	90.43	73.76	6.67	7.36	21.43	0	2.21	9.81
C880	82.84	83.21	36.33	16.9	16.34	44.27	0.26	0.45	19.4
C1355	98.71	93.88	82.20	1.29	6.12	15.92	0	0	1.88
C1908	72.41	70.22	35.40	25.68	27.37	59.42	1.91	2.41	5.18
C2670	68.50	53.79	38.27	28.88	39.64	44.31	2.62	6.57	17.42
C3540	60.31	60.21	34.89	38.51	37	61.68	1.18	2.79	3.43
C5315	90.19	88.77	29.44	9.47	9.43	43.05	0.34	1.8	27.51
C6288	59.43	46.6	27.27	37.88	45.1	64.58	2.69	8.3	8.15
C7552	78.91	61.9	57.48	19.77	30.95	31.89	1.32	7.15	10.63
Average	77.03	70.76	44.35	21.24	25.34	44.03	1.73	3.90	12.12

3.5 Power Estimation with the Power Model

After the power model of a circuit is built, the average power consumption for any test sequence can be estimated as shown in Figure 3-5. First, we use a logic-level simulator (e.g. Verilog-XL) to calculate the CDC values of pattern pairs in the test sequence. With the CDC values, we can find their corresponding groups in the lookup table for those pattern pairs in the sequence. If a pattern pair belongs to a CDC interval, its power consumption will be set as the value of the corresponding table entry in the lookup table, and the total power is equal to the summation of total values of every pattern pairs. For the average power, it can be obtained from dividing the total power by the number of pattern pairs in the test sequence.

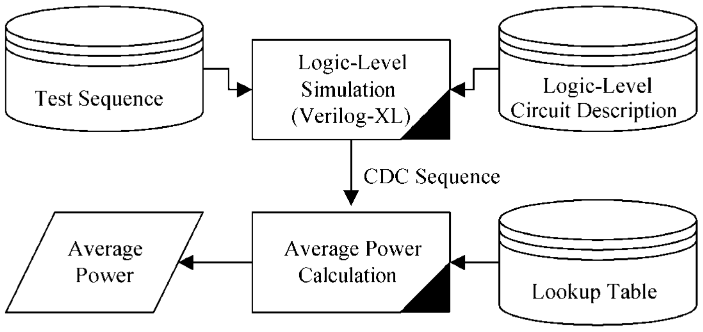

Test Sequence
Logic-Level Simulation (Verilog-XL)
Logic-Level Circuit Description
CDC Sequence
Average Power
Average Power Calculation
Lookup Table

Figure 3-5. Block diagram of average power calculation

The lookup table may not cover whole CDC distribution of all possible pattern pairs because we did not simulate all pattern pairs in the characterization process. In this case, we can use extrapolation to estimate the power consumption of those pattern pairs whose CDC values are out of the range of the lookup table. The average power consumption can be expressed as Equation (3-2). In Equation (3-2), N is the total number of pattern pairs in the test sequence. g is the number of entries of the lookup table. P_i is the average power recorded in the i^{th} entry of the lookup table. n_i is the number of pattern pairs in the test sequence whose CDC values are involved in the CDC interval of i^{th} entry. $P_{out\_of\_range}$ is the total power consumption of those pattern pairs whose CDC values are out of the range of the lookup table. As shown in Equation (3-3), we can use extrapolation to estimate the power consumption of those pattern pairs. P_1, P_2, P_{g-1} *and* P_g are defined as P_i in Equation (3-2). CDC_1, CDC_2, CDC_{g-1} *and* CDC_g are the largest CDC values of entries *1*, *2*, *g-1* and *g* in the lookup table. k_r and k_l are the numbers of pattern pairs which are out of the smallest and largest range of CDC values in the lookup table.

$$P_{avg} = \frac{\sum_{i=1}^{g} P_i \times n_i + P_{out\_of\_range}}{N} \qquad (3\text{-}2)$$

$$P_{out\_of\_range} = \sum_{i=1}^{k_r}\left[P_1 - \left(\frac{P_2 - P_1}{CDC_2 - CDC_1}\right)\times\left(CDC_1 - CDC_i\right)\right]$$

$$+\sum_{i=1}^{k_l}\left[P_g + \left(\frac{P_g - P_{g-1}}{CDC_g - CDC_{g-1}}\right)\times\left(CDC_i - CDC_g\right)\right] \qquad (3\text{-}3)$$

3.6 Experimental Results

In this section, we will show some experimental results about the accuracy and efficiency of the proposed power model. The experiments are obtained on a SUN UltraSPARC II workstation. The test circuits are ISCAS'85 benchmark circuits with 0.35*um* cell library. The experimental results are shown in Table 3-2. The table sizes for the benchmark circuits are listed in the 2nd row under the names of circuits in the 1st row. According to the results, table sizes are only 42 to 107 for those circuits. It is very small and almost independent to the circuit size.

In order to show that our approach can be applied to various input sequences, we test the accuracy of our method by estimating the average power consumption of circuits with 3 different sequences. They are pseudo random sequence, counter sequence (up/down counter) and LFSR (Linear Feedback Shift Register) sequence with 50,000 pattern pairs respectively. The 3rd row to 7th row give the comparison between the simulation results with PowerMill and the estimation results from table lookup for pseudo random sequence. The next 5 rows give the comparison results with counter sequence and the last 5 rows give the results with LFSR sequence. The estimation time listed in Table 3-2 for table-lookup method includes the Verilog-XL simulation time and average power calculation time. The maximum error is

7.48% for C2670 with LFSR sequence. The overall average error is only 2.99%. The experimental results show that our power model still has high accuracy for different input sequences.

Table 3-2. The experimental results

		Circuits	C432	C499	C880	C1355	C1908	C2670	C3540	C5315	C6288	C7552
		Table Size	63	45	75	51	42	90	65	103	107	103
Random Sequence	PowerMill	I (uA)	56.135	149.718	106.480	161.386	144.083	261.945	340.913	611.173	4841.000	830.832
		Time (Sec)	3072	8626	5826	9494	7649	14667	19017	33918	290751	43169
	1-D Table	I (uA)	57.083	148.698	103.818	158.006	141.921	253.659	327.412	598.521	4722.450	816.488
		Time (Sec)	23.4	70.3	46.2	65.9	54.5	112.5	114.9	205.4	388.5	267.3
	Error (%)		1.69	0.68	2.50	2.09	1.50	3.16	3.96	2.07	2.45	1.73
Counter Sequence	PowerMill	I (uA)	13.243	34.493	38.119	38.270	50.477	6.402	191.720	20.911	351.290	68.626
		Time (Sec)	690	1958	1773	2163	2417	479	9347	1275	19385	3631
	1-D Table	I (uA)	13.971	33.468	37.153	37.456	53.483	5.923	192.283	20.703	355.533	70.526
		Time (Sec)	19.7	61.9	39.4	57.2	48.6	84.9	105.1	159.3	257.6	210.5
	Error (%)		5.50	2.97	2.53	2.13	5.96	7.48	0.29	1.00	1.21	2.77
LFSR Sequence	PowerMill	I (uA)	71.436	167.103	118.619	182.053	169.728	286.910	374.463	669.812	5014.680	975.568
		Time (Sec)	4050	9496	6786	11007	9460	16789	21914	38744	309071	52882
	1-D Table	I (uA)	66.274	161.111	114.371	178.741	162.510	282.068	361.522	652.722	4820.264	936.464
		Time (Sec)	24.4	71.8	47.5	68.4	56.4	117.1	117.3	210.9	398.3	279.3
	Error (%)		7.23	3.59	3.58	1.82	4.25	1.69	3.46	2.55	3.88	4.01
Average Error (%)			4.80	2.41	2.87	2.01	3.90	4.11	2.57	1.87	2.51	2.83

3.7 Summary

In this work, we proposed an efficient IP-Level power model with a small lookup table for complex CMOS circuits. The lookup table has only one-dimension that maps the zero-delay charging and discharging capacitance (CDC) to the real power consumption of input pattern pairs but still has high accuracy. This power characteristic is selected according to several experimental results shown in this work within three popular power characteristics. In order to reduce the table size, we proposed a dynamic grouping algorithm to collect those pattern pairs with similar CDC values to be a group and only set an entry in the lookup table for each group. For improving the efficiency of characterization process, the Monte Carlo approach is used during the estimation for the average power of each group to skip the samples that will not increase the accuracy too much. The experimental results show that our power model can estimate the average power of IP-level complex designs very efficiently and accurately for various test sequences.

Chapter 4

High-Level Power Modeling with Neural Networks

4.1 High-Level Power Modeling

Lookup table (LUT) is the most commonly used high-level power model. In order not to increase the table size too much, most of the LUT-based approaches [65~68] use the aggregate signal statistics (average input signal probability, average input signal transition density, average output signal transition density, input signal correlation coefficient, etc.) of the primary inputs and outputs of circuits to be the indexes of lookup tables. In [65], the lookup tables with 2 dimensions, 3 dimensions and 4 dimensions were compared. The results showed that the estimation errors are decreased when the dimensions of tables are increased, but the sizes of tables are also exponentially increased. For large circuits, the table size may increase very fast in order to meet the accuracy requirement.

In Chapter 3, we proposed a one-dimension lookup table using the zero-delay charging and discharging capacitance as table index. An efficient method is proposed to divide power characteristics of pattern pairs into several groups and fill the lookup table with the average power consumption of each group. Although this approach can build smaller lookup tables with reasonable accuracy, it still requires gate-level descriptions and node capacitance information to obtain the total charging and discharging capacitance in the circuits, which may not provided by IP vendors.

There are also some approaches [73~75] that use equations instead of lookup tables to be the power models. After identifying suitable variables for the power equations of a circuit, those equation-based approaches will use some numerical methods such as linear regression to find out the best parameter for each variable to form the equations. Compared to LUT-based approaches, equation-based approaches often have fewer data to be recorded for a power model because the distribution of an equation often requires many points to describe. However, because the power distribution is often a very irregular curve as illustrated in Figure 4-1, it is hard to use only a single equation to describe this curve. Therefore, in order to improve the accuracy of their power models, those equation-based approaches may increase the order of the power equations (more variable) or use piece-wise power equations (more equations) to approximate the power distribution, which will significantly increase the complexity of the power models.

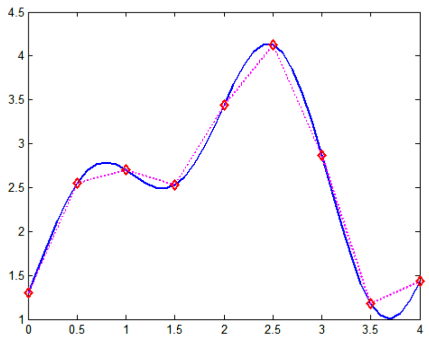


Figure 4-1. Irregular power distribution and piece-wise approximation

Most of the above techniques focus on estimating the average power consumption over a long input sequence, which are referred to as *cumulative power models*. However, in some applications, the average power is not sufficient. One of the other important tasks is to understand the power consumption of a circuit due to a given pattern pair, which is often referred to as *cycle-accurate power models* [74,75]. This information is crucial for circuit reliability analysis, dc/ac noise analysis, and design optimization. Of course, those cycle-accurate power models can also provide the information of average power consumption by just computing the average of the power consumed at each cycle in the given input sequence. Therefore, cycle-accurate power models are considered to have more use than cumulative power models. Because it is not feasible to build a lookup table for every possible combinations at each cycle, most of those cycle-accurate power models use equation-based approaches to record the power distributions.

In other research areas, neural networks play as a powerful tool in many applications such as classification, clustering, pattern recognition, control application, etc. Because of the self-learning capability of neural networks, they can recognize complex characteristics by using several simple computation elements with proper training. For irregular distributions such as the power distribution shown in Figure 4-1, neural networks can still have good efficiency because they use the combination of some non-linear curves to fit the multi-dimensional non-linear surface instead of increasing the recording points to reduce the errors as in the traditional power models. Therefore, several researches [72][81] tried to use neural networks to solve the power estimation problem. The authors in [81] proposed a symbolic neural network model to estimate the power consumption of circuits. Based on Hopfield neural network [82], they built another representation for the gate-level description of circuits and stored the structure information of the neural networks in algebraic decision diagrams [83] to reduce the memory usage. In that approach, neural networks were only used to replace the gate-level structures and to perform a gate-level simulation for estimating the power consumption of circuits with fewer resources. Therefore, the simulation time was only similar to the gate-level simulation, which will be very slow for large circuits.

The authors in [72] proposed a power modeling approach for library circuits using Bayesian inference and neural networks. They divided the leakage and switching power distributions of circuits into a limited number of classes and trained two neural networks to classify an input state or transition into the corresponding class. After classification, the leakage power of an input state and the switching power of an input transition can be estimated by the average power consumption of that class, which has been stored in a lookup-table as in the traditional table-based approaches. Although the classification of an input state or transition can be more accurate by using neural networks, the number of classes may limit the accuracy of this power modeling approach over the entire power spectrum. Therefore, they may also have to increase the number of table entries to reduce the estimation error, which is similar to the problem of traditional table-based approaches. In addition, if the table entries are increased, the number of outputs of the neural networks is also increased. If there are too many outputs in a neural network, it will often become much harder to converge and sacrifice the classification accuracy. However, the authors did not show the experimental results for the cases with wide power distribution and large number of classes.

In this work, we propose a quite different approach for high-level power modeling of complex digital circuits that uses a 3-layer fully connected feedforward neural network [84] to learn the power characteristics during simulation without any lookup tables. By considering all possible types of state transitions separately in the input data, both the state-dependent leakage power and transition-dependent switching power are still recorded well in our power model. In addition, because the numbers of input and output neurons in our neural power model are fixed as 8 and 1 respectively, the complexity of our neural power model has almost no relationship with circuit size and the numbers of primary inputs and outputs such that this power model can be kept very small even for complex circuits. Unlike the piece-wise equations in the equation-based approaches, only one simple neural model is enough for those test circuits to provide similar accuracy thus reducing the modeling complexity.

4.2 Background about Neural Networks

4.2.1 Feedforward Neural Networks

The basic unit in a neural network is an artificial neuron as shown in Figure 4-2. In Figure 4-2, x_1 to x_N are the input data for the neuron, w_1 to w_N are the weights of input x_1 to x_N individually that represent the contribution from each input, and s is the summation of x_1w_1 to x_Nw_N and the bias factor x_0w_0 as represented in Equation (4-1). In most cases, x_0 is fixed as 1 such that the training algorithm only adjusts the weight w_0 to w_N. Function f is the transfer function that converts s into output y as represented in Equation (4-2). According to the relationship between input data and output data, users can choose different transfer function f for different cases.

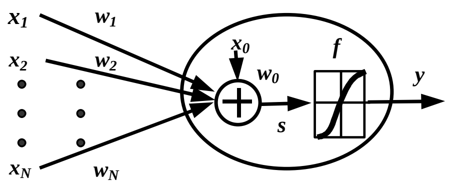


Figure 4-2. An artificial neuron

$$s = \sum_{i=0}^{N} w_i x_i \tag{4-1}$$

$$y = f(s) \tag{4-2}$$

A neural network is a set of interconnected neurons, where the outputs of neurons act as the inputs of other neurons or the final outputs of the neural network. Feedforward neural network [84] is one of the most popular models of neural networks. Although any architecture of neural network with learning capabilities can be used in this work, we use the 3-layer fully connected feedforward neural network, as shown in Figure 4-3, to learn the relationship between power consumption and statistic information of input patterns because the 3-layer feedforward neural network has a quite simple structure with good performance in many applications. In addition, the fully connected configuration can automatically consider the correlation between all inputs by properly adjusting the weights for their interconnections. The accuracy of this power model can thus be improved because the correlation is also an important factor that affects power consumption.

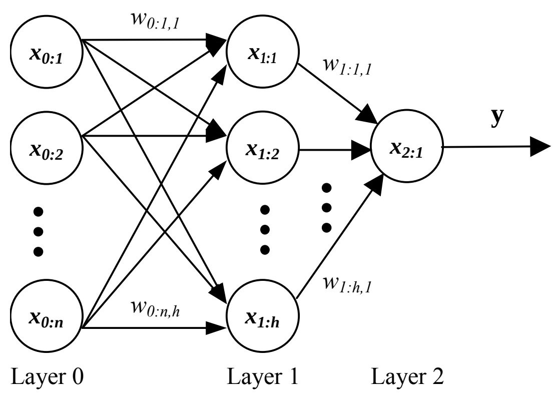


Figure 4-3. A fully connected 3-layer feedforward neural network

In Figure 4-3, $x_{l:i}$ represents the ith neuron in the lth layer, $w_{l:i,j}$ represents the weight of the interconnection between neuron $x_{l:i}$ and $x_{l+1:j}$, and y is the final output of this neural network. The number of input neurons is n and the number of hidden neurons is h. In order to simplify the graph, the biases, the summations, and the outputs of all neurons are not labeled in Figure 4-3. The weight of the bias in neuron $x_{l+1:j}$ is denoted as $w_{l:0,j}$, the input of the bias in neuron $x_{l+1:j}$ is denoted as $a_{l:0}$, the summation s in neuron $x_{l:i}$ is denoted as $s_{l:i}$, and the output of neuron $x_{l:i}$ is denoted as $a_{l:i}$. The output y in this neural network, which is denoted as $a_{2:1}$, is defined as Equation (4-3).

$$y = a_{2:1} = f(s_{2:1}) = f\left(\sum_{i=0}^{h} w_{1:i,1} a_{1:i}\right) \qquad (4\text{-}3)$$

4.2.2 Training Process

Before using a neural network, we have to train the neural network with proper strategies such that it can learn as many experiences as it needs. Given a 3-layer neural network η with N input neurons, H hidden neurons and 1 output neuron, we can denote the training set as $\boldsymbol{T}=\{(\mathbf{x}_i, t_i), i=1:P\}$, where $\mathbf{x}_i$ is a column vector of the ith input vector, t_i is the expected output for the ith input vector, and P is the size of training set. The target of this training process is going to minimize an error function or metric using this training set and the corresponding weight matrix in the neural network. In this work, the error function is chosen as the mean square error defined in Equation (4-4) because it is widely used in many applications and there are many existing training algorithms for minimizing this error function. In Equation (4-4), $\boldsymbol{W} = [w_1\ w_2\ \ldots\ w_Q]^T$ consists of all weights including biases of the network, y_i is the output value of the ith input vector and Q is the number of weights.

$$F(\boldsymbol{W}) = \sum_{i=1}^{p} (y_i - t_i)^2 \tag{4-4}$$

There are many training algorithms for feedforward neural networks that can select suitable weights to minimize the error function in Equation (4-4). Some methods such as steepest descent algorithms, conjugate gradients algorithms and quasi-Newton algorithms [84] are general optimization methods. In this work, we choose Levenberg-Marquardt algorithm [84~86] to train our neural power models because it is very suitable to minimize the error functions that arise from a squared error criterion.

Typically, the training process will minimize the user-specified error function iteratively until the neural network can satisfy user-specified error criterion. When this criterion is satisfied, the neural network is considered as having learned the behavior between the input data and the expected output values. In some cases, it is very possible that the learning capability of this neural network is not enough to learn the required behavior such that the training process cannot satisfy the stop criterion even the network has been trained for many iterations. In such cases, we have to stop the training process and enhance the learning capability of the neural network such as increasing its hidden neurons.

In some cases, the neural network is probably over fitting the training set thus producing a neural network that has poor generalization performance. Typically, an extra validation set $\boldsymbol{V}=\{(\boldsymbol{x}_i, t_i), i=1:M\}$ will be used to prevent this situation. A validation set is similar to a training set, but its size M is larger than the size of training set P. In general, it is stochastically independent of the training set but has the same distribution. This set is used to determine when to stop the training process according to the value of a user-defined validation error function $F_v(\boldsymbol{V}, \boldsymbol{W})$. In each training iteration, say the r^{th} iteration, we will hold the weight matrix $\boldsymbol{W}_r$ and calculate the value of $F_v(\boldsymbol{V}, \boldsymbol{W_r})$, which is also called the validation error. When the validation error starts to increase gradually but the value of error function is still decreasing, it is considered as over fitting. At that moment, the training process should be stopped and the complexity of this neural network may have to be reduced. In this work, we start from a small neural network architecture and increase the complexity of the neural network until it can satisfy the error requirement. According to our experiences in those benchmark circuits, this strategy seldom results in over fitting. The details of this strategy will be described in Section 4.3 with several preliminary experimental results.

Since those training algorithms have been extensively discussed in neural network researches with many good solutions, the most important problems for us about the training process are designing a good training set and setting a good stop criterion such that the trained neural network can be applied to most cases in the input space. Because it is hard to train a neural network with the entire input space in many applications, the size and distribution of the training set and the stop criterion will have great influence on the accuracy of the trained neural network. We will discuss the details of this problem and our strategies in

Section 4.3.

4.2.3 Evaluating the Accuracy of a Trained Neural Network

In order to evaluate the accuracy of a trained neural network, we also need a test set that is independent to the training set and the validation set if it is used. We denote the test set as $\mathbf{Z}=\{(\mathbf{x}_i, t_i), i=1:K\}$, where $\mathbf{x}_i$ and t_i are the same as defined in the training set and K is the size of this test set. The output of neural network for input $\mathbf{x}_i$ is denoted as y_i. In order to verify our power model can be used on a wide distribution on the input space, the test set is composed of many short test sequences. Those short test sequences are denoted as $\mathbf{S_j}=\{(\mathbf{x}_i, t_i), i=1:K_j\}$, where $j=1:L$ and $K_1+K_2+\ldots+K_L=K$, which are widely distributed in the input space. The details of our strategies to generate those test sequences will be discussed in Section 4.3.

Because our neural power models are used to estimate the average power consumption of circuits under a specific input sequence, we will use typical evaluation criterion for power estimation instead of traditional methods in neural networks to evaluate the quality of our power model. We define the error in estimating the power consumption of the j^{th} test sequence (ESP_j) as Equation (4-5). The average error $AESP$, which is the average of ESP_j and the maximum error $MAXESP$, which is the maximum value of ESP_j, are defined as Equation (4-6) and (4-7). The root mean square error ($RMSESP$) and standard deviation errors ($STDESP$) of those test sequences are defined as Equation (4-8) and (4-9) to show the distribution of the estimation errors. Those metrics will be used to evaluate the quality of our neural power models in the following experiments.

$$ESP_j = \frac{\sum_{i=1}^{K_j} y_i - \sum_{i=1}^{K_j} t_i}{\sum_{i=1}^{K_j} t_i} \tag{4-5}$$

$$AESP = \frac{1}{L}\sum_{j=1}^{L}\left|ESP_j\right| \tag{4-6}$$

$$MAXESP = Max\left(ESP_j\right) \quad where\ j=1\ldots L \tag{4-7}$$

$$RMSESP = \left(\frac{1}{L}\sum_{i=1}^{L}\left(ESP_i\right)^2\right)^{1/2} \tag{4-8}$$

$$STDESP = \left(\frac{1}{L}\sum_{i=1}^{L}\left(ESP_i - \frac{1}{L}\sum_{j=1}^{L}ESP_j\right)^2\right)^{1/2} \tag{4-9}$$

4.3 Power Model Construction with Neural Networks

Since neural networks have been used for many years in other areas, determination of those parameters in neural networks is still mostly done by heuristic approaches. In typical experience, these parameter determinations are application-oriented problems. Different applications might have different suitable parameters of modeling construction. Therefore, the primary goal in this work is trying to build a systematic procedure to build the power models using neural networks. The best parameters in the neural networks might still be different from circuit to circuit.

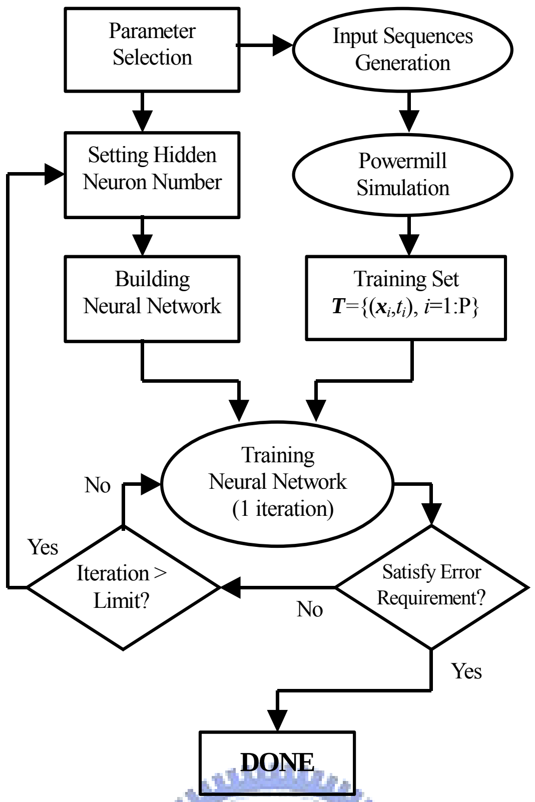

Parameter Selection
Input Sequences Generation
Setting Hidden Neuron Number
Powermill Simulation
Building Neural Network
Training Set $\boldsymbol{T}=\{(\boldsymbol{x}_i,t_i),\ i=1{:}P\}$
Training Neural Network (1 iteration)
No
Yes
Iteration > Limit?
Satisfy Error Requirement?
No
Yes
DONE

Figure 4-4. The workflow of building a neural power model

The overall construction procedure of the proposed neural power model is illustrated in Figure 4-4. This procedure consists of three major steps: building a neural network, generating training sets, and training the neural network. In order to make good decisions at each step, we will use several simple experiments to explain the decision strategies in the following discussions.

Because there are a lot of good training methods for neural networks, we will directly use them and focus our discussions in the first two phases in the following sections. The only two things that have to be decided are the target error and the maximum number of iterations in the training process. Because our target is to use the neural power model to estimate the average power of a test sequence, we decide to use the mean square error (MSE) as the validation error function. During the training phase, we will hold the temporal weight matrix after each iteration and estimate the validation error according to this weight matrix. If the validation error is smaller than 0.0036, which can roughly imply that the estimation error using the validation set is near 6%, we will stop the training process. Otherwise, we will train the neural network again using the same training set. According to our experiences, the validation errors are often saturated after 15 training iterations for those benchmark circuits. Therefore, we set the upper bound of training iterations as 15. If the stop criterion is still not satisfied after 15 iterations, we will add more hidden neurons and train the new neural network again using the same training set.

4.3.1 Building Neural Network

As described in Section 4.2, we decide to use the 3-layer fully connected feedforward neural network structure and the Levenberg-Marquardt training algorithm with the mean square error function in our power model. In the first step, we have to decide the input data type of this neural network, the number of hidden neurons and the transfer function of internal neurons. In typical experiences, the best decisions might be different in different cases, which are hard to be theoretically analyzed. Therefore, we will use a simple experiment on the circuit C1355, which is arbitrarily chosen in ISCAS'85 benchmark circuits, to explain our decisions for those parameters. Because it is not feasible to show all detailed analysis for each circuit, we will try to verify the feasibility of our approach with complete benchmark set in Section 4.4 by using the metrics defined in Section 4.3.

A. Input Data Type and Transfer Function

In this work, the input data type of neural networks and the transfer function of internal neurons are decided together because the most suitable transfer function depends on the behavior between the input data and the output values of the training set. If the relationship between input data and output values has non-linear characteristics, the accuracy of the trained neural networks would be lost when using a linear transfer function for internal neurons because it is similar to use a piece-wise linear curve to fit a non-linear curve as illustrated in Figure 4-1.

Since we are building a high-level power model, the input data of this neural power model can only use primary input and output information of circuits. The most straightforward idea is to use the complete 4 states of bit transitions ($0\rightarrow0$, $0\rightarrow1$, $1\rightarrow0$, $1\rightarrow1$) at the input and output pins of circuits as the input data of the neural networks because the effects of both state-dependent leakage power and transition-dependent switching power can be considered. If we use such bit-level transition data (bit-level statistics) to be the input data of our neural power model, the number of neurons in the input layer will be fixed as $4*(n+m)$ for a circuit with n inputs and m outputs, which are $TI_{i,00}$, $TI_{i,01}$, $TI_{i,10}$, $TI_{i,11}$, $TO_{j,00}$, $TO_{j,01}$, $TO_{j,10}$ and $TO_{j,11}$. Here, $TI_{i,xy}=1$ represents that the i^{th} input pin changes from logic state x to y and $TO_{j,xy}=1$ represents that j^{th} output bit changes from logic state x to y in a pattern pair. Instead of using bit-level statistics, we could use the word-level statistics as the input data of our neural power model. If we consider the input statistics and the output statistics separately, the number of neurons in the input layer will be fixed as 8, which are TI_{00}, TI_{01}, TI_{10}, TI_{11}, TO_{00}, TO_{01}, TO_{10} and TO_{11}. Here, TI_{xy} represents the ratio of input signals change from logic state x to y in a input pattern pair and TO_{xy} represents the ratio of output signals change from logic state x to y in a output pattern pair. For example, given a circuit with 10 inputs and 10 outputs, we assume that its corresponding output signals will change from 0101101100 to 0110110111 when the input signals change from 0001110101 to 1010101011. For this pattern pair, both the bit-level and word-level statistics are shown in the 4th to 7th rows in Figure 4-5(a) and Figure 4-5(c) respectively. According to the definitions, their input data will be formed as shown in Figure 4-5(b) and Figure 4-5(d).

Input vectors

Bit i	0	1	2	3	4	5	6	7	8	9
Pattern$_t$	0	0	0	1	1	1	0	1	0	1
Pattern$_{t+1}$	1	0	1	0	1	0	1	0	1	1
$TI_{i,00}$	0	1	0	0	0	0	0	0	0	0
$TI_{i,01}$	1	0	1	0	0	0	1	0	1	0
$TI_{i,10}$	0	0	0	1	0	1	0	1	0	0
$TI_{i,11}$	0	0	0	0	1	0	0	0	0	1

Corresponding output vectors

Bit j	0	1	2	3	4	5	6	7	8	9
Pattern$_t$	0	1	0	1	1	0	1	1	0	0
Pattern$_{t+1}$	0	1	1	0	1	1	0	1	1	1
$TO_{j,00}$	1	0	0	0	0	0	0	0	0	0
$TO_{j,01}$	0	0	1	0	0	1	0	0	1	1
$TO_{j,10}$	0	0	0	1	0	0	1	0	0	0
$TO_{j,11}$	0	1	0	0	1	0	0	1	0	0

Figure 4-5(a). An example of the bit-level statistics characterization

total length=40 *bit-level statistics*

Input vector part					Corresponding output vector part				
$TI_{0,00}$	$TI_{0,01}$	. . .	$TI_{9,10}$	$TI_{9,11}$	$TO_{0,00}$	$TI_{0,01}$	. . .	$TI_{9,10}$	$TI_{9,11}$
0	1	…	0	1	1	0	…	0	0

Figure 4-5(b). Input data format with bit-level statistics

Input vectors

Bit i	0	1	2	3	4	5	6	7	8	9	
Pattern$_t$	0	0	0	1	1	1	0	1	0	1	
Pattern$_{t+1}$	1	0	1	0	1	0	1	0	1	1	
TI_{00}		√									1/10
TI_{01}	√		√				√		√		4/10
TI_{10}				√		√		√			3/10
TI_{11}					√					√	2/10

Corresponding output vectors

Bit j	0	1	2	3	4	5	6	7	8	9	
Pattern$_t$	0	1	0	1	1	0	1	1	0	0	
Pattern$_{t+1}$	0	1	1	0	1	1	0	1	1	1	
TO_{00}	√										1/10
TO_{01}			√			√			√	√	4/10
TO_{10}				√			√				2/10
TO_{11}		√			√			√			3/10

Figure 4-5(c). An example of the word-level statistics characterization

total length=8 (fixed) *word-level statistics*

Input vector part				Corresponding output vector part			
TI_{00}	TI_{01}	TI_{10}	TI_{11}	TO_{00}	TO_{01}	TO_{10}	TO_{11}
0.1	0.4	0.3	0.2	0.1	0.4	0.2	0.3

Figure 4-5(d). Input data format with word-level statistics

If we use bit-level statistics to be the input data of our neural power model, the neural network may recognize the individual contribution to the total power consumption from each input transition such that the estimation error of the power model is more possible to be reduced. However, the size of this power model will increase very fast especially for the circuits with large amount of I/O pins. Moreover, because the complexity of bit-level statistics is too high, it will become much harder to learn such complex relationship between the bit-level statistics and the power consumptions for a neural network. If we use word-level statistics as the input data of neural power model, some individual characteristics of each possible input transition may be lost, especially for the control-dominated circuits with significantly different power modes. However, it is a common heuristic method used in many power models [46][65,66][73] to reduce the modeling complexity with reasonable error loss. According to their experimental results, the induced errors are indeed in an acceptable range in most cases.

When we are selecting the transfer function, we also have to consider the output format of our neural power model. The output of our neural power model is expected to represent the estimated power consumption of pattern pairs. Because the values of power consumptions often continuously distribute on a wide range, those transfer functions that use discrete values, such as the unit-step or the sign functions are not suitable. In our observations, the three commonly used functions, logarithmic sigmoid (*logsig*), hyperbolic tangent sigmoid (*tansig*) and linear (*linear*) functions, are more suitable for our application, which are defined in Equation (4-10), (4-11) and (4-12) respectively. However, it should be noted that the values of power consumption have to be normalized between 0 to 1 in both the training set and the test set if logarithmic sigmoid and hyperbolic tangent sigmoid functions are used in the output neuron. In order to save the normalization effort, we use the *linear* function as the transfer function of the output neuron. In the following discussions, we will focus on the comparison between those three transfer functions for hidden neurons.

logsig transfer function: $$f(s)=\frac{1}{1+e^{-s}} \tag{4-10}$$

tansig transfer function: $$f(s)=\frac{2}{1+e^{-2s}}-1 \tag{4-11}$$

linear transfer function: $$f(s)=s \tag{4-12}$$

In order to help us making better decisions, we perform an experiment to compare the accuracy and performance of 6 combinations between 2 input data types (bit-level and word-level statistics of the input and output pattern pairs) and 3 transfer functions (*logsig, tansig, linear*). Because this experiment is only used for evaluating the input data types and suitable transfer functions, many parameters in the neural network are arbitrarily chosen and fixed in this experiment. All the training process will be stopped after 15 iterations instead of using the validation error checking as the stop criterion. The number of hidden neurons is fixed as 4. The training set includes 20 sequences, which are uniformly distributed over the population of the average signal probability (P_{in}) and the average signal transition density (D_{in}) and each sequence includes 1,000 random input pattern pairs with the chosen *PD* combination. The comparison using *AESP* and *STDESP* for those test cases on C1355 is shown in Table 4-1.

The neural power model using bit-level statistics has higher complexity than that of the neural power model using word-level statistics, which can be observed in the number of weight |**W**| and the constructing time. However, as shown in the experimental results, the neural power model using bit-level statistics does not have many improvements in terms of

AESP and *STDESP*. According to the analysis above, we select word-level statistics as the input data of our neural power model. While checking the neural power model using word-level statistics, we can find that the neural power model using *tansig* function as the transfer function of hidden neurons provides better results on *AESP* and *STDESP*. Therefore we select *tansig* function as the transfer function of internal hidden neurons in this work.

Table 4-1. Comparison of input data types vs. transfer functions

Circuits	Input Data & |**W**|	*f(s)*	Average Power AESP (%)	Average Power STDESP (%)	Construction Time (sec)
C1355 PI=41 PO=32	Bit-Level |**W**|=1177	***linear***	4.40	6.32	443.53
		logsig	4.92	4.68	704.73
		tansig	4.38	7.80	555.81
	Word-Level |**W**|=41	***linear***	5.09	7.19	54.17
		logsig	4.71	4.16	55.83
		tansig	3.81	2.81	57.17

B. Number of Hidden Neurons

Another issue to be decided is the number of hidden neurons required in the neural power model. Typically, the minimal number of hidden neurons depends on the complexity of the relationship between the input data and output values in the training set. However, according to the experience in neural network researches, there is no easy or general way to determine the optimal solution for the number of hidden neurons to be used [87]. As mentioned in Section 4.2, our strategy is initially using a neural network with a small number of hidden neurons and increasing the hidden neurons until the stop criterion has satisfied. In the following discussions, we will explain the reason of using this strategy through a simple experiment.

We first build a neural power model for C1355 and set the initial number of hidden neurons as 2. The training set is the same one as used in the experiment of Section 4.3.1.A. The validation set includes 20 sequences with the same *PD* distribution as in the training set, but and the size of each sequence is increased to 3,000. In the following experiment, we increase the number of hidden neurons from 2 to 15 and test the accuracy of the neural networks after 15 training iterations using the same test set as used in the experiment of Section 4.3.1.A. The experimental results about the effects of number of hidden neurons are shown in Table 4-2.

Table 4-2. The effects of the number of hidden neurons

Circuits	Hidden Neurons	Average Power		Validation Error	Construction Time (sec)
		AESP (%)	STDESP (%)		
C1355 PI=41 PO=32	2	4.96	3.57	0.004264	50.51
	3	4.92	3.56	0.004255	53.25
	4	3.81	2.81	0.004098	57.17
	5	3.93	3.06	0.004272	59.78
	6	5.67	3.21	0.004550	62.97
	7	3.25	3.38	0.004252	66.41
	8	3.62	3.30	0.004205	71.49
	9	3.20	2.24	0.004011	73.14
	10	2.96	2.64	0.003969	78.99
	11	3.79	2.89	0.004406	81.83
	12	3.72	2.65	0.004357	88.52
	13	3.36	2.81	0.004675	90.28
	14	3.26	2.41	0.004179	96.56
	15	3.31	2.59	0.004163	99.76

According to the results in Table 4-2, increasing the number of hidden neurons does improve both *AESP* and *STDESP* when the number is small. However, when the number of hidden neurons is larger than 10, we could find that the validation error, *AESP*, and *STDESP* may become worse due to the over fitting problem mentioned in Section 4.2.2. Therefore, for this case, the best choice is to set the number of hidden neurons as 10.

4.3.2 Design of Training Sets

Typically, a power model is expected to be used for different test sets with various input distributions. In order to achieve this target, the neural power model should be trained over a wide range in the input space such that it can learn enough experiences. *Therefore, we will randomly decide the P_{in} and D_{in} while generating each test sequence.* Because an input signal is assumed to make at most one single transition per cycle, there is a relationship between P_{in} and D_{in} as shown in Equation (4-13), whose detailed proof can be found in [65]. Therefore, while generating those training sets over a wide range of P_{in} and D_{in} distribution, we can use only the *PD* combinations that satisfy Equation (4-13) such that neural networks could learn the correct characteristics between the input signal statistics and the power consumption of circuits.

$$\frac{D_{in}}{2} \le P_{in} \le 1 - \frac{D_{in}}{2} \tag{4-13}$$

The size of a training set is also an important issue while training the neural power model. According to the related study [88], it suggested to determine the size of training set according to Equation (4-14), in which P is the number of samples, $|\mathbf{W}|$ is the number of weights to be adjusted and a is the expected accuracy. In this work, our target is set as $a \ge$ 95%. According to this error requirement, we suggest generate the training set with size $P >> 20\,|\mathbf{W}|$. A larger training set is supposed to produce a more accurate neural power model. However, the characterization time of this power model is also increased. In the following experiment, we will show the observation of the relationship between the size of training set and the modeling accuracy.

$$P > \frac{|\mathbf{W}|}{1-a} \tag{4-14}$$

In this experiment, we use the best neural network structure for C1355 decided in Section 4.3.1, which is a neural network that uses 10 hidden neurons and word-level statistics. As mentioned above, the samples of the training sets should be distributed over the space with a wide range of P_{in} and D_{in}. Therefore, we generate 4 training sets consisting of 20 sequences in each, which have the same uniformly distribution on the input space that satisfies the P_{in} and D_{in} constrains in Equation (4-13). However, the length of each sequence is different, which are 500, 1,000, 2,000 and 5,000 pattern pairs respectively. In other words, the total sizes of the training sets are 10,000, 20,000, 40,000 and 100,000 respectively. The validation sets consist of 20 sequences that have the same distribution as those test sequences but their sequence lengths are multiplied by 3. Therefore, the sizes of validation sets are 30,000, 60,000, 120,000 and 300,000 respectively. The experimental results of those 4 neural networks under different training conditions after 15 training iterations, which are evaluated using the same test set that consists of 20 sequences with 3,000 pattern pairs are shown in Table 4-3.

Table 4-3. The effects of the size of training set

Circuit	**C1355**			
Size of Training Sets	Average Power		Validation Error	Construction Time (sec)
	AESP (%)	STDESP (%)		
10,000	4.05	2.20	0.003952	59.47
20,000	2.63	2.43	0.003804	99.67
40,000	2.05	1.99	0.003510	114.21
100,000	3.67	2.86	0.004203	233.95

The experimental results show that the size of training set will not affect the accuracy too much on accuracy if the training set is large enough. According to this observation, we generate 20 sequences with 3,000 pattern pairs in each sequence to be the training set of our neural power model in the following experiments to make a trade off between the size of test sequences and accuracy. Of course, those 20 sequences will have a distribution that covers a wide range of the input space.

4.4 Experimental Results

In this section, we will demonstrate the accuracy and efficiency of our power model with ISCAS'85 benchmark circuits and one real design, a combinational divider with 32-bit dividend/quotient and 12-bit divisor/remainder. All circuits in our experiments were synthesized by using 0.35 μm cell library. The accuracy will also be compared with traditional 3D-LUT power modeling methodology, which uses P_{in}, D_{in} and average output signal transition density (D_{out}) as its three dimensions, and the interval size of each dimension is set to 0.1. Our neural power models including the training algorithms were all constructed on MATLAB by using an Intel Pentium III 1GHz mobile CPU and 384M RAM.

In the model construction phase, the input training sequences are generated over a wide range of input distribution as described in Section 4.3.2. The real power of those input sequences is simulated by a transistor-level simulator, PowerMill such that the measured power consumptions can include switching power and leakage power and can be characterized in the power model. In order to show that the power models can be used for various input distribution, we test those models by using 200 test sequences with 3,000 pattern pairs. Each sequence has different P_{in} and D_{in} that are randomly selected over a wide range satisfies the condition in Equation (4-13). After simulation, the estimated average power consumption with this power model is also compared to the simulation results from PowerMill.

All the test circuits will be tested using the two power estimation approaches with the same information: traditional 3D-LUT power model and our neural power model. The same training and test sequences will be used for both approaches to make a fair comparison. The performances of both power models are summarized in Table 4-4. The construction time of neural power models includes the data pre-loading time of training and validation sets, the establishing time of neural network and the elapsed time of network training process. The simulation time of transistor-level simulation is not included.

According to Table 4-4, the average values of AESP and STDESP are 17.58% and 18.59% respectively while we use the traditional 3D-LUT power models. The convergences of this approach are quite poor that can be observed from the large values of STDESP. It implies that using the LUT-based power model may have large errors in some cases. Compared to the traditional 3D-LUT power model, we only have 4.72% error for all cases on average, and the largest *AESP* is only 8.93% for the 32-bit divider. The improvement of our neural power model can be shown in the *STDESP*. The largest *STDESP* is only 5.88% for the 32-bit divider using our approach, which shows a good agreement with real powers. The combined scatter plots of all ISCAS'85 circuits by using our approach and the 3D-LUT approach are shown in Figure 4-6 and Figure 4-7 respectively. In order to examine all circuits on the same plot, the power consumptions of all circuits are normalized with the circuit size and operating frequency. Comparing the two plots, we can see that our approach can really provide better trend of estimation accuracy.

Table 4-4. The comparison between traditional 3D LUT power model and our neural power model

Circuits		C432	C499	C880	C1355	C1908	C2670	C3540	C5315	C6288	C7552	Divider	|Average|
I/O Pin Number	Input	36	41	60	41	33	233	50	178	32	207	44	-
	Output	7	32	26	32	25	140	22	123	32	108	45	-
Gate Count		116	324	245	362	328	447	700	1102	1640	1135	2726	-
3D LUT Power Model	# of table entries	500											
	MAXESP (%)	77.63	27.66	68.39	36.71	71.24	65.76	99.05	44.46	88.71	71.08	43.94	63.28
	AESP (%)	18.56	11.72	22.81	11.19	16.81	14.14	24.41	16.60	29.57	15.00	12.46	17.58
	STDESP (%)	17.21	9.09	23.92	9.72	16.32	17.50	24.38	14.08	35.37	18.64	18.07	18.59
	RMSESP (%)	24.22	14.37	30.39	14.16	22.82	21.86	32.41	20.34	39.23	22.51	18.44	23.72
Our Neural Power Model	Neurons in Hidden Layer	8	6	8	8	7	9	9	7	8	8	8	7.82
	|W|	81	61	81	81	71	91	91	71	81	81	81	79.18
	Construction Time (sec)	313.07	216.28	310.83	323.51	262.48	390.29	391.53	267.34	320.19	324.29	325.23	313.19
	Training Iterations	102	36	101	105	90	120	120	88	105	105	105	97.91
	MAXESP (%)	15.13	-11.19	32.72	11.83	22.09	24.74	24.14	-17.28	16.34	-17.49	26.63	20.06
	AESP (%)	3.36	2.92	5.34	2.10	5.29	5.72	4.57	3.83	4.14	5.47	8.93	4.72
	STDESP (%)	3.11	2.31	5.07	1.66	4.45	6.07	3.15	3.99	3.47	2.82	5.88	3.85
	RMSESP (%)	3.80	3.72	7.39	3.82	6.81	6.60	7.68	3.91	3.95	7.28	9.13	5.85

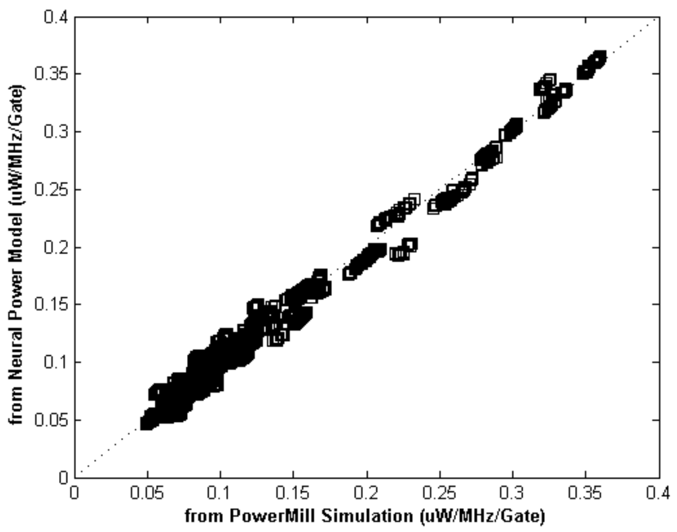


Figure 4-6. Scatter plot of neural power model estimation versus PowerMill simulation in ISCAS'85 benchmarks

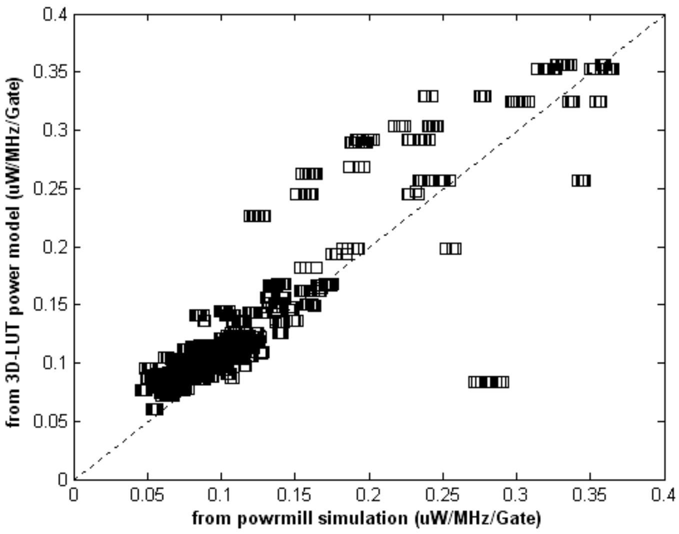


Figure 4-7. Scatter plot of 3D-LUT estimation versus PowerMill simulation in ISCAS'85 benchmarks

The storage requirements are also much less in our approach. According to the results shown in Table 4-4, the maximum number of hidden neurons is 9 in our experiments. It means that we only use up to 9 hidden neurons structure with 91 elements in the weight matrix |W| to record the power characteristics, which is quite small compared to the lookup tables, which require 500 (=10*10*10/2) numbers to record the tables. These experimental results have also shown that the complexity of our neural power model has almost no relationship with circuit size and number of inputs and outputs. Even for large circuits such as the 32-bit divider, the complexity of its power model is still the same as the complexity of smaller circuits such as C432. Besides, the construction of neural power model is rapid that can be observed from the short construction time and the total training iterations of each neural power model in Table 4-4. Therefore, using such a power model can be very efficient even for complex circuits and also has high accuracy.

Another important information not shown in Table 4-4 is the estimation time while using our power model. Actually, the estimation time of our neural power model is dominated by the functional simulation time with a logic simulator, which simulates the circuits with specific input vectors to obtain the corresponding output vectors. If we assume that the corresponding output values under specific input sequences are also provided by users, the estimation time of our neural power model is always less than one second for all ISCAS'85 circuits. Therefore, the estimation time is not shown in Table 4-4 because it is almost equal to the logic simulation time, which is quite small compared with low-level power estimation methods such as PowerMill.

In order to demonstrate that the neural power model can handle specific functional patterns in practical use, we also test the practical design, the 32-bit divider design, with user-given functional patterns. The functional sequence consists of 1,000 pattern-pairs only. However, the average estimation error is only 5.98% compared to the PowerMill results.

4.5 Summary

In this work, we propose a novel power model for complex digital circuits, which uses neural networks to learn the power characteristics during simulation including both leakage power and switching power. Unlike the power characterization process in traditional approaches, our characterization process is very simple and straightforward. The complexity of our neural power model is also smaller than that of the traditional 3D LUT power model which is almost no relationship with circuit size and the number of inputs and outputs. More importantly, using the neural power model for power estimation does not require any detailed circuit information of the circuits, which is very suitable for IP protection. In this work, we have tested our neural power model on all ISCAS'85 benchmark circuits and one real design. The experimental results demonstrate that our neural power model can accurately estimate the power consumption of combinational circuit for different test sequences with wide range of input distributions.

Chapter 5

Conclusions and Future Works

In this dissertation, we proposed a series of power estimation and power modeling methods for combinational IPs. Because IP vendors may release only limited design information to protect their knowledge, we proposed corresponding methods for the designs with different information. We divide the design information into three levels. The first level is transistor-level information. The second level is gate-level information. The third level is functional-level information.

In Chapter 2, we proposed two consecutive sampling techniques to improve the losing of performance in those vector compaction methods with random sampling techniques. According to the experimental results on ISCAS'85 benchmark circuits, the speedups of random sampling, single-sequence sampling and multi-sequence sampling approaches are 94.21, 119.08 and 147.03 respectively. The multi-sequence approach improves 56% on speed compared to random sampling approach. The single-sequence approach only improves 26% on speed. The average compaction ratio achieved in random sampling approach is 129.22 and the average error is 4.14%. The average compaction ratio achieved in single-sequence approach is 179.41 and the average error is 3.32%. The average compaction ratio achieved in multi-sequence approach is 251.04 and the average error is 4.19%. It shows that the multi-sequence approach can dramatically reduce the useless transitions in the random sampling method such that it can almost keep the desired compaction ratio exactly.

In Chapter 3, we proposed a power model for IPs with only gate-level design information. We build a lookup table for each IP that maps the zero-delay charging and discharging capacitance during an input pattern transition to an estimative value of real power consumption. The experimental results on ISCAS'85 benchmark circuits show that the table sizes are only 42 to 107 for ISCAS'85 benchmark circuits. It is very small and almost independent to the circuit size. We test the accuracy of our method by estimating the average power consumption of circuits with 3 different sequences, which are pseudo random sequence, counter sequence (up/down counter) and LFSR (Linear Feedback Shift Register) sequence with 50,000 pattern pairs respectively. The maximum error is 7.48% for C2670 with LFSR sequence. The overall average error is only 2.99%. The experimental results show that our power model still has high accuracy for different input sequences.

In Chapter 4, we proposed a quite different approach for high-level power modeling of complex digital circuits that uses a 3-layer fully connected feedforward neural network to learn the power characteristics during simulation without any lookup tables. According to the experimental results on ISCAS'85 benchmark circuits and a 32-bit divider circuit, the average values of AESP and STDESP are 17.58% and 18.59% respectively while we use the traditional 3D-LUT power models. Our proposed neural network power model has 4.72% error for all cases on average, and the largest *AESP* is only 8.93% for the 32-bit divider. The improvement of our neural power model can be shown in the *STDESP*. The largest *STDESP* is only 5.88% for the 32-bit divider using our approach, which shows a good agreement with real powers. Our experimental results have also shown that the complexity of our neural power model has almost no relationship with circuit size and number of inputs and outputs.

Besides, the construction of neural power model is rapid and it can be observed from the short construction time and small number of the total training iterations of each neural power model in our experimental results. Therefore, using such a power model can be very efficient even for complex circuits to achieve high accuracy.

In the future, we have to extend our experiments on sequential circuits and asynchronous input signals. We will also try to integrate the neural network power model into HDL simulator such that the power estimation could be done in only one simulator. We also have to think about how our methods could be applied on memory block. If those issues can be solved, the power estimation flow for a system will be more complete.

Reference

[1] C. Small, "Shrinking Devices Put the Squeeze on System Packaging," EDN, Vol. 39, No. 4, pp. 41-46, 1994.

[2] S. Wang, "Fundamentals of Semiconductor Theory and Devices Physics," Prentice Hall, 1989.

[3] L. Benini and G.D. Micheli, "System-level Power Optimization: Techniques and Tools," in Proceeding of International Symposium on Low Power Electronics and Design, pp. 288-293, 1999.

[4] H. Yu and J.D. Cho, "Low-power Design and Architecture," IEEE Potentials, Vol. 20, Issue 3, pp. 18-22, 2001.

[5] F.N. Najm, "Low-power Design Methodology: Power Estimation and Optimization," in Proceeding of the 34th Design Automation Conference, pp. 76-81, 1997.

[6] E. De Angel and E.E. Swartzlander, Jr, "Survey of Low Power Techniques for VLSI Design," in Proceeding of the Eighth Annual IEEE International Conference on Innovative Systems in Silicon, pp. 159-169, 1996.

[7] R. Mehra and S. Malik, "Low Power Digital System Design: An Overview," in Proceeding of IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control, Vol. 1, pp. 118-123, 1998.

[8] E. Macii, M. Pedram and F. Somenzi, "High-Level Power Modeling Estimation, and Optimization," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 17, Issue 11, pp. 1061-1079, 1998.

[9] F. N. Najm, “A Survey of Power Estimation Techniques in VLSI Circuits,” IEEE Transactions on VLSI Systems, Vol. 2, Issue 4, pp. 446-455, 1994.

[10] F.N. Najm, “Power Estimation Techniques for Integrated Circuits,” in Proceeding of IEEE/ACM International Conference on Computer-Aided Design, pp. 492-499, 1999.

[11] F. N. Najm, “Simulation and Modeling – Estimating Power Dissipation in VLSI Circuits,” IEEE Circuit and Device Magazine, Vol. 10, Issue 4, pp. 11-19, 1994.

[12] P. Landman, “High-Level Power Estimation,” in Proceeding of International Symposium on Low Power Electronics and Design, pp. 29-35, 1996.

[13] A.C. Deng, “Power Analysis for CMOS/BiCMOS Circuits,” in Proceeding of 1994 International Workshop on Low Power Design, pp. 3-8, 1994.

[14] X. Huang, B. Zhang, C. Deng, and B. Swirski, “The Design and Implementation of PowerMill,” in Proceeding of 1995 International Workshop on Low Power Design, pp. 105-109, 1995.

[15] A. Salz and M. A. Horowitz, “IRSIM: An Incremental MOS Switch-level Simulator,” in Proceeding of 26th ACM/IEEE Design Automation Conference, pp. 173-178, 1989.

[16] A. Bogiolo, L. Benini and B. Ricco, “Power Estimation of Cell-Based CMOS Circuits,” in Proceeding of ACM/IEEE Design Automation Conference, pp. 433-438, 1996.

[17] C.V. Schimpfle, S. Simon and J.A. Nossek, “High-level Circuit Modeling for Power Estimation,” in Proceeding of The 6th IEEE International Conference on

Electronics, Circuits and Systems, Vol. 2, pp. 807 – 810, 1999.

[18] P. Israsena, and S. Summerfield, "Novel Pattern-based Power Estimation Tool with Accurate Glitch Modeling," in Proceeding of IEEE International Symposium on Circuits and Systems, Vol. 4, pp. 721 – 724, 2000.

[19] W.Z. Shen, J.Y. Lin and J.M. Lu, "CB-Power: a Hierarchical Cell-based Power Characterization and Estimation Environment for Static CMOS Circuits," in Proceeding of the Asia and South Pacific Design Automation Conference, pp. 189 – 194, 1997.

[20] W.W. Bachmann and S.A. Huss, "Efficient Algorithms for Multilevel Power Estimation of VLSI Circuits," IEEE Transaction on VLSI Systems, Vol. 13, Issue 2, pp. 238-254, 2005.

[21] J.Y. Lin, W.Z. Shen and J.Y. Jou, "A Structure-oriented Power Modeling Technique for Macrocells," IEEE Transactions on VLSI Systems, Vol. 7, Issue 3, pp. 380 – 391, 1999.

[22] M. Y. Sum, S.Y. Huang, C.C. Weng, K.S. Chang, "Accurate RT-level Power Estimation Using Up-down Encoding," in Proceeding of Asia-Pacific Conference on Circuits and Systems, pp. 69-72, 2004.

[23] C.Y. Tsui, R. Marculescu, D. Marculescu and M. Pedram, "Improving the Efficiency of Power Simulators by Input Vector Compaction," in Proceeding of Design Automation Conference, pp. 165-168, 1996.

[24] C.Y. Hsu, C.W. W. and W.Z. Shen, "A Pattern Compaction Technique for Power Estimation Based on Power Sensitivity Information," in Proceeding of International

Symposium on Circuits and Systems, pp. 467-470, 2001.

[25] A. Pina and C.L. Liu, "Power Invariant Vector Sequence Compaction," in Proceeding of International Conference on Computer-Aided Design, pp. 473-476, 1998.

[26] R. Marculescu, D. Marculescu and M. Pedram, "Sequence Compaction for Power Estimation:Theory and Pratice," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 18, Issue 7, pp. 973-993, 1999.

[27] H.L. Huang, Y.R. Chen, J.Y. Jou and W.Z. Shen, "Grouped Input Power Sensitive Transition – An Input Sequence Compaction Technique for Power Estimation," in Proceeding of International Symposium on Circuits and Systems, pp. 471-474, 2001.

[28] N. Dragone, R. Zafalon, C. Guardiani and C. Silvano, "Power Invariant Vector Compaction Based on Bit Clustering and Temporal Partitioning," in Proceeding of International Symposium on Low Power Electronics and Design, pp. 118-120, 1998.

[29] S.Y. Huang, K.C. Chen, K.T. Cheng and T. C. Lee, "Compact Vector Generation for Accurate Power Simulation," in Proceeding of Design Automation Conference, pp. 161-164, 1996.

[30] R. Radjassamy and J.D. Carothers, "Faster Power Estimation of CMOS Designs Using Vector Compaction - a fractal approach," IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 33, Issue 3, pp. 476-488, 2003.

[31] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo Approach for Power Estimation," IEEE Transaction on VLSI System, Vol. 1, Issue 1, pp. 63-71,

1993.

[32] V. Saxena, F. N. Najm, and I. N. Hajj, “Monte-Carlo Approach for Power Estimation in Sequential Circuits,” in Proceeding of European Design and Test Conference, pp. 416-420, 1997.

[33] I. R. Miller, J. E. Freund and R. Johnson, Probability and Statistics for Engineers. Englewood Cliffs, NJ: Prentice-Hall, 1990.

[34] M. H. DeGroot, “Probability and Statistics,” Addison-Wesley, 1984.

[35] Y. S. Chow and H. Teicher, “Probability Theory,” Springer, 1997.

[36] G. S. Fishman, “Monte Carlo: Concepts, Algorithms, and Applications,” Springer, 1995.

[37] C. S. Ding, Q. Wu, C. T. Hsieh, and M. Pedram, “Stratified Random Sampling for Power Estimation,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 17, Issue 6, pp. 465-478, 1998.

[38] L. Benini, G. D. Micheli, E. Macii, M. Poncino and R. Scarsi, “Fast Power Estimation for Deterministic Input Streams,” in Proceeding of International Conference on Computer Aided Design, pp. 494-501, 1997.

[39] C.S. Ding, C.T. Hsieh, and M. Pedram, “Improving Sampling Efficiency for System Level Power Estimation,” in Proceeding of International Symposium on Low Power Electronics and Design, pp. 115-117, 1998.

[40] W.L. Hsu, W.Z. Shen and J.Y. Lin, “A Mixed-level Power Estimator for CMOS Circuit Using Pattern Compaction Techniques,” in Proceeding of Asia-Pacific Conference on Circuits and Systems, pp. 771-774, 1998.

[41] M. Nemani and F.N. Najm, "Towards a High-level Power Estimation Capability," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 15, Issue 6, pp. 588-598, 1996.

[42] D. Marculescu, R. Marculescu, and M. Pedram, "Information Theoretic Measures for Power Analysis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 15, Issue 6, pp. 599-610, 1996.

[43] C.H. Hwang and A.C.-H. Wu, "An Entropy Measure for Power Estimation of Boolean Functions," in Proceeding of Asia and South Pacific Design Automation Conference, pp. 101-106, 1997.

[44] F. Ferrandi, F. Fummi, E. Macii, and M. Poncino, "Power Estimation of Behavioral Descriptions," in Proceeding of Design, Automation and Test in Europe, pp. 762-766, 1998.

[45] M. Nemani and F.N. Najm, "High-level Area and Power Estimation for VLSI Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 18, Issue 6, pp. 697-713, 1999.

[46] I.B. Dhaou, and H. Tenhunen, "Efficient Library Characterization for High-level Power Estimation," IEEE Transactions on VLSI Systems, Vol. 12, Issue 6, 2004.

[47] E. Macii and M. Poncino, "Exact Computation of the Entropy of a Logic Circuit," in Proceeding of the Sixth Great Lakes Symposium on VLSI, pp. 162-167, 1996.

[48] F. N. Najm, "Transition Density: A New Measure of Activity in Digital Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 12, Issue 2, pp. 310-323, 1993.

[49] S. Devadas, K. Keutzer and J. White, “Estimation of Power Dissipation in CMOS Combinational Circuits Using Boolean Function Manipulation,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 11, Issue 3, pp. 373-383, 1992.

[50] H. Mehta, M. Borah, R. M. Owens, M. J. Irwin, “Accurate Estimate of Combinational Circuit Activity,” in Proceeding of ACM/IEEE Design Automation Conference, pp. 618-622, 1995.

[51] D. Kim, and T. Ambler, “Robust Transition Density Estimation by Considering Input/Output Transition Behavior,” in Proceeding of IEEE International Symposium on Circuits and Systems, pp. 403-406, 2001.

[52] R. Marculescu, D. Marculescu and M. Pedram, “Switching Activity Analysis Considering Spatiotemporal Correlations,” in Proceeding of International Conference on Computer Aided Design, pp. 294-299, 1994.

[53] T.L. Chou, K. Roy and S. Prasad, “Estimation of Circuit Activity Considering Signal Correlations and Simultaneous Switching,” in Proceeding of International Conference on Computer Aided Design, pp. 300-303, 1994.

[54] J.M. Jou, S.C. Chen and C.L. Wang, “Fast Delay Dependent Power Estimation of Large Combinational Circuits,” in Proceeding of IEEE International Symposium on Circuits and Systems, pp. 53-56, 1998.

[55] A. Ghosh, S. Devadas, K. Keutzer and J. White, “Estimation of Average Switching Activity in Combinational and Sequential Circuits,” in Proceeding of ACM/IEEE Design Automation Conference, pp. 253-259, 1992.

[56] J. Monterio, S. Devadas, B. Lin, C.Y. Tsui, and M. Pedram, "Exact and Approximate Methods of Switching Activity Estimation in Sequential Logic Circuits," in Proceeding of International Workshop on Lower Power Design, pp.117-122, 1994.

[57] J. Monteiro, S. Devadas and B. Lin, "A Methodology for Efficient Estimation of Switching Activity in Sequential Logic Circuits," in Proceeding of ACM/IEEE Design Automation Conference, pp. 12-17, 1994.

[58] C. Y. Tsui, M. Pedram and A. M. Despain, "Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs," in Proceeding of ACM/IEEE Design Automation Conference, pp. 18-23, 1994.

[59] J. C. Costa, J. C. Monterio and S. Devadas, "Switching Activity Estimation Using Limited Depth Reconvergent Path Analysis," in Proceeding of International Symposium on Low Power Electronics and Design, pp. 184-189, 1997.

[60] D. I. Cheng, M. M. Sadowska and K. T. Cheng, "Speeding Up Power Estimation by Topological Analysis," in Proceeding of IEEE Custom Integrated Circuits Conference, pp. 623-626, 1995.

[61] S. Bhanja and N. Ranganathan, "Switching Activity Estimation of VLSI Circuits Using Bayesian Networks," IEEE Transaction on VLSI Systems, Vol. 11, Issue 4, pp. 558-567, 2003.

[62] Z. Chen, K. Roy, and T. L. Chou, "Power Sensitivity – A New Method to Estimate Power Dissipation Considering Uncertain Specifications of Primary Inputs," in Proceeding of International Conference on Computer Aided Design, pp. 40-44,

1997.

[63] Z. Chen, K. Roy and T.L. Chou, "Efficient Statistical Approach to Estimate Power Considering Uncertain Properties of Primary Inputs," IEEE Transaction on VLSI System, Vol. 6, Issue 3, pp. 484-492, 1998.

[64] Z. Chen, K. Roy, and K.P. Chong, "Estimation of Power Sensitivity in Sequential Circuits with Power Macromodeling Application," in Proceeding of International Conference on Computer Aided Design, pp.468-472, 1998.

[65] S. Gupta and F. N. Najm. "Power Modeling for High-Level Power Estimation," IEEE Transactions on VLSI Systems, Vol 8, Issue 1, pp. 18-29, 2000.

[66] G. Jochens, L. Kruse and W. Nebel, "A New Parameterizable Power Macro-Model for Datapath Components," in Proceeding of European Design and Test Conference, pp. 29-36, 1999.

[67] R. Corgnati, E. Macii and M. Poncino. "Clustered Table-Based Macromodels for RTL Power Estimation," in Proceeding of 9th Great Lakes Symposium on VLSI, pp. 354-357, 1999.

[68] A. Bogliolo, R. Corgnati, E. Macii and M. Poncino. "Parameterized RTL Power Models for Soft Macros," IEEE Transactions on VLSI Systems, Vol. 9, Issue 6, pp. 880-887, Dec. 2001.

[69] Z. Chen, K. Roy and E. K. Chong. "Estimation of Power Dissipation Using a Novel Power Macromodeling Technique," IEEE Transactions on CAD, Vol. 19, Issue 11, pp. 1363-1369, Nov. 2000.

[70] Z. Chen and K. Roy, "A Power Macromodeling Technique Based on Power

Sensitivity," in Proceeding of ACM/IEEE Design Automation Conference, pp.678-683, 1998.

[71] H. Mehta, R. M. Owens and M. J. Irwin. "Energy Characterization based on Clustering," in Proceeding of 33rd Design Automation Conference, pp. 702-707, 1996.

[72] L. Cao, "Circuit Power Estimation Using Pattern Recognition Techniques," in Proceeding of IEEE/ACM International Conference on Computer-Aided Design, pp. 412-417, 2002.

[73] S. Gupta and F. N. Najm, "Analytical Models for RTL Power Estimation of Combinational and Sequential Circuits," IEEE Transactions on Computer-Aided Design, Vol. 19, Issue 7, pp. 808-814, Jul. 2000.

[74] Q. Wu, Q. Qiu, M. Pedram, and C.S. Ding, "Cycle-Accurate Macro-Models for RT-Level Power Analysis", IEEE Transactions on VLSI Systems, Vol. 6, Issue 4, pp. 520-528, 1998.

[75] S. Gupta and F. N. Najm, "Energy-Per-Cycle Estimation at RTL," in Proceeding of ACM/IEEE Internal Symposium on Low Power Design, pp. 121-126, 1999.

[76] C.Y. Hsu, C.N. Jimmy Liu and J.Y. Jou, "Efficient Vector Compaction Methods for Power Estimation with Consecutive Sampling Techniques," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E87-A No.11 pp.2973-2982, 2004.

[77] C.Y. Hsu, C.N. Jimmy Liu and J.Y. Jou, "An Efficient Power Model for IP-Level Complex Designs," IEICE Transactions on Fundamentals of Electronics,

Communications and Computer Sciences, Vol.E86-A No.8 pp.2073-2080, 2003.

[78] C.Y. Hsu, W.T. Hsieh, C.N. Jimmy Liu and J.Y. Jou, "A Tableless Approach for High-Level Power Modeling Using Neural Networks," Journal of Information Science and Engineering, 2005. (Accepted)

[79] C.Y. Hsu and W.Z. Shen. "Vector Compaction for Power Estimation with Grouping and Consecutive Sampling Techniques," in Proceeding of International Symposium on Circuits and Systems, Vol. II, pp. 472-475, 2002.

[80] Michael R. Garey and David S. Johnson, "Computers and Intractability: A Guide to the Theory of NP Completeness," W. H. Freeman, San Francisco, 1979.

[81] E. Macii and M. Poncino, "Estimating Power Consumption of CMOS Circuits Modelled as Symbolic Neural Networks," IEE Proceedings on Computers and Digital Techniques, Vol. 143, Issue 5, pp. 331-336, 1996.

[82] J.H. Hopfield, "Artificial Neural Networks," IEEE Circuits and Devices Magazine, Vol. 4, Issue 5, pp. 3-10, 1988.

[83] R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Algebraic Decision Diagrams and Their Applications," in Proceeding of IEEE/ACM International Conference on Computer-Aided Design, pp. 188-191, 1993.

[84] T. L. Fine, "Feedforward Neural Network Methodology," New York: Springer, 1999.

[85] K. Levenberg, "A Method for the Solution of Certain Problem in Least Square," Quarterly of Applied Mathematics, Vol. 11, pp. 164-168, 1944.

[86] D.W. Marquardt, “An Algorithm for Least Squares Estimation of Non-linear Parameters,” Journal of the Society for Industrial and Applied Math., Vol. 11, pp. 431-441, 1963.

[87] K. Mehrotra, C. K. Mohan and S. Ranka, “Elements of Artificial Neural Networks,” Cambridge, Massachusetts: MIT Press, 1997.

[88] E. B. Baum and D. Haussler, “What Size Net Gives Valid Generalization?” Neural Computation, Vol. 1, pp. 151-160, 1989.