Multilevel Power Estimation Of VLSI Circuits Using Efficient Algorithms

A Thesis Submitted In Partial Fulfillment of the Requirements for the Award of the Degree of

Master of Technology

In

Electronics and Communication Engineering
(VLSI Design and Embedded System)

by

Bikash Chandra Rout

Roll No: 209EC2136

Department of Electronics & Communication Engineering
National Institute of Technology Rourkela

June 2011

Multilevel Power Estimation Of VLSI Circuits Using Efficient Algorithms

A Thesis Submitted In Partial Fulfillment of the Requirements for the Award of the Degree of

Master of Technology

In

Electronics and Communication Engineering
(VLSI Design and Embedded System)

by

Bikash Chandra Rout
Roll No: 209EC2136

Under the Supervision of

Dr. Kamala Kanta Mahapatra

Department of Electronics & Communication Engineering
National Institute of Technology Rourkela

June 2011

Abstract

New and complex systems are being implemented using highly advanced Electronic Design Automation (EDA) tools. As the complexity increases day by day, the dissipation of power has emerged as one of the very important design constraints. Now low power designs are not only used in small size applications like cell phones and handheld devices but also in high-performance computing applications.

Embedded memories have been used extensively in modern SOC designs. In order to estimate the power consumption of the entire design correctly, an accurate memory power model is needed. However, the memory power model commonly used in commercial EDA tools is too simple to estimate the power consumption accurately.

For complex digital circuits, building their power models is a popular approach to estimate their power consumption without detailed circuit information. In the literature, most of power models are built with lookup tables. However, building the power models with lookup tables may become infeasible for large circuits because the table size would increase exponentially to meet the accuracy requirement.

This thesis involves two parts. In first part it uses the Synopsys power measurement tools together with the use of synthesis and extraction tools to determine power consumed by various macros at different levels of abstraction including the Register Transfer Level (RTL), the gate and the transistor level. In general, it can be concluded that as the level of abstraction goes down the accuracy of power measurement increases depending on the tool used. In second part a novel power modeling approach for complex circuits by using neural networks to learn the relationship between power dissipation and input/output characteristic vector during simulation has been developed. Our neural power model has very low complexity such that this power model can be used for complex circuits. Using such a simple structure, the neural power models can still have high accuracy because they can automatically consider the non-linear power distributions. Unlike the power characterization process in traditional approaches, our characterization process is very simple and straightforward. More importantly, using the neural power model for power estimation does not require any transistor-level or gate-level description of the circuits. The experimental results have shown that the estimations are accurate and efficient for different test sequences with wide range of input distributions.

Acknowledgement

This project is by far the most significant accomplishment in my life and it would be impossible without people who supported me and believed in me.

This is a unique opportunity for me to express our abysmal sense of gratitude, reverence and indebtedness to Dr. Kamala Kanta Mahapatra, Professor of department of Electronics and Communication Engineering, NIT Rourkela, for his aurulent guidance, unceasing supervision, sustained enthusiasm, keen interest and constructive criticism during the period of preparation of this project manuscript. His trust and support inspired me in the most important moments of making right decisions and I am glad to work with him.

It is my pleasure to refer VHDL, Verilog, Acrobat Reader and Microsoft Word exclusive of which the whole process, right from simulation to compilation of this report would have been impossible.

I would also like to mention Mr. Ayaskanta Swain, Mr. Jagannath Prasad Mohanty for their cooperation and constantly rendered assistance.

I would like to thank all my friends and especially my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious. I've enjoyed their companionship so much during my stay at NIT, Rourkela.

Last but not the least; I am highly indebted to our adored parents and my elder brother for having provided us with the needed and not needed instructive insight and encouragement during the completion of the project.

Table of Contents

List of tables

List of figures

Chapter 1

Introduction

1.1 Motivation

With the increasing usage of electronics devices and Internet appliances, there is a corresponding increased need for employing low-power design methodologies. One of the important requirements to know during a design process is how much power the circuit should dissipate considering its application. So after the designer writes the required code, keeping in mind all the specifications that have been given to him, a power calculation needs to be done to confirm if the design meets the required specification. This is done prior to sending the chip for fabrication. So it is extremely important to get accurate power values using power determining tools running them at certain input conditions.

Numerous EDA (Electronic Design Automation) tools have been developed to not only determine power but also help in power reduction. The usage of these tools is classified depending on the layer of abstraction they are used in. The three main layers of abstraction include the RTL (Register Transfer Level), the gate and the transistor level. Though there are numerous tools that can be used at each of these levels, this thesis mainly concentrates on using Synopsys tools.

System-on-a-chip (SOC) is a trend of system integration in recent years. 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 on operating time of portable devices. Traditionally, power estimation is often performed at transistor-level by SPICE-liked simulation at the end of design flow, as shown in Figure 1-1. At this moment, it is often too late to obtain the information of power dissipation at transistor-level. In order to avoid costly redesign steps for such complex design, designers have to estimate the power consumption at higher design stage to understand whether more improvements are required. Furthermore, this SPICE-liked approach will become unpractical for SOC designs because the transistor-level description of whole designs is often too

large to be simulated and IP vendors may not provide such low-level description for an IP to protect their knowledge.

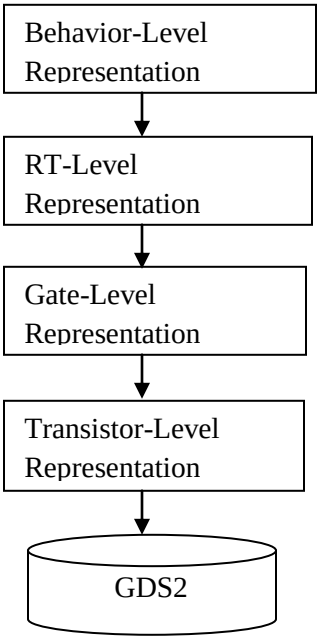

Figure 1-0-1A Design Flow of CMOS Digital Circuit

For this application, power models may provide an efficient solution to estimate power consumption of complex circuits without transistor-level even gate-level circuit descriptions. After a power characterization process with detailed circuit information, we can build a power model that describes the relationship between high-level power characteristics and real power consumption under specific input sequences or input/output signal statistics. With this power model, users can obtain the power consumption of the circuits without detailed circuit information because it can be derived from the power model and the high-level power characteristics directly. Lookup table is the most commonly used power model. In other research areas, neural networks are widely used 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. Therefore, in this thesis, we propose a novel power model for complex digital circuits that uses neural networks to learn the power characteristics during simulation. The complexity of our neural power model has no relationship with circuit size and

number of inputs and outputs such that this power model can be kept very small even for complex circuits. More importantly, using the neural power model for power estimation does not require any transistor-level or gate-level description of the circuits, which is very suitable for IP protection.

1.2 Goals and contributions

The main goal of this thesis is to calculate the power of several digital circuits which vary in complexity from a 500-transistor net-list to one containing more than 150,000 transistors. The next goal of this thesis is to develop a power model to calculate the dynamic power dissipation, which is based on neural network.

For each of the benchmark circuit, power will be calculated at various levels of abstraction using two EDA tools supplied by Synopsys: Power Estimator, Power Compiler. The purpose and functionality of each of tools will be discussed in the later chapters. Scripts will be developed to implement the various results. The following Figure 1.2 shows the design flow involved in the thesis in calculating the power values at different levels of abstraction.

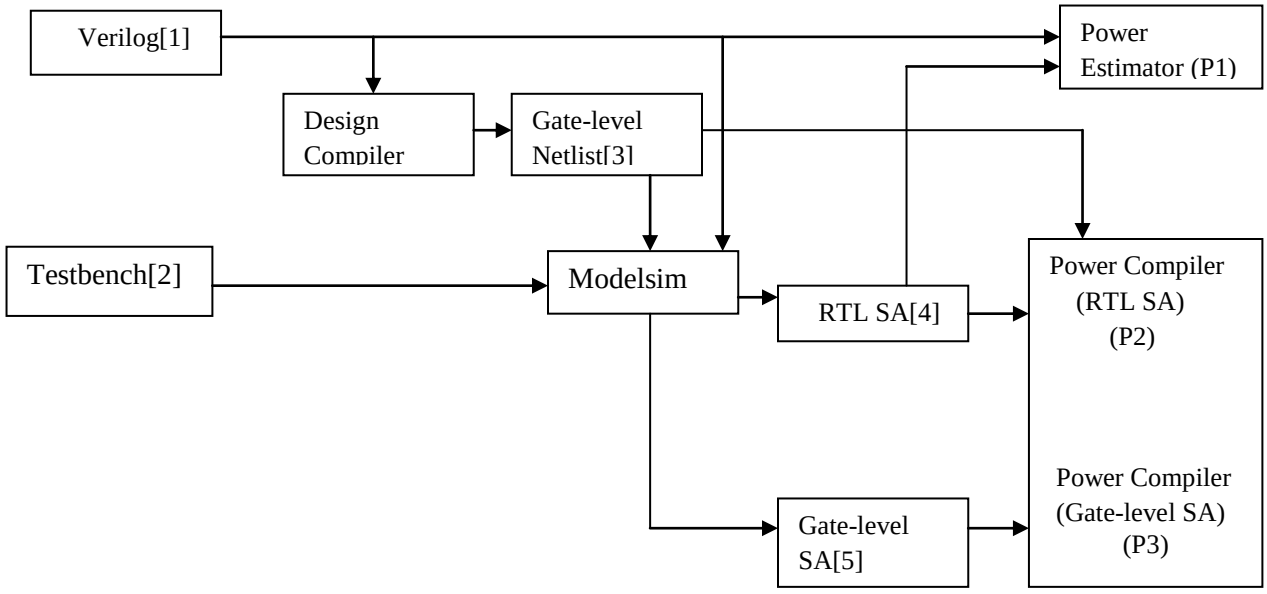

Figure 1-0-2Design Methodology showing power calculation using different power tools

Then a novel power modeling approach will be developed based on neural network back propagation algorithm to calculate the power by taking the gate level power report as the basic building for power model.

Thesis Organization

Chapter 2 mainly reviews the literature related to the various tools that have been used in this work and briefly discusses about different types of power along with high level power estimation. Chapter 3 discusses the artificial neural network and different training algorithms. Chapter 4 covers with the power modeling approach using neural network. Chapter 5 presents the experimental design of benchmark circuit C 432. In chapter 6 results from power compiler and different algorithms will be compared. Finally the conclusions, and future works are given in chapter 7.

Chapter 2

Background

2.1 Need for Low Power Design

In the early 1970's designing digital circuits for high speed and minimum area were the main design constraints. Most of the EDA tools were designed specifically to meet these criteria. Power consumption was also a part of the design process but not very visible. The reduction of area of digital circuits is not as big issue today because with new IC production techniques, many millions of transistors can be fit in a single IC. However, shrinking sizes of circuits have paved the way for reduced power consumption in order to have an extended battery life. Also in submicron technologies, there is a limitation on the proper functioning of circuits due to heat generated by power dissipation. Market forces are demanding low power for not only better life but also reliability, portability, performance, cost and time to market. This is very true in the field of personal computing devices, wireless communications systems, home entertainment systems, which are becoming popular now-a-days. Devices that are also used for high-performance computing particularly need to dissipate less power to function correctly and for a long period of time [1]. Keeping all these in mind, low power design has become one of the most important design parameters for VLSI (Very Large Scale Integration) systems.

2.1.1 Design Flow with and without Power

A top-down ordinary VLSI design approach is illustrated in Figure 2.1. The figure summarizes the flow of steps that are required to follow from a system level specification to the physical design. The approach was aimed at performance optimization and area minimization. However, introducing the third parameter of power dissipation made the designers to change the flow as shown in the right-hand side of the Figure 2.1.

In each of the design levels are two important power factors, namely power optimization and power estimation. Power optimization is defined as the process of obtaining the best design knowing the design constraints and without violating design specifications. In order to meet the design and required goal, a power optimization technique unique to that level should be employed. Power estimation is defined as the process of calculating power and energy dissipated with a certain percentage of accuracy and at different phases of the design process. Power

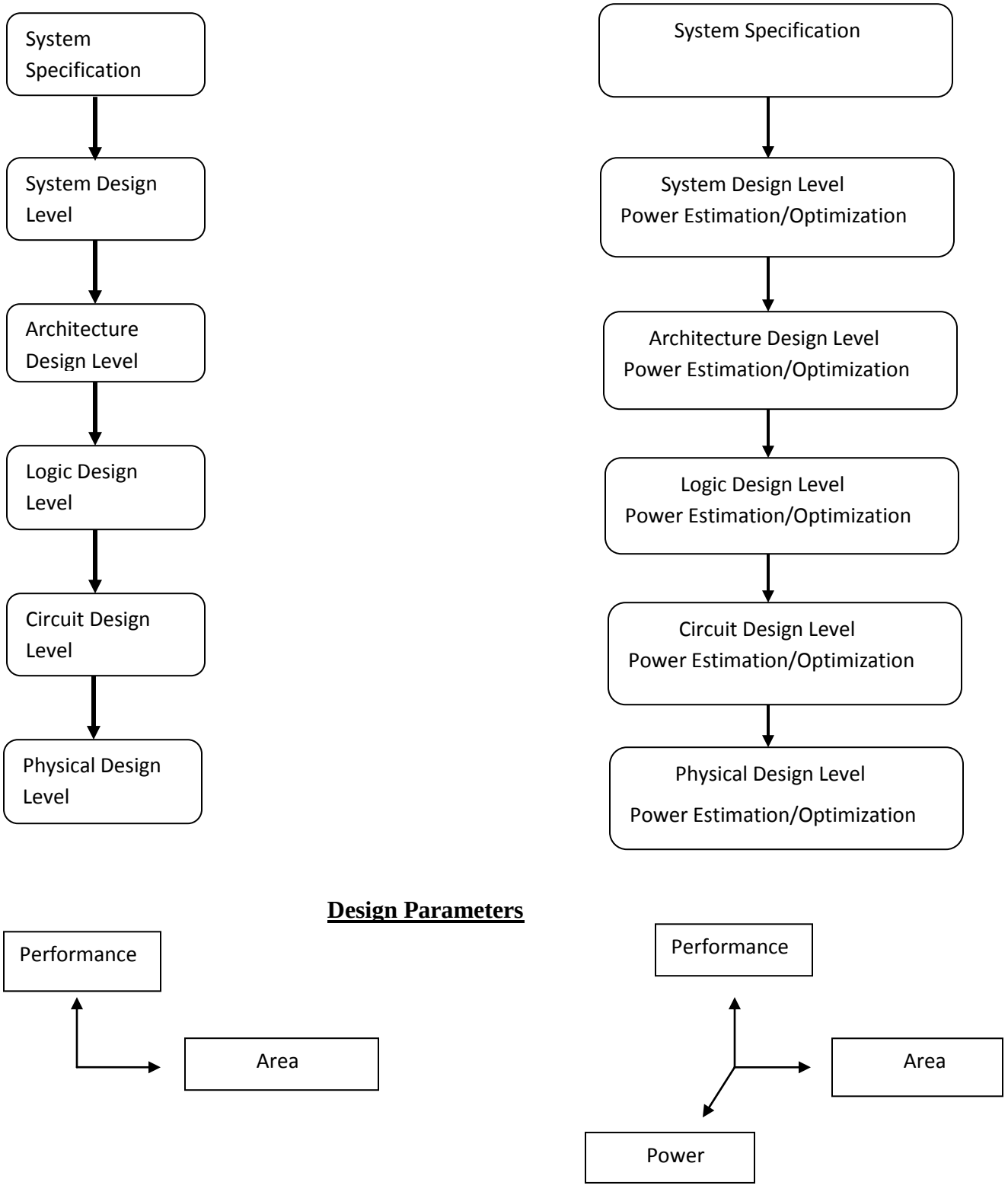

Figure 2-0-3 VLSI Design Flow

estimation techniques evaluate the effect of various optimizations and design modifications on power at different abstraction levels.

Generally a design performs a power optimization step first and then a power estimation step, but within a certain design level there is no specific design procedure. Each design level includes a large collection of low power techniques. Each may result in a significant reduction of power dissipation. However, a certain combination of low power techniques may lead to better results than another series of techniques.

Generally, power is consumed when capacitors in the circuits are either charged or discharged due to switching activities. So at higher levels of a system this power dissipation is conserved by reducing the switching activities which is done by shutting down portions of the system when they are not needed. Large VLSI circuits contain different components like a processor, a functional unit and controllers. The idea of power reduction is to stop any of the components of the processor when they are not needed so that less power will be dissipated when the processor is operating [2].

2.2 Relationship Between Different Abstraction Levels

The relationship between design abstraction level and power estimation techniques is shown as Figure 2.2. The power estimation at higher level is much faster, but the accuracy will become worse due to the limited design information A number of CAD techniques for power estimation at lower levels of abstraction, such as transistor-level [2-4] or gate-level [5], have been proposed.

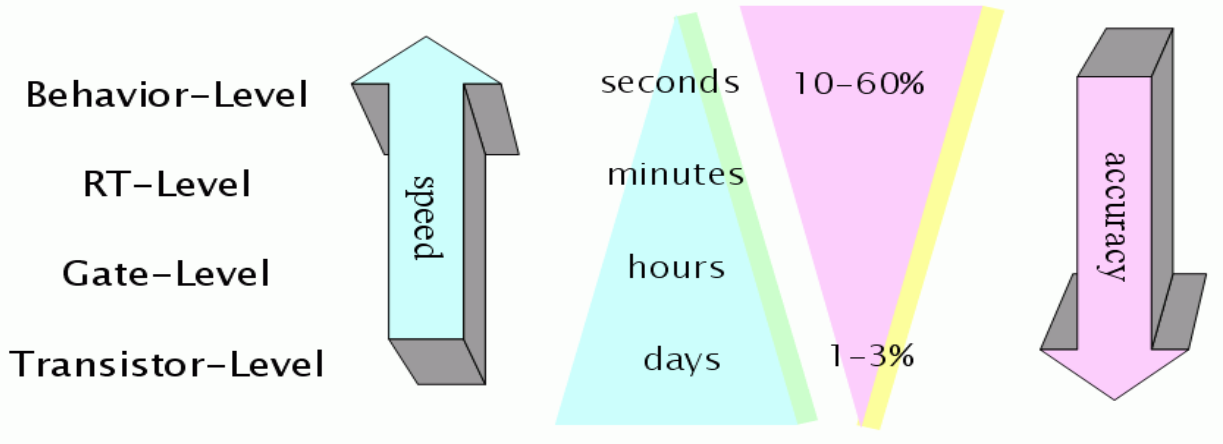

Figure 2-0-4Relationship between different abstraction level & Power estimation techniques

Generally speaking, they can provide more accurate estimation results. However, they may become unpractical for complex designs due to the whole system simulation requires too much computation resources in such low abstract levels. In addition, when the design has been specified down to gate level or lower, it may be too expensive to go back to fix high-power problems. Most importantly, IP vendors may not provide such low-level description for an IP to protect their knowledge.

2.3 Basic Concepts for Power

The power dissipation of digital CMOS circuits can be described by

$$P_{avg} = P_{dynamic} + P_{short\text{-}circuit} + P_{leakage} + P_{static}$$

P_{avg} is the average power dissipation, $P_{dynamic}$ is the dynamic power dissipation due to switching of transistors, $P_{short\text{-}circuit}$ is the short-circuit current power dissipation when there is a direct current path from power supply down to ground , $P_{leakage}$ is the power dissipation due to leakage currents, P_{static} and is the static power dissipation [2][4]

2.3.1 Static Power

Static power is the power dissipated by a gate when it is not switching that is, when it is inactive or static. Ideally, CMOS (Complementary Metal Oxide Semiconductor) circuits dissipate no static (DC) power since in the steady state there is no direct path from V_{dd} to ground. This scenario can never be realized in practice, since in reality the MOS transistor is not a perfect switch. There will always be leakage currents, sub threshold currents, and substrate injection currents, which give rise to the static component of power dissipation. The largest percentage of static power results from source-to-drain sub threshold voltage, which is caused by reduced threshold voltages that prevent the gate from completely turning off [2][4].

2.3.2 Dynamic Power

Dynamic power is the power dissipated when the circuit is active. A circuit is active anytime the voltage on net changes due to some stimulus applied to the circuit. In other words, dynamic power dissipation is caused by the charging. Because voltage on an input net can change without necessarily resulting in logic transition in the output, dynamic power can be

dissipated even when an output net doesn't change its logic state. This component of dynamic power dissipation is the result of charging and discharging parasitic capacitances in the circuit [2][4].

Dynamic power of a circuit is composed of

a) Switching power

b) Internal power

2.3.2.1 Switching power

The switching power of a driving cell is the power dissipated by the charging and discharging of the load capacitance at the output of the cell. The total load capacitance at the output of a driving cell is the sum of the net and gate capacitances on the driving output. The charging and discharging are result of logic transitions. Switching power increases as logic transitions increase. Therefore, the switching power of a cell is a function of both the total load capacitance at the cell output and the rate of logic transitions. Switching power comprises 70-90 percent of the power dissipation of an active CMOS circuit [2][4].

2.3.2.2 Internal power

Internal power is any power dissipated within the boundary of a cell. During switching, a circuit dissipates internal power by the charging or discharging of any existing capacitances internal to the cell. Internal power includes power dissipated by a momentary short circuit between the P and N transistors of a gate, called short-circuit power.

2.3.3 Short-Circuit Power

The short-circuit power consumption, $P_{short\text{-}circuit,}$ is caused by the current flow through the direct path existing between the power supply and the ground during the transition phase.

2.3.4 Leakage Power

The PMOS and NMOS transistors used in a CMOS logic circuit commonly have non-zero reverse leakage and sub-threshold currents. These currents can contribute to the total power dissipation even when the transistors are not performing any switching action. The leakage power dissipation, $P_{leakage}$ is caused by two types of leakage currents.

The leakage power dissipation, $P_{leakage}$ is caused by two types of leakage currents

a) Reverse-bias diode leakage current

b) Sub threshold current through a turned-off transistor channel

2.4 Overview of Power Estimation Techniques

In our research, we focus on estimating the dynamic power dissipation of digital circuit, which is directly related to chip heating and battery lifetime. This is quite different from estimating the worst case of instantaneous power. Because this is a strongly input pattern dependent problem, several solutions [3][6][20] are proposed to overcome this problem by using the probabilistic measures. In those approaches, they use probabilities as a compact way to describe a large set of possible logic signals. Another approach for average power estimation is to obtain the current waveform by performing a simulation. We refer these methods as simulation-based techniques. In the literature, many simulation-based approaches have been proposed at various kinds of abstraction level [21]. Generally speaking, the comparison of the accuracy and speed among those approaches can be summarized in figure 2.2.

Those most accurate power estimation approaches is to perform transistor-level simulation, because the detailed information of the whole design is known. However, it has the worst because it requires too much computation resources efficiency and it takes too much time for simulation. Gate-level power simulation techniques can provide a better trade-off between accuracy and efficiency, but it may still cost a lot of redesign time to solve power problems when the design is already at gate-level. Compared to other approaches, high-level power estimation is much harder to obtain high accurate results, because the detail information of the design is already loss too much. However, if the accuracy can be improved to an acceptable region, high-level power estimation techniques will become very useful because we can detect the power problems much earlier and more quickly. In following section, the high-level power estimation will be introduced.

2.5 High-Level Power Estimation

In order to avoid costly redesign steps for such complex design, designers have to estimate the power consumption at higher design stage to understand whether more improvements are

required. It is unpractical for SOC designs to use the traditional SPICE-liked simulation at transistor-level as mentioned in Chapter 1. Therefore, a number of CAD techniques have been proposed for gate-level power estimation [3]. However, when the design has been implemented to the gate level, it may still too late or too expensive to improve the design for power consumption problems. It implies that high-level power estimation techniques are essential for designing such a complex design to shorten redesign cycles.

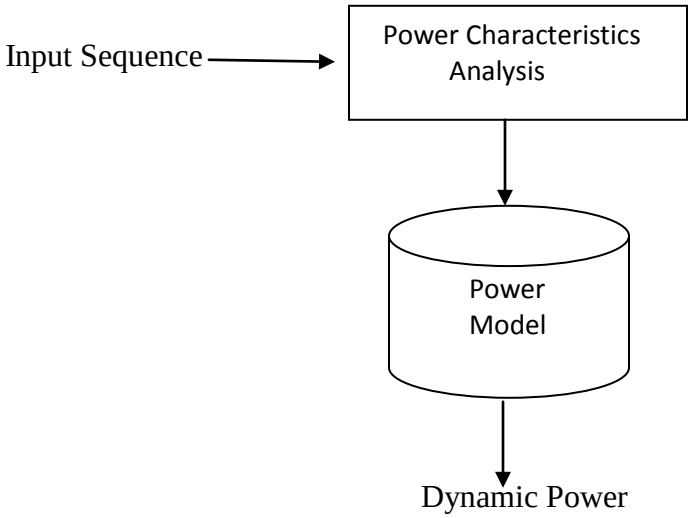

Figure 2-3 A usage of high-level power model

A number of high-level power estimation techniques have been proposed as surveyed in [5]. They are often classified as top-down and bottom-up styles [6]. In the top-down techniques, a circuit is specified as a Boolean function without detail information of the circuit structure. Top-down methods usually use some abstract measurements such as entropy to measure of the amount of information change as the power consumption values [4][18]. They would be useful when designing a logic block that was not previously designed.

High-level power estimation techniques can be roughly divided into two categories: top-down and bottom-up. In the top-down techniques, a combinational circuit is specified only as a Boolean function without information on the circuit implementation.

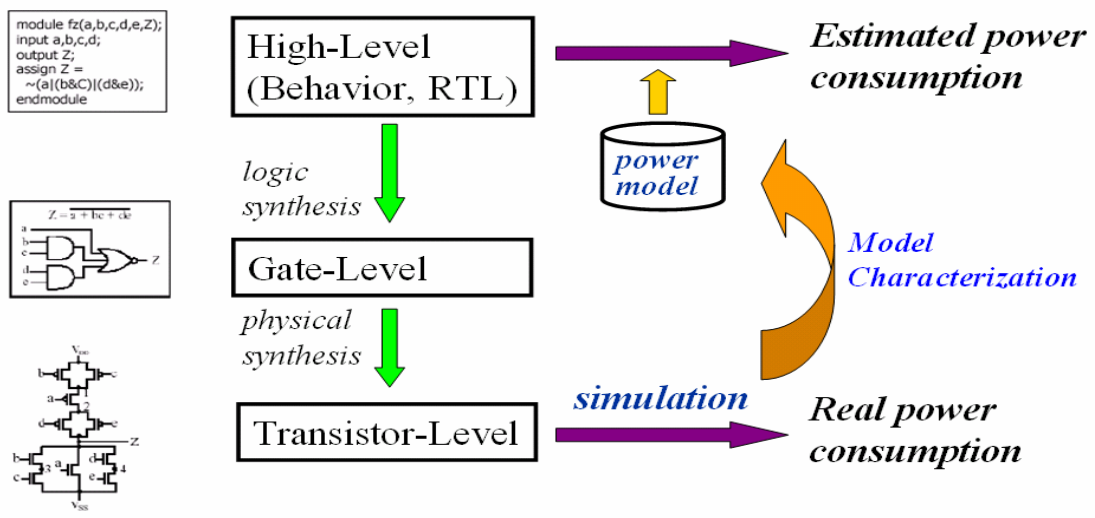

Figure 2-4 High Level power modeling concept

Normally, they will estimate the switching activity of circuits by using entropy. Entropy is a characterization of a random variable or a random process which is commonly used in the information theory [17] as a measure of information-carrying capacity. These kind of top-down techniques are useful when one is designing a logic block that was not previously designed because they can provide a rough measurement about the trend of power consumption before implementation. However, they may not have very good accuracy due to the lack of implementation details.

In contrast, bottom-up methods are useful when reusing a previously designed logic block so that all detailed internal structures of the circuit are known. A power macro-model will be built for such logic blocks in this kind of methods. When this logic block is used in another application, the corresponding power macro-model can be used to estimate the power dissipation of this block without performing any simulation at gate-level or transistor-level. The usage of power model has been showing Figure 2.3. This kind of power modeling approach will be very useful in the IP-based SOC designs.

2.6 Tools Used

There has been a variety of tools involved in this thesis. Even though, this thesis is all about power calculations of macros which are done using tools; there are other tools that have been used prior to the usage of power tools to give the required input to the power tools. More

emphasis is given to these tools that are mainly involved in power estimation. The usage of tools has been classified as Power tools and Non-Power tools.

2.6.1 Non-Power Tools

Non-power tools include Simulation tools, Synthesis tools, Layout tools, Extraction tools and Waveform viewers. The tools that are discussed in this chapter are some of the non-power tools involved in the entire design flow. A short description of each of these tools along with their working flow is given in this chapter to understand their functionality. The subsequent chapter discusses each of the power tools in detailed manner as most of the thesis involves the use of these power tools. The following chapter also discusses the design flow from code writing to spice net-list simulation, clearly explaining the usage of these tools at the respective level.

2.6.1.1 Simulation Tool

Initially, Verilog or VHDL code for a particular design is written and tested. Simulation is done using Mentor's Modelsim for both VHDL Verilog and other Verilog simulators. ModelSim is a simulation and a debugging tool for VHDL, Verilog, and other mixed-language designs from Mentor Graphics [21]. The basic simulation flow is as shown in Figure 2.5. Initially, a working library is created and the code is compiled using the commands depending upon whether the code is VHDL or Verilog.

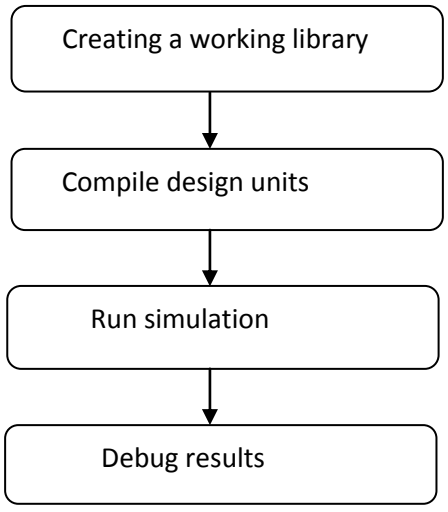

Figure 2-5Modelsim simulation flow

Verilog Compiled Simulator (VCS) [22] from Synopsys is a high-performance, high-capacity Verilog simulator that incorporates advanced high-level abstraction, verification into an open platform. The basic work flow for VCS consists of two basic steps:

a) Compiling source files into executable binary files

b) Running the executable binary file

This two step approach simulates the design faster and uses less memory than other interpretive simulators. The basic design flow is given in Figure 2.

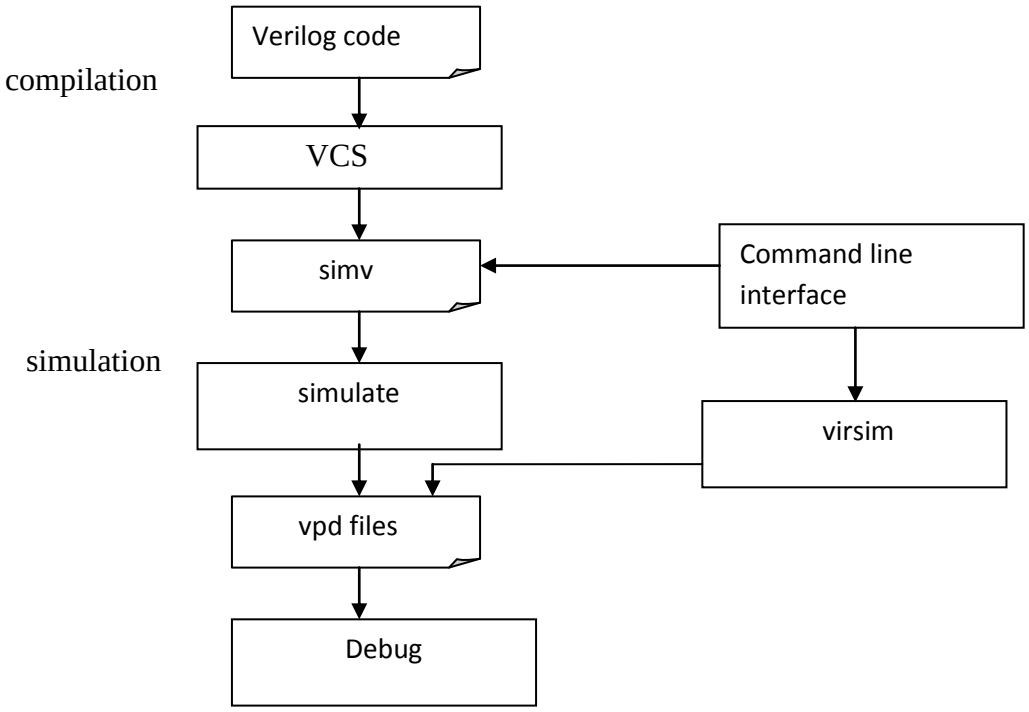

Figure 2-6VCS work flow

2.6.1.2 Synthesis Tool

Design Compiler [24] is the core of the Synopsys synthesis software products. It comprises tools that synthesize HDL designs into optimized technology-dependent, gate-level designs. It supports a wide range of hierarchical design styles and can optimize both combinational and sequential designs for speed, area, and power.

The basic Design Compiler(Design Vision) synthesis process is given in Figure 2.7.

The Design Compiler is a powerful tool that other products can be run inside its environment using specific commands. Some of the products that can be accessed are HDL compiler,

Automated chip synthesis, FPGA compiler, Behavioral compiler and Power Compiler. HDL compiler reads and writes Verilog or VHDL design files. The Verilog or VHDL compiler reads the HDL files and performs translation and architectural optimization of the designs. The appropriate HDL compiler is automatically called by Design Compiler when it reads an HDL design file.

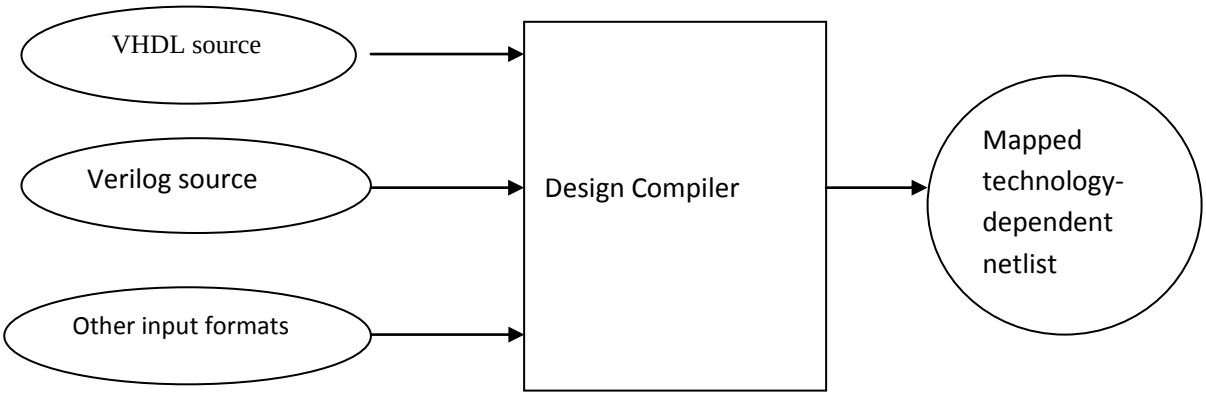

Figure 2-7Design compiler synthesis process

2.6.2 Power Tools

This thesis involves the usage of Synopsys power tools. The power products are tools that comprise a complete methodology for low-power design. Synopsys power tools offer power analysis and optimization throughout the design cycle, from RTL to the gate level. Analyzing power early in the design cycle can significantly affect the quality of the design. Improvements made to the design while it is at RTL level can get even better results eventually. Not only these power tools do accurate measurements but also can help in calculating power quicker.

Power consumption is calculated at three levels of abstraction. The tools used at these levels are

a) RTL Level - RTL Power Estimator

b) Gate Level – Power Compiler (based on switching activity),

c) Transistor Level – NanoSim

2.6.2.1 Power Compiler

Power Compiler [28] is an add-on product to Design Compiler. The Power Compiler tool optimizes the design for power. Working in conjunction with the Design Compiler tool, Power Compiler provides simultaneous optimization for timing, power and area. In addition to the standard inputs to synthesis (RTL or gate-level net-list, technology library, design constraints, and parasitics), Power Compiler uses two other inputs: Switching activity of design elements and power constraints. It contains all the analysis capabilities of Design Power.

Power Compiler uses the same power analysis engine as Design Power. This allows Power Compiler to the use the same switching activity for optimization that Design Power uses for analysis. It accepts either user-defined switching activity, switching activity from simulation, or a combination of both. It provides RTL clock gating and optimizes the circuit based on circuit activity, capacitance, and transition times. Power Compiler cannot only be used as a standalone product but also can be used in coordination with Design Compiler, Module Compiler, Physical Compiler and Floor plan Manager.

2.6.2.1.1 Power Compiler Methodology

Power Compiler is used at RTL and Gate level to calculate power and do power optimization depending on the need. At each level of abstraction, simulation, analysis and optimization can be performed to refine the design before moving to the next lower level. Simulation and the resultant switching activity gives the analysis and optimization the necessary information to refine the design before going to next lower level of abstraction. The higher the level of design abstraction, the greater the power savings can be achieved. The following Figure 2.8 describes the power flow at each of the abstraction level. Figure 2.9 shows power flow from RTL to Gate level.

Cell internal power and net toggling directly affect dynamic power of a design. To report or optimize power, Power Compiler requires toggle information for the design. This toggle information is called Switching Activity.

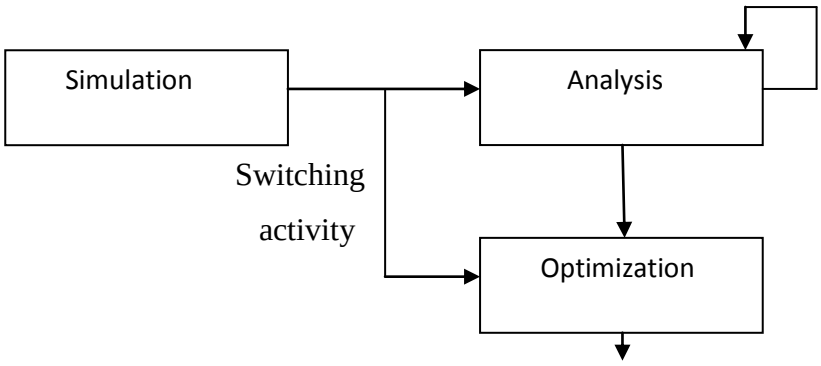

Figure 2-8 Power flow at each of the abstraction level

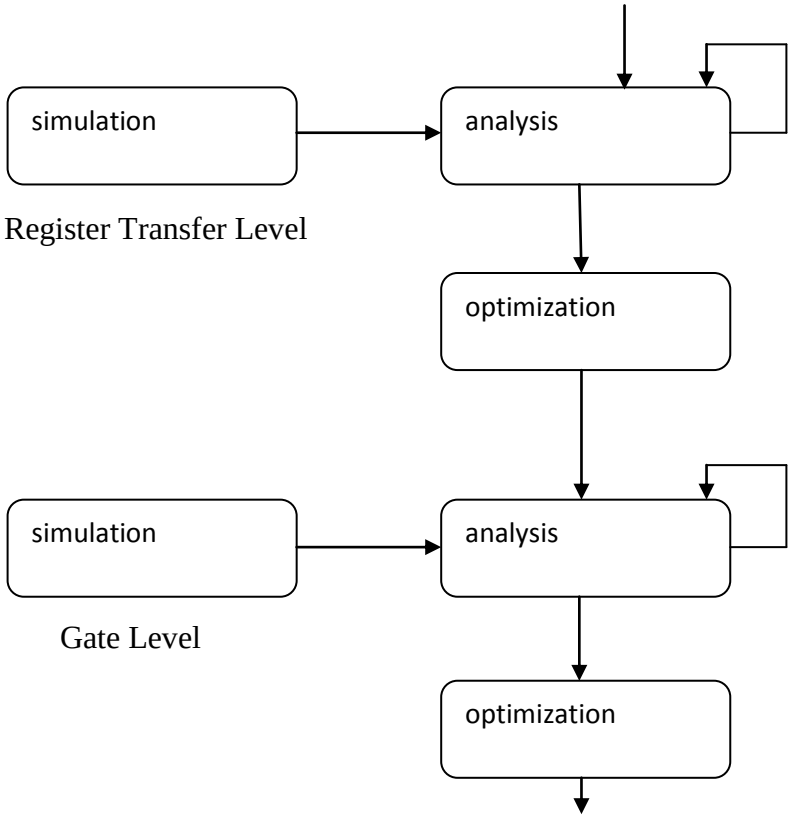

Figure 2-9power flow from RTL to Gate level

Power Compiler models switching activity in terms of static probability and toggle rate. Static probability is the probability that a signal is at a certain logic state and is expressed as a number between 0 and 1. It is calculated during simulation of the design by comparing the time of a signal at a certain logic state to the total time of the simulation. Toggle rate is the number of logic-0-to-logic-1 and logic-1-to-logic-0 transitions of a design object per unit of time.

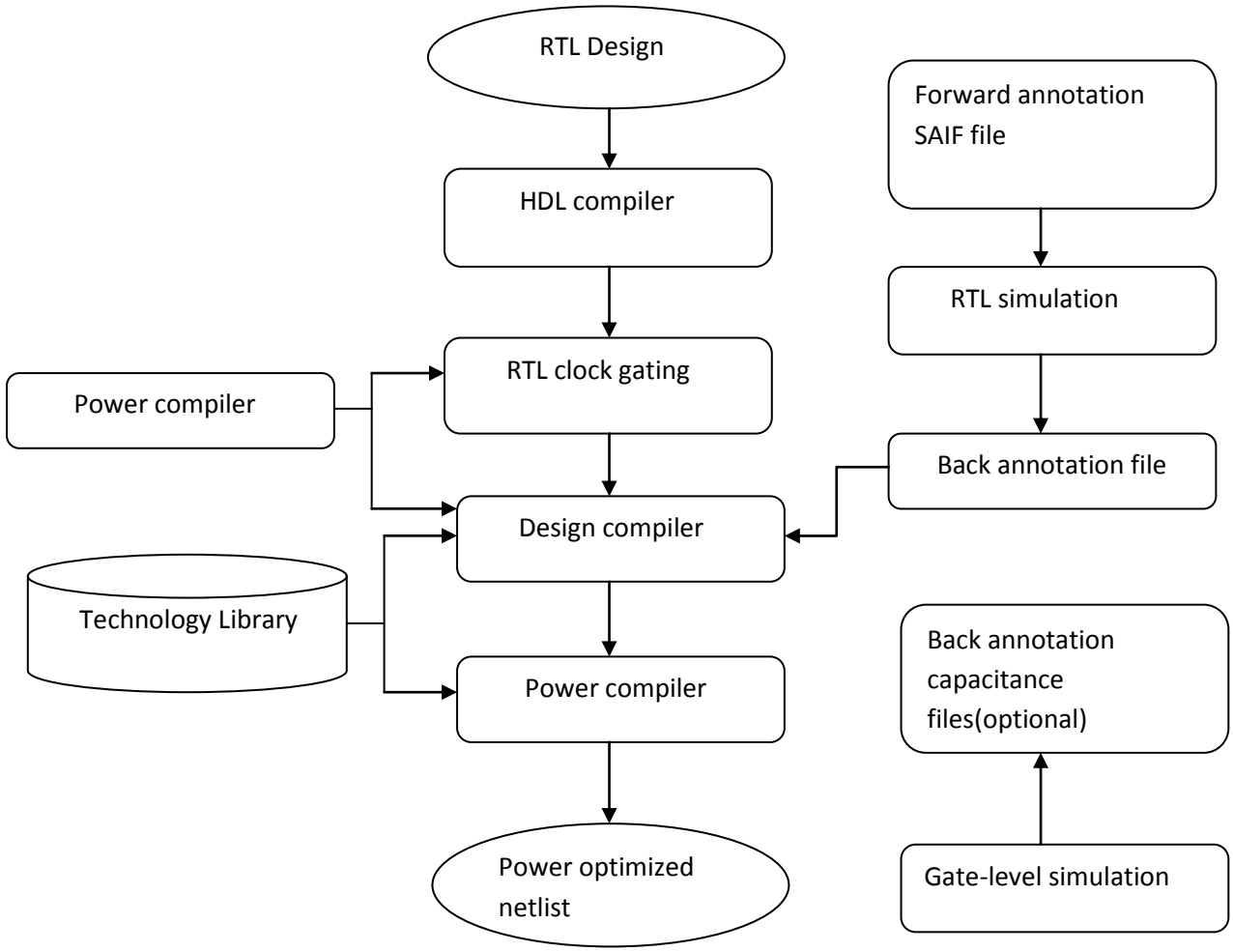

Figure 2-0-50Power methodology in power compiler

The following Figure 2.10 shows the methodology of power calculation using the combination of Power Compiler and Design Compiler. The flow of data between the different steps and tools used are also shown. Before starting to calculate power using Power Compiler the desired gate-level net-list of the design should be first generated. The power methodology starts with the RTL design and finishes with a power-optimized gate-level net-list. Ultimately, Power Compiler is used to calculate power using the gate-level net-list produced by the Design Compiler or power-optimized gate net-list produced by Power Compiler itself. As seen in the figure most of the processes that take place are using Design Compiler, but the simulation process that is shown is outside Design Compiler tool and is done as part of power calculation. The main purpose of simulation is to generate information about the switching activity of the

design and create a file called Back-annotation. This file can contain switching activity from RTL simulation or gate-level simulation. Initially, the RTL design is given to the HDL compiler to create a technology-independent format called as GTECH design. This is as a result of analyzing and elaborating the design by HDL compiler. This formatted design is given as an input to Design Compiler. Before it is compiled by the Design Compiler, "**rtl2saif**" command is used to create forward-annotation file which is later used for simulation. The formatted design GTECH is later given as input to Design Compiler which produces an output which is given to Power Compiler.

The Forward-annotation SAIF file is given as an input to do RTL simulation which gives a back-annotation SAIF file which is used by Power Compiler. This forward annotated file contains directives that determine which design elements to be traced during simulation. Gate-level simulation can also use a library forward-annotation file. This forward-annotation file used for gate level simulation has different information compared to RTL forward-annotation file. This file contains information from the technology library about cells with state and path-dependent power models. "**Lib2saif**" command is used to get this forward-annotation file.

During power analysis, Power Compiler uses the annotated switching activity to evaluate the power consumption of the design. During power optimization, Power Compiler uses the annotated switching activity to make decisions about the design.

Chapter 3

Artificial Neural Network

3.1 Introduction

Although today's computers are extremely fast and precise, there are still many tasks that the human brain can compute more efficiently than a computer in real world. For example: reading handwritten characters automatically, recognizing the words spoken by any speaker, driving a car, walking and running as an animal or human being, etc. These works are easy for us but not for computers because of the special ability of biological brain – recognition and learning. This is why we called it "computer", not "electric brain".

In a conventional computer, the instructions are executed sequentially in a fast and complicated single processor. The speed of the processor in a computer is more than 100 times faster than the basic processing element of the brain called **neuron**. It is worthy to note that even the neurons are slower than electrical processor, the brain can still perform many tasks much faster than any conventional computer. This is because the brain has a massively parallel structure of biological neural networks.

3.2 Biological Neurons vs. artificial neurons

3.2.1 Biological Neurons

It is claimed that the human brain consists of a large number (approximately 10^{11}) [8][9] of highly connected (approximately 10^4 connections per element) elements called neurons. They communicate through a connection network of axons and synapses [8]. It can be considered that the brain is a densely connected electrical switching network that is operated by the biochemical processes. These neurons have three principle components [9]: the cell body, which is also called the soma, the axon, and the dendrites as shown in Figure 3-1. The dendrites are tree-like receptive networks of nerve fibers that carry electrical signals into soma. Soma sums these incoming signals and judges the threshold effectively. The axon is a single long fiber that carries the electrical signal from soma to other neurons. The contact point between an axon of one cell and a dendrite of another cell is called a synapse. The information transfer is across a synapse,

which is controlled by biochemical agents [10] a process that is modeled in electronic neurons by the changing of synaptic weights. Thus, the process of learning is to alter these various synapses. It is believed that the new memories are formed by modification of these synaptic strengths.

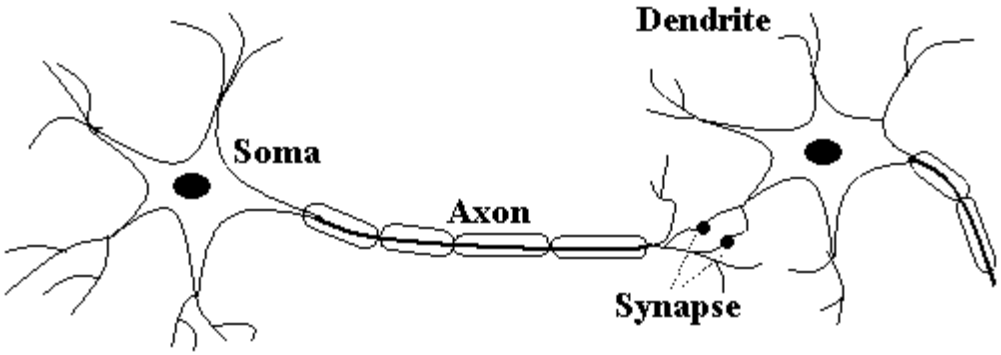

Figure 3-0-6 A simplified schematic diagram of two biological neuron

3.2.2 Artificial Neurons

After we briefly described the operations of biological neurons, we will introduce the simplified mathematical model of the neurons and will explain how these artificial neurons operate. Since the neural computing is a mathematical model inspired by biological models, this computing system is also made up of a number of artificial neurons and a huge number of interconnections between them.

The basic unit in a neural network is an artificial neuron as shown in Figure 3-2. In Figure 3-2, 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. In most cases, x_0 is fixed as 1 such that the training algorithm, which will be discussed on later chapter, only adjusts the weight w_0 to w_N. f is the transfer function that converts s into y, which is often a non-linear function and can be arbitrarily decided by users. As a summary, the output of a neuron can be expressed as Equation (3-1).

$$y = f\left(\sum_{i=1}^{P} w_i x_i\right) \qquad 3.1$$

Comparing the both models, inputs (x) in the artificial model are similar to the input electrical signals in biological neurons. Outputs (y) in the artificial model are similar to the output of biological neurons. Weights (w) in the artificial model correspond to the synaptic

strength connections in biological neurons. And the transfer function is analogous to the firing frequency of the biological neurons.

3.3Feed-Forward Neural Network

A neural network is a set of interconnected neurons, where the outputs of neurons act as the inputs of other neurons. Although there are many connection configurations for neural networks, we choose the multi-layer feed-forward network architecture as the first study case for the new application, high-level power estimation, in VLSI/CAD field.

Feed-forward neural network is one of the most popular models of neural networks. Basically, it is a layered acyclic network in which the neurons are separated into several layers and connections are only allowed from the neurons in layer l to the neurons in layer l+1. For example, a full-connected 3-layered feed-forward neural network is illustrated as Figure 3-3. In this architecture, the neurons in one layer get their inputs from the previous layer and feed their output to the next layer. The input layer is made up of special input neurons that transmit the applied external inputs to their outputs. The last layer of neurons is called the output layer and the layers between the input and output layers are called the hidden layers. If there are only input layer and output layer in a network, it is called a single layer network. If there are one or more hidden layers, such networks are called multi-layer networks. For a feed-forward network, there always exists an assignment of indices to neurons so that the weight matrix can be kept triangular to have smaller indices. Furthermore, if the diagonal entries are zero, it means that there is no self-feedback on the neurons.

3.4 Operations of Neural Network

In this section, we are going to explain the detailed operations of neural networks. They can be separated into two phases: learning (also called training) phase and recalling phase. The learning approaches of neural networks can be further divided into two categories: supervised learning and unsupervised learning structures. In supervised learning [8][10], the network is "taught" what response it should make to each input that it received. We assume that at each instant of time when the input **x** is applied, the desired response **d** of the system is provided by the teacher as illustrated in Figure 3-4a. The distance ρ[**d**,**o**] between the actual response (**o**) and the desired response (**d**) serves as an error measurement to modify the parameters in the network. Those

parameters, including weights and bias factor matrix, will be modified according to the error measurement such that the error can be decreased.

This mode of learning is commonly used in many situations of natural learning. A set of inputs and desired output patterns, which are called training set, are required for this learning mode. In our case, we use the supervised learning method, backpropagation [8][9], to learn the relationship between the power dissipation and the status of primary input/output signals. The learning capability can also be built in the networks without teachers. Unsupervised networks[8][10] can also learn by using built-in rules for self-modification. In other words, the weights and biases are modified in response to the inputs of network only as illustrated in Figure 3-4b. There are no target outputs to act as a teacher that can help us to modify the network parameters. While the weights and bias matrixes are fixed after learning phase, the computation of **o** for a given **x** performed by the network is the recalling phase. The operation in this phase can be viewed as a simple number calculation process to decode the stored contents that have been encoded in the network at learning phase.

3.5 Training Algorithms

The target of training algorithms is to minimize an error function by adjusting the corresponding weight matrix in the neural network. In this work, the error function is chosen as the mean square error defined in Equation (3-2) because it is widely used and there are many existed training algorithms for minimizing this error function. In Equation (3-2), $\boldsymbol{W} = [w_1\ w_2\ \ldots\ w_Q\ ]^T$ consists of all weights including biases of the network, y_i is the output value of the i_{th} input vector and Q is the number of weights. There are many training algorithms for feedforward neural networks that can select suitable weights to minimize the error function in Equation (3-2). Some methods such as steepest descent algorithm, conjugate gradients algorithm and quasi-Newton algorithm [12] are general optimization methods. In this work, we choose both steepest descent approach and Levenberg-Marquardt algorithm [12][14][15] to train our neural power models because they are very suitable to minimize the error functions that arise from a squared-error criterion. In the following descriptions, we will try to briefly explain the operations of both training algorithms.

3.5.1 Levenberg Marquardt algorithm

For Levenberg-Marquardt algorithm the equations are shown below. In this algorithm basically the input to hidden layer transfer function log sigmoid, whereas hidden to output layer transfer function is pure linear.

$$\text{Log sigmoid---------- } f(s)=\frac{1}{1+e^{-s}}$$

$$F(W)= \sum_{i=1}^{P} (y_i-t_i)^2 \qquad 3.2$$

Equation (3-2) can be rewritten as Equation (3-3), where $\boldsymbol{E}=[e_1\ e_2\ \ldots\ e_P]T$ and $ei=yi-ti$, $i=1,\ldots,P$. If we define the Jacobian matrix as Equation (3-4), the weights in Equation (3-4) can be calculated iteratively using Equation (3-5), where $\boldsymbol{I}$ is the identify unit matrix, u is learning parameter, and r is the number of iterations. A large value of u will lead to a faster learning process but the weights may become more unstable. On the contrary, a small value of u implies a slower learning process but the results are more stable. In our work, we will fixed this parameter as 0.003.

$$F(\boldsymbol{W}) = \boldsymbol{E}^T \text{ E} \qquad 3.3$$

$$W_{r+1}=W_r-(J_r^T J_r+u_r I)^{-1} J_r^T E \qquad 3.4$$

$$J=\begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \frac{\partial e_1}{\partial w_3} \ldots\ldots & \frac{\partial e_1}{\partial w_Q} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \frac{\partial e_2}{\partial w_3} \ldots\ldots & \frac{\partial e_2}{\partial w_Q} \\ \frac{\partial e_3}{\partial w_1} & \frac{\partial e_3}{\partial w_2} & \frac{\partial e_3}{\partial w_3} \ldots\ldots & \frac{\partial e_3}{\partial w_Q} \\ \frac{\partial e_p}{\partial w_Q} & \frac{\partial e_p}{\partial w_Q} & \frac{\partial e_p}{\partial w_Q} \ldots\ldots & \frac{\partial e_p}{\partial w_Q} \end{bmatrix} \qquad 3.5$$

$\boldsymbol{W}^{m}$ is weight of the mth layer of the network, and $\boldsymbol{b}^{m}$ is bias of the mth layer of the network.

Steepest Descent Algorithm

Backpropagation algorithm is used as the training method of the designed artificial neural network. The backpropagation algorithm includes the following steps:

1. Initialize weights and biases to small random numbers.

2. Present a training data to neural network and calculate the output by propagating the input forward through the network using (11).

3. Propagate the sensitivities backward through the network:

s^{M}=-2$F^{M}(n^{M})$(t-a)

4. Calculate weight and bias updates

$$\Delta W^{m}(k) = -\alpha S^{m}(a^{m-1})^{r}$$

$$\Delta b^{m}(k) = -\alpha S^{m}$$

Where α is learning rate

5. Update the weights and biases

W^{m}(k+1)=W^{m}(k)+ $\Delta W^{m}(k)$

b^{m}(k+1)=b^{m}(k)+ $\Delta b^{m}(k)$

6. Repeat step 2 – 5 until error is zero or less than a limit Value.

3.6 The Properties of Neural Network

i. Fast processing time

Because of the parallel structure, all neurons will perform their computation concurrently in the ideal artificial neural networks. Therefore, the processing time of a neural network is very fast. If we refer the speed of the processing time as intelligence, the comparison of "intelligence" is shown in Figure 3-5[13][10].

ii. High storage capacity

Because of the highly connected neurons, it can have amazing large memory storage according to the Kolmogorov theory [11]. It showed that every continuous function of several variables with a closed and bounded input domain can be represented as the superposition of a small number of functions of one variable.

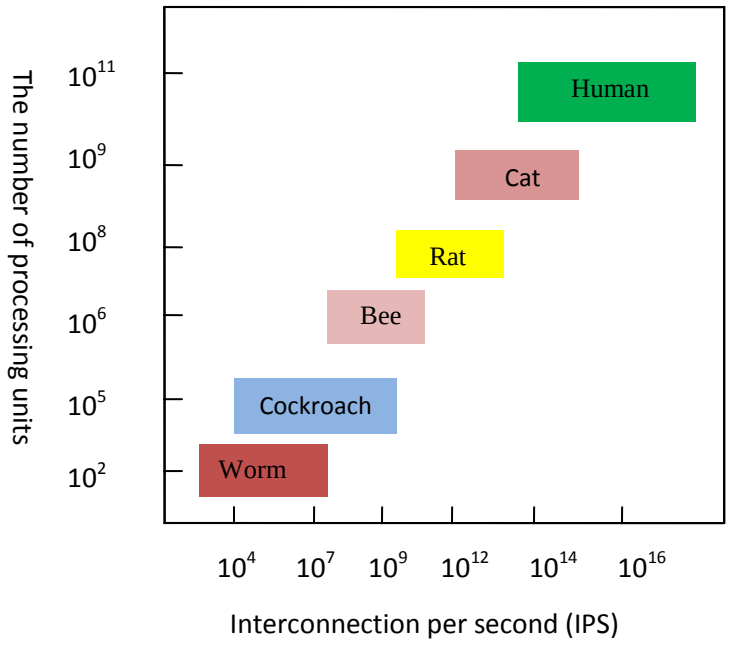

Figure 3-0-7 The comparison of neural creatures

iii.Learning capability

A neural system may learn the rules simply from a set of examples. The learning capability of a neural network enables it to give a satisfactory response for an input which is not part of the set of training examples.

iv. Distributed memory and fault tolerance

In neural networks, “memory” corresponds to an activation map of the neurons. Memory is thus distributed over many units that gives resistance to noise. In distributed memories, such as neural networks, it is possible to start with noisy data and recall the correct data. Distributed memory also has a benefit for fault tolerance. In most neural networks, if some PEs are destroyed or altered slightly on their connections, the behavior of the entire network is not changed too much.

Summary

It is noted that neural network is not an all-purpose solution. The premise to use this algorithm is that the problem to solve is a learnable case. It means that there must exists a relationship between the inputs and outputs of neural network; otherwise, it will become hard to be trained in contradiction cases. In our case, we are going to build the power model by using neural network to learn the relationship between real power dissipation and input/output status. In the CMOS digital circuit, the total power dissipation is dominated by dynamic power dissipation, which is input pattern-dependence. It implies that this kind of relationship should be able to be learned.

Chapter 4

Power Modeling with Neural Network

4.1 Introduction

In chapter 3 we have discussed the basic feed forward neural network and training algorithms. After detailed analysis, we have realized that the logic level of the unchanged signal must be considered because they might be a control signal that controls the internal signal switching propagation. We will focus not only on the the novel modeling approach but also look into the parameters of neural network that are required for the basic building block of the algorithm. power modeling research is to develop a model that can be used easily and efficiently and has enough accuracy. It means that, the characterization process must be as simple as possible.

As mentioned in Chapter 3, we choose the fully feed forward connection configuration to be our neural network architecture in our first study.The first parameter to be decided is the input data format, which is mentioned as the characteristic value of the input patterns in the circuit. Because the total power dissipation of CMOS circuit is dominated by dynamic power dissipation that depends on the circuit input switching activity, the total power dissipation will also depend on the switching activity. Therefore, in the first try of our work, we choose the transition status of every input pin and output pin as the input data of our power model.

The minimal number of neurons in the hidden layer depends on the complexity of the relationship between input data and output data. 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 neurons to be used [17]. Therefore, in this work, we start from a small number and add more neurons until the neural network can learn the properties with desired accuracy. In our experiment, h will be small than 15. It shows that, the complexity of our neural power model is only linearly proportional to the number of inputs and outputs. Finally, the output layer has only one neuron. Its output is the estimation for the power consumption of this circuit under given input patterns. The overall picture of this neural power model is shown in Figure 4.1.

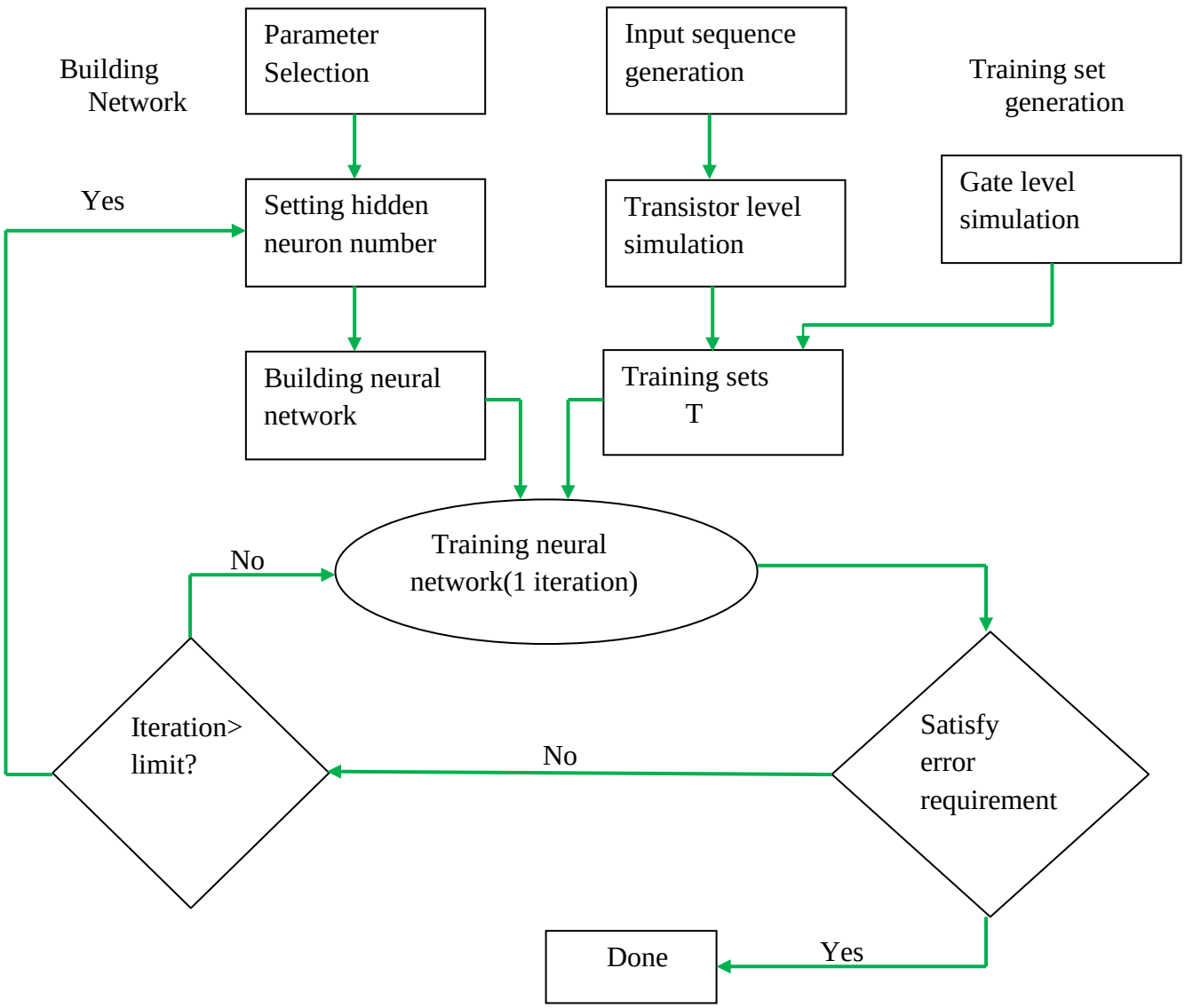

Figure 4-1Power model construction procedure

4.2 Parameters of Neural Network

According to the experience we learned by reading different articles, we will try to consider the signal transition statistics as well as probability values at the inputs and outputs pins. We set the inputs of the neural network as some real numbers between one and zero, which are the signal transition statistics of an input pattern pair and its corresponding output pattern pair individually. Therefore, the number of neurons in the input layer is fixed as 8, which are PI_{00}, PI_{01}, PI_{10}, PI_{11}, PO_{00}, PO_{01}, PO_{10} and PO_{11} that represent the ratio of each case in this pattern pair. Here, PI_{xy} represents the ratio of input signals change from logic state x to y and PO_{xy} represents the ratio of output signals change from logic state x to y in a pattern pair. It can be noted that $PI_{00}+PI_{01}+PI_{10}+PI_{11}=1$ and $PO_{00}+PO_{01}+PO_{10}+PO_{11}=1$. An example of the input data format is shown in Figure 4.2

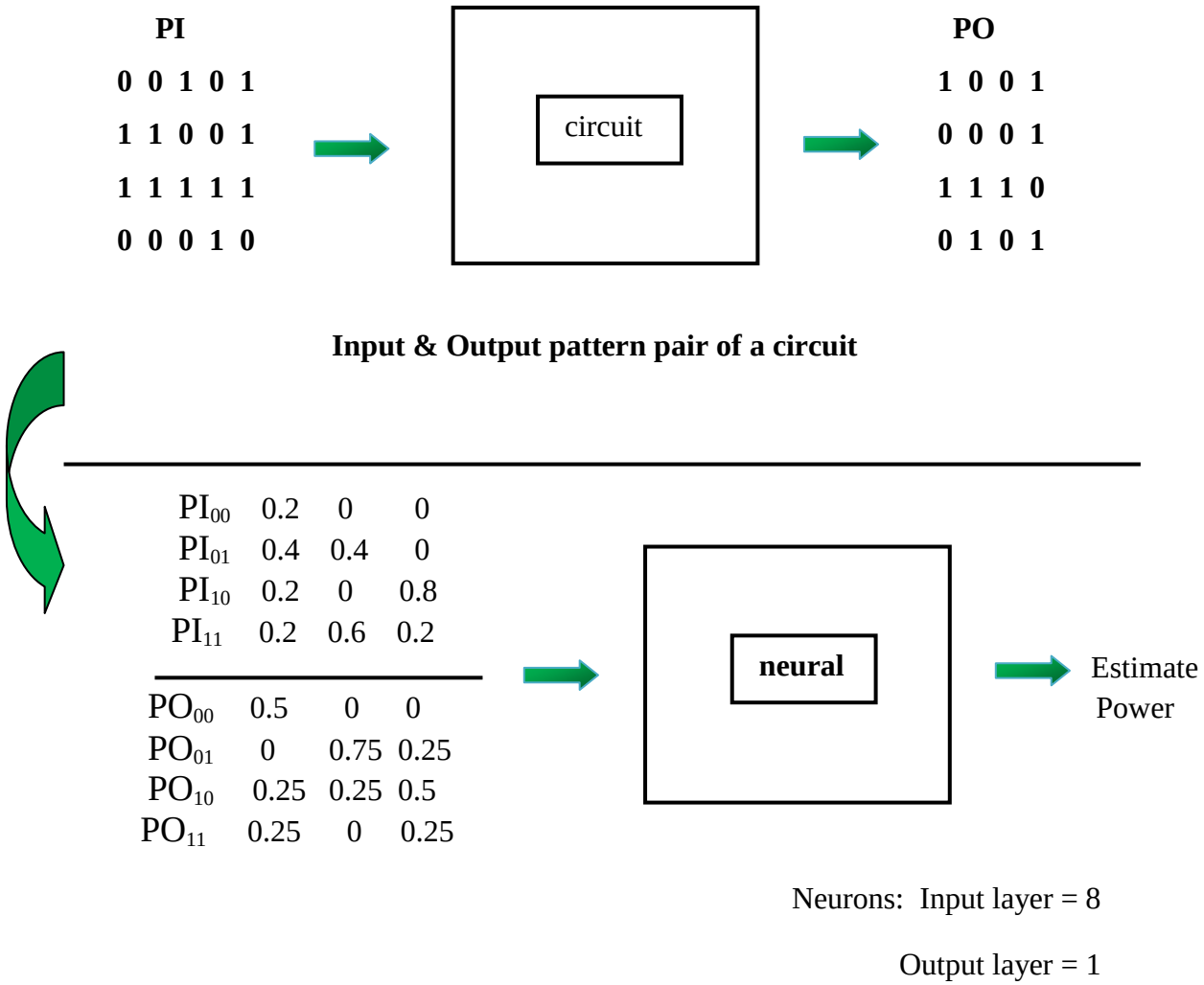

Figure 4-2An example of the signal transition statistics

Therefore, the model will be built as shown in Figure 4.3 . The complexity of this neural power model has no relationship with circuit size and number of inputs and outputs such that this power model can be kept very small even for complex circuits.

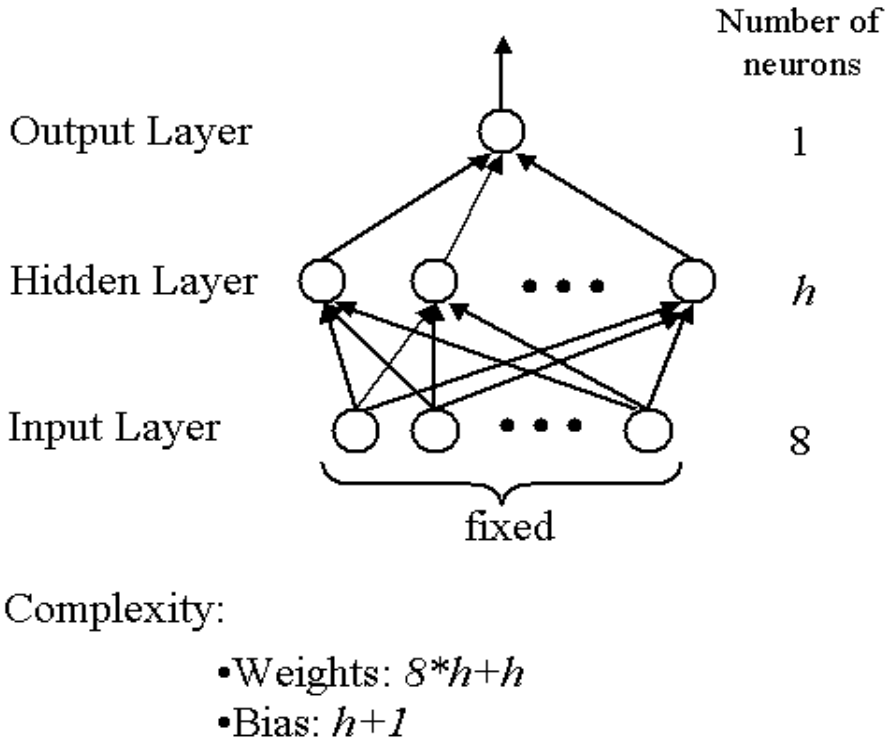

Figure 4-**3**Illustration of the neural power model

Chapter 5

Experimental Design

5.1 Introduction

In this section, we will demonstrate the accuracy and efficiency of our power model with ISCAS'85 benchmark circuit C432, which is synthesized by Synopsys Design Compiler using 0.065*u*m cell library. The neural networks for the power models of those circuits are built on TURBO C++ and performed on a laptop with Intel **Centrino** Duo 1.6GHz CPU and 1GB RAM.

In the training phase, the input sequences are randomly generated by iteratively changing the average signal transition density such that the neural models can learn many different cases. In our experiments, the neural networks of the circuit are trained with 65 input patter pairs. The real power of those input pattern pairs is estimated by Power Compiler such that dynamic power dissipation can be characterized in the power model.

The mean square error and the learning rate of the training target is set as 10^{-8} and 0.003 respectively. When the training process goes through the whole training set, we will check the mean square error of the estimation results. If the error is not small enough, the training process will be executed again until the training target is satisfied.

5.2 Bench mark circuit description(C432)

Statistics: 36 inputs; 7 outputs; 160 gates; bus translations

Function: c432 is a 27-channel interrupt controller. The input channels are grouped into three 9-bit buses (we call them A, B and C), where the bit position within each bus determines the interrupt request priority. A forth 9-bit input bus (called E) enables and disables interrupt requests within the respective bit positions. The figure above concisely represents the circuit. The figure above contains the modules labeled M1, M2, M3, M4, and M5, which contain the underlying logic.

The interrupt controller has three interrupt request buses A, B and C, each having nine bits or channels, and one channel-enable bus D. The following priority rules apply: A[i] > B[j] > C[k], for any i, j, k; i.e., bus A has the highest priority and bus C the lowest. Within each bus, a

channel with a higher index has priority over one with a lower index; for example, A[i] > A[j], if i > j. If D[i] = 0, then the A[i], B[i], and C[i] inputs are disregarded.

The seven outputs PA, PB, PC and out[3:0] specify which channels have acknowledged interrupt requests. Only the channel of highest priority in the requesting bus of highest priority is acknowledged. One exception is that if two or more interrupts produce requests on the channel that is acknowledged, each bus is acknowledged. For example, if A[4], A[2], B[6] and C[4] have requests pending, A[4] and C[4] are acknowledged. Module M5 is a 9-line-to-4-line priority encoder. The output line numbered 421 actually produces the inverted out[3] response of that shown in the truth table. We have taken the liberty of adding an inverter to output 421 to form out[3] for this table (but not in the models).

I/O bus	Function	ISCAS-85 Netlist numbers
A[8:0]	Highest priority input bus	1, 11, 24, 37, 50, 63, 76, 89, 102
B[8:0]	Middle priority input bus	8, 21, 34, 47, 60, 73, 86, 99, 112
C[8:0]	Lowest priority input bus	14, 27, 40, 53, 66, 79, 92, 105, 115
D[8:0]	Channel enable input bus	4, 17, 30, 43, 56, 69, 82, 95, 108
PA,PB,PC	Requesting bus output	223, 329, 370
Out[3:0]	Requesting channel output	421, 430, 431, 432

Table 5-0-1C432 benchmark circuit pin description

5.3 Power Estimation Techniques

Power values for each of these macros are done using four power tools of Synopsys spread through three levels of abstraction, RTL level, Gate level and Transistor level and in overall 5 different values for a macro being calculated. Power calculation for each of the tools at a specific level is done using a different methodology and with other non-power tools involved. One of the major non-power tools involved in this is an extraction tool. A table is built summarizing all the values.

The first method of power calculation is done using Power Estimator which is used at the RTL level. The second method involves using Power Compiler with RTL level switching activity and the third method involves using Power Compiler with Gate Level switching activity. The fourth method is by using Prime Power which also comes at Gate level. The final and the most accurate fifth method is by using NanoSim which is at the transistor level. The accuracy of the power values obtained using these tools gets better as we move from RTL level to transistor level. This is because the information required for calculating accurate power of a macro is given in more detail as the level goes to the lower levels of abstraction and also the tools involved get more complex at those levels. Finally, a table is made with power values filled for each of the macros together with the simulation time required to get those.

5.4 Basic design flow

The following Figure 3.1 gives a basic idea of the design flow that takes place from code writing of the macro to sending the final macro output for fabrication. Initially to start with the VHDL or Verilog hardware description language is used to describe the design. The design is verified using one of the different simulators to test its functionality. Once the test is fine, the next process of creating the net-list is carried out. The gate-level net-list is created using Design Compiler. Additionally, the power tools are used to estimate power at different levels depending on the tool used at a specific level of abstraction. The next sections in this chapter describe the process and methodology used in each of the power tools and how the power is calculated.

The following are the different methods of calculating power

a) Power Estimator using RTL level switching activity (Pre-Synthesis)

b) Power Compiler using Gate level net-list with RTL level switching activity

c) Power Compiler using Gate level net-list with Gate-level switching activity

5.5 Power Estimation at the Register Transfer Level

The RTL Power Estimator enables to obtain design power estimates early in the design process. Its pre-synthesis simulation capabilities enable to analyze the power consumption of the design at the RTL. These Architectural or RTL level tools can be used to quickly understand which modules in the entire design consume the largest amount of power. This is also the best level to evaluate the usage of clock gating strategies which are primarily used to reduce power consumption. The run time efficiency of running the tools at this level is also used to calibrate the fastness of the tool. Some of the features of using Power Estimator are

a) Obtain quick power estimation early in the design

b) Perform architectural tradeoffs early in the design flow

5.5.1 Methodology

The following is the approach that has been followed to calculate power using Power Estimator which is part of the Power Compiler tool. Figure5.1 gives the flow. As shown in the figure, the RTL design is first taken. There are two flows from the RTL design. One is the RTL code which is simulated using ModelSim simulator to get Back Switching SAIF file which contains the switching activity of the design and it is used to create power model for the design using "**create_power_model**" command. Then the design is annotated using the back annotated switching activity and power is reported using "**report_rtl_power**" command. All the commands can be added up in a script which can be used by invoking "**pp_shell**" command.

5.5.2 Capturing Forward and Backward Switching Activity

Power Compiler requires information about the switching activity of the design to do power analysis. The forward and back-annotation files are in SAIF format. SAIF is an ASCII format developed at Synopsys to facilitate the interchange of information between simulators and Synopsys power tools. Some of the power tools cannot understand SAIF file so in that case VCD file is used. Depending on the tool, either RTL level switching activity or Gate-level switching activity is used. Power Compiler has a methodology that enables the use of switching activity from RTL simulation as well as from Gate-level simulation. Using gate-level simulation the power values are much more accurate but doing that is time consuming. During RTL and gate level simulation the designer can direct the simulator to monitor and write out the switching

activity of certain important elements in the design. For accurate analysis, synthesis-invariant elements should be closely monitored during RTL simulation. These are the elements that are not changed during simulation like primary inputs, sequential elements, black boxes, three-state devices and hierarchical ports.

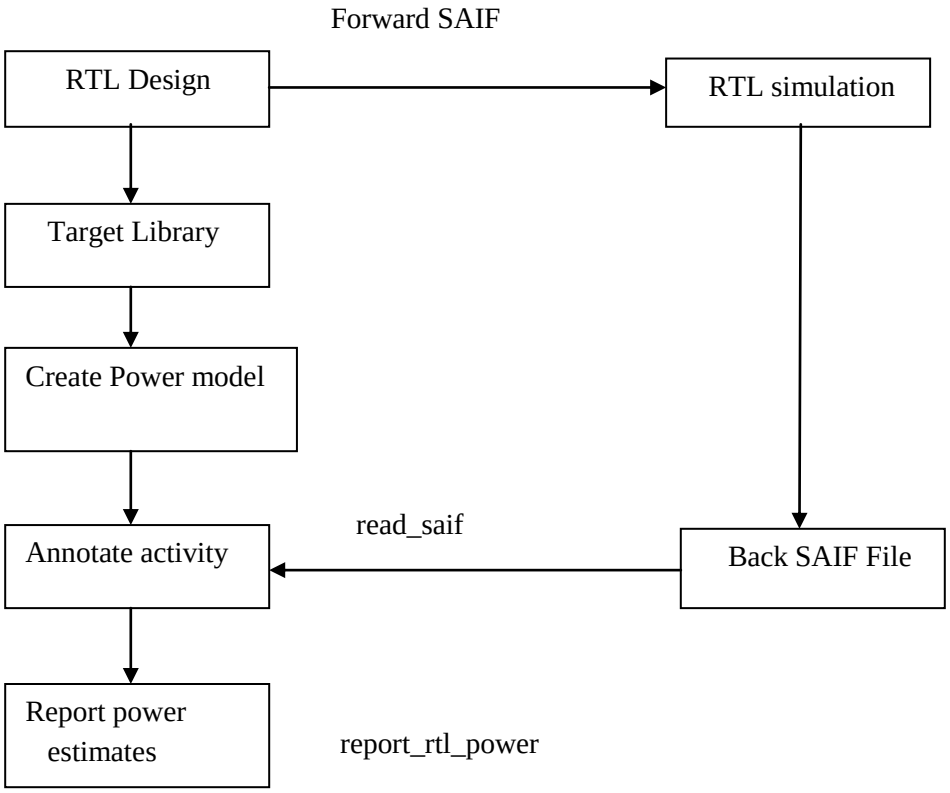

Figure 5-0-8Power Analysis flow in Power Estimator

5.5.2.1 SAIF file and RTL simulation

A SAIF forward-annotation file directs the simulation to monitor primary inputs and other synthesis-invariant elements. The backward SAIF file generated from the simulation contains the resultant switching activity of the elements monitored during the RTL simulation. Synopsys power tools can read the information in the back-annotation file and annotate it on the compiled design. The following steps as shown in the Figure 5.2 are done to get forward and finally the back switching activity file

a) Set the variable "**power_preserve_rtl_hier_name = true**"

b) Create a SAIF forward-annotation file from "**dc_shell**"

c) Include the SAIF forward-annotation file in simulation using ModelSim

d) Write a SAIF back-annotation file from simulation

e) Read the SAIF back-annotation file to annotate the design from "**dc_shell**"

As the design is analyzed and elaborated, HDL compiler creates a technology-independent design called GTECH design. Using GTECH design, HDL compiler creates the SAIF forward-annotation file when invoking the "**rtl2saif**" command.

The following is the methodology followed using RTL simulation and SAIF files.

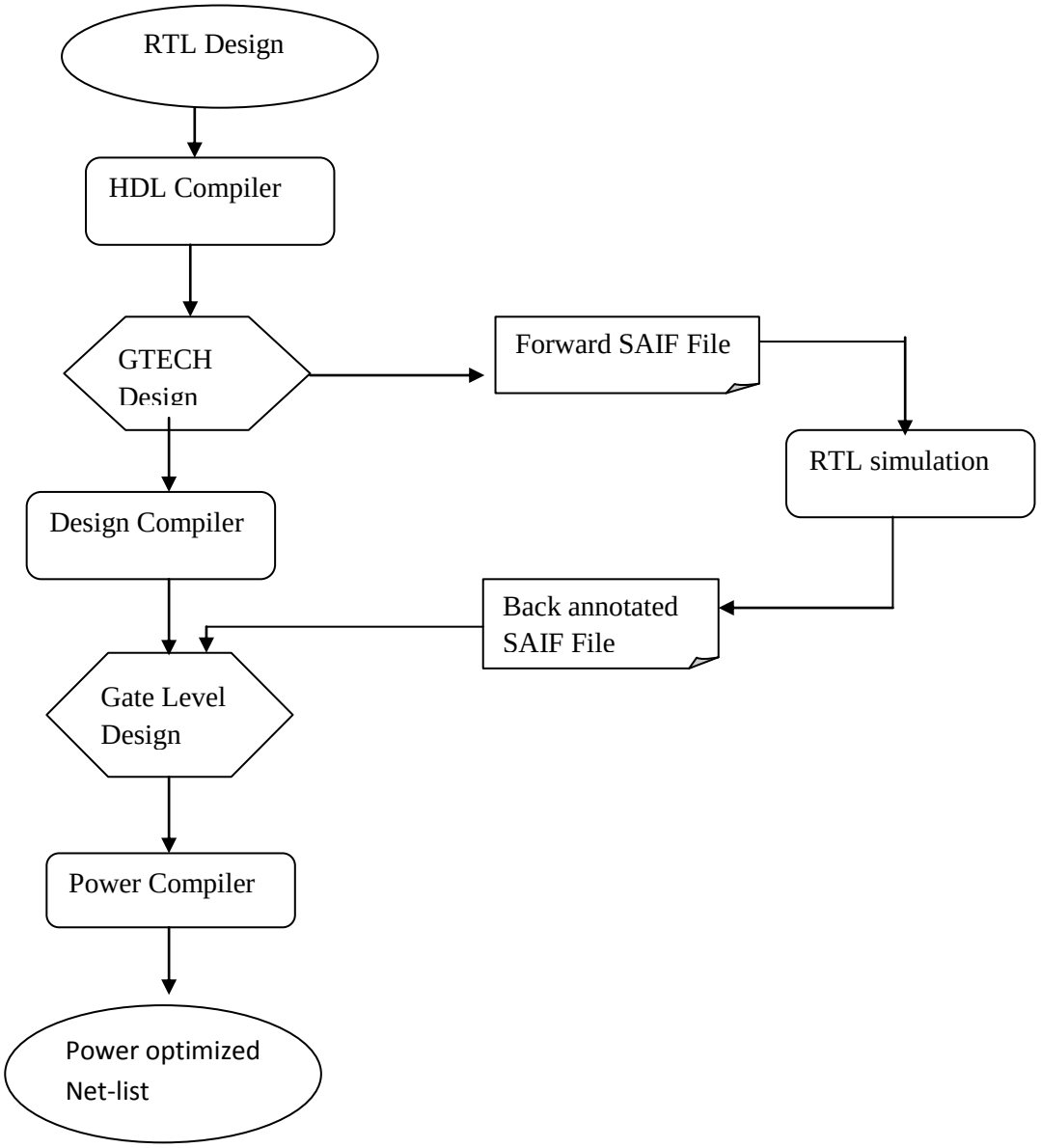

Figure 5-0-9Methodology using RTL simulation and SAIF file

5.5.2.2 SAIF forward-annotation file

The following script has been used to create forward annotation file for "adder" design.

" **power_preserve_rtl_hier_names = true**
analyze -f verilog {c432.v}
elaborate c432
link
rtl2saif -output c432_forward.saif -design c432 "

The following is the explanation of each of the command lines in the script. To start with the "**dc_shell**" command is used to invoke the Design Compiler.

a) **power_preserve_rtl_hier_names = true**

This variable is set true to preserve the hierarchy information of the RTL objects in the RTL design.

b) **analyze -f verilog {c432.v}**

elaborate c432

The analyze and elaborate commands read the RTL design into active memory and converts it to a technology-independent format called the GTECH design.

c) **link**

The link command resolves instantiated references of the sub designs.

d) **rtl2saif -output c432_fw.saif -design c432**

The **rtl2saif** command creates the forward-annotation file using the GTECH format created during the analysis and elaboration of the RTL design. Here "**c432_fw.saif**" is the forward-annotation file for adder.

5.5.2.3 Creating Backward SAIF file

Now for Power Estimator to report power, Backward SAIF file is required which is obtained using Forward SAIF file. Modelsim simulator is used to create the backward SAIF file. First, the Verilog of C432 along with the test bench are compiled and then the ModelSim simulator is invoked. Forward switching activity file generated by "**rtl2saif**" command as part of the Design Compiler is also fed to the simulator. The "**read_rtl_saif**" command reads the SAIF forward-annotation file and registers design objects for monitoring. The next subsection describes about the toggle command methodology in detail. The "**toggle_report**" command creates a SAIF back-annotation file from simulation. The back-annotation file contains information about the switching activity of the synthesis-invariant elements in the design. The "**read_saif**" dc_shell command back-annotates the information from the SAIF file onto the current design. Figure 5.3 shows the steps involved in creating the backward SAIF file.

5.6 Power Estimation using Power Compiler with RTL switching activity

Power estimation at gate level using gate level power estimation tools is the next accurate method in calibrating power. These tools operate on the gate level net-list of the design together with the gate level power library. The power library consists of power models for each of the gates like inverters, NAND gates, and flip-flops

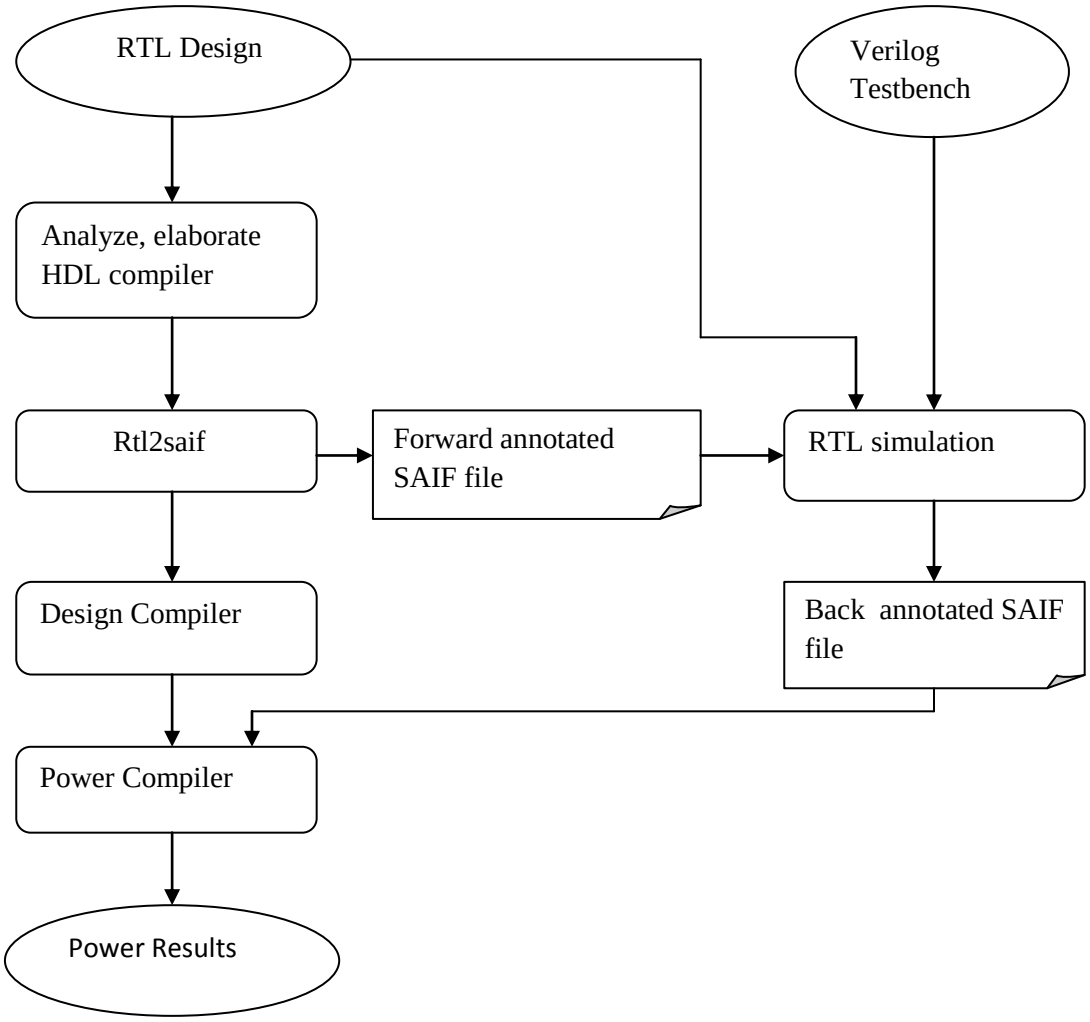

Figure 5-3RTL backward switching activity using ModelSim

These models consists information about the parameters that contribute to power dissipation in each of the standard cells. In this thesis, Power Compiler is used as the gate level power estimation tool. Power Compiler not only estimates the power but also helps in optimizing the design for lower power. The gate level power consumption checks the power being

consumed by logic transitions on wires and by capacitances and short circuits internal to gates during an input transition. In the case of smaller design, the designer can do some gate level changes to reduce power after estimating. If it is a larger design then it would be difficult for the designer to check all the gate-level changes. At this point, Power optimization tools come in handy. Power Compiler is also an optimization tool

5.6.1 Methodology

In this method of power estimation, Power Compiler is used with the same RTL back-annotation switching activity used for power estimation using Power Estimator but instead of RTL code, it uses gate-level net-list of the design.

Also for getting better power result, parasitic information of the system is also provided. In this case DSPF is obtained from Place and route tool, Soc encounter using HyperExtract Extraction tool. The gate-level net-list is obtained from Design Compiler. The following script has been used to report power.

" **read -f verilog -net-list c432_syn.v**
current_design c432
create_clock -name clk -period 100 -waveform {0 50}
read_parasitics -format DSPF c432_syn.dspf -elmore
read_saif c432_fw.saif -instance c432
report_power > power_report_RTL "

As shown in the script first the gate-level net-list of the adder design obtained from Design Compiler is read inside the "**dc_shell**" environment. Depending on the clock frequency used, it has been assigned using the "**create_clock**" command. The parasitic is read in the form of DSPF file using "**read_parasitics**" command. Then the backward SAIF file is loaded using "**read_saif**" command. Then finally "**report_power**" command is used to report the power. Depending on the design, extra commands may be required in this script especially for designs having a clock tree. Designs having clock tree will report high fanouts when run in this environment. Additional commands will enable to remove the high fanouts.

5.7 Power Estimation using Power Compiler with Gate-level Switching activity

Another method of calculating power of a design which is more accurate than the previous Power Compiler method is to use gate-level net-list with gate-level switching activity. This

method is better than the previous method because it uses the gate level net-list to get the switching activity of the design, but the time taken to do this procedure is more than previous two methods.

5.7.1 Creating Gate-level Switching Activity

The following Figure 3.6 shows the flow required to get the Back annotation gate level switching activity which will be later used to calculate power. The main difference between RTL back annotation switching activity and gate-level switching activity is that here gate level net-list is given as the input to the ModelSim simulator along with the testbench and the do file which contains all the toggle region definition and the actual running of the simulation and the reporting of the toggle activity. The resultant back-annotation SAIF file is read back to Power Compiler and power is reported. The do file that is used to capture switching activity follows the same procedure as RTL switching activity like defining the reading the forward SAIF file, defining the region for counting toggle information, starting and stopping the monitoring switching activity and finally using "**toggle_report**" command to report the activity in a SAIF file format.

read -f verilog -net-list c432_syn.v
current_design c432
create_clock -name clk -period 100 -waveform {0 50}
read_parasitics -format DSPF c432_syn.dspf -elmore
read_saif -input c432_bw.saif -instance testbench/design
report_power > power.rpt "

First the gate level net-list is read into dc_shell environment. Once the net-list is read the top level of the design is made as the current design to work on it. Then the clock is created depending on the frequency is run while calculating the power. Then a certain load is given to the output port which in this case is SUM. Then the parasitic values are read into as DSPF form. Then the backward annotation file is read which has the switching activity of the design. The switching activity file gives information to the tool at which points there is switching in the design.

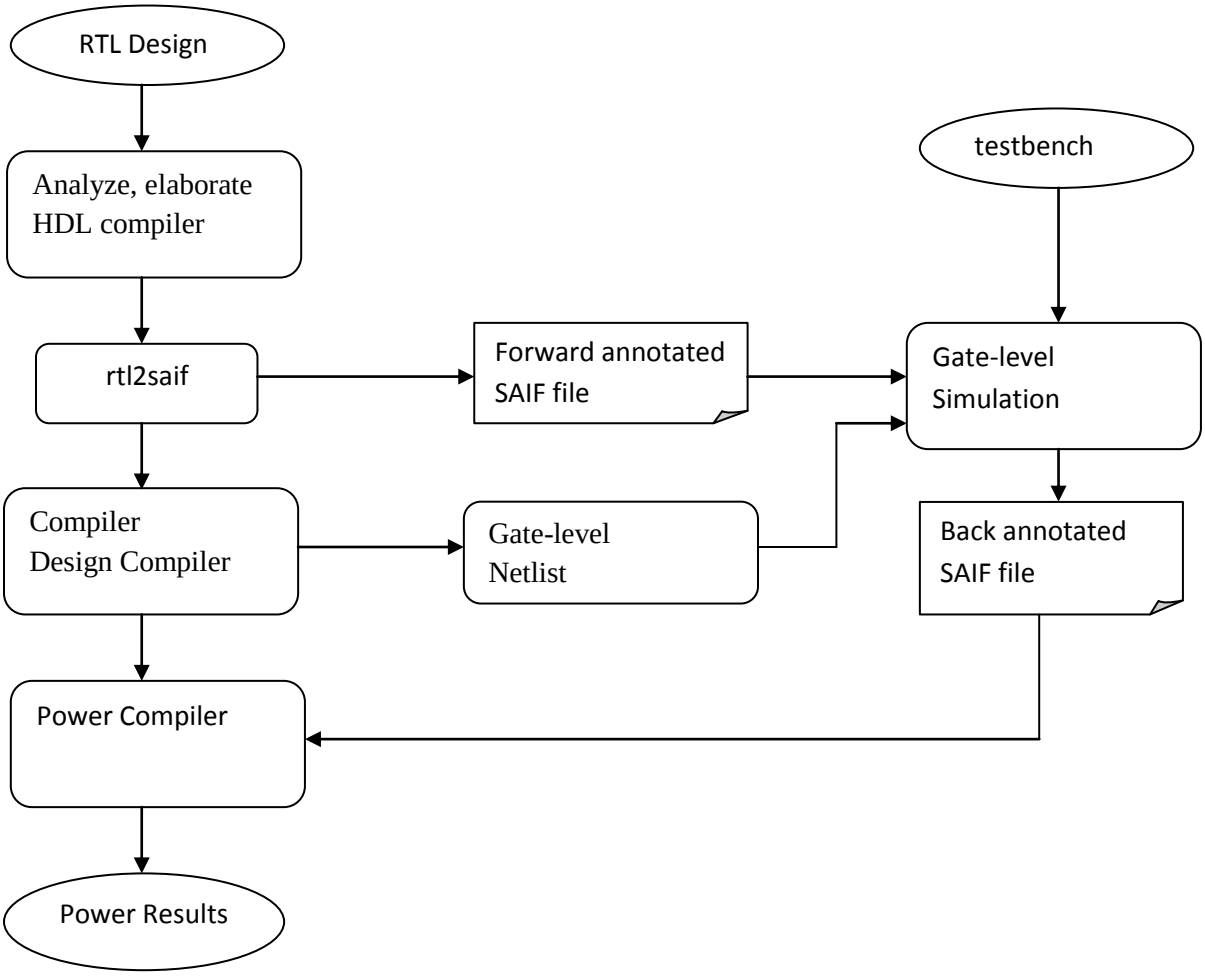

Figure 5-4Gate-level backward switching activity using ModelSim

This is useful to report power of the design. "**report_power**" command is used to report the power of the design. This method gives power values much more accurate the other previous methods. Next method discussed is by using another Gate-level Power Estimator using almost the same input files except that it takes in the switching activity as VCD format. This tool supposed to give almost equal power compared to Power Compiler using Gate level switching activity.

Chapter 6

Results & Discussions

This chapter gives details on the various results that have been obtained using the benchmark circuit C432 that was discussed earlier.

The figure1.2 shows the methodology of calculating different power values at the different levels of abstraction. Detailed methodology of how different power values are calculated using these tools has been discussed in chapter 2.

The following section discusses the different power values that are obtained using different power tools as shown in the figure1.2.

1. **Power Estimator – P1 (RTL) :**

Power Estimator is used to calculate power at the RTL level. The inputs for Power Estimator are Verilog[1], RTL switching activity[2]. The input [4] is got from ModelSim giving [1] + [2] as inputs.

2.Power Compiler – P2 (RTL) :

The second power value is calculated using Power Compiler at the RTL level. The inputs to calculate power are Gate-level Net-list [3], RTL switching activity [4]. [3] is obtained from Design Compiler giving [1] as the input. [4] is obtained from ModelSim giving [1] + [2] as inputs.

3. **Power Compiler – P3 (Gate-level) :**

The third power value is calculated using Power Compiler at the Gate-level. The inputs to calculate power are Gate-level Net-list [3], Gate-level switching activity [5], Parasitic information [6]. [3] is obtained from Design Compiler giving [1] as the input. [5] is obtained from Modelsim using [2] + [3] as inputs. [3] is given as input to Silicon Ensemble to do the Place and Routing and after that [6] is obtained from Soc encounter.

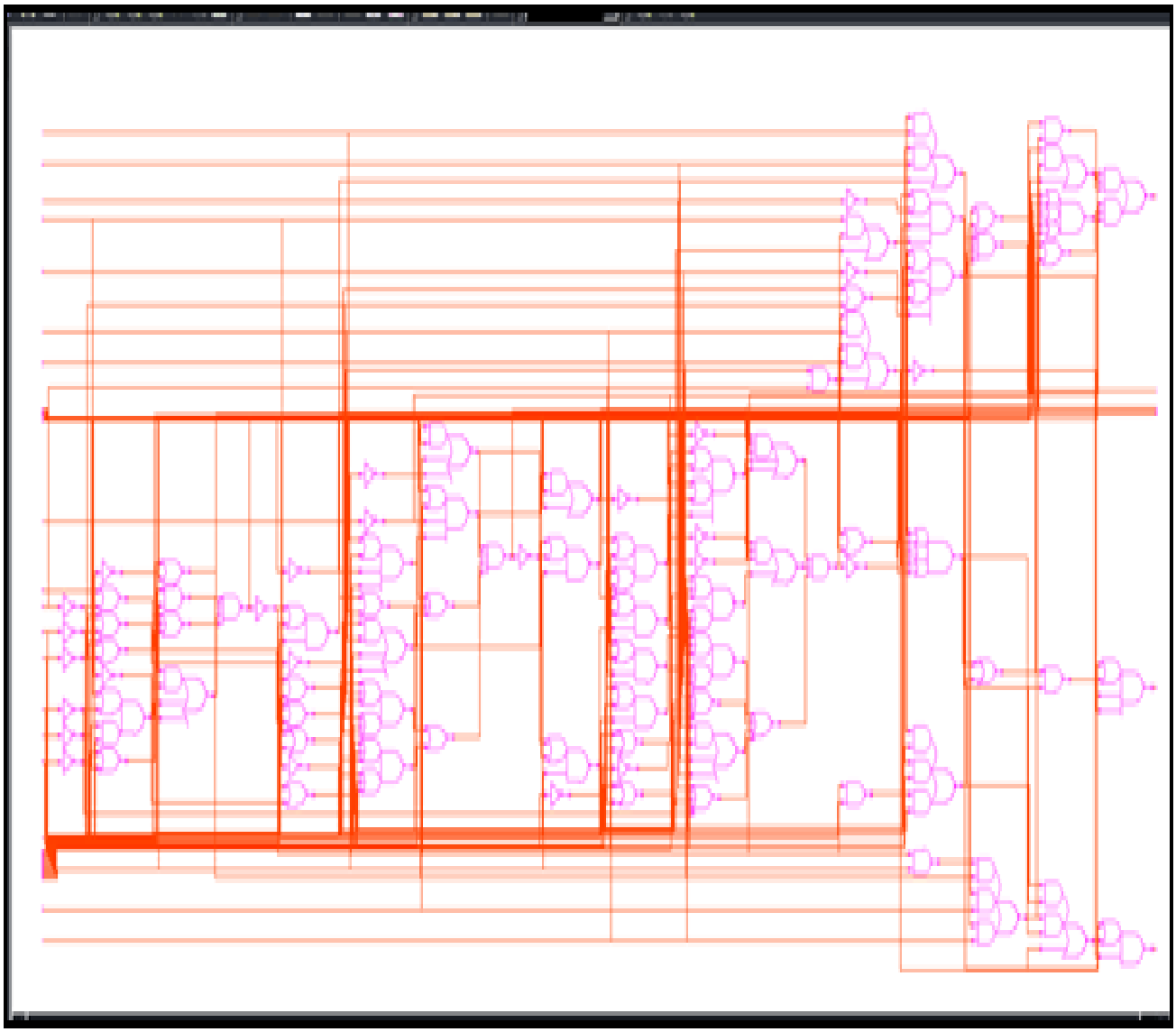

Figure 6-1Gate level net-list of benchmark circuit c432

6.1 Power Report

6.1.1 RTL power report

```
****************************************
Report : power
        -analysis_effort low
Design : c432
Version: B-2008.09
Date   : Mon Mar 28 10:03:22 2011
****************************************


Library(s) Used:

    tcbn65gplustc (File: /home/NIS/tcbn65gplustc.db)



Operating Conditions: NCCOM   Library: tcbn65gplustc
Wire Load Model Mode: segmented

Design        Wire Load Model            Library
------------------------------------------------
c432                   ZeroWireload      tcbn65gplustc



Global Operating Voltage = 1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW
--------------------------------------------------------------------------------
Hierarchy           Switch     Int      Leak        Total
                    power      power    power       power        %
--------------------------------------------------------------------------------
c432              277.8240nW 420.4745nW 345.7192nW 698.2985nW    100
```

6.1.2 Power Report using Power Compiler with RTL Switching Activity

```
Information: Updating design information... (UID-85)
Information: Propagating switching activity (low effort zero delay
simulation). (PWR-6)


****************************************
Report : power
        -analysis_effort low
Design : c432
Version: B-2008.09
Date   : Mon Mar 28 10:03:22 2011
****************************************


Library(s) Used:
    tcbn65gplustc (File: /home/NIS/tcbn65gplustc.db)


Operating Conditions: NCCOM   Library: tcbn65gplustc
Wire Load Model Mode: segmented

Design        Wire Load Model            Library
------------------------------------------------
c432                   ZeroWireload      tcbn65gplustc


Global Operating Voltage = 1

Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


Cell Internal Power  = 310.8485 nW   (57%)
  Net Switching Power  = 235.3477 nW   (43%)
                         ---------
Total Dynamic Power    = 546.1962 nW  (100%)
Cell Leakage Power     = 560.6182 nW
```

6.1.3 Gate Level Power Report

```
Information: Updating design information... (UID-85)
Information: Propagating switching activity (low effort zero delay
simulation). (PWR-6)


****************************************
Report : power
        -analysis_effort low
Design : c432
Version: B-2008.09
Date   : Mon Mar 28 10:03:22 2011
****************************************


Library(s) Used:
    tcbn65gplustc (File: /home/NIS/tcbn65gplustc.db)

Operating Conditions: NCCOM   Library: tcbn65gplustc
Wire Load Model Mode: segmented


Design        Wire Load Model            Library
------------------------------------------------
c432                   ZeroWireload      tcbn65gplustc



Global Operating Voltage = 1

Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


Cell Internal Power  = 216.8613 nW   (48%)
  Net Switching Power  = 233.3269 nW   (52%)
                         ---------
Total Dynamic Power    = 450.1882 nW  (100%)

Cell Leakage Power     = 560.6182 nW
```

The following figure 6.2 shows some input and output result for benchmark circuit C432, which was simulated in modelsim and virsim**.**

I/P Data	**O/P data**
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	**0 0 0 0 0 0 0**
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1	**0 1 0 1 0 1 0**
0 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1	**1 1 0 0 0 0 0**
1 0 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 1	**1 1 1 1 0 1 0**
1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 0 1 0 1	**1 1 0 1 0 1 1**
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	**0 0 0 0 0 0 0**
0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 1 0 1 1	**1 1 1 1 0 1 0**
0 1 0 1 1 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 0 1 1 1 0 0 0 0	**1 1 0 0 0 1 1**
0 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1	**1 1 0 0 0 0 0**
1 0 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 0 0 1	**1 0 0 1 1 0 1**
1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 0 1 0 1	**1 1 0 1 0 1 1**
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	**0 0 0 0 0 0 0**
0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	**1 0 0 1 1 1 1**
0 1 0 1 1 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 0 1 1 1 0 0 0 0	**1 1 0 0 0 1 1**
0 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0	**1 1 1 1 0 1 1**

Figure 6-0-10I/P & O/P for circuit C432

6.2 Input data for Neural power model

Transition data between two input patterns	Power Dissipation(nw)
0.083, 0.917, 0, 0, 0.5714, 0.4286, 0, 0	0.0218746
0,0.083 ,0.194, 0.722, 0.4286, 0.1428, 0.2857, 0.1428	0.0195852
0.166, 0.083, 0.27, 0.472 ,0.2857 ,0.4286, 0 ,0.2857	0.0198687
0.25,0.194,0.138 ,0.416,0.1428,0.1428 ,0.1428, 0.5714	0.0181678
0.3889 ,0 ,0.6111 ,0, 0.2857, 0 ,0.7143 ,0	0.0253467
0.3055 ,0.6944, 0, 0, 0.286 ,0.714 ,0, 0	0.0285293
0.111,0.194,0.278 ,0.417 ,0.143, 0.143 ,0.286, 0.428	0.0191845
0.194 ,0.194, 0.056, 0.556, 0.428, 0, 0.286, 0.286	0.0107953
0.111,0.139, 0.222, 0.528 ,0.286, 0.428, 0.143 ,0.143	0.0206794
0.194,0.139, 0.194, 0.472 ,0.143,0.286 ,0.143, 0.428	0.0207974
0.3889 ,0 ,0.6111 ,0 ,0.2857 ,0 ,0.7143 ,0	0.0261919
0.528 ,0.472, 0, 0, 0.286, 0.714, 0, 0	0.0118949
0.194, 0.333 ,0.167 ,0.306 ,0.143,0.143, 0.286 ,0.428	0.0195477
0.278 ,0.111, 0.111, 0.5 ,0.143 ,0.286 ,0, 0.571	0.0094198
0.139 ,0.25, 0.194 ,0.417 ,0, 0.1428 ,0.4286, 0.4286	0.0276176

Table 6-1Data pattern for power modeling approach

The above table 6-1 shows the input patterns which we have used for the training of our neural network power model. 100 numbers of patterns have been taken for the training purpose. A c++ code has been written to find the data pattern for power modeling approach.

6.3 Power comparison

Patterns	Steepest-Descent	Levenberg-Marquardt	Power Compiler
Pattern-1	0.0231747	0.0222549	0.020974
Pattern-2	0.02962	0.02662	0.0195852
Pattern-3	0.0224852	0.024875	0.0208771
Pattern-4	0.0224259	0.0195867	0.0198687
Pattern-5	0.020683	0.020034	0.0181678

Table 6-2 Power comparison

The above table 6-2 shows the power comparison having the unit of nano Watt, between different neural network backpropagation training algorithms and power compiler. These algorithms are tested for 5 different patterns. It is observed that Levenberg-Marquardt training algorithm gives better performance compared to steepest-descent training algorithm. The comparison result shows that small amount of error is occurred, when it is compared with power compiler. This error occurs due to small number of patterns for training. So to get less error we should take more number of samples [35].

Chapter 7

Summary, Conclusions and Future Work

7.1 Summary

Power Estimation for different circuits from RTL level to Gate level using different power estimation tools has been performed. The methodology involving the usage of these tools at different levels of abstraction has been shown with examples. Scripts have been developed for each of these levels to automate the flow for each of the digital circuit involved.. However, these results are still very impressive on the reduction of the power model complexity and the feasibility for a wide range of input signal distribution. The lower complexity can also reduce the characterization time and estimation time sufficiently. We will try to improve this model in the future such that the maximum error can be further reduced.

7.2 Conclusions

It can be concluded from these power estimations at different levels of abstraction how inaccurate values at RTL are compared to Transistor level. The power results obtained using Power Estimator (P1), Power Compiler using RTL Switching Activity (P2), Power Compiler using Gate-Level Switching Activity (P3) used power technology file from TSMC18.

In this thesis, we propose a novel power modeling approach for complex digital circuits, which uses neural networks to learn the power characteristics during simulation. Our neural power model has very low complexity such that this power model can be used for complex circuits. Because of the structures of neural networks, the neural power models can still have high accuracy with simple architectures because they can automatically consider the non-linear power distributions. Unlike the power characterization process in traditional approaches, our characterization process is very simple and straightforward. 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 only test our neural power model on ISCAS'85 benchmark circuits, which are all combinational circuits. The experimental results have shown that the estimations are accurate for wide input range. We may also try to extend our neural power model to the power estimation of sequential circuits such that this approach can be used for any kinds of complex circuits.

How many samples are needed for a good training while building the neural power model for each circuit? This is also an open problem for neural networks. According to the related study [35], it suggested to determine the number of samples according to Equation 7.1 , in which *P* is the number of samples, |**W**| is the number of weights to be trained and *a* is the expected accuracy. In this work, our target is set as $a \geq 95\%$. Therefore, we have to generate the training set with size $P >> 20\mathbf{W}$.

$$P >> \frac{W}{1-a} \qquad 7.1$$

7.3 Future Work

In this dissertation, there are still some improvements could be done in the future. In the power modeling for accurate result, we need to estimate the power at transistor level using Synopsys(Nanosim).For more accurate result we will consider number of samples to be increased according to the equation 7.1.Then we will compare this power modeling approach to other benchmark circuits.

Chapter 8

REFERENCES

[1] W. Goh, S. Rofail, and K. Yeo, “Low-Power Design: An Overview”, Prentice Hall.

[2] D. Soudris, C. Piguet, and C. Goutis, ”Designing CMOS Circuits for Low Power”, Kluwer Academic Publishers, 2002.

[3] F. Najm, “A Survey of Power Estimation Techniques in VLSI Circuits”, IEEE Transaction on VLSI Systems, vol. 2, pp. 446-455, 1994.

[4] K. Roy and S. Prasad, “Low Power CMOS VLSI: Circuit Design”, John Wiley & Sons, 1999.

[5] P. Landman, “High-level power estimation,” *in Proc. Int. Symp. Low Powerb Electronics and Design, Monterey, CA, Aug. 12-14, 1996, pp. 29-35*.

[6] Subodh Gupta and Farid N. Najm. “Power Modeling for High-Level Power Estimation,” *IEEE Transactions on VLSI Systems, pp. 18-29*, Feb. 2000.

[7] A. Chandrakasan, and R.W. Brodersen, “Low Power Digital CMOS Design”, Kluwer Academic Publishers, 1995.

[8] A. Raghunathan, N. K. Jha and S. Dey, “High-Level Power Analysis and Optimization” , Kluwer Academic Publishers, 1998.

[9] M. Nemani and F. Najm, “Towards a High-Level Power Estimation Capabillity,” IEEE Trans. On Computer-Aided Design, vol. 15, pp. 588-598, Jun. 1996.

[10] D. Marculescu, R. Marculescu, and M. Pedram, “Information Theoretic Measures of Energy Consumption at Register Transfer Level,” in Proc. of ACM/IEEE Internal Symp. on Low Power Design, pp. 87-92, 1995.

[11] P. Landman, R. Mehra, and J. Rabaey, “An Integrated CAD Environment for Low-Power Design”, IEEE Design and Test of Computers, pp. 72-82, Summer 1996.

[12] K. Keutzer, and P. Vanbekbergen, "The Impact of CAD on the Design of Low Power Digital Circuits", IEEE Symposium on Low Power Electronics, San Diego CA, Oct. 1994

[13] C-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, A. M.Despain, and B. Lin, “Power Estimation in Sequential Logic Circuits", IEEE Trans. on VLSI Systems , Vol. 3, No. 3, pp. 404-416, 1995.

[14] C. M. Huizer, "Power Dissipation Analysis of CMOS VLSI Circuits by means of Switch Level Simulation", IEEE European Solid State Circuits Conf., pp. 61-64, 1990.

[15] M. Nemani, and F. Najm, "Towards a High-Level Power Estimation Capability ", IEEE Trans. on CAD, Vol. 15, No. 6, pp. 588-598, 1996

[16] L. Cao, "Circuit Power Estimation Using Pattern Recognition Techniques," IEEE/ACM International Conf. on Computer-Aided Design, pp. 412-417, 2002.

[17] T. M. Cover and J. A. Thomas, "Elements of Information Theory," A Wiley-Interscience publication, 1991.

[18] Marculescu, D., Marculescu, R., Pedram, M.,"Information theoretic measures for power analysis"**,***Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , Volume: 15 Issue: 6 , Page(s): 599 –610,* Jun 1999.

[19] Martin T. Hagan, Howard B. Demuth, Mark Beale, "Neural Network Design", *PWS Publishing Company,* 1995.

[20] Jacek M. Zurada, "Introduction to Artificial Neural Systems", *WESTPUBLISHING COMPANY,* 1992.

[21] ModelSim SE User's Manual, Version 5.71, June 2008.

[22] VCS/VCSi User Guide, Version 7.0.2, September 2008.

[23] Michael Chester, "Neural Networks – A Tutorial", *PTR Prentice Hall* 1993

[24] Synopsis's, Design Compiler http://www.synopsys.com/products/logic/design_compiler.html.

[25] Simon Haykin, "Neural Network", Prentice Hall.

[26] W. T. Hsieh, "A Novel High-Level Power Model Using Neural Network," Master Thesis, Dept. Elec. Eng., National Central University, Taiwan, Jun., 2003.

[27] W.-T. Hsieh, C.-C. Shiue and C.-N. Liu, "A Novel Approach for High-Level Power Modeling of Sequential Circuits Using Recurrent Neural Networks," IEEE International Symp. on Circuits and Systems, pp. 3591-3594, 2005

[28] Synopsys's Power Compiler, http://www.synopsys.com/products/power/power_ds.pdf.

[29] Power Compiler User Guide, Version U-2003.03, March 2008.

[30] ModelSim SE User's Manual, Version 5.71, June 2008.

[31] Christopher M. Bishop, “Neural Networks for Pattern Recognition”, *Oxford University Press Inc*., 1995.

[32] Terrence L. Fine, “Feedforward Neural Network Methodology,” *New York: Springer*, 1999.

[33] K. Levenberg, “A method for the solution of certain problem in least square,” *Quarterly of Applied Mathematics*, vol. 11, pp. 164-168, 1944.

[34] 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.

[35] Eric B. Baum and David. Haussler, “What Size Net Gives Valid Generalization?” *Neural Computation*, Vol. 1, pp. 151-160, 1989.

[36]. Chih-Yang Hsu, Wen-Tsan Hsieh, Chien-Nan Jimmy Liu, and Jing-Yang Jou, " A Tableless Approach for High-Level Power Modeling Using Neural Networks", Journal of Information Science and Engineering (**SCI, EI**), vol. 23, no. 1, pp.71-90, January 2007.

[37] W.Bachmann, S.Huss,” Efficient Algorithms for Multilevel Power Estimation of VLSI Circuits,” IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 13, NO. 2, FEBRUARY 2005.

[38] Martin T. Hagan, *Neural Network Design.* Boston: PWS Publishing, 1996.

[39] S. Haykin, Neural Networks: A Comprehensive Foundation (2nd Edition). Prentice Hall, 1999.