MODELING OF GLITCHING EFFECTS IN ESTIMATION OF DYNAMIC POWER CONSUMPTION

A Project Report

submitted by

K NITISH KUMAR & PRAKRUTHI P

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

(ELECTRICAL ENGINEERING)

&

MASTER OF TECHNOLOGY

(MICROELECTRONICS AND VLSI)

DEPARTMENT OF ELECTRICAL ENGINERRING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

June 2014

THESIS CERTIFICATE

This is to certify that the thesis titled **MODELING OF GLITCHING EFFECTS IN ESTIMATION OF DYNAMIC POWER CONSUMPTION**, submitted by **K Nitish Kumar and Prakruthi P**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology and Master of Technology**, is a bona fide record of the research work done by them under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Nitin Chandrachoodan
Research Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: June 13, 2014

ACKNOWLEDGEMENTS

"It is good to have a happy end to a journey; but it is the journey that matters in the end."

Foremost, we would like to express our sincere and heartfelt gratitude to our project guide Dr. Nitin Chandrachoodan for his invaluable mentorship, motivation and support throughout the ups and downs of our project. The stimulating weekly discussions and bimonthly reviews helped us stay enthusiastic and head in the right direction all along. We learnt how to be careful about our assumptions and claims, and the importance of not ignoring the outliers in our analysis. Most importantly, he taught us how to cater to every small detail in the project while having the big picture at the back of our minds.

We are overwhelmed to be a part of one of the exemplary and well-knit departments of IIT Madras. We are grateful to all the professors for bestowing the finest knowledge upon us. We also thank the IE lab and DCF staff for their technical support and resources. It would be incomplete without a special mention to the department coffee shop where a lot of bonds were built and memories were made.

We were also privileged to have a really invigorating and supportive lab environment which kept us focused and simultaneously entertained. We are indebted to Celia and Seetal for their valuable inputs to our project at crucial times. A special note goes to Ananth, Nikhil and Jobin for reminding us to rightly pace ourselves throughout the course of our project. We would also like to credit our beloved lab-head, Karthikeyan, for all the encouragement from resolving minor technical problems to getting late-night coffees and mangoes.

We are particularly thankful to a bunch of special people who have made the good times better and the hard times easier - Meghana and Chitra for making the most of every moment, Jayant and Soorya for all the exhilarating technical discussions, Ankit and Anuja for all the emotional abutment, Shikha and Uday for being there as understanding friends at all times and Abraham for inspiring us to work harder and smarter.

Finally, words cannot express how grateful we are for our parents for all their blessings, their sacrifices made on our behalf and their constant belief in us to strive for excellence. We also thank our siblings for all the love and affection they showered upon us to rejoice every moment of our project experience.

ABSTRACT

KEYWORDS: Process Variation; Dynamic Power; Glitches; Monte Carlo Analysis; Input Variation; Ambiguity Interval Propagation; Probability Propagation

Process variation has an implicit impact on switching energy consumption by altering the propagation delays of gates in the circuit and thereby, resulting in glitches. We perform Monte Carlo analysis to identify the impact of process variation on switching energy consumption. We also estimate the contribution of glitches resulting due to input and process variation to overall switching energy consumption and signify the need to reduce their propagation to subsequent combinational logic in the design. A new approach to identify the nets in the circuit which result in maximal power reduction if glitches at that net are blocked is presented. Implementation of two algorithms, ambiguity interval propagation and probability propagation to estimate the average number of transitions at each net is also shown. A validation of the approach is done by blocking glitch propagation at nets using latches and estimating power consumption.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DC	Design Compiler
VCD	Value Change Dump
SAIF	Switching Activity Information File
SDF	Standard Delay Format
DAG	Directed Acyclic Graph
CMOS	Complimentary Metal Oxide Semiconductor
CAD	Computer Aided Design
SoC	System on Chip
RTL	Register Transfer Level
SPICE	Simulation Program with Integrated Circuit Emphasis
BDD	Binary Decision Diagrams

CHAPTER 1

INTRODUCTION

1.1 Motivation

The desirability of portable electronic systems has increased significantly over the years. The primary aspect of size of such a device is contributed by its battery, which has direct implications by power consumption of the system. The cost of cooling systems has also increased. Operation of systems at higher frequencies impose limitations on the overall power consumption of the system. Increasing density of CMOS circuits has added additional concerns to the problem of power dissipation. Various factors of system design like power-supply sizing, current requirements, heat sink/cooling arrangements and choice of devices within the system are directly determined by the system power consumption. Reduction of power is known to increase reliability of operation of the device. Hence, the power consumption of digital CMOS circuits has become one of the primary design constraints and is being actively researched for several decades now.

Power consumption is no longer a secondary issue in designing CMOS devices. The designer today has to design systems targeting all the three major design requirements simultaneously: *low power, high performance, small area*. The increasing complexity and high-performance requirements makes the task of designing low power systems challenging. This results in a need to make design decisions early and in accordance with power specifications of the system. System designers are now looking at architectural improvements and automated design methodologies from the stand point of implementation, circuit and logic topologies, digital architectures and algorithms used for low power design. This necessitates estimation of power consumption at every stage of system design cycle to ensure meeting of design specifications.

Apart from meeting average power specifications by the design, estimation of peak power consumption is also needed for better circuit reliability. Large amounts of current dissipation in a short span of time may cause problems of excessive heat generation, resulting in circuit damage in a few cases. It may also cause temporary changes in

voltage lines causing operational failure of the circuit. Hence, estimation of maximum power consumption by the design is also crucial for exploring appropriate packaging and cooling options or optimizing power grid networks.

The fact that early and accurate power estimation is continuing to assume greater importance has been the strongest motivation for the present work.

1.2 Problem Statement

The key objectives of our work are:

- To identify and analyze the impact of process variation on the Dynamic Power Consumption of combinational CMOS circuits.
- To demonstrate the contribution of glitches to Dynamic Power Consumption and hence signify the need to minimize the propagation of glitches through large combinational blocks.
- To devise a novel time-efficient approach to identify the high glitch nets in a combinational circuit without Monte Carlo Simulations.
- To propose a heuristic to compute the set of nets in a combinational circuit which would result in maximal reduction of total power consumption of the circuit on blocking the glitches at those nets.

1.3 Related work

Several researchers have looked at the problem of reduction of CMOS power dissipation using varied approaches. Algorithms, both deterministic and probabilistic approaches have been proposed to efficiently estimate power consumption. Impact of process variation on switching energy consumption has also been actively researched for the past few years.

The paper by Najm (1994) presents a survey of various power estimation techniques. It describes the simulation based techniques for power estimation. It also enumerates the drawbacks like time-inefficiency and high dependence on input vectors by such approaches. It then describes the probabilistic approach to power estimation using signal probabilities and Binary Decision Diagrams (BDDs). The paper by Raghunathan

et al. (1996) looks at early power estimation and design optimization at Register Transfer Level (RTL). It analyzes glitch generation and propagation in data path blocks as well as control logic and suggests design optimization techniques for glitch reduction. Another paper by Najm (1995) also looks at power estimation at RTL. It calculates entropy of gates, given only its boolean implementation function and uses this metric to estimate power consumption at RTL. The paper by Anderson and Najm (2004) suggests a method to predict switching activities at nets and node capacitances early for FPGA designs. It presents a mathematical prediction model to estimate switching activities at nets by considering every glitch at a node to be generated at that node or propagated by its fan-in nodes. It also presents a mathematical model to estimate node capacitances early in the design along with noise considerations.

The paper by Benini *et al.* (1998) looks at power modeling at system level. The power manager interacts with the resources in the system and receives requests from the environment. This abstract model is used to estimate system level power information of the design. The paper by Dhanwada *et al.* (2012) presents a new approach called Power Contributor Modeling. In this paper, they separate independent components of power like capacitative switching, short circuit power and leakage power and models are built to estimate the contributions of these components. All the components are summed for individual cells to estimate power consumption. The paper by Qu *et al.* (2000) elaborates a method to estimate power consumption of microprocessors. They empirically build a power data bank, which contains the power information of various instructions and built-in functions. The execution information of the software run on the microprocessor is used along with the power data bank to estimate power consumption. There is some active research being done in behavioral level power estimation (Qu *et al.* (2000)). The paper by Buyuksahin and Najm (2005) suggests a high-level estimation of average switching activity, total capacitance and area by using a Boolean network representation of the design.

The paper by Hao *et al.* (2011) considers spatial correlation in process variation and suggests a methodology to estimate dynamic power consumption. It first establishes that variations in process parameters like channel length affect dynamic power consumption significantly. It then presents a method to compute dynamic power statistically at every net in the circuit using orthogonal polynomials and virtual grid based variables for process parameters with spacial correlation. The paper by Dinh *et al.* con-

siders impact of process variation on dynamic power consumption by modeling transition waveforms at inputs of every gate in the design. This paper also considers partial glitches by capturing probability and timing of events at the nets by segmenting these transition waveforms. The paper by Harish *et al.* (2007) presents a simulation based approach to estimating power in the presence of process variation. The process variation is assumed to have a Gaussian distribution. Simultaneous dependence on multiple process parameters is modeled using the statistical technique of Design of Experiments by performing a 3-level Face Centered Central Composite design.

The paper by Alexander and Agrawal (2009) suggests a simulation based approach to estimate bounds on the number of transitions at every net in the circuit. It assumes a bounded delay model for every gate in the design and propagates ambiguity intervals, which represent the minimum and maximum times during which the gate can have probable transitions due to an applied input vector pair. In this thesis, we implement some of the ideas discussed in this paper and propose some applications for the same. The paper by Meixner and Noll (2014) looks at power estimation at a higher level of abstraction. It proposes a methodology to pre-characterize glitch activities in functional blocks of combinational logic using signal statistics at every net in the design. It introduces new metrics like glitch occurrence, glitch width and glitch time to quantify statistical glitching properties on circuit nets. The major drawback of such an approach would be the necessity for time-consuming characterizations. Hence, this methodology is more suited for generic designs.

Some papers by Tsui *et al.* (1993) and Ghosh *et al.* (1992) suggest probabilistic approaches to computing power consumption in CMOS circuits. The paper by Tsui *et al.* (1993) explains the methodology to estimate static and transition probabilities at every net of the circuit and use these parameters to estimate dynamic power consumption of the circuit. It also accounts for correlation at the inputs of intermediary gates in the design while computing power. One major advantage of their methodology is that the algorithm uses real propagation delays for all the gates in the circuit. An improvisation of this approach has been suggested by the paper by Ghosh *et al.* (1992) where the static and transition probabilities are computed by calculating Boolean functions (disjoint covers) that cause switching at every gate in the design. We combine the ideas discussed in both the papers and implement them in this thesis.

The paper by Monteiro *et al.* (1993) looks at retiming algorithms from the perspective of low power. It suggests an iterative methodology to optimally place flip-flops in the circuit such that the timing constraints are not violated, the intended functionality is not altered and total power consumption is minimized.

1.4 Contributions

- We have analyzed the impact of process variation on switching energy consumption of combinational CMOS circuits using Monte Carlo analysis. We have also differentiated the contributions of variation in total switching energy due to process variation and due to variation in input stimuli.
- We have highlighted the contribution of glitches to switching energy consumption. For this, we have implemented a time-efficient power extraction algorithm for estimating switching power due to each input vector pair, given the gate-level netlist, technology library file and the `VCD` file. With this analysis, we signify the importance of reduction of glitches and their propagation in the circuit for achieving low power designs.
- We have implemented two algorithms, ambiguity propagation algorithm and probability propagation algorithm to estimate the average number of transitions at every net in the design and compute bounds on switching energy consumption. We also built an efficient event-driven simulation tool for zero delay simulation of combinational circuits.
- We have proposed a methodology to estimate the total switching energy of the circuit due to a glitch at one of the intermediary nets in the circuit. This analysis would highlight those nets in the circuit where the glitches need to be blocked for maximum switching energy reduction. We validate these results by estimating power reduction in the circuit when glitches at a net are blocked by a latch.

Scope of the work

- The circuits that we have considered for our analysis are combinational and have a typical gate count ranging between 100 and 1000. We have analyzed the circuits at $45\,\mathrm{nm}$ and $250\,\mathrm{nm}$ technologies.
- We neglect partial glitches and non-linear input waveforms in all the gate level simulations.
- We have assumed inertial delay model for all the gates in the circuit.
- Process variation is modeled by introducing a standard normal variation in propagation delays of all gates and by varying their standard deviations. We have not analyzed the effect of individual process parameters and we have assumed a constant source voltage ($\mathrm{V_{dd}}$).

- Most of our analysis uses only switching power to demonstrate dynamic power and is based on an assumption that internal power is either insignificant or proportional to the switching power. Also, since there is no clock for the combinational circuits, the energy dissipated per transition was considered as an indicator of power consumed, since power consumed is just average energy dissipated per clock period multiplied by the clock frequency.

1.5 Organization of the thesis

The reminder of the thesis is organized as follows:

Chapter 2: Power dissipation in CMOS circuits

This chapter describes the various components of power dissipation in CMOS circuits. It also describes the typical VLSI Design Flow and the need for power estimation at various stages of chip design. An introduction to glitches and impact of process variation on switching energy consumption is also presented in this chapter.

Chapter 3: Transistor Level Analysis

This chapter details the analysis of power dissipation in circuits at the transistor level. Spice characterization of gates is explained with an example. The results of transistor level power dissipation for an AND gate and a combinational circuit have been presented.

Chapter 4: Contribution of glitches to dynamic power consumption

In this chapter, we present the detailed analysis of contribution of glitches to switching energy consumption of combinational circuits. We also show the individual and combined impact of process variation and variation in input stimuli on switching energy dissipation. We describe the time-efficient power extraction algorithm used for Monte Carlo analysis of the circuit.

Chapter 5: Ambiguity Interval Propagation

We discuss the ambiguity interval propagation algorithm used to estimate bounds on switching energy consumption of a combinational circuit. We suggest some improvisations on the existing algorithm for better estimation of maximum number of transitions at every net in the design. We also present the results obtained for power estimation by evaluation of multiple ambiguity intervals.

Chapter 6: Probability Propagation

We discuss the probabilistic approach to estimation of switching energy consumption of a combinational circuit in this chapter. We present the results obtained by probability propagation under both zero delay and real delay scenarios. We also show how correlation in input signals of gates can impact power estimation using this approach.

Chapter 7: Glitch Power Reduction

In this chapter, we discuss the methodology for estimating power reduction by blocking glitches at various nodes in a combination circuit. We present the results of sensitivity analysis of each node to glitch power reduction and validate the same by latch-based glitch reduction approach. We hence, identify the high glitch activity nodes in the circuit.

Chapter 8: Conclusion

We present the conclusions and inferences drawn from the work done in the thesis. We propose the scope of the work ahead.

CHAPTER 2

POWER DISSIPATION IN CMOS CIRCUITS

We begin with a review of various components of power dissipation in digital CMOS circuits. We, then explain the typical VLSI Design flow and the need for power estimation at each level. The theory behind glitches and how they are generated is explained with an example. The final section explains process variation and how it impacts power dissipation in detail.

2.1 Components of Power

The need for low power devices led to the development of CMOS technologies. CMOS power consumption is primarily determined by two components of power: *Static power consumption* and *Dynamic power consumption*. CMOS devices have very low static power dissipation, owing to lower leakage power compared to other technologies. This power dissipation occurs in steady state, when all the signals in the design do not undergo any switching. However, an increase in operating frequencies of the systems has resulted in increased switching activity of the design. Hence, dynamic power consumption contributes majorly to overall power dissipation of the system. Charging and discharging of capacitances in the design adds primarily to this component of power dissipation.

The power dissipated by a digital CMOS circuit can be broadly classified into two categories.

- Static Power
- Dynamic Power

Figure 2.1 represents the various components of CMOS power consumption.

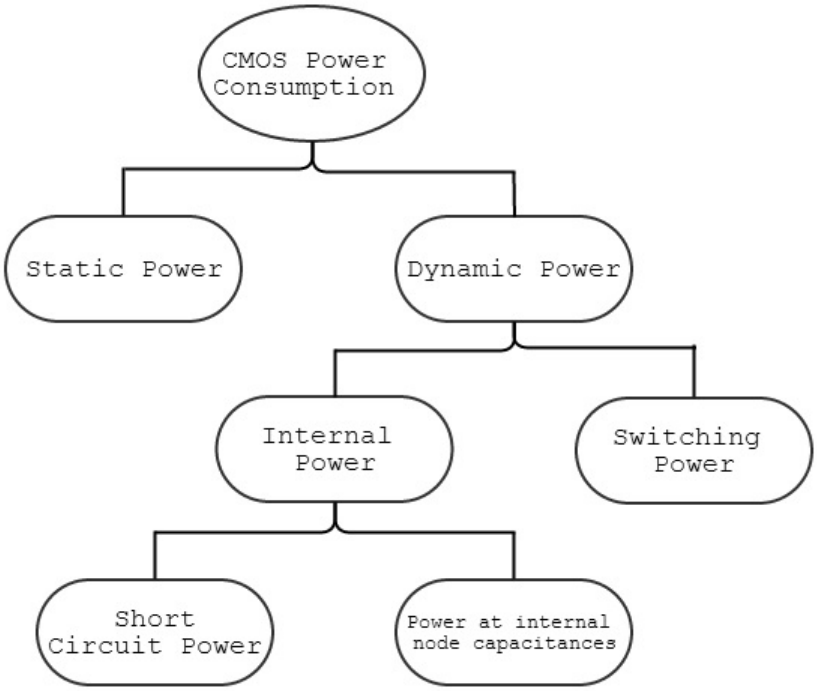


Figure 2.1: CMOS Power Consumption

2.1.1 Static Power

Static power is otherwise called leakage power. Leakage power is the power dissipated by a gate when it is not switching. This happens when the gate is static or inactive in the design. The primary component of leakage power is source-to-drain sub-threshold leakage, which is caused by reduced threshold voltages that prevent the gate from completely turning off. It is caused by currents that flow through the transistors in the gate, even when they are turned off. This component of power is becoming significant in newer technologies. Many-a-times, all sources of leakage power are lumped together into a single value and specified in the technology library for modeling purposes.

Leakage power is caused primarily by reverse biased `pn` junctions, the sub-threshold leakage current and the gate tunneling effect. This power is dissipated as long as the supply voltage is on. Several other factors like temperature also affect leakage power, due to its direct dependence on thermal voltage and threshold voltage.

2.1.2 Dynamic Power

Dynamic power is the power dissipated by a gate when it is active (or switching). A gate is active when the voltage at the net changes during a low to high transition or a high to low transition due to an applied stimulus. Dynamic power is further divided into two components.

- Switching Power
- Internal Power

Switching Power

Switching power of a gate is the power dissipated by charging and discharging of the load capacitance at the output of the gate. The load capacitance is composed of interconnect wire capacitance as well as the input capacitances of the fan-out gates the output is driving. The amount of switching power depends on the switching activity of the gate. This in turn also depends on the operational frequency of the design.

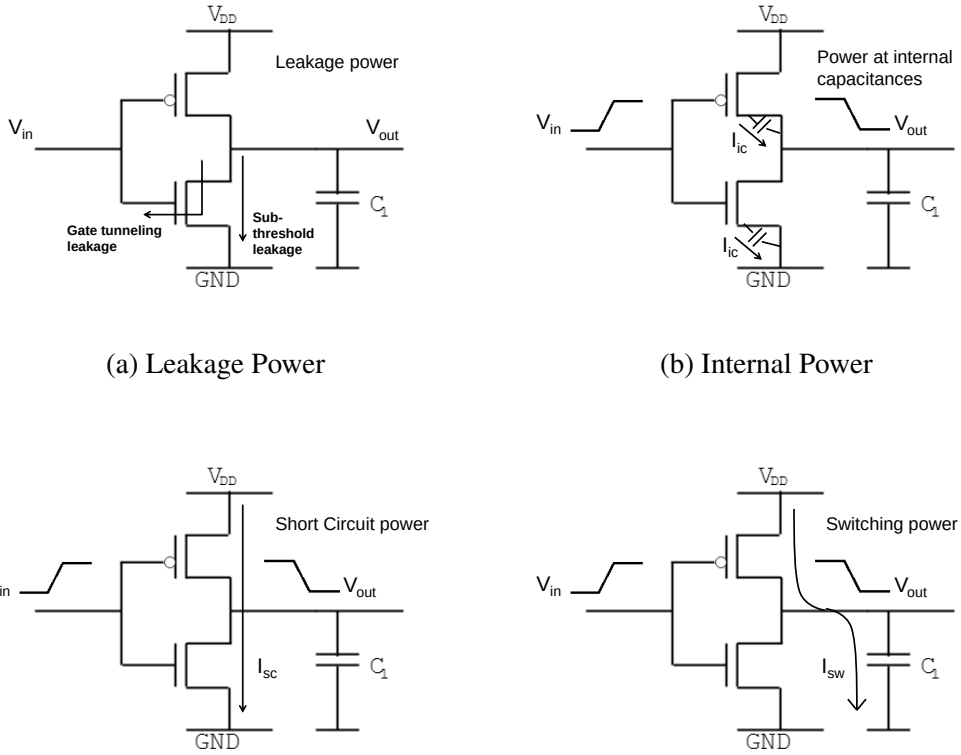
$$SwitchingPower = f \times C_l \times V^2 \tag{2.1}$$

where f is the transition frequency, C_l is the output load capacitance and V is the supply voltage. Switching power is resulted by two kinds of transitions: *steady state transitions* and *glitches*. Steady state transitions are expected transitions at the nodes in the circuit due to applied stimulus. Glitches are unwanted transitions at the nodes. We will see more about glitches and how they are generated in further sections.

Internal Power

Internal power is the power dissipated within the boundary of the gate. There are two components of internal power.

- Short Circuit Power
- Power dissipated at internal capacitances



(a) Leakage Power

(b) Internal Power

(c) Shortcircuit Power

(d) Switching Power

Figure 2.2: The four different components of power in a CMOS inverter.

Short Circuit Power

During the period when the transistors in a gate switch, there is a momentary short circuit current which flows through them. For a small amount of time, the transistors are all in the ON state and thus, provide a conducting path from supply line to ground. This happens for a small duration of time within the rise or fall time of the input signal to the gate.

Power dissipated at internal capacitances

During switching, the gate dissipates some amount of power by charging and discharging of capacitances internal to the gate. The intrinsic gate capacitances depend on the region of operation of the transistors.

Figure 2.2 explains the various components of power dissipation using an inverter as an example gate.

2.2 VLSI Design Flow

The VLSI design flow primary comprises of three stages: the *behavioral level*, describing the functionality of the design, the *gate level*, describing the form of implementation and the *physical level*, describing the physical implementation of the design. A simplified view of the design flow is represented in the flow chart (Figure 2.3) below.

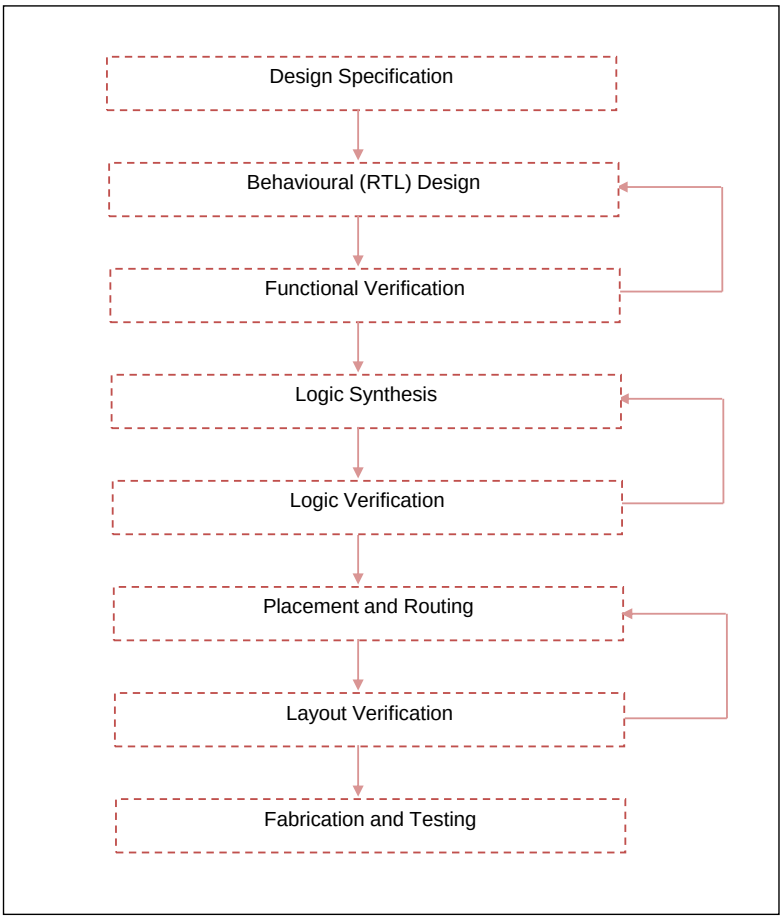


Figure 2.3: VLSI Design Flow

In a typical VLSI design flow, we start with system specifications, which signify the intent of the design. These specifications are then converted to Register Transfer Level by architectural synthesis tools. Once the RTL is verified to meet the design specifications, logic synthesis is done by CAD tools to convert it to a gate level netlist. This netlist is then used for placement and routing procedure, after which the chip will be sent for fabrication. The verification of the design at every step plays a significant part of the design process. At every stage, all the design constraints, including power

consumption of the design have to be met to the specifications. In today's SoC designs, several circuit blocks are integrated together into a single system. While designing such complex systems, there is always a trade off between accuracy and computational speed in estimation of power consumption. Accurate estimates are obtained at later stages, when the gate level information is ready. Power estimation is traditionally performed at the transistor level by SPICE simulation at the end of the design flow. However, as the systems get more and more complex, this becomes a computationally intensive task. Moreover, power estimates from the later stages make redesigning of the system difficult and thereby, increasing the time-to-market. Gate level power estimation has become increasingly popular and active research is being done on the same. Power estimates at earlier levels like RTL are less accurate than gate level estimates. Algorithms to estimate power at earlier levels are being developed.

2.3 Glitches

Glitches are unwanted transitions at the output of a gate due to the skew in its input signals. This arises primarily due to unbalanced path delays in the circuit. When several paths arriving at one internal gate have different propagation delays, the gate undergoes several unnecessary transitions before settling to the correct logic level.

Consider the example shown in Figure 2.4. Due to path delays at earlier levels in the circuit, the two transitions at the inputs do not occur at the same time. They are shifted in time by a small amount. This results in a $0 - 1 - 0$ transition at the final levels in the circuit. Observe that the glitch at the output do not undergo a complete $0 - 1 - 0$ transition. Such transitions are called *partial glitches*. Partial glitches are those glitches whose pulse width is too short in comparison to gate delay. They are even more difficult to evaluate in the circuit. Since we assume inertial delay model for the gates in the design, we will not be considering partial glitch transitions in our analysis.

Switching energy consumption is altered by two kinds of transitions: *steady state transitions* and *glitches*. As we will see in further chapters, glitches contribute significantly to switching energy dissipation of the circuit. If glitches are generated at earlier levels in the circuit, they propagate further through the gates till the output. Hence,

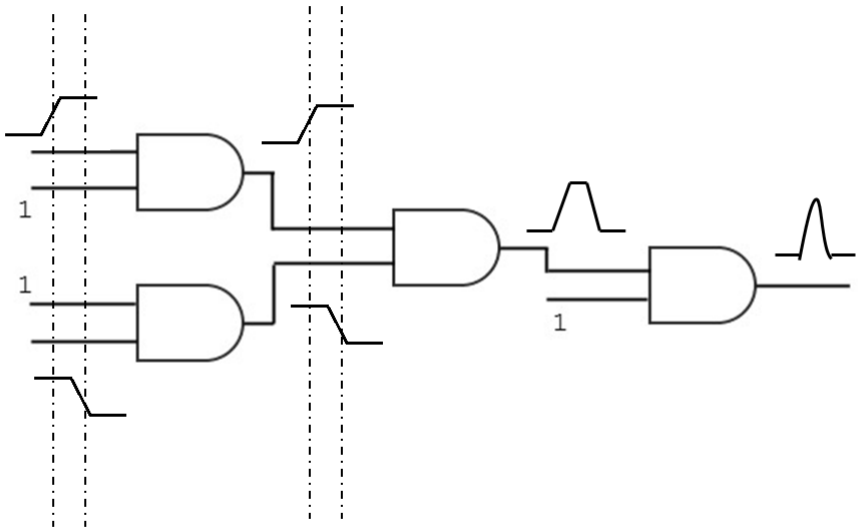


Figure 2.4: An example of generation of glitches in a circuit

glitch transitions can give rise to several other glitch transitions further down the network. In larger designs, this increases switching power of the circuit significantly.

2.4 Process Variation

Process variations are the variations in parameters like oxide thickness, threshold voltage, channel length and interconnect wire properties introduced during the semiconductor fabrication process. They can be broadly classified into two categories: *inter-die* and *intra-die* variations. Inter-die variations are the variations on the same wafer lot across several dies. In this case, the parameters within an individual die remain constant. Intra-die variations are variations introduced within the same die during the fabrication process. In this case, some of the parameters within the same die vary across the surface area of the die. In today's nano-scale technologies, there is wider amount of process variation. Process variations, typically alter one or more of threshold voltage, channel length or operating temperature. This generates uncertainty in gate delays, thereby, altering switching energy consumption of the design. This effect varies from device to device. Glitch transitions are primarily dependent on gate delays of the circuit. Power supply fluctuations, interconnect noise and temperature fluctuations are some of

the factors which affect gate delays in the design. Contribution of glitches to switching energy consumption is increasingly becoming significant in nano-scale technologies.

The sub-threshold leakage current, which is a primary component of leakage power can be calculated by Equation (2.2).

$$I_{sub} = \mu_0 C_{ox} \frac{W}{L} V_t^2 e^{(\frac{V_{gs}-V_{th}}{nV_{th}})}(1 - e^{(\frac{-V_{ds}}{V_t})}) \tag{2.2}$$

where μ_0 is effective mobility, C_{ox} is gate oxide capacitance per unit area, W is gate width, L is channel length, V_{gs} and V_{ds} are gate to source and drain to source voltages respectively, V_{th} is the threshold voltage, $V_t = \frac{kT}{q}$ is the thermal voltage and n is a technology parameter.

As we can see from Equation (2.2), leakage power depends on threshold voltage, temperature and channel length. Hence, process variation has a direct impact on leakage power. However, it doesn't affect dynamic power directly. It alters the gate delays in the circuit, thereby increasing the number of glitch transitions in many cases. This in turn increases switching power as the number of transitions increases.

CHAPTER 3

TRANSISTOR LEVEL ANALYSIS

One of the primary objectives of our work is to identify the impact of process variation on switching energy consumption, and hence, estimate the contribution of glitches to switching energy. One of the most accurate ways of doing this is by a simulation at the transistor level across several inputs. However, an exhaustive simulation like this would be computationally time intensive. As explained in earlier sections, there is always a trade off between accuracy and computational speed in estimation of power. So, typically a gate level simulation is done to estimate the overall power consumption. The simulations at this level are done usually by CAD tools. These estimates will not consider partial glitches and non-linearity of waveforms at all the nets in the design. Hence, there will be some loss of accuracy with these estimates.

In this chapter, we will look at characterization of gates and simulation of larger combinational circuits at transistor level and how they compare to gate level power estimates using CAD tools.

3.1 Spice Characterization of gates

Power consumption by a logic gate depends on various environmental and process parameters. Logic family, topology and sizing of transistors, rail voltages, the output load capacitance the gate drives and the input stimulus are some of the primary factors which directly influence the power dissipated by the gate. Typically, CAD tools use the technology library file for the power characteristics of individual gates. The library file has information about the power consumed by each gate across several load capacitance values. In order to ensure that the power estimation at gate level by CAD tools adequately captures the impact of process variation on power consumption of the circuit, we first need to validate the same at transistor level using spice simulation. And to do that, the power characterization values in the library used by the CAD tools also need to be validated against values by the spice simulator being used for simulations.

This section deals with characterization of a logic gate in order to predict its energy consumption for various driving load capacitances.

3.1.1 Characterization of AND gate

Characterization of a gate for power consumption is done for several values of input transition times and total output net capacitances. A rising or falling transition is given as one of the inputs while keeping the other input constant. 0 V voltage sources are used at various signals to measure the current flowing through those signals. Figure 3.1 shows the circuit used for the characterization.

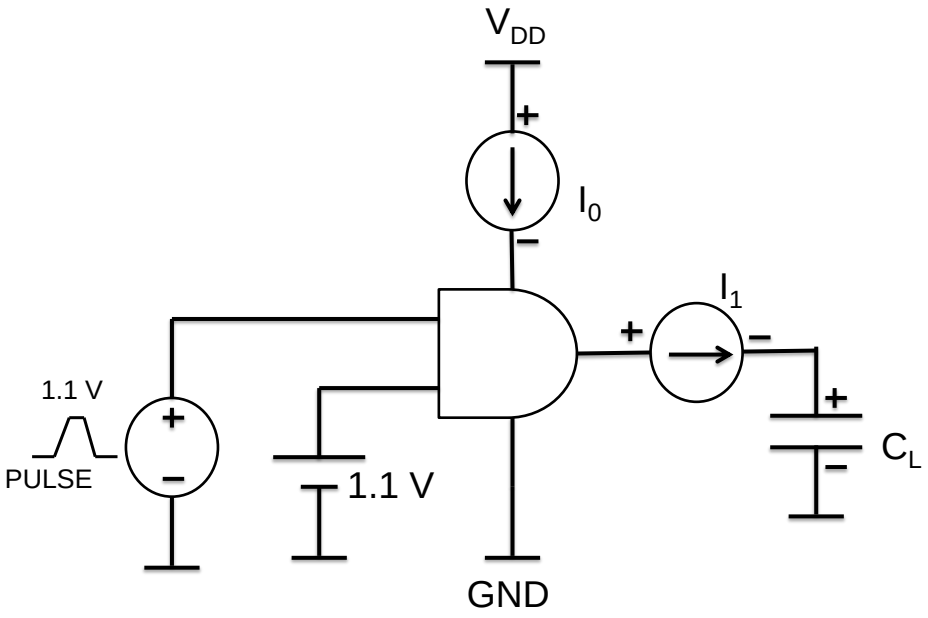


Figure 3.1: Spice characterization of AND gate

The analysis is done on a two input AND gate using *Spice Opus* software. A pulse of voltage $1.1\,V$ is used to give a rising or a falling transition at one of the inputs. The other input of AND gate is fed to a $1.1\,V$ DC voltage source. The $\mathrm{V_{DD}}$ rail is kept at $1.1\,V$ and `GND` rail at $0\,V$. Simulation is done for several values of load capacitance $\mathrm{C_{LOAD}}$. The $0\,V$ voltage sources at the $\mathrm{V_{DD}}$ rail and at the output of the AND gate measure the current flowing in those signals. A rising transition with a rise time of `0.00117378 ns` is given to one of the inputs of the AND gate. `PTM` $45\,\mathrm{nm}$ `technology` model file is used for spice simulation.

Table 3.1: Spice Characterization of AND gate

Output Capacitance Capacitance	Transition Time (ps)	Peak Current (mA)	Total Energy Consumed (fJ)	$E_{consumed} - E_{load}$ (fJ)	E_{load} (fJ)	$0.5 \times C\,V^2$ (fJ)
0.365616	13.6	0.154	2.37	2.13	0.242	0.221198
1.89304	23.7	0.191	4.14	2.88	1.25	1.14529
3.78609	36.3	0.207	6.41	3.90	2.51	2.29058
7.57217	62.0	0.221	10.8	5.96	4.87	4.58116
15.1443	114	0.232	20.5	10.4	10.1	9.1623
30.2887	220	0.241	38.8	18.7	20.1	18.3247
60.5774	431	0.244	74.4	35.3	39.0	36.6493

The following inferences are drawn from Table 3.1.

- The transition time almost proportionally increases with load capacitance.
- The peak current is relatively constant for all loads.
- The value E_{load}, which is obtained by spice simulation is nearly equal to theoretical value of energy consumed by the load capacitance, $0.5\,C\,V^2$.
- Power dissipated in the internal capacitances of the gate is negligible in comparison to short circuit power.

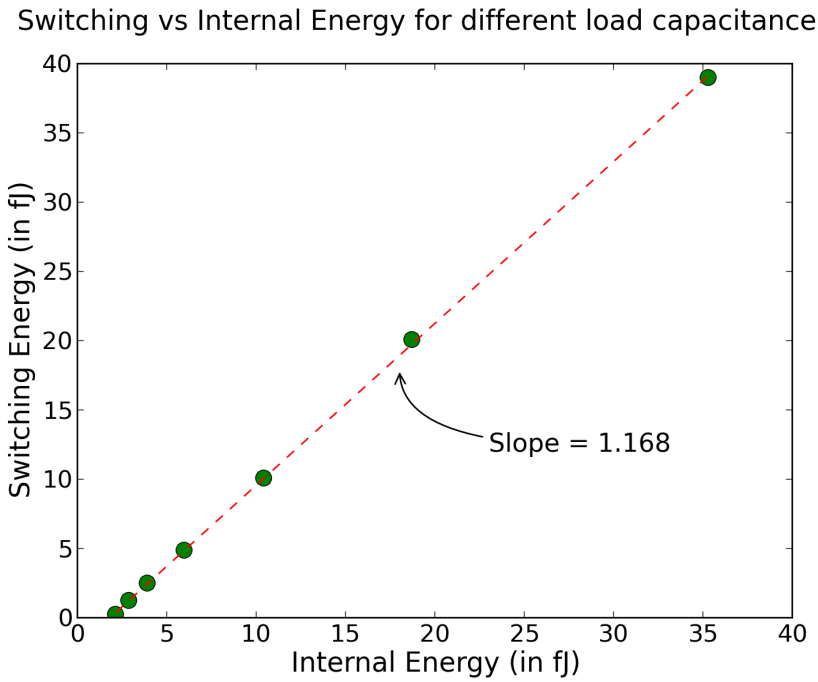


Figure 3.2: A comparison of contributions of internal energy and switching energy to dynamic power for different load capacitances for a 2-input AND gate

Figure 3.2 shows the contribution of internal energy and switching energy to dynamic power consumption of AND gate for various values of output load capacitance. We observe that both the components, switching energy and internal energy significantly contribute to overall dynamic power consumption of the gate. They also increase with increase in output load capacitance of the gate.

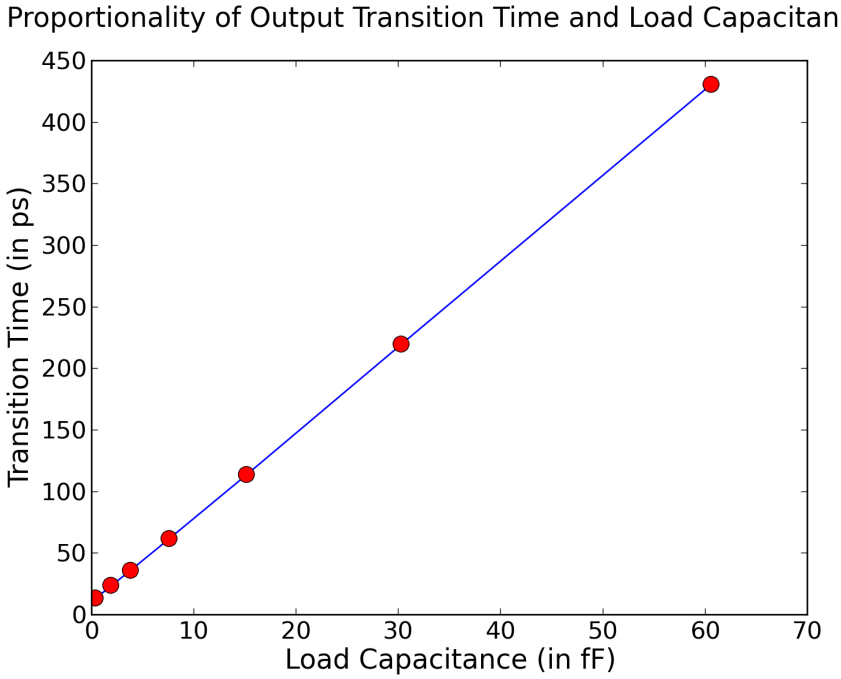


Figure 3.3: Transition time of the signal at the output node versus output load capacitance for a 2-input AND gate

Figure 3.3 shows the transition times of the signal at the output node of the AND gate for several values of output load capacitances. We observe that the transition time is nearly proportional to the output load capacitance.

Figure 3.4 compares the energy estimate for the load obtained by spice simulation and its theoretical estimate of $0.5\,\mathrm{C}\,\mathrm{V}^2$. We observe that the two values are nearly equal.

3.2 Transistor level simulation of a combinational circuit

In the previous section, we characterized a two input AND gate and analyzed the contribution of various sources of power dissipation. In this section, we look at a combi-

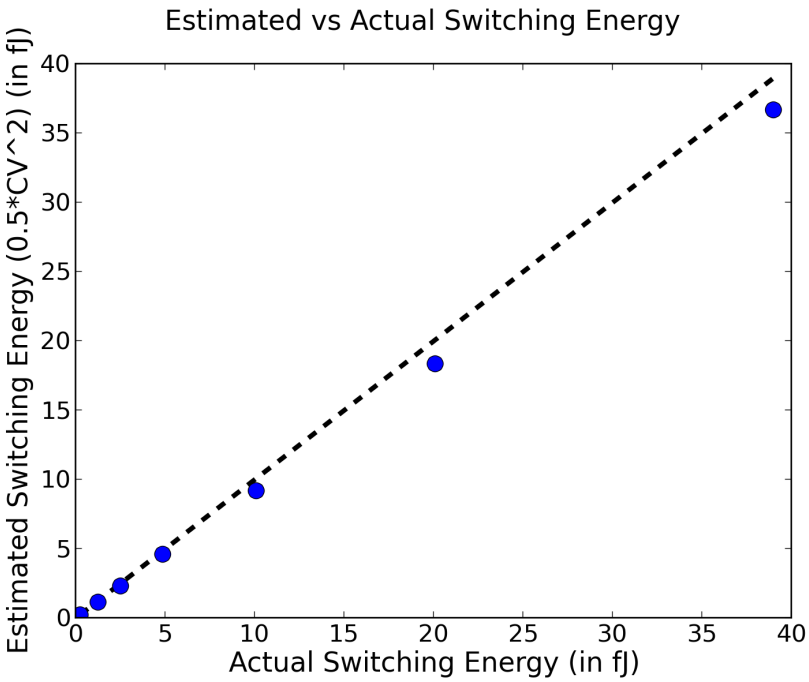


Figure 3.4: A comparison of the energy estimate obtained by spice simulation and theoretical value of energy consumed by output load capacitance for a 2-input AND gate

national circuit and estimate power consumed by the circuit at transistor level. We also analyze the pros and cons of power estimation at transistor level.

A combinational circuit is modeled as a network of transistors, voltage and current sources at the transistor level. Spice simulator solves for the node current and voltage equations using Kirchoff's laws. The spice simulation then captures the voltage and current waveforms at all the signals in the circuit due to the applied input stimulus. A gate level power estimate doesn't account for partial glitches and non-linearity in waveforms at the nodes of the circuit. Hence, the estimates obtained at gate level aren't as accurate as transistor level estimates.

The primary advantage of transistor level analysis is the high accuracy of power estimates as compared to gate level estimates. Transistor level estimation considers non-linearity of waveforms at inputs of the gates and partial glitches in the circuit. However, it is highly time intensive. Gate level estimation, on the other hand, is time efficient comparatively and easier to model. There is loss of accuracy at gate level owing to negligence of partial glitches and non-linearity of waveforms at inputs of gates in the circuit.

The circuit that is simulated is an 8×8 `Booth Multiplier` circuit. `NanGate` $45\,\mathrm{nm}$ `technology` file is used. *Synopsys Design Compiler* converts the verilog RTL netlist of the circuit into a gate level netlist. This is then converted into an equivalent transistor level netlist using the tool *Calibre*. Since the netlist file rendered by *Calibre* will not have input sources, the netlist is then modified using a `Python` script to form an *Eldo* spice simulation file. Appropriate pulse waveforms are applied at the input voltage sources to match with the input pairs used by *Synopsys Design Compiler*. The total power dissipation by the circuit is estimated then by *Eldo* spice simulator. *Eldo* spice simulator converts every logic gate in the circuit into its corresponding transistor level sub-circuit. These sub-circuits are then simulated at transistor level for an applied stimulus. *Synopsys Design Compiler* uses the characterized power estimates from `NanGate` $45\,\mathrm{nm}$ `technology` file for each gate in its power estimation.

We observed that *Eldo* spice simulation takes about $10 - 15$ minutes for simulating `Booth Multiplier` circuit for one input vector pair. Monte Carlo analysis of switching energy consumption of circuits would be highly time-consuming and hence, transistor level simulation was not taken forward in our analysis.

CHAPTER 4

CONTRIBUTION OF GLITCHES TO DYNAMIC POWER CONSUMPTION

Power consumption of a CMOS circuit is contributed primarily by static leakage power and dynamic power due to switching activities at nodes in the circuit. The switching activities at various nodes in the circuit are caused by two kinds of transitions. The first kind of transition is the expected steady state transition at each node due to an applied stimulus to the circuit. The second kind of transitions are caused by unbalanced delays in various paths in the circuit. These transitions are called glitches and they contribute significantly to the switching power of the circuit. When several stages of combinational logic are pipelined together, glitches generated at earlier stages propagate further down the network, resulting in significant increase in overall power consumption of the design. Glitch estimation at Register Transfer level is highly inaccurate. Typically, it is estimated at gate level.

As seen in chapter 2, process variation also significantly impacts the number of glitch transitions generated in the network. Process variations are the variations in parameters like oxide thickness, threshold voltage, channel length and interconnect wire properties introduced during the semiconductor fabrication process. We also saw that process variation does not have a direct impact on switching energy consumption, but alters delays of gates in the circuit, thereby generating glitches. Some of parameters majorly altered by process variation includes Threshold Voltage $(\mathrm{V_t})$, Temperature (T) and Channel Length (L).

In this chapter, we will see the impact of process variation on combinational circuits and contribution of glitches to overall switching energy consumption of the circuit in detail.

4.1 Impact of process variation on switching energy

One of the most accurate ways of modeling process variation is by introducing variation in individual parameters like temperature or threshold voltage independently and estimating power. Process variation can cause both intra-die and inter-die variations. Intra-die variations can have spacial correlation between several parameters within a die. Hence, modeling process variation at a parametric level, inclusive of several correlations is difficult. We assume that the variation on individual parameters are independent. Process variation alters delays of gates in the circuit. Hence, we lump the effect on individual parameters together and model the variation at gate level. We do so by introducing a standard normal variation on the propagation delays of all the gates in the circuit. We do a `Monte Carlo` analysis by simulating the circuit many times over in order to obtain the distribution of switching energy consumption.

4.1.1 Monte Carlo Analysis using Design Compiler

Process variation is modeled by introducing a standard normal variation in propagation delays of all the gates in the circuit. The circuit is simulated many times , each time modifying the delays of all the gates in the circuit. The simulation is also done for several values of standard deviation of the normal distribution being introduced. The detailed procedure of the simulation is explained below.

Procedure

Step 1: The RTL netlist of the circuit is converted into a gate level netlist using *Synopsys Design Compiler (DC)*. DC takes a technology library file as input. The RTL netlist is read in and compiled by the compiler. It then translates the design into a technology-independent design. It also optimizes the design at architectural level, logic level and gate level. Next, the cells in the technology-independent design is mapped to the corresponding cells in the technology library file. The propagation delays of all the gates and interconnect delays of all the nodes of the circuit are annotated in the `Standard Delay Format` file. DC generates the gate level netlist and the `SDF` files as the output.

Step 2: The standard normal variation in propagation delays of all the gates in the circuit is introduced by modifying the SDF file, generated as output in the previous step. For every gate, the SDF file contains annotations of the minimum, typical and maximum propagation delay for a rising transition as well as for a falling transition for each input of the gate to its output. A normal variation is induced on each of these propagation delay values using a Python script in every iteration of the simulation.

Step 3: The circuit is then simulated by a CAD tool *ModelSim* from *Mentor Grpahics*. The gate level netlist, the SDF file and a test bench of the circuit are given as inputs to *ModelSim*. The test bench contains the input stimulus applied to the circuit. The tool then generates a Value Change Dump (VCD) file after simulation. This file contains a series of time-ordered value changes of all the signals in the circuit in the given simulation.

Step 4: The VCD file is converted into a SAIF file. SAIF is the format compatible with *Synopsys DC*.

Step 5: The SAIF file, gate level netlist and the technology library file are given as inputs to *Synopsys DC*. This then estimates the various components of power dissipated in the circuit using the transitions of the signals annotated in the SAIF file.

Step 6: The circuit is simulated many times, by repeating steps 2-5. We then obtain the profile of power dissipation in the circuit for the introduced process variation.

Figure 4.1 depicts a flowchart of the procedure used for Monte Carlo analysis of power consumption using Design Compiler.

Results

An 8×8 Booth Multiplier circuit is simulated for one input vector pair. The SDF file is modified and simulated 1000 times. NanGate $mathrm45\,nm$ technology library is used as the library file for linking the design.

Figures 4.2 and 4.3 show the histograms for Monte Carlo analysis of power

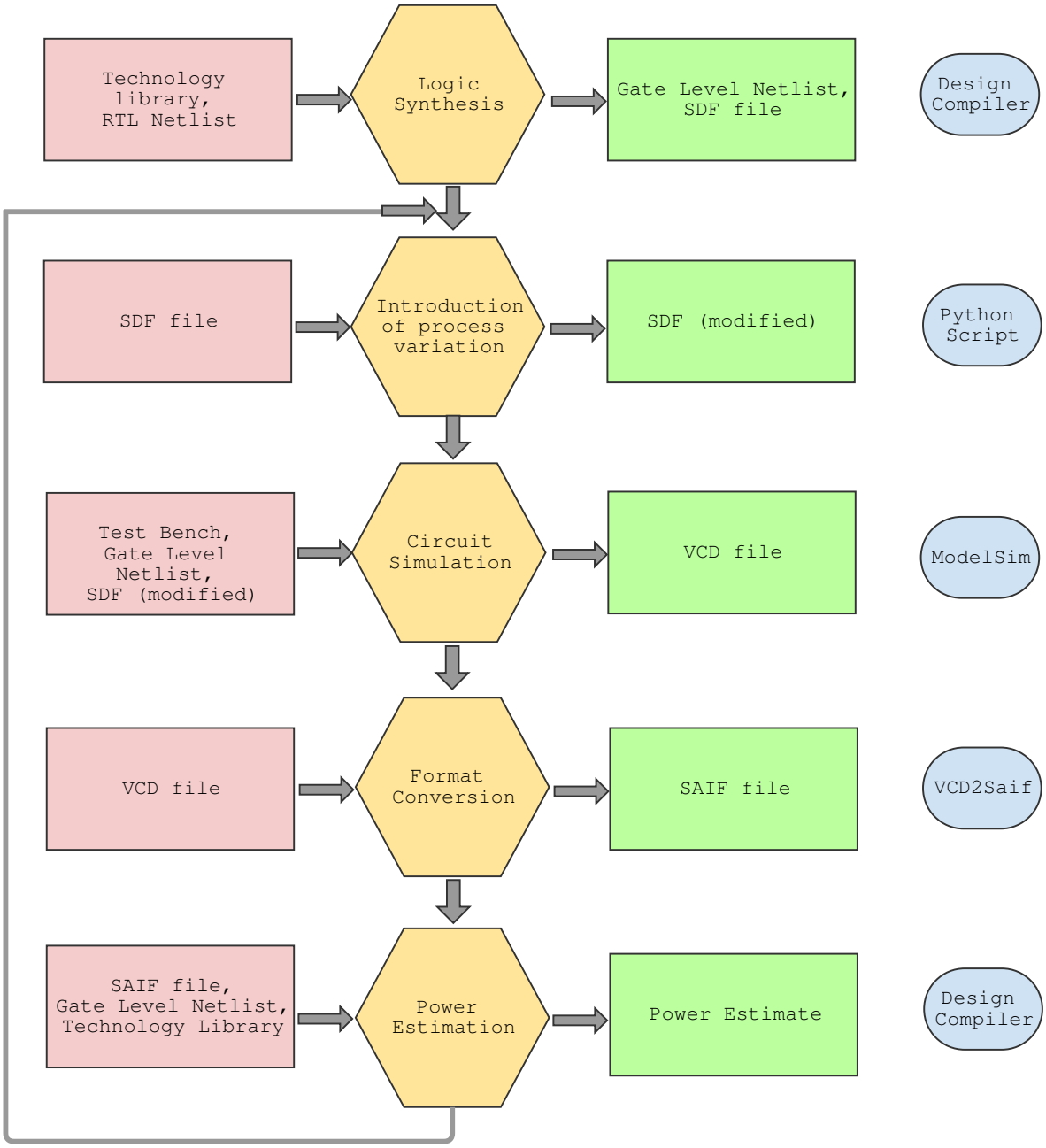


Figure 4.1: Flowchart of the procedure used for Monte Carlo Power Estimation Analysis using Design Compiler

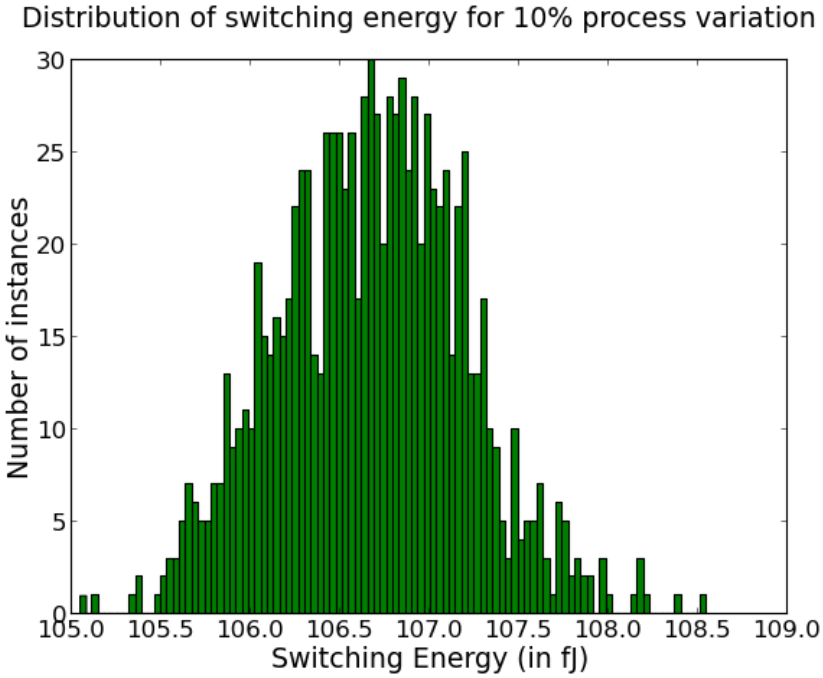


Figure 4.2: Histogram for Monte Carlo analysis of energy dissipation in Booth Multiplier circuit for a 10% standard deviation in process variation. SDF file is modified 1000 times and simulated for one input vector pair. Circuit is synthesized using NanGate 45 nm library.

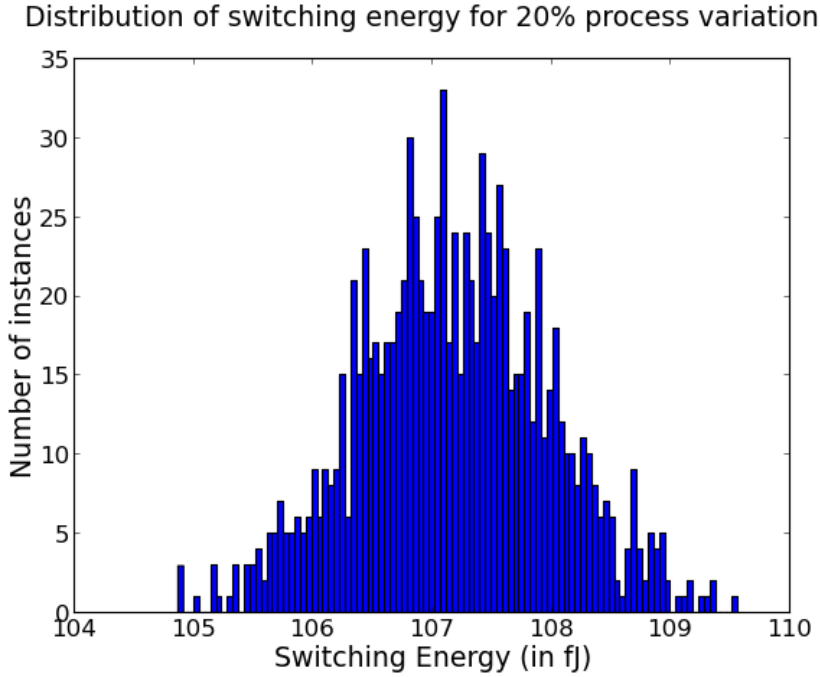


Figure 4.3: Histogram for Monte Carlo analysis of energy dissipation in Booth Multiplier circuit for a 20% standard deviation in process variation. SDF file is modified 1000 times and simulated for one input vector pair. Circuit is synthesized using NanGate 45 nm library.

dissipation in `Booth Multiplier` circuit for a 10% and 20% standard deviation in process variation respectively. One input vector pair was simulated for 1000 different `SDF` files.

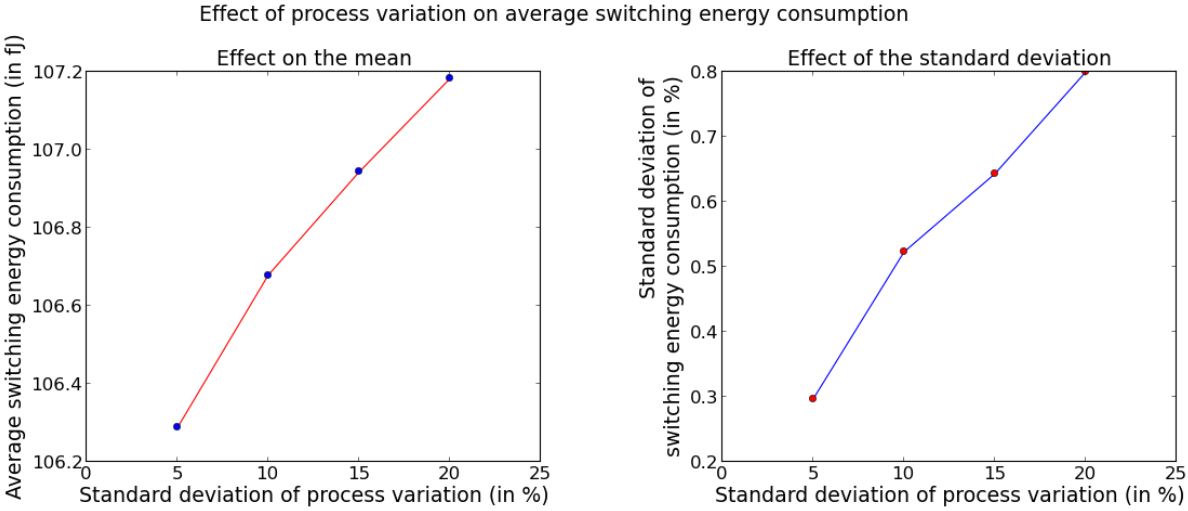


Figure 4.4: Effect of process variation on average switching energy consumption in Booth Multiplier circuit for different values of standard deviation in process variation. SDF file is modified 1000 times and simulated for one input vector pair for each value of standard deviation. Circuit is synthesized using NanGate 45 nm library.

Figure 4.4 shows the effect of process variation on overall average switching energy consumption of the circuit. The circuit was simulated for 5%, 10%, 15% and 20% standard deviation in process variation. As we can see from the figure, both mean and standard deviation of total switching energy consumption of the circuit increases with increase in standard deviation in process variation. The mean and standard deviation of total switching energy consumption of the circuit were computed for 1000 simulations of modification of `SDF` file for one input vector pair.

Different input vector pairs cause a different number of transitions in the nodes of the circuit, thereby resulting in different total switching energy consumption values. The analysis so far was done for one specific input vector pair. In order to eliminate the effect of variation in switching energy consumption due to different inputs, we need to simulate the circuit for several input vector pairs. The bottleneck of the procedure described in this section is that the power estimation of the circuit using *Design Compiler* is computationally a time intensive step. Invoking and running the tool for 10^6 runs (1000 different inputs and 1000 modifications of `SDF` file each) would lower the computational efficiency of the process. This necessitates an alternative efficient ap-

proach to estimate power using the VCD output generated by *ModelSim*. The standard deviation in process variation was set as 20% for all the analysis in the reminder of the chapter.

4.1.2 Power Extraction algorithm for Monte Carlo Analysis

We propose a time-efficient algorithm to primarily estimate switching energy consumption of a circuit. This algorithm also allows us to selectively compute power dissipated in a specified time interval (or for a particular input vector pair) and even segregate the power consumption due to glitches from the total switching energy. It is based on the simple principle that the switching power at any node can be independently computed just by knowing the number of transitions and capacitance of that node.

$$Switching\ Energy = \sum_{i \in allnodes} t_i \times \frac{1}{2} C_i V^2 \tag{4.1}$$

where t_i is the number of transitions at node i, C_i is the capacitance at node i and V is the supply voltage.

Total switching energy of the circuit can be evaluated by equation (4.1). It is equal to the sum of switching energies at every net in the design.

Inputs of the algorithm

- VCD file generated by *ModelSim* after simulation of the circuit.
- Gate level netlist of the circuit
- Technology library used for logic synthesis of the design

Output of the algorithm

The primary output of the algorithm is the total switching energy dissipated by the circuit. The algorithm also allows us to estimate switching energy for each input vector pair. It can also output the switching energy estimates for a particular net in the circuit.

Procedure

The algorithm assumes that the steps 1-3 of the procedure described in the above subsection are carried out and the `VCD` file has been generated after simulation of the circuit by *ModelSim*.

Step 4: The number of rising and falling transitions at every net in the design is evaluated by parsing the `VCD` file by a `Python` script. This estimates the total number of rising transitions and the total number of falling transitions at every net in the circuit, thereby estimating the total number of toggles in the circuit.

Step 5: The rising and falling input capacitances of all the fan-out gates of a net are summed individually and a look-up table is created for rising and falling capacitances at every net in the circuit. This is done once for a particular circuit. The rising and falling capacitances for each net are then extracted from this look-up table in each simulation.

Step 6: The value of number of rising transitions at a net is multiplied with the corresponding rising capacitance value and the number of falling transitions is multiplied with the corresponding falling capacitance. These two values are added and multiplied with V^2. This value is then evaluated for every net in the circuit and summed together to obtain the total switching energy of the circuit.

Note: In this approach, a `Python` script is used to estimate switching power dissipation of the circuit. The circuit is simulated many times for several modifications of the `SDF` file. However, unlike in the previous approach where the circuit had to be simulated for each applied input for each `SDF`, this approach enables us to provide all the inputs together in the test bench and simulate the circuit just once for a particular `SDF` file.

Results

The power extraction algorithm is validated by simulating an 8×8 `Booth Multiplier` circuit for 100 different input vector pairs and using the `SDF` file generated by *Synopsys*

Design Compiler during logic synthesis. The `NanGate` 45 nm library is used as technology library. The overall switching energy of the circuit for each applied input vector pair is evaluated by both the Design Compiler approach and by our power extraction algorithm.

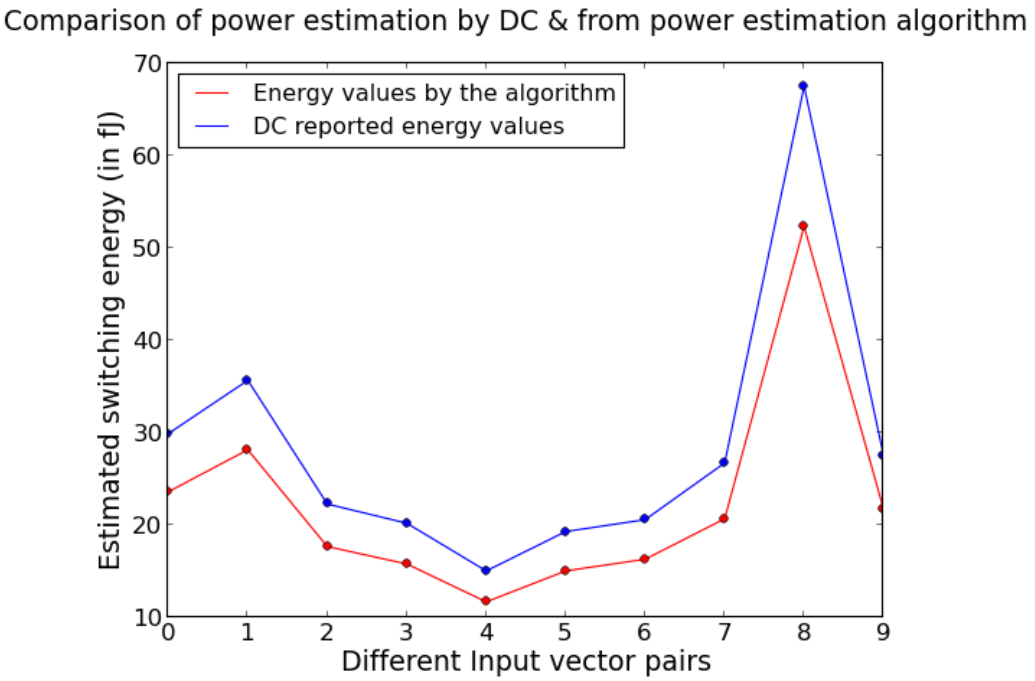


Figure 4.5: Comparison of overall switching energy consumption of Booth Multiplier circuit by Design Compiler and power extraction algorithm for different input vector pairs, considering a standard SDF file and NanGate 45 nm technology library.

Figure 4.5 shows the comparison between the switching energy of the circuit obtained by both the approaches for some applied input vector pairs. This figure indicates that the values obtained by both the approaches are almost proportional. This is more clearly observed from the scatter plot shown in figure 4.6. In this figure, the energy estimates of the circuit for 100 input vector pairs obtained by both the approaches have been depicted. The deviation from the power reported by DC is because DC also includes wireloads for the nodes which are ignored in our computation.

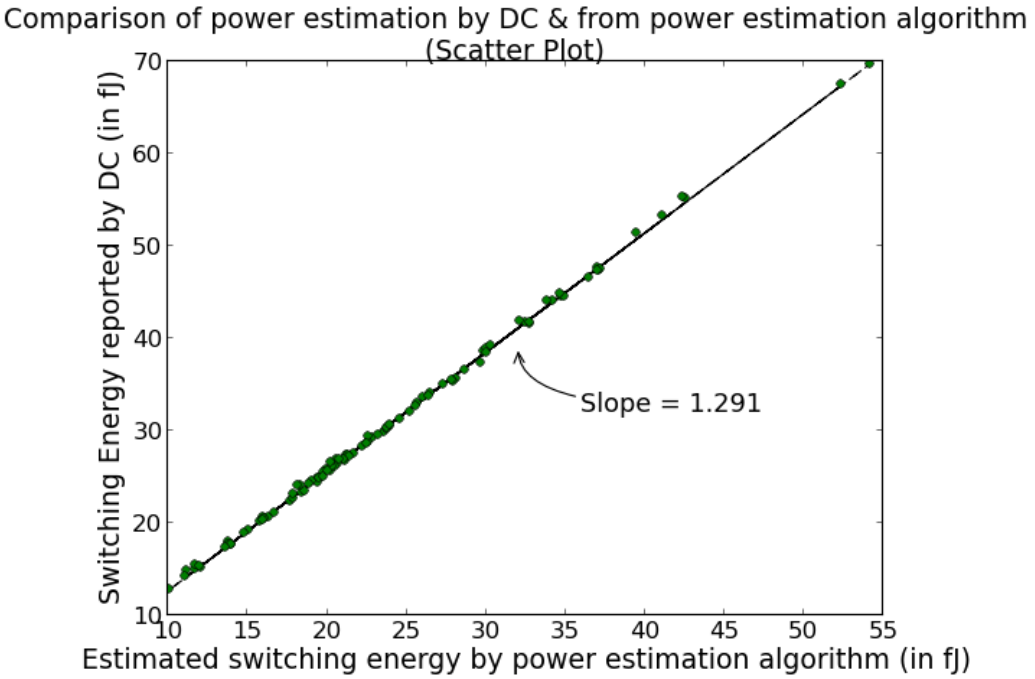


Figure 4.6: Comparison of overall switching energy consumption of Booth Multiplier circuit by Design Compiler and power extraction algorithm for 100 input vector pairs, considering a standard SDF file and NanGate 45 nm technology library.

4.2 Analysis of impact of input variation and process variation on switching energy consumption

4.2.1 Combined impact of input variation and process variation

Total switching energy dissipation in a circuit is affected by variation in input stimulus as well as process variation. In this subsection, we see the impact of simultaneous variation in input stimulus and process variation. We also observe this impact across two different technologies, 45 nm and 250 nm for two circuits, an 8×8 `Booth Multiplier` and an `ISCAS85` benchmark circuit `c880`, which is an `8-bit ALU`. The circuits are simulated by power extraction algorithm using the approach mentioned in the above subsection. 1000 input vector pairs are applied to the circuits and for each input vector applied, `SDF` files are modified 1000 times. Two technology libraries, 45 nm and 250 nm were used for simulating the circuits.

Figure 4.7 shows the distributions of power dissipation for two circuits, `Booth Multiplier` and `c880` at two different technologies, 45 nm and 250 nm. The his-

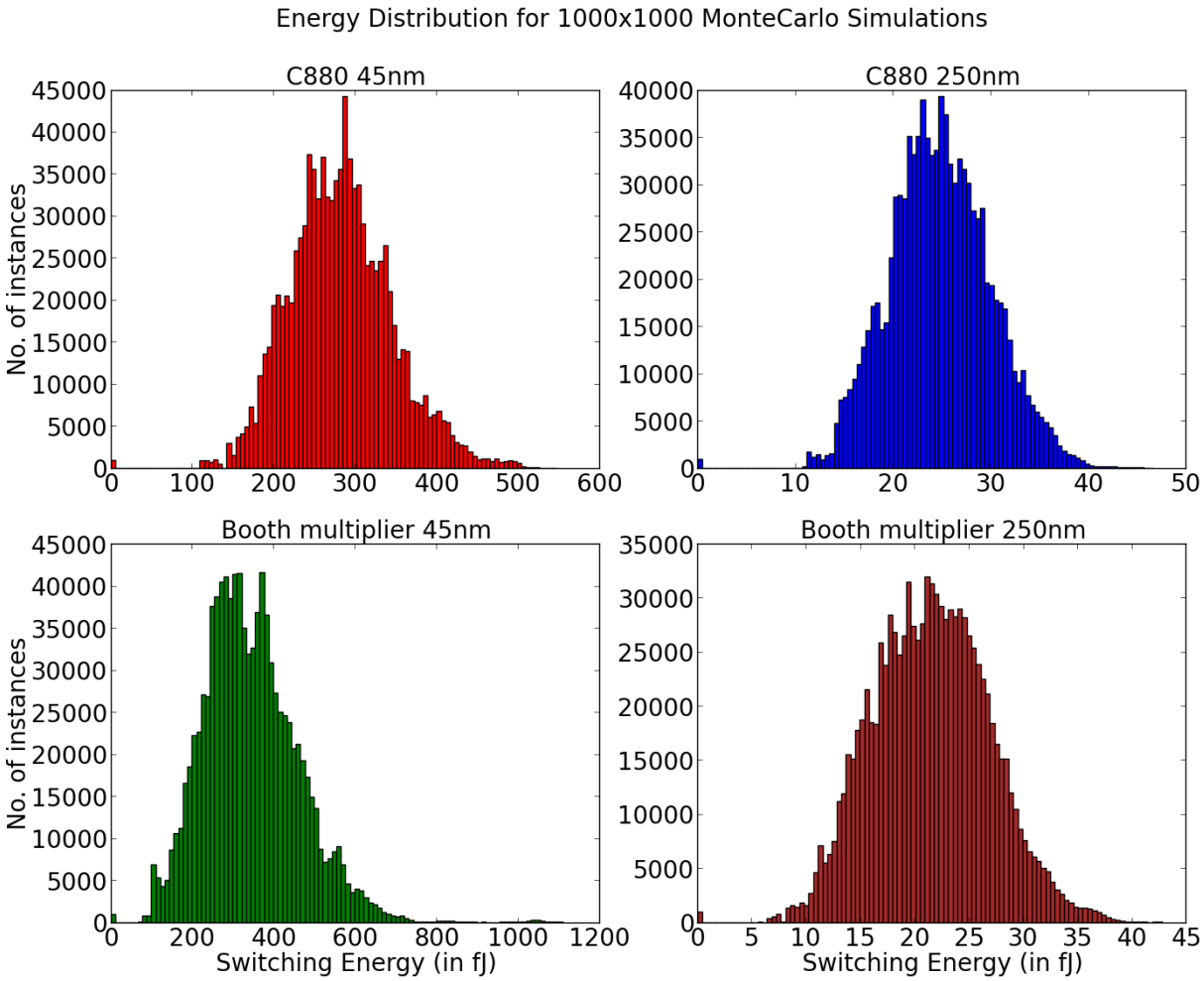


Figure 4.7: Histograms of total switching energy distribution for Booth Multiplier and C880 circuit at $45\,\mathrm{nm}$ and $250\,\mathrm{nm}$. Circuits are simulated for 1000 input vector pairs and SDF files are modified 1000 times.

Table 4.1: Mean and Standard Deviations of Total Switching Energy due to both input and process variation for `ISCAS '85` benchmark circuits

Circuit	Total Switching Energy Consumption (fJ)		
	Mean	Standard Deviation	Overall Range
c17	3.49	2.08	0 - 9.66
c1908	237.81	69.09	57.81-580.18
c3540	688.80	199.88	94.43 - 1413.14
c432	115.01	40.43	23.53 - 349.36
c499	182.99	35.92	70.14 - 372.91
c6288	19172.87	2822.48	5019.21 - 27702.34
c880	160.90	49.79	44.61 - 391.08

tograms represent the total switching energy distribution combining the effects of both process variation and input variation. This analysis is done for the `ISCAS '85` benchmarks, and the statistical parameters & range of the variation in switching energy distribution have been tabulated in table 4.1.

As we can infer from these distributions and the table, switching energy consumption does have a signification variation (when inputs and process parameters are varied) regardless of the circuit or the technology.

4.2.2 Impact of process variation versus input variation

We have seen earlier that the variation in power dissipation of the circuit is caused by both variation in applied input stimulus as well as process variation. In the previous section, we observed their combined impact on switching energy consumption of circuits. In this section, we separate and analyze the impact of both the variations on total switching energy of the circuit.

Just as in the previous simulation, an 8×8 `Booth Multiplier` circuit is simulated using $45\,\mathrm{nm}$ technology library file, 1000 input vector pairs are applied to the circuit and for each pair the `SDF` file is modified 1000 times (an over-all of 10^6 simulations).

For a particular input vector pair, we plot the distribution of switching energy of the circuit for 1000 modifications of the `SDF` file. This would represent the impact of process variation alone on switching energy. We can do this for many input pairs to capture the overlap. We also take a standard `SDF` file, generated during logic synthesis by *Synopsys Design Compiler* and simulate the circuit for 1000 different applied input vector pairs. Hence, we identify the impact of process variation by keeping the input stimulus constant and the impact of variation in input stimulus by keeping the gate delays constant.

We observe that the impact of process variation for each input vector pair follows a normal distribution and the mean of the distribution is equal to the total switching energy of the circuit for that input vector pair for the standard `SDF` file. We also observe that the impact of variation in input stimuli on switching energy is far more significant than the impact of process variation on the same.

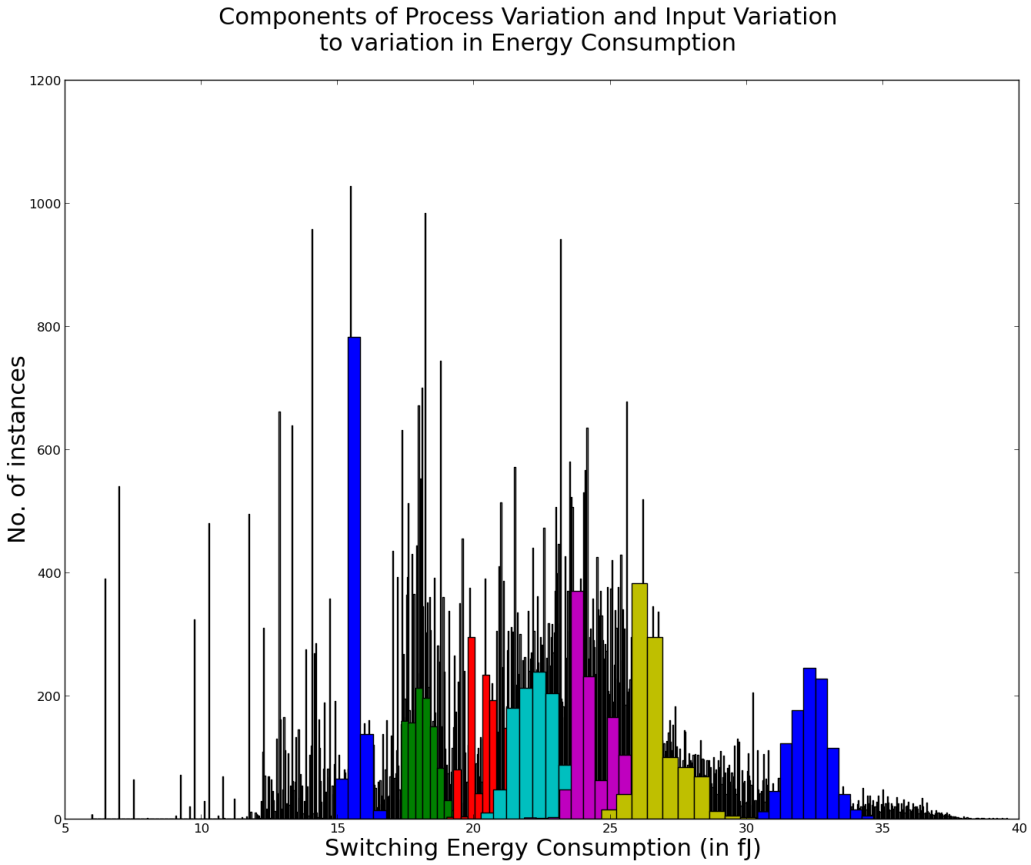


Figure 4.8: Histograms of impact of process variation for specific input vector pairs on total switching energy for Booth Multiplier circuit at $45\,\mathrm{nm}$ technology. SDF file is modified 1000 times for each input vector pair.

Figure 4.8 shows the histograms of switching energy distribution for different input vector pairs due to process variation. The different colored histograms correspond to switching energy variation with just process variation for different input vector pairs. The overall distribution of switching energy due to the combined effects of process variation and input stimuli variation is also shown in the background in black color. This figure not only indicates that the variation in switching energy due to process variation is a normal distribution for every input vector pair but also that the overall distribution of switching energy is a superposition of many small normal distributions for several input vector-pairs (where the mean of power distribution of all the input-vector pairs spans over a wide range).

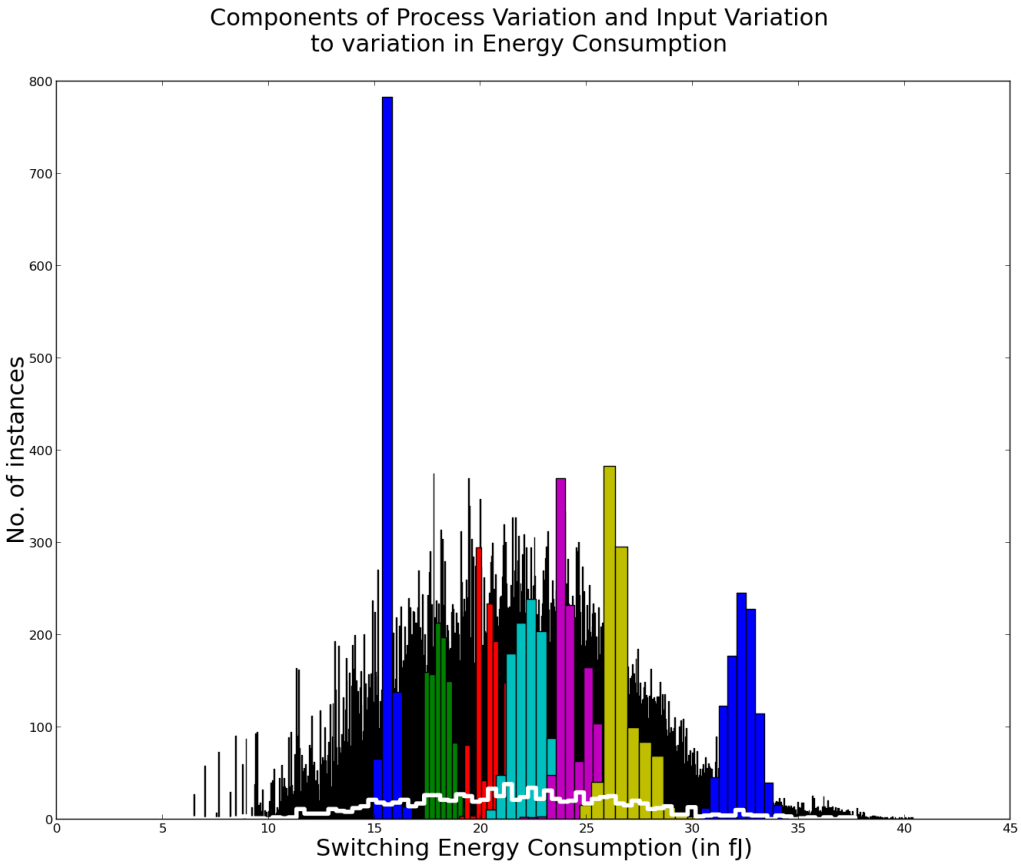


Figure 4.9: Histograms of impact of process variation versus input variation on total switching energy for Booth Multiplier circuit at $45\,\mathrm{nm}$ technology. `SDF` file is modified 1000 times for each input vector pair to identify impact of process variation. 1000 input vector pairs are applied to the circuit and simulated using standard `SDF` file to identify impact of input variation. Outline of impact input variation (for a particular `SDF` file) is shown in white.

Figure 4.9 shows the impact of process variation versus impact of variation in input stimuli on total switching energy consumption of the circuit. The impact of variation

in input stimuli (for a particular SDF file) on switching energy of the circuit is plotted in white and superimposed on the histograms of the impact of process variation. We can see that the standard deviation for power distribution with input variations (for one SDF file) is much greater than that of power distribution with process variation (for one input vector pair).

The standard deviation of total switching energies of the circuit for one input vector pair and 1000 modifications of the SDF file is computed. This value of standard deviation signifies the impact of process variation on total switching energy of the circuit (for one applied input vector pair). This process is repeated for 1000 input vector pairs. The maximum and average of all the obtained values of standard deviation are computed for both the circuits across the two technologies (45 nm and 250 nm).

Table 4.2 shows the maximum and average of all the standard deviation values obtained by the process explained above. The table also shows the range of total switching energy of the circuits. From the values of maximum standard deviation mentioned in the table, we infer that for some input vector pairs, there is a significant impact of process variation on switching energy. Since the difference between average and maximum values indicated in the table is small, we infer that there is definitely a significant impact of process variation on switching energy for every applied input vector pair.

Table 4.2: Standard deviation of distribution of switching energy due to process variation on Booth Multiplier and C880 circuits using 45 nm and 250 nm technology library

Circuit	Technology	Maximum standard deviation	Average standard deviation
Booth Multiplier	45 nm	124.21 fJ	115.96 fJ
Booth Multiplier	250 nm	5.63 pJ	5.37 pJ
C880	45 nm	67.19 fJ	63.33 fJ
C880	250 nm	5.32 pJ	5.16 pJ

Table 4.3 shows the average switching energy of the circuit along with its gate count. We observe that although Booth Multiplier has higher gate count as compared to C880 circuit, the average switching energy of the circuit decreased for 250 nm technology. We infer that the switching energy of a circuit not just depends on the total number of gates that are toggling in the circuit, but also depends on the type of gates that are

toggling. We also observe that the gate count not just depends on the technology used for logic synthesis of the circuit but also on the type of circuit and the logic functionality of the circuit.

Table 4.3: Analysis of gate count and average switching energy for `Booth Multiplier` and `C880` circuits using $45\,\mathrm{nm}$ and $250\,\mathrm{nm}$ technology library

	Booth Multiplier		**C880**	
	$45\,\mathrm{nm}$	$250\,\mathrm{nm}$	$45\,\mathrm{nm}$	$250\,\mathrm{nm}$
Total Gate Count	320	293	180	213
Average Switching Energy (fJ)	343.65	21498.12	282.40	24794.96

4.3 Analysis of glitch power variation

The switching energy consumption of the circuit is contributed by two kinds of transitions: expected steady state transitions and glitch transitions. In this section, we estimate the contribution of glitches to overall switching energy consumption of the circuit.

4.3.1 Procedure

It is assumed that the `SDF` file is modified and the circuit is simulated by *ModelSim*. `VCD` file is generated by the simulation.

Step 1: The total number of steady state transitions at every net in the circuit is computed by parsing the `VCD` file. The steady state values of all the nets before and after the application of the second input vector in the input vector pair are identified. All the rising and falling transitions among those values are summed together for each net in the circuit. This step estimates the total number of steady state toggles in the circuit.

Step 2: The number of rising and falling transitions at every net in the design is evaluated by parsing the `VCD` file by a `Python` script. This estimates the total number of

rising transitions and the total number of falling transitions at every net in the circuit, thereby estimating the total number of toggles in the circuit.

Step 3: The rising and falling input capacitances of all the fan-out gates of a net are summed individually and a look-up table is created for rising and falling capacitances at every net in the circuit. This is done once for a particular circuit. The rising and falling capacitances for each net are then extracted from this look-up table in each simulation.

Step 4: The value of number of rising transitions (obtained in step 2) at a net is multiplied with the corresponding rising capacitance value and the number of falling transitions (obtained in step 2) is multiplied with the corresponding falling capacitance. These two values are added and multiplied with V^2. This value is then evaluated for every net in the circuit and summed together to obtain the total switching energy of the circuit. This step is repeated for corresponding steady state transitions obtained in step 1. Hence, we estimate the total switching energy of the circuit as well as the switching energy of the circuit considering steady state transitions only. These two values are subtracted to obtain the contribution of glitches to overall switching energy of the circuit.

4.3.2 Results

An 8×8 `Booth Multiplier` circuit and `c880` circuit are simulated by the power extraction algorithm following the steps described in the procedure above. The `SDF` files are modified 1000 times and the circuits are simulated for 1000 input vectors. This is done for both the $45\,\mathrm{nm}$ and $250\,\mathrm{nm}$ technologies.

Figures 4.10 to 4.13 shows the contributions of glitches to overall switching energy of the circuits. The figures (a) show the distribution of total switching energy of the circuits and the switching energy contribution by glitches. We observe that the histogram of the contribution of glitches is close to a normal distribution. The areas under these histograms indicate the sum total of switching energies consumed by the circuit across all inputs and the corresponding contributions of glitches. From the area under the glitch energy histograms, we observe that they contribute significantly to overall switching energy of the circuits.

The figures (b) show the percentage contribution of glitch power to the total power with percentages on X-axis and number of inputs on Y-axis. From these figures, we observe that glitches contribute on an average, $30 - 40\%$ to the overall switching energy for `Booth Multiplier` circuit and $10 - 20\%$ for `C880` circuit. This has also been plotted for some `ISCAS '85` benchmark circuits as shown in figure 4.14. We can also see that for certain inputs, glitches account for even $70 - 80\%$ to the total switching energy. This reiterates over hypothesis that reducing the number of glitches in a circuit can significantly reduce the overall switching energy consumption of the circuit.

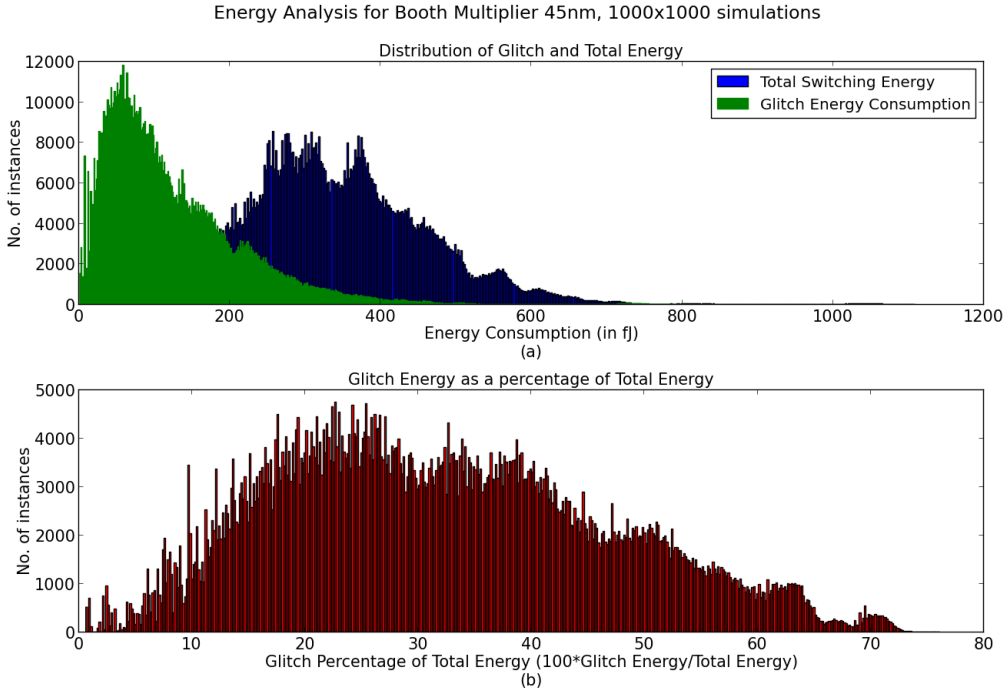


Figure 4.10: Booth Multiplier circuit, $45\,\mathrm{nm}$, 1000 input vectors, 1000 modifications of SDF file. Contribution of glitches to switching energy of the circuit: (a) Histograms of overall switching energy and contribution of glitches (b) Histogram of contribution of glitches as a percentage of total switching energy

The simulation across a large number of input variations was carried out for the `ISCAS '85` benchmark circuits, and the mean percentage contribution of the glitch power consumption to the total switching energy consumption (along with the range of percent contribution) has been tabulated in table 4.4. From the table, we can infer that though the mean percent contribution of glitches to switching energy might vary across circuits, it is significant for all of them. We can also observe that for certain inputs, the percentage contribution can go as high as 70% of the total switching energy. Hence, reduction of glitches can be one of the essential factors for low power design of circuits.

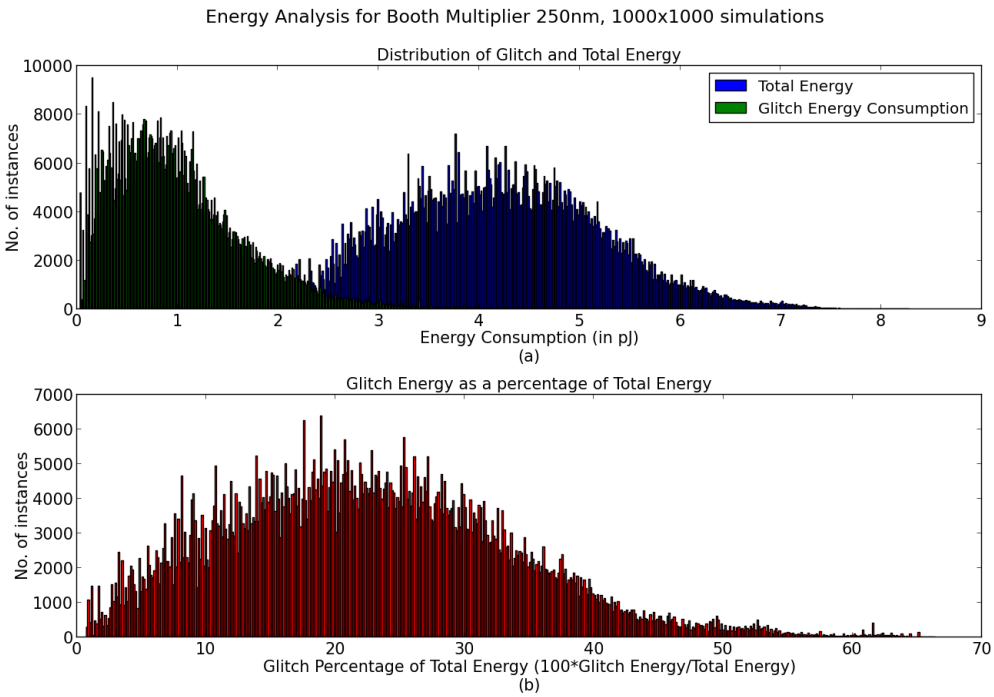


Figure 4.11: Booth Multiplier circuit, $250\,\mathrm{nm}$, 1000 input vectors, 1000 modifications of SDF file. Contribution of glitches to switching energy of the circuit: (a) Histograms of overall switching energy and contribution of glitches (b) Histogram of contribution of glitches as a percentage of total switching energy

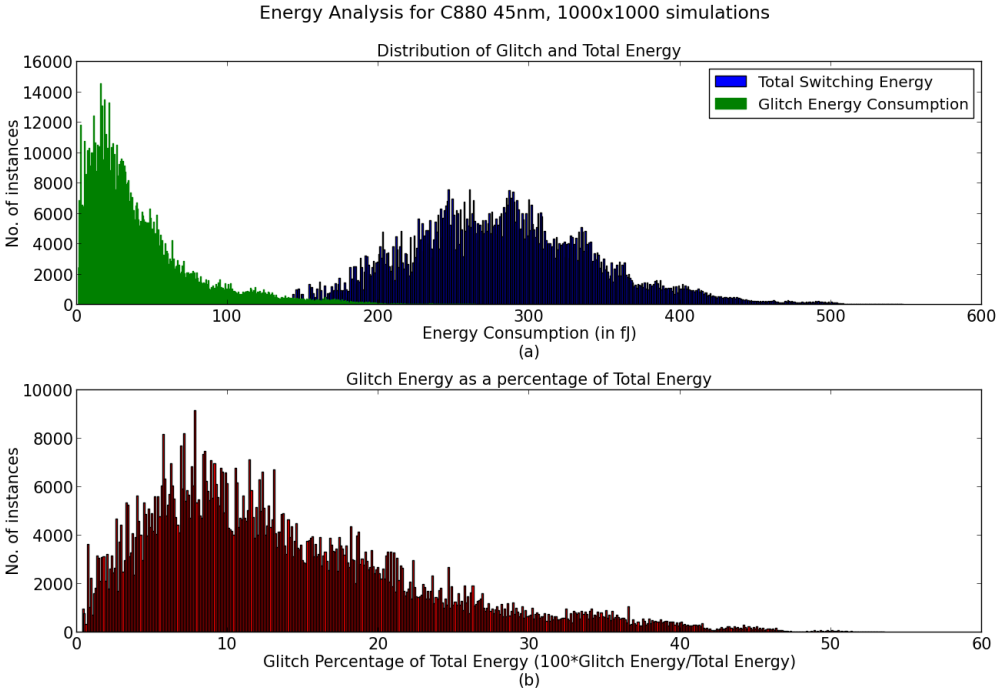


Figure 4.12: C880 circuit, $45\,\mathrm{nm}$, 1000 input vectors, 1000 modifications of SDF file. Contribution of glitches to switching energy of the circuit: (a) Histograms of overall switching energy and contribution of glitches (b) Histogram of contribution of glitches as a percentage of total switching energy

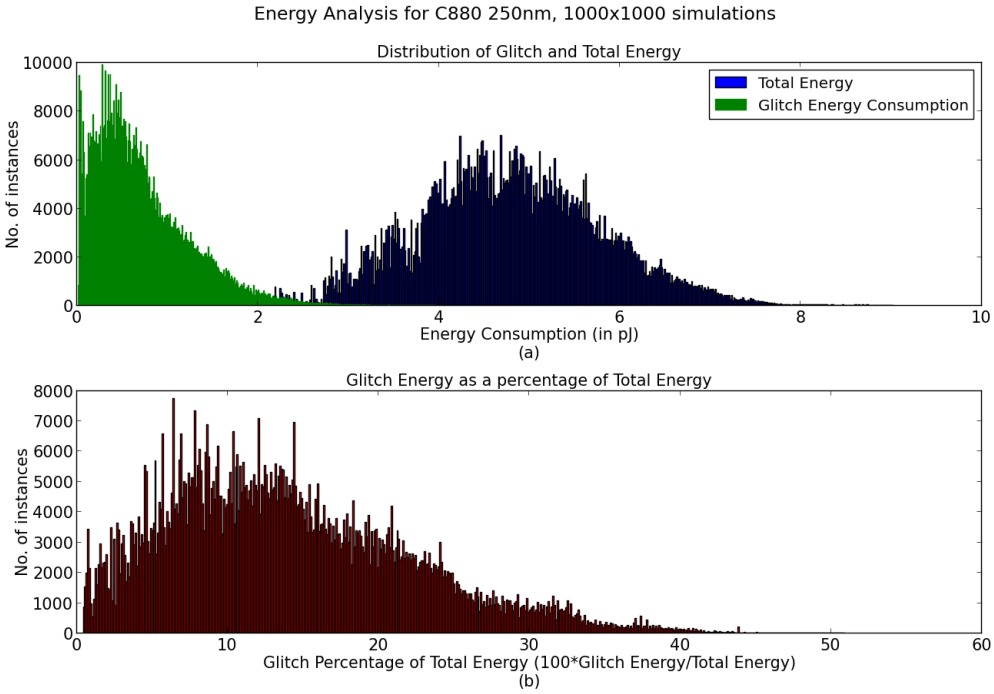
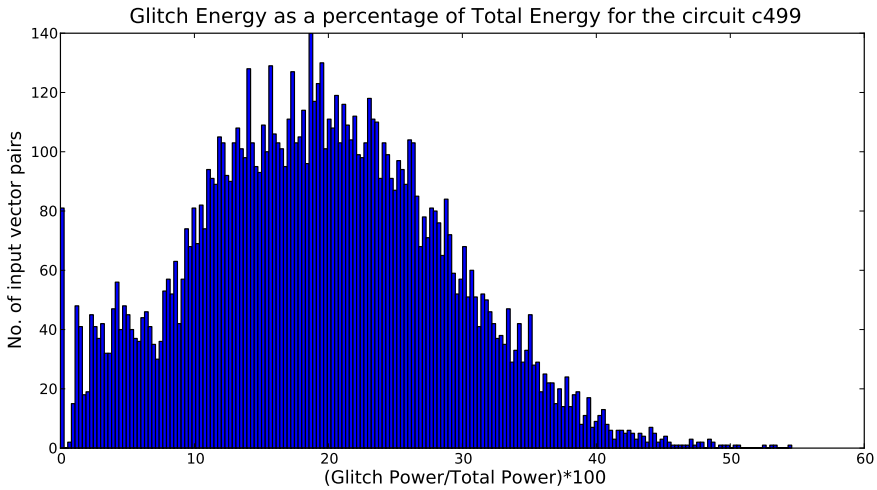


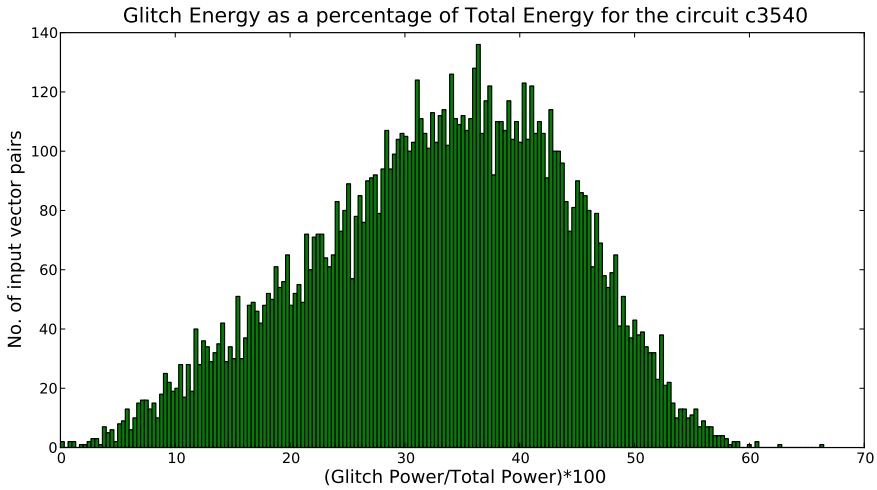
Figure 4.13: C880 circuit, $250\,\mathrm{nm}$, 1000 input vectors, 1000 modifications of SDF file. Contribution of glitches to switching energy of the circuit: (a) Histograms of overall switching energy and contribution of glitches (b) Histogram of contribution of glitches as a percentage of total switching energy

Table 4.4: The percentage of glitch power to Total Dynamic Power for `ISCAS '85` benchmark circuits

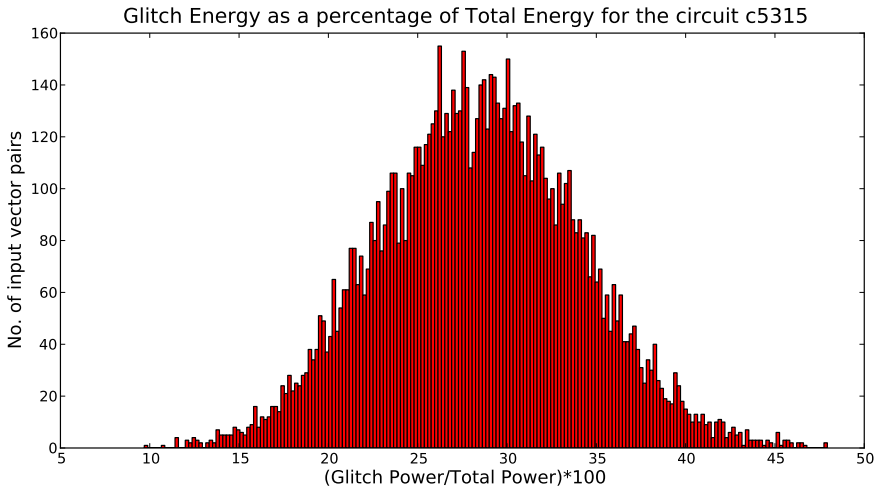
Circuit	Percentage of Glitch Power Contribution to Total Dyanmic Power (in %)	
	Mean Percentage	Standard Deviation of Percentage
c432	15.55	12.33
c880	16.88	9.65
c7552	37.45	8.92
c499	19.40	9.16
c2670	23.03	7.86
c1908	31.88	12.47
c5315	28.42	5.58
c3540	32.87	11.02



(a) Circuit: c499



(b) Circuit: c3540



(c) Circuit: c5315

Figure 4.14: Histogram of contribution of glitches as a percentage of total switching energy for some ISCAS '85 benchmark circuits.

CHAPTER 5

AMBIGUITY INTERVAL PROPAGATION

Switching energy consumption of a CMOS circuit primarily depends on the number of transitions that signals make at the gate outputs. These transitions are accounted by steady state logic transitions as well as glitches. While steady state logic transitions can be easily evaluated through logic simulation, it is difficult to estimate the average number of glitches at every node without Monte Carlo simulations. This makes the process time-consuming and the accuracy depends on the number of input vectors that are being simulated.

Ambiguity Interval Propagation is an algorithm to estimate bounds on the number of transitions that signals can make in a combinational circuit for a given input vector pair. This algorithm has been originally proposed in the paper by Alexander and Agrawal (2009). The algorithm identifies the time intervals in which transitions can possibly occur at each net, given the gate delays and time instances at which transitions at its inputs occur. It then evaluates the minimum and maximum number of transitions that can occur within these intervals. This approach thereby allows us to estimate bounds on the switching energy consumed by individual nets in the circuit.

A primary advantage of this algorithm lies in its inclusion of process variation in gate delays. In today's nano-scale technologies, there is a significant impact of process variation on gate delays and thereby on switching energy consumption, as established in the earlier sections. The bounded delay model for every gate permits us to incorporate this impact in its specification.

In the reminder of the chapter, the implementation details of the algorithm, its underlying assumptions and results have been elaborated.

5.1 Assumptions

5.1.1 Delays of gates

The algorithm assumes an inertial delay model for every gate. Accordingly, those glitch transitions at the input of a gate whose width is less than the minimum delay specified for the gate are not propagated to the output. Hence, partial glitches in the circuit are not considered by the algorithm. The delay of every gate is specified as a bounded interval of minimum and maximum delay (min, max) permitted by all the inputs of the gate. Logic synthesis of the circuit by the tool Design Compiler (DC) generates a Standard Delay Format (SDF) file, which has annotations of gate delays for the netlist. For every gate, a minimum and a maximum delay are extracted from this file, by considering the minimum and maximum delays from every input of the gate to the output. Rise and fall delays are not considered separately.

5.1.2 Process Variation

As discussed in earlier sections, process variation affects gate delays. This effect is implicitly incorporated by the algorithm through gate delay specification. A standard normal variation in the delays is assumed and the standard deviation is set to 20%. So the minimum delay (min) is reduced by 20% $(min \times 0.8)$ and the maximum delay is increased by 20% $(max \times 1.2)$. Since the worst case estimate for the bounds on delays are considered, the algorithm gives the theoretical minimum and maximum switching energy for every net. This doesn't necessarily signify the actual minimum and maximum power consumption for the net.

5.1.3 Primary inputs

The primary inputs are assumed to have no ambiguity. Hence, transitions at those nets are assumed to occur at the start of our reference time $(t = 0)$. The previous and current state of inputs are also assumed to be known.

5.2 Definitions

5.2.1 Controlling and Non-controlling value

The *controlling value* of an input of a gate is the boolean value of the input which determines the output to a known value, irrespective of the state of other inputs of the gate. This depends on the logic function of the gate and such a value need not exist for every input of a gate. The compliment of controlling value is the *non-controlling value*.

Table 5.1: Controlling and non-controlling values for various gates

Gate	Controlling value	Non-Controlling value
AND	0	1
OR	1	0
NAND	0	1
NOR	1	0
Other	x	x

5.2.2 Ambiguity Interval

Ambiguity interval for a net is defined as a tuple of two time values, called the *Earliest Arrival Time* (E_t) and the *Latest Stabilization Time* (L_t). For every signal, we define the Earliest Arrival Time as the time before which the signal would remain steady and the Latest Stabilization Time as the time after which the signal would be steady. (E_t, L_t) specify the time bounds during which a signal can have transitions. $(E_t, L_t) = (\infty, -\infty)$ signifies that the signal has no ambiguity or is in steady state condition. There can be multiple ambiguity intervals within this interval, depending on the steady state values of inputs of the gate. The evaluation of this has been detailed in further sections.

Apart from E_t and L_t values for every signal, we also define few other parameters.

E_{cn} The earliest arrival time of a signal that causes the input of a gate to change from controlling value to non-controlling value.

E_{nc} The earliest arrival time of a signal that causes the input of a gate to change from a non-controlling value to a controlling value.

L_{nc} The latest stabilization time of a signal that causes the input of a gate to change from non-controlling value to controlling value.

L_{cn} The latest stabilization time of a signal that causes the input of a gate to change from controlling value to non-controlling value.

5.3 Methodology

The first step to the propagation of ambiguity intervals is the evaluation of steady state values for every net in the design. For every applied input vector pair, the steady state values of every signal is evaluated using a zero-delay logic simulation. The ambiguity intervals of primary inputs are initialized. Then, the ambiguity intervals of every net is propagated from the primary inputs to the outputs, covering every net in the design. Using these ambiguity intervals and steady state values of every signal, the minimum and maximum number of transitions are evaluated for the design.

5.3.1 DAG representation of the netlist

The netlist of the design is read and stored as a Directed Acyclic Graph (DAG). For every gate in the design, its functionality, controlling and non-controlling value, fan-out gates, inputs and delays are stored in the graph. The library file is parsed for obtaining the functionality of every gate in the design. The delays are obtained from the SDF file as mentioned above. A *breadth first search* is done to parse through the graph for every step ahead.

5.3.2 Evaluation of steady state values

The input vectors are randomly generated and stored in a file. These are then read in as vector pairs. The evaluation is started from the gates adjoining the primary inputs. The

output value of every gate is evaluated using the corresponding functionality. A gate is evaluated only when the steady state values of all its inputs are evaluated. A breadth first search on the netlist DAG is done to propagate and evaluate steady state values till outputs are reached. This process is done for each input vector in the vector pair.

5.3.3 Evaluation of Ambiguity Intervals

The ambiguity interval of the output of a gate is determined from the ambiguity intervals of its inputs, their steady state values and the bounded gate delay of the gate.

The following equations are used to compute the ambiguity interval for a signal.

$\forall i \in inputs(gate)$,

$$E_{cn}(out) = maximum(E_{cn}(i)) \tag{5.1}$$

$$E_{nc}(out) = minimum(E_{nc}(i)) \tag{5.2}$$

$$L_{nc}(out) = minimum(L_{nc}(i)) \tag{5.3}$$

$$L_{cn}(out) = maximum(L_{cn}(i)) \tag{5.4}$$

$$E'_t = maximum(E_{cn}(out), E_{nc}(out)) \tag{5.5}$$

$$L'_t = minimum(L_{cn}(out), L_{nc}(out)) \tag{5.6}$$

Then, for the output

$$(E_t, L_t) = \begin{cases} (E'_t + mindel, L'_t + maxdel) & \text{if } (L'_t - E'_t) \geq maxdel \\ (\infty, -\infty) & \text{if } (L'_t - E'_t) < mindel \\ (E'_t + mindel, L'_t + maxdel) & \text{if } \textit{steady state transition at the output} \end{cases} \tag{5.7}$$

The above equations are slightly different from what has been proposed in the paper by Alexander and Agrawal (2009), in the cases when the ambiguity intervals are zero for some inputs when their steady states are different (like primary inputs). In that case, the outputs should still have non-zero ambiguity intervals.

Equation (5.1) indicates that the maximum of E_{cn} values among all inputs is consid-

ered when the inputs go from controlling value to a non-controlling value. This ensures that the output will not have a transition even if one of the input has a controlling value.

Equation (5.2) indicates that the minimum of E_{nc} values among all inputs is considered when the inputs go from non-controlling value to a controlling value. This ensures that the output will have a transition as soon as one of the inputs toggles from a non-controlling value to controlling value.

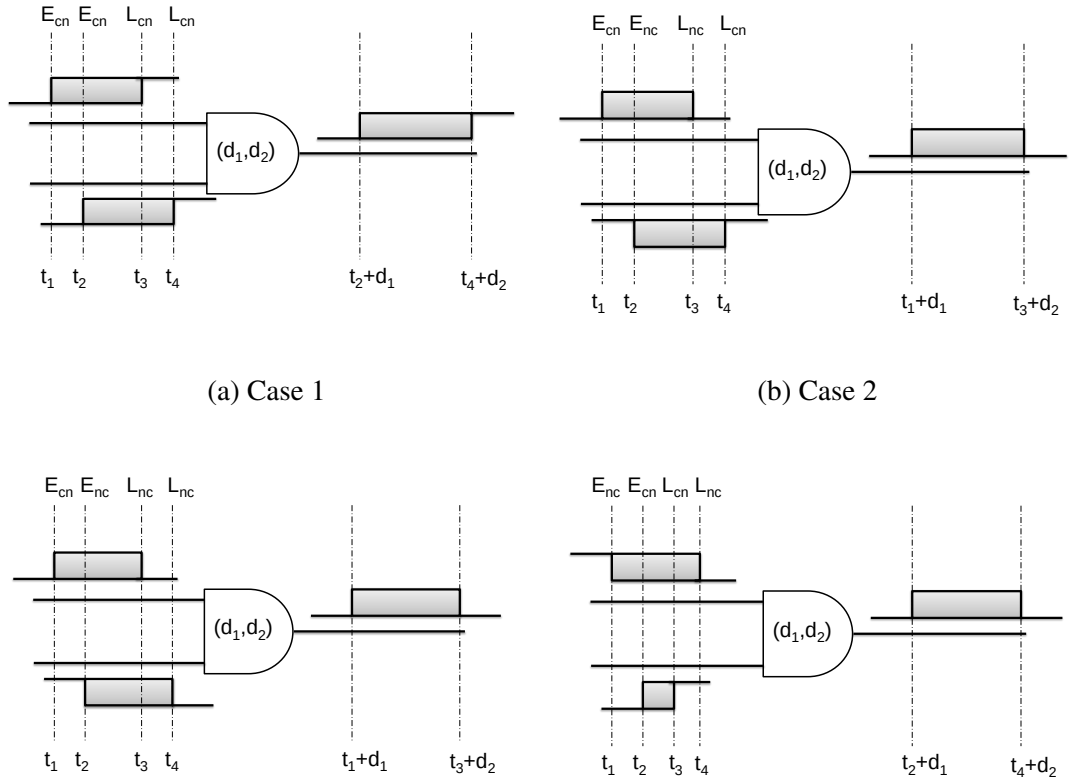
Equation (5.3) indicates that the minimum of L_{nc} values among all inputs is considered when the inputs go from non-controlling value to controlling value. The minimum is taken to ensure the final transition at the output will occur when one of the inputs transition from non-controlling value to controlling value.

Equation (5.4) indicates that the maximum of L_{cn} values among all inputs is considered when the inputs go from controlling value to a non-controlling value. The maximum is taken to ensure that the final transition at the output will not occur as long as atleast one input has a controlling value.

E'_t and L'_t are the temporary E_t and L_t values before adding the $mindel$ and $maxdel$ values to them. Equation (5.5) indicates that the maximum of E_{cn} and E_{nc} is taken as E'_t value.

Note: Here, the expression for evaluating E'_t and L'_t (equations (5.5) and (5.6)) can seem counter-intuitive, since we have used *maximum* for E'_t and *minimum* for L'_t when earliest values and the last values should likely be the minimum and maximum of all the transition times respectively. The catch here, however, is in the way we have defined E_{nc} in the case of controlling to non-controlling transition and E_{cn} in the case of non-controlling to controlling transition, and correspondingly for L_{cn} and L_{nc}.This will be clearer once we discuss further (in the same section) the assignment of these values for every case at a node.

Equation (5.7) is used to propagate the E'_t and L'_t values to the output from the input ambiguity intervals using gate delays. If the difference between L'_t and E'_t is less than the minimum delay of the gate, the inertial delay model doesn't permit that glitch to propagate to the output. Hence, E_t and L_t values are set to ∞ and $-\infty$, signifying no ambiguity at the output. If the difference is more than the minimum delay, the transitions are recognized by the gate and propagated to the output. The



(a) Case 1

(b) Case 2

(c) Case 3

(d) Case 4

Figure 5.1: Evaulation of Earliest Arrival time and Latest Stabilization time for different cases for an AND gate

earliest transition at the inputs will propagate to the output earliest by $E'_t + mindel$ and the last transition at the inputs will propagate to the output latest by $L'_t + maxdel$. If the steady state values at the output of the gate due to the applied input vector pair signify a transition, then the ambiguity interval is propagated from the input to output irrespective of the difference $L'_t - E'_t$ being greater or lesser than $mindel$.

The E_{cn}, E_{nc}, L_{cn} and L_{nc} values for the inputs of a gate are computed from the E_t and L_t values computed for those signals in the previous level and the steady state values of those nets.

Figure 5.1 shows the computation of E_t and L_t values for an AND gate for different cases of input waveforms. Here, (d_1,d_2) is the range of propagation delays of the gate. Since this is just a two input gate, the computation of Earliest Arrival time and Latest Stabilization time for this is straight-forward.

The following is done for every input signal before computation of the ambiguity interval for the gate for different cases.

Case (1) Steady state value of the input signal remains at the controlling value of the gate on applying the input vector pair

The value of E_{cn} is assigned to E_t value computed for that signal in the previous iteration. E_{nc} is equated to $-\infty$. The L_{nc} value is assigned to L_t value computed for the signal. L_{cn} value is equated to ∞.

Case (2) Steady state value of the input signal goes from controlling value of the gate to non-controlling value on applying the input vector pair

The value of E_{cn} is assigned to E_t value computed for that signal in the previous iteration. E_{nc} is equated to $-\infty$. The value of L_{cn} is assigned to L_t value computed for the signal. The value of L_{nc} is equated to ∞.

Case (3) Steady state value of the input signal goes from non-controlling value to controlling value on applying the input vector pair

The value of E_{nc} is assigned to E_t value computed for that signal in the previous iteration. E_{cn} is equated to $-\infty$. The value of L_{nc} is assigned to L_t value computed for that signal in the previous iteration. L_{cn} is equated to ∞.

Case (4) Steady state value of the input signal remains at the non-controlling value of the gate on applying the input vector pair

The value of E_{nc} is assigned to E_t value computed for that signal in the previous iteration. E_{cn} is equated to $-\infty$. The value of L_{cn} is assigned to L_t value computed for that signal in the previous iteration. L_{nc} is equated to ∞.

Case (5) Controlling and non-controlling value of the gate is 'x'

The value of E_{nc} is assigned to E_t value computed for that signal in the previous iteration. E_{cn} is equated to $-\infty$. The value of L_{cn} is assigned to L_t value computed for that signal in the previous iteration. L_{nc} is equated to ∞.

In each of these cases, since we know the steady state values of a node, the nature of first and last transition is also known (whether it is controlling to non-controlling or vice-versa).

Earliest Arrival Time Evaluation

If the first transition at a node is controlling to non-controlling, we assign $-\infty$ to E_{nc} and if the first transition is non-controlling to controlling, we assign $-\infty$ to E_{cn}. Such an assignment is paramount for the equations (5.1) to (5.6) to hold true. Because of this, only if there is no controlling to non-controlling transition, $max(E_{cn})$ will be $-\infty$, $min(E_{cn})$ will be the binding term in Equation (5.5) and hence E'_t will be equal to the *minimum* of all non-controlling to controlling transition times as expected. Also, if there is at least one controlling to non-controlling transition, then $min(E_{nc})$ will be

$-\infty$, $max(E_{cn})$ will be the binding term in Equation (5.5) and hence E'_t will be equal to the *maximum* of all controlling to non-controlling transition times.

Because of the way we have assigned E_{cn} and E_{nc} in each case, exactly one term in the evaluation expression in Equation(5.5) will be $-\infty$.

Latest Stabilization Time Evaluation

Analogous to evaluation of earliest arrival time, in this case, if the final transition at a node is controlling to non-controlling, we assign ∞ to L_{nc} and if the final transition is non-controlling to controlling, we assign ∞ to L_{cn}. Therefore only if there is no controlling to non-controlling transition, $min(L_{cn})$ will be ∞, $max(L_{cn})$ will be the binding term in Equation (5.6) and hence L'_t will be equal to the *maximum* of all non-controlling to controlling transition times as expected. Also, if there is at least one controlling to non-controlling transition, then $max(L_{nc})$ will be ∞, $max(L_{cn})$ will be the binding term in Equation (5.6) and hence L'_t will be equal to the *minimum* of all controlling to non-controlling transition times.

On a similar observation, exactly one term in the evaluation expression in Equation(5.6) will always be ∞.

In the figure 5.2, we observe that the transitions of all the nets (for one particular input) are all contained in the ambiguity intervals evaluated by the algorithm. It only remains to estimate the maximum and minimum possible number of transitions in these intervals.

5.4 Power Bounds Evaluation

5.4.1 Maximum number of transitions

The maximum number of transitions at every net in the design is computed using the ambiguity interval for that net and the delay specification of the gate. The maximum number of transitions for the primary inputs is set to 1 if there is a transition at the input node due to the applied input vector pair. It is set to 0 otherwise. This is due to the fact that primary inputs are assumed to be glitch free. Since the inertial delay

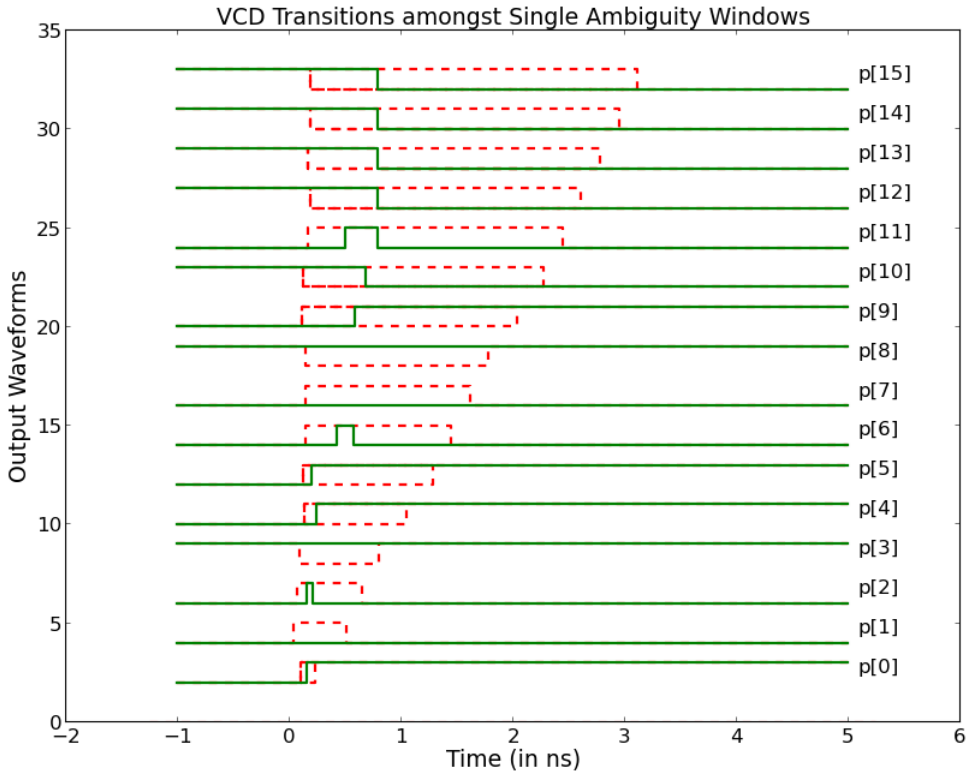


Figure 5.2: The simulated waveforms of outputs of `Booth Multiplier` Circuit P[15:0] at $45\,\mathrm{nm}$ technology, along with the single ambiguity intervals shown in dashed lines predicted by ambiguity propagation method

model restricts the minimum width of a glitch to the $mindel$ of the gate, the maximum number of transitions in an ambiguity interval is when all the transitions are spaced exactly $mindel$ apart. Here, for evaluating the maximum, we estimate two parameters $Ndelay_{tran}$ and $Ninput_{tran}$.

$\boldsymbol{Ndelay_{tran}}$

The maximum number of transitions will occur at the output when the glitch transitions are evenly spaced within the output ambiguity interval.

$$Ndelay_{tran} = \begin{cases} \frac{(L_t - E_t)}{mindel} & \text{if } (L_t - E_t) \geq 0 \\ 0 & \text{if } (L_t - E_t) = -\infty \end{cases} \tag{5.8}$$

$Ninput_{tran}$

The maximum number of transitions is also limited by maximum number of transitions at all the inputs of the gate. This is because of a simple observation that the number of transitions at the output cannot be more than the sum total of all the input transitions. $Ninput_{tran}$ is the sum of maximum number of transitions at all the inputs of the gate. For every input, the maximum number of transitions there is added to the variable $Ninput_{tran}$ if the conditions

$$E_t(input) \leq (L_t(output) - mindel)$$
$$L_t(input) \geq (E_t(output) - maxdel)$$

are satisfied for the input. These two conditions ensure that the input transitions are added only when the ambiguity interval of the corresponding input contribute to the ambiguity interval of the output i.e. there is at least some overlap between the ambiguity window of the input(after propagation) and that of the output.

Once the two upper bounds are computed, the maximum number of transitions at the output is computed as follows.

$$maxtran(output) = minimum(Ndelay_{tran}, Ninput_{tran}) \tag{5.9}$$

The minimum of the two upper bounds is taken in equation (5.9) because $maxtran$ is limited either by the total number of transitions happening at the inputs of the gate or by number of transitions permitted within the output ambiguity interval due to gate delay.

Further, the value of $maxtran$ is incremented by 1 if the computed value of $maxtran$ is odd but the steady state values at the output node indicate no transitions due to applied input vector pair. In this case, the value of $maxtran$ is made even since only even number of transitions can happen. The value of $maxtran$ is also incremented by 1 if the computed value of $maxtran$ is even but the steady state values at the output indicate a transition due to applied input vector pair. In this case, we expect the number of transitions to be odd and hence, is incremented.

5.4.2 Minimum number of transitions

The minimum number of transitions at any node in the design is evaluated using the steady state values at the node due to the applied input vector pair. If the steady state values at the node indicate a transition at that node, the value of $mintran$ is set to 1. It is set to 0 otherwise.

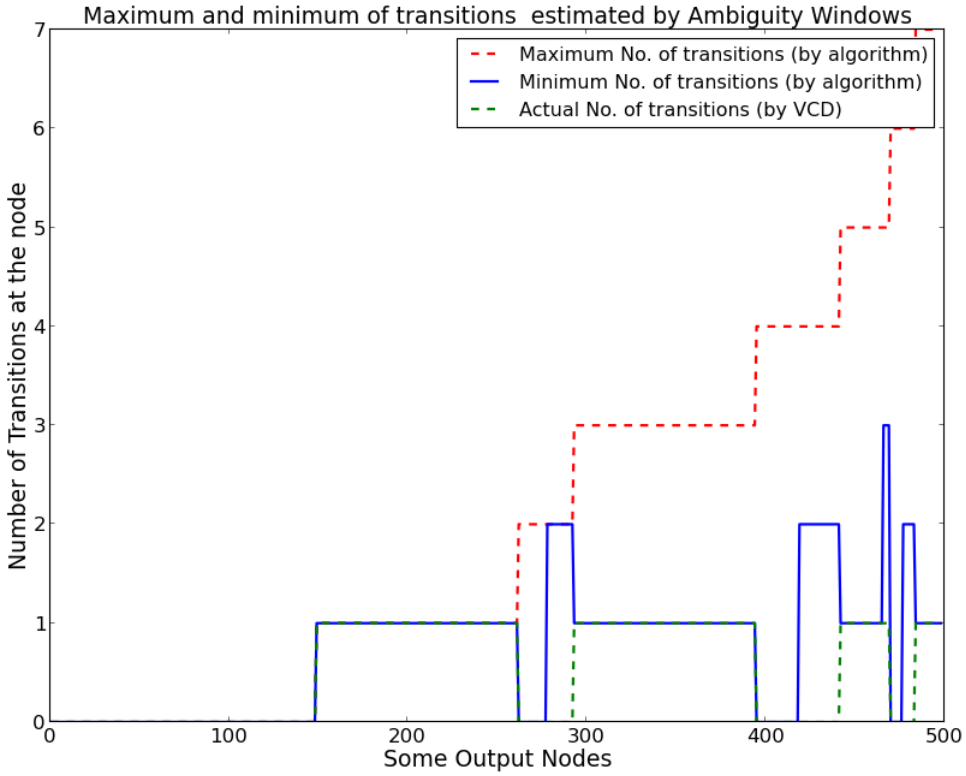


Figure 5.3: The actual number of transitions by *ModelSim* simulation at outputs of `Booth Multiplier` Circuit P[15:0] at 45 nm technology, along with the predicted maximum and minimum number of transitions shown in dashed lines predicted by ambiguity propagation method

5.4.3 Estimation of node capacitances

The rising and falling input capacitances of all the fan-out gates of a net are summed individually and a look-up table is created for rising and falling capacitances at every net in the circuit. This is done once for a particular circuit.

5.4.4 Estimation of switching energy

The switching energy of a node depends on the number of transitions at that node, the node capacitances and supply voltage of the design. It is estimated using the formula given below.

$$E_{switching} = \frac{1}{2} \times (No.\ of\ transitions) \times C \times V^2 \tag{5.10}$$

Equation (5.10) is used to estimate the bounds on switching energy for every node in the design. The capacitance C is taken as an average of rise and fall capacitances of that gate. The voltage V depends on the technology used for the logic synthesis of the design. The lower bound for switching energy is obtained using $mintran$ as the value for number of transitions. Similarly, the upper bound for the switching energy is obtained by using $maxtran$ as the value for number of transitions.

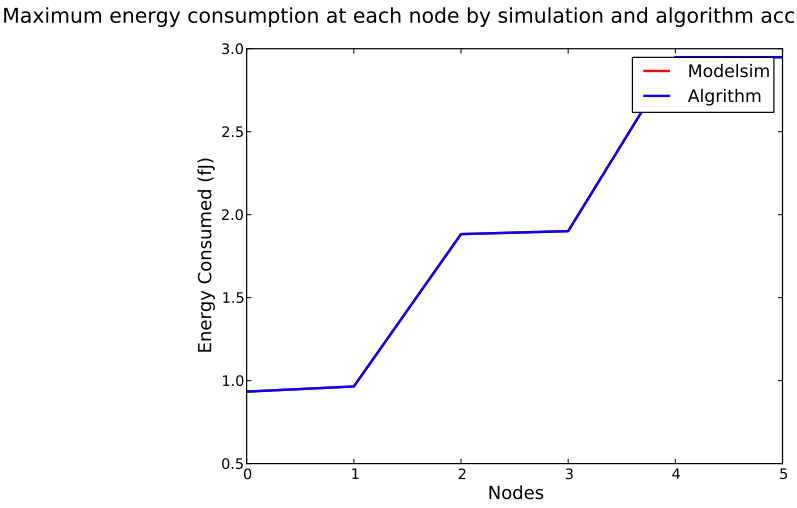
5.5 Results

We now use the methodology described in the previous sections to compute the power consumption bounds for some of the `ISCAS '85` benchmark circuits. Also, we simulate these circuits for 500 inputs and 500 process variations for each file (250000 simulation runs each) in order to extract the maximum power consumed at each node. We plot the actual maximum by simulations and the maximum bound by ambiguity propagation method in the Figures 5.4 and 5.5.

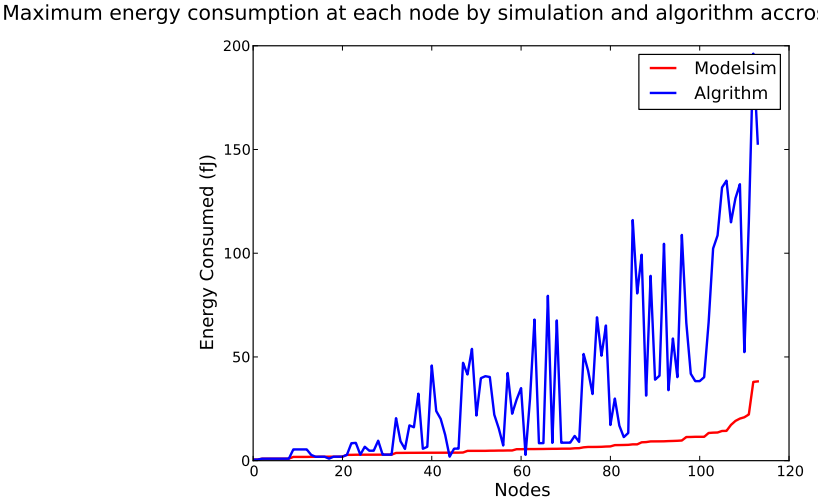
In these figures, we can observe that the maximum bound by ambiguity window is mostly either equal to or larger than the actual maximum by simulations, and for the small circuits like c17, they are the same.

We can also observe that there is a large correlation between both the maximums i.e. most of the nodes with highest maximum power consumption by ambiguity windows are also the ones with actual maximum power consumption.

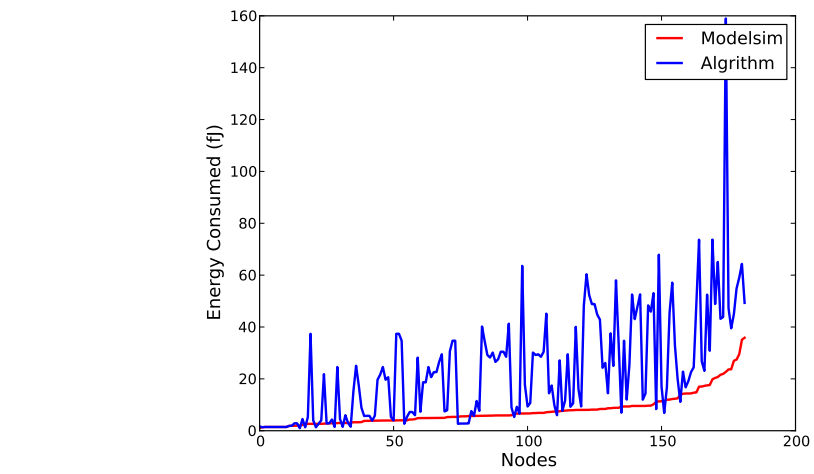
Table 5.2 consists of the time taken for complete simulation across all inputs and process variation and the time taken by ambiguity interval algorithm for the same number of inputs for some `ISCAS '85` benchmark circuits. We can see that the Ambiguity



(a) Circuit: c17

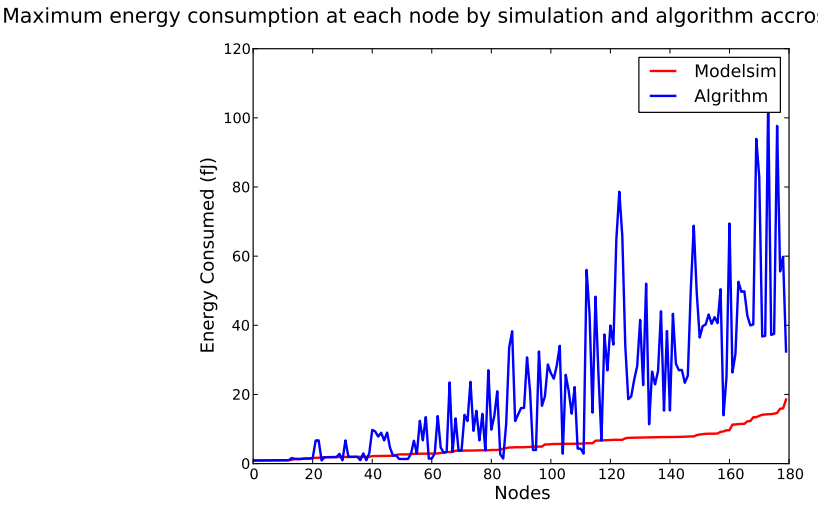


(b) Circuit: c432

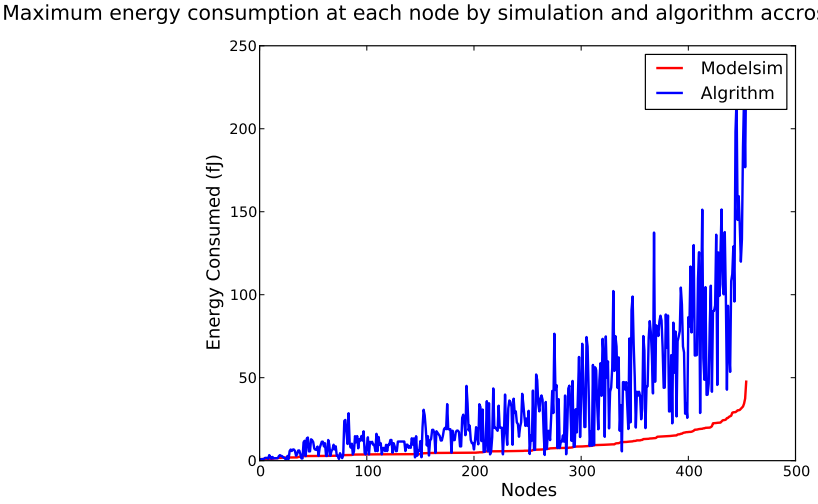


(c) Circuit: c1908

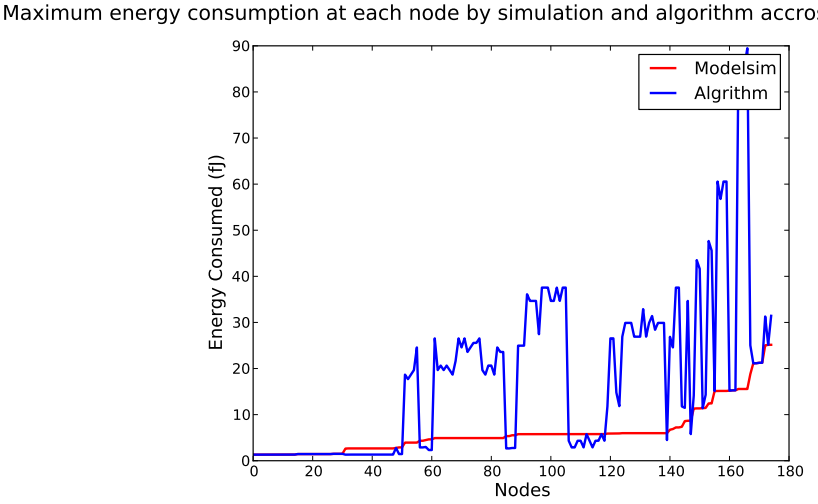
Figure 5.4: The maximum energy dissipation at each net of the circuit as estimated by ambiguity propagation approach and ModelSim simulation at $45\,\mathrm{nm}$ Nangate technology for different `ISCAS` benchmark circuits: C17, C432, and C1908 (Nets on x-axis sorted as per their energy values by ModelSim simulation)



(a) Circuit: c880



(b) Circuit: c3540



(c) Circuit: c499

Figure 5.5: The maximum energy dissipation at each net of the circuit as estimated by ambiguity propagation approach and ModelSim simulation at $45\,\mathrm{nm}$ Nangate technology for different `ISCAS` benchmark circuits: C880, C3540, and C499 (Nets on x-axis sorted as per their energy values by ModelSim simulation)

Table 5.2: Comparison of computational time taken by complete Simulation and Ambiguity Window propagation for 500 input vector pairs and for 20% process variation

Circuit	Gate Count	Computational Time (in seconds)	
		Ambiguity Interval Propagation	Monte Carlo Simulation
c17	11	0.2340	11650.11
c1908	215	13.91	11721.67
c3540	505	49.69	16545.55
c432	150	11.56	11708.85
c499	216	9.34	11746.97
c880	240	12.48	11644.55

Interval Propagation algorithm implementation significantly reduces the computational time for identifying the high-glitch nodes of a circuit.

5.6 Multiple Ambiguity Windows

In the above sections, we saw how to estimate a single ambiguity interval for every node in the design. The single ambiguity interval was then used to estimate the minimum and maximum number of transitions at each node. In many cases, this gives a pessimistic bound for the switching energy. This is because the above algorithm takes care of all ambiguity intervals and combines them into one single interval. For example, the ambiguity intervals at the inputs might cause non-overlapping intervals at the output of the gate. In this scenario, combining both intervals will give a pessimistic upper bound on the number of transitions. Hence, multiple ambiguity intervals are needed to better estimate the upper bound on the number of transitions.

5.6.1 Evaluation of Multiple Ambiguity Intervals

Assigning just one Earliest Arrival time and one Latest Stabilization time might be inefficient and very pessimistic in some cases. When there is very less overlap between the ambiguity windows of the inputs, then the bounds might be much closer to the actual bounds when multiple ambiguity intervals are considered, i.e. a list of (E_t,L_t) tuples.

The multiple ambiguity intervals at the output can be obtained once the multiple

ambiguity intervals at the inputs of the gate are known. All the multiple ambiguity windows at each of the inputs of the gate are stored in one single array. The E_t and L_t values at each input for all the multiple ambiguity intervals at that node are modified as $E_t + mindel$ and $L_t + maxdel$ respectively and stored. The several intervals are then sorted. This will give rise to several overlapping intervals as well. So, a union of all these intervals is taken to segregate them into discrete intervals, without any overlap. An intersection of these intervals with the single ambiguity interval from the above section is taken to give rise to multiple ambiguity intervals at that node. If the single ambiguity interval indicate no ambiguity at the output node, ($(E_t, L_t) = (\infty, -\infty)$) and if it is at controlling value, then the multiple ambiguity intervals are discarded and the output is set to have no ambiguity.

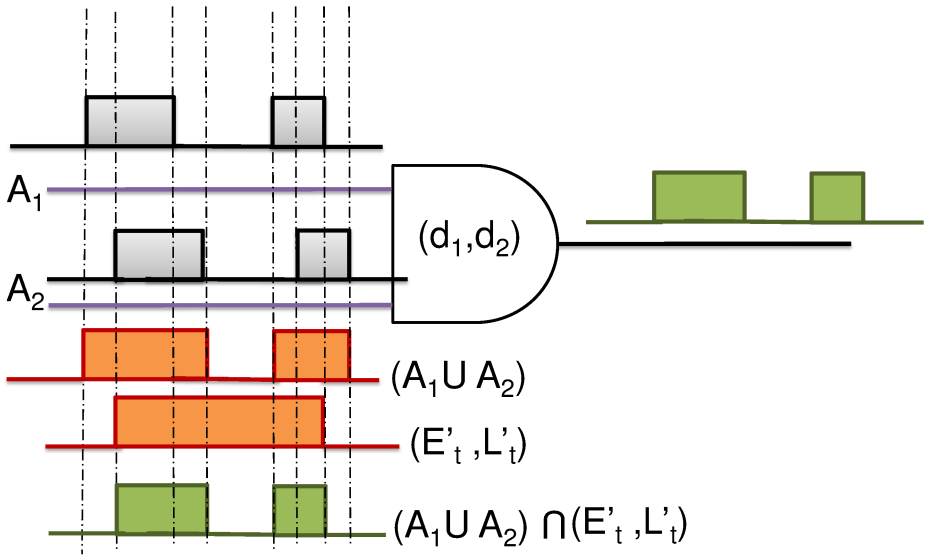


Figure 5.6: Evaluation of multiple ambiguity windows for an AND gate

$$(E_t, L_t) = (E'_t + \delta_m in, L'_t + \delta_m ax) \bigcap \left\{ \bigcup_{i \in \{Inputs\}} [(E_t, L_t)]_i \right\} \tag{5.11}$$

In the Equation 5.11, (E'_t, L'_t) are evaluated as described in the previous section, and $[(E_t, L_t)]_i$ is the list of multiple ambiguity intervals for input i of the gate. This is clearly depicted in the figure 5.6 for an AND gate.

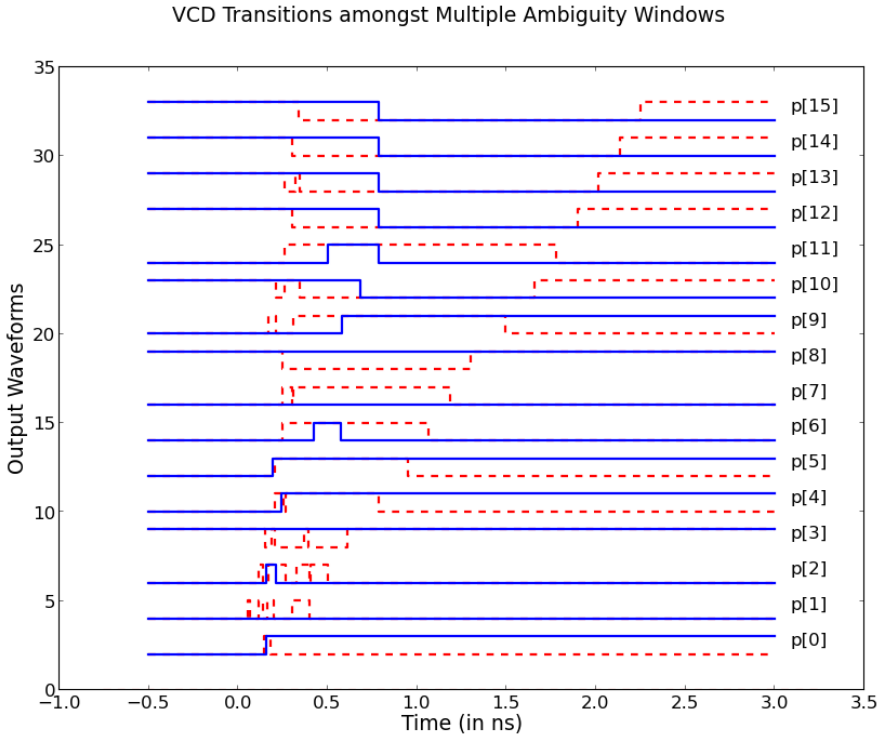


Figure 5.7: The simulated waveforms of outputs of `Booth Multiplier` Circuit P[15:0] at $45\,\mathrm{nm}$ technology, along with the multiple ambiguity intervals shown in dashed lines predicted by ambiguity propagation method

The figure 5.7 shows how the transitions as simulated by *ModelSim* lie within the reduced multiple windows estimated by the algorithm. It is also important here to note that the algorithm uses a bounded delay model while the process variation induced on date delays during simulation is normal and hence has no bounds. So, the windows will contain all the transitions for most of the cases but not necessarily all the cases.

CHAPTER 6

PROBABILITY PROPAGATION

6.1 Introduction

The extent of switching energy consumption depends heavily on the inputs applied. This has been demonstrated in the Section 4.2 in the Chapter 4, and can be observed in the figure 4.9. In the previous chapter, we implement an algorithm that estimates the maximum and minimum switching energy consumed at every net of the circuit for a particular input vector pair. However, these bounds might themselves vary significantly for different input vector pairs. Since the time complexity of the algorithm is better than full simulation (for large number of process variations) in orders of magnitude, one can estimate the over-all average power consumed by a Monte Carlo analysis using the algorithm for large number of input-vector pairs. But this approach might not be feasible if the circuit is to be incrementally modified based on the outputs of algorithm i.e. if modifications are done at the node with highest power consumption.

In this Chapter, we implement an algorithm to estimate the average switching energy consumption at a net across all input vector pairs without Monte-Carlo analysis, using a probabilistic approach. We assign static and transition probabilities to all the primary inputs of a combinational circuit and then propagate them through the subsequent gates to the outputs to evaluate the static and transition probabilities at every net of the circuit. This method is inspired by similar approaches implemented in the paper by Tsui *et al.* (1993) and in the paper by Ghosh *et al.* (1992). Probabilistic approaches are not just impressive in largely reducing simulation times in estimate average switching activities, but also provide for additional analysis to be performed on the nets, for example, evaluating the conditional probabilities as required in the paper by Monteiro *et al.* (1993).

For circuits with large number of primary inputs, the Monte Carlo Simulations can be largely time-intensive or inaccurate, where this approach can be particularly time saving. While this method, as described in this thesis, is restricted to combinational

circuits, it can also be extended to sequential circuits treating the outputs of each register as primary inputs for the corresponding combinational block. Also, one can assign unequal input static and transition probabilities for different primary inputs and evaluate the corresponding expected switching power.

One of the biggest challenges in probabilistic approach is to model the correlation of the inputs. For the beginning part of the algorithm description we have assumed that the inputs are uncorrelated, and then we later explain how to consolidate even the correlation between the inputs.

6.2 Assumptions

6.2.1 Gate Delays

The general gate delay model is used for this algorithm. The bounded gate delay model is ignored, and a specific gate delay is assigned to each gate in the circuit. While the propagation delay can vary for each input of the gate, it does not vary for different kinds of transition i.e. the rising and the falling transitions are averaged for each input. Also, for the evaluation of transition probabilities, the *Inertial Delay Model* is ignored, i.e. any transition at the input of a gate can potentially propagate to the output regardless of whether there is another transition within a time interval of the propagation delay.

6.2.2 Primary Inputs

The inputs are assumed to have a transition at time $t = 0$. The static and transition probabilities at that time instance the inputs are known and are provided as inputs to the algorithm.

6.2.3 Gate Outputs

All the gates are assumed to have just one output. If a gate has more than one output, for example a *Full Adder*, it is split into two separate gates with the same primary inputs but different and unique outputs. The logic functions for these split gates are also assigned

accordingly.

6.2.4 Correlation of inputs

The inputs are assumed to be uncorrelated for initial description of the algorithm, and modification of the algorithm in order to account for correlation is explained later.

6.3 Zero Gate Delays

In this approach, all the transitions at all the nets in the circuit are treated as probabilistic events. But since we are assuming that the propagation delays of gates are deterministic, the set of time instances where there can be a possible transitions is countably finite and deterministic.

We first implement the algorithm with the assumption of zero gate delays for all the gate delays and then improvise it to account for non-zero gate delays. With this assumption, all all the transitions in the circuits will occur at $t = 0$, and there will be at most one transition at every node.

6.3.1 Terminology

The following symbols will be used to describe the method:

$P_i^s(0)$ Static probability of the node i being zero at $t = 0^+$

$P_i^s(1)$ Static probability of the node i being one at $t = 0^+$

$P_i^t(\theta)$ Transition probability of the node i having a transition θ at $t = 0$, where $\theta \in \{00, 01, 10, 11\}$

M_i Set of all input state (state of inputs) for the gate with output node i

S_i Set of all possible input state transitions for the gate with output node i, i.e. $S_i = \{(m_i m_j)\, where\, i, j \in M_i\}$

$P_i(\sigma)$ Probability that the inputs for the gate with output node i have transition σ, where $\sigma \in S_i$

$T_i(\sigma)$ The set of transitions at the inputs for the gate with output node i corresponding to the input state transition σ

$X_i(\theta)$ The set of input state transitions transitions for the gate with output node i corresponding to the output transition θ, where $\theta \in \{00, 01, 10, 11\}$

Here, the static probabilities are defined as the probability of being at a particular state right after the time instance (or right after the transition at the time instant).

An input state for a gate is a combined state of the inputs. For example, for a 2-input gate, the set of all the input states would be $\{00, 01, 10, 11\}$. Hence, the set of all input state for a gate with n inputs will be of the size 2^n.

An input state transition for a gate would be a pair of any two input states, one belonging to 0^- and the other belonging to 0^+. It follows that the set of all input state transitions for a gate with n inputs will be of the size 2^{2n}.

Every input state transition for a particular gate with output i corresponds to a unique transition at each of its inputs, i.e. $\sigma \equiv \{\sigma_i \text{ where } \sigma_i \text{ is transition at net } i\}, \sigma \in S_i$, and we call this set $T_i(\sigma)$.

It can also be observed that every state transition for a gate causes a unique transition at the output, one from $\{00, 01, 10, 11\}$. So, the entire set S_i can be further divided into four sets $X_i(\theta)$ where $\theta \in \{00, 01, 10, 11\}$, such that

$$X_i(\theta) = \{\sigma \text{ where } \sigma \in S_i \text{ and } T_i(\sigma) \equiv \theta \text{ at } i\}$$

Note: Since we have assumed that all gates have a unique output, there is a one-to-one mapping between all the gates and the nets in the circuits (except for the primary inputs). So, we can refer to a gate by the index of its output node.

6.3.2 Methodology

From the definitions of static and transition probabilities, it can be inferred that

$$P_i^s(0) = P_i^t(00) + P_i^t(10) \tag{6.1}$$

$$P_i^s(1) = P_i^t(01) + P_i^t(11) \tag{6.2}$$

Now, we shall consider a particular gate with output i and with n inputs, and assume

that the static probabilities and transition probabilities of its inputs are known. We can generate all possible input states and form the set M_i with 2^n elements and from M_i, we can generate the set S_i with 2^{2n} elements i.e. all possible pairs of elements in M_i. Using the logic function of the gate, we can find out the output transition corresponding to each element in S_i, and we can thus generate the four sets $X_i(\theta)$ where $\theta \in \{00, 01, 10, 11\}$.

If we assume that the inputs are uncorrelated (a transition at one input is independent of transitions at all other inputs), then the probability of input state transition σ can be evaluated from equation (6.3).

$$P_i(\sigma) = \prod_{\theta \in T_i(\sigma)} P_j(\theta), \text{ where } j \text{ is the input corresponding to } \theta \tag{6.3}$$

Now we have the probability of every input state transition, and a mapping of every input state transition to an output transition (the X_is). Since we assume that inputs are uncorrelated, all the input states are also independent of each other. Hence, we can evaluate the transition probabilities of the outputs by just summing up the input state transition probabilities corresponding to that transition at the output, as implied by equation (6.4).

$$P_i(\theta) = \sum_{\sigma \in X_i(\theta)} P_i(\sigma), \text{ for } \theta \in \{00, 01, 10, 11\} \tag{6.4}$$

This equation gives us all the transition probabilities at the output node i with just the transition probabilities of the input nodes and the logic function of the gate corresponding to the node i. Using the transition probabilities and the equations (6.1) and (6.2), we can compute the static probabilities at each node.

Thus, we can start at the primary inputs (whose static and transition probabilities are provided) and propagate the static and transition probabilities through the gates at each subsequent level till the output when we will have the static and transition probabilities of all the nets in the circuit in just one simulation run.

6.3.3 Power Estimation

After having computed the transition probabilities, the expected number of transitions at a node i for zero-delay case is given by,

$$\begin{aligned} E_i &= 0 \times P_i(00) + 1 \times P_i(01) + 1 \times P_i(10) + 0 \times P_i(11) \\ &= P_i(01) + P_i(10) \end{aligned}$$

Now that we have expected number of transitions at each node, and we have the node capacitances at each node as computed in section 5.4.3 in Chapter **??**, we can compute the average switching power consumption at each node and hence the total average switching power consumption of the circuit using the Equation .

$$E_{switching} = \sum_{i \in \{\text{All Nets}\}} \frac{1}{2} \times E_i \times C_i \times V^2 \tag{6.5}$$

6.3.4 Results

The zero-delay probability propagation was performed on two circuits : Booth Multiplier and c880 and the expected number of transitions at each net in the circuit was evaluated using the algorithm. Also, the expected number of transitions was also evaluated by Simulation for large number of inputs for each of the circuits to verify the performance of the algorithm.

We can observe from figures 6.1 and 6.2 that the estimated expected number of transitions at all the nodes of the circuits are almost as much as the values evaluated by *ModelSim* Simulation. Since there can be at most one transition at each node, we can see that all expected values lie between 0 and 1.

6.4 Non-zero Gate Delays

Until now, we looked at how to compute the transition probabilities at all the nets in the circuit and expected power consumption of the circuit in the case of zero gate delays, where all the transitions in the circuit happened exactly at $t = 0$. We shall now allow

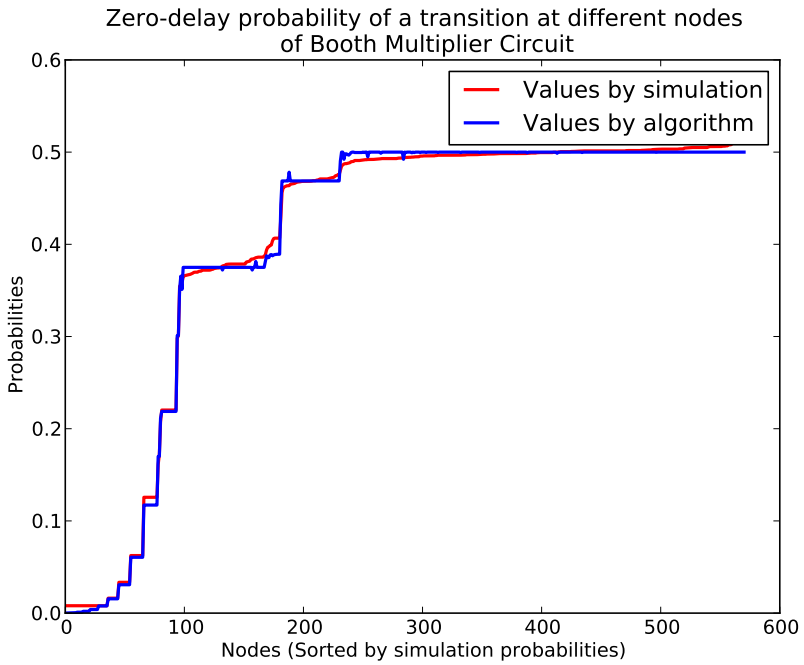


Figure 6.1: The expected number of transitions for all the nodes by the probability propagation algorithm as well as by Modelsim Simulation (for 10000 inputs) for a Booth Multiplier circuit

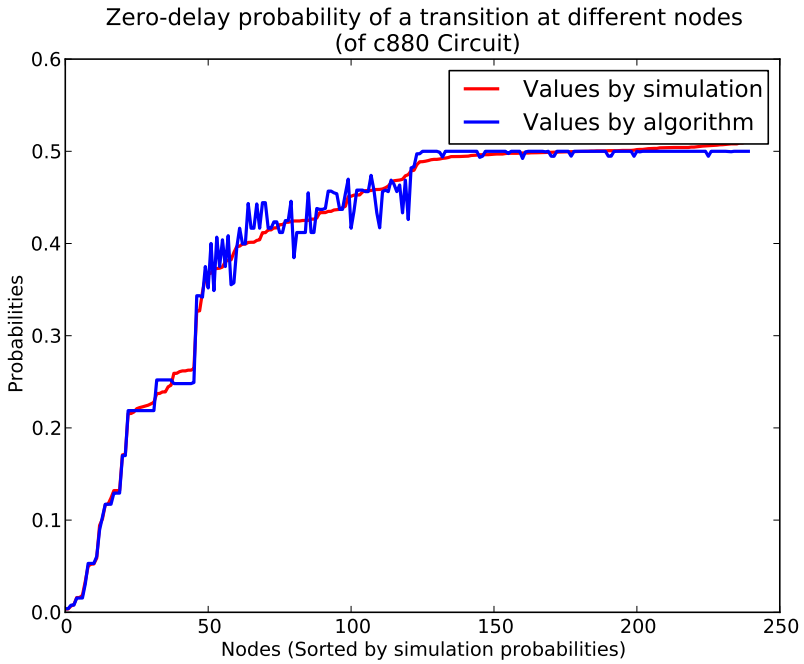


Figure 6.2: The expected number of transitions for all the nodes by the probability propagation algorithm as well as by Modelsim Simulation (for 10000 inputs) for a c880 circuit

the gates to have non-zero (but fixed) gate delays. In this case, there can be more than one time instance where an event occur and hence there can be more than one transition at a node. We improvise on the algorithm explained in the previous section to compute the expected number of transitions at each net of the circuit.

6.4.1 Terminology

Since there are multiple time instances where a transition can occur, we have slightly a modified terminology for this analysis.

T^i_{tran} The set of all time instances when a transition can occur at a node i

$I_i(t)$ The set of all inputs that can cause a transition at node i at time instance t, where $t \in T^i_{tran}$

For every node, we have multiple time instances when there can be a possible transition at the node. So, we carry out a similar procedure as the last section, but this time, at every time instance t.

$P^s_i(t)(0)$ Static probability of the node i being zero at time instance t

$P^s_i(t)(1)$ Static probability of the node i being one at time instance t

$P^t_i(t)(\theta)$ Transition probability of the node i having a transition θ at time instance t, where $\theta \in \{00, 01, 10, 11\}$

$M_i(t)$ Set of all input state (state of inputs) for the gate with output node i at time instance t

$S_i(t)$ Set of all possible input state transitions for the gate with output node i at time instance t,
i.e. $S_i(t) = \{(m_i m_j)\, where\, i, j \in M_i\}$

$P_i(t)(\sigma)$ Probability that the inputs for the gate with output node i have transition σ at time instance t, where $\sigma \in S_i(t)$

$T_i(t)(\sigma)$ The set of transitions at individual inputs for the gate with output node i corresponding to the input state transition σ at time instance t

$X_i(t)(\theta)$ The set of input state transitions transitions for the gate with output node i corresponding to the output transition θ, at time instance t, such that the inputs not in $I_i(t)$ do not change in the input state transition, where $\theta \in \{00, 01, 10, 11\}$

We have just created a copy of all the parameters for each time instance at a node.

6.4.2 Methodology

The static probabilities of a node a time instance t can be obtained from the transition probabilities from the following equations.

$$P_i^s(t)(0) = P_i^t(t)(00) + P_i^t(t)(10) \tag{6.6}$$

$$P_i^s(t)(1) = P_i^t(t)(01) + P_i^t(t)(11) \tag{6.7}$$

It is also important to note here that if $t1$ and $t2$ are two consecutive time instances in the set T_{tran}^i, then the static probabilities of the node i at any time instance in the interval $(t1, t2)$ are same as the static probabilities of the net at time instance $t1$.

We must also observe the modification in definition of $X_i(t)(\theta)$, which is now the set of only those input state transitions where the inputs that do not cause a transition at time instance t do not change (the ones that do cause a transition might or might not change).

Now, let us consider a gate with n inputs whose transition times, all static and transition probabilities are known. The first step is to compute T_{tran}^i which can be obtained by summing up each transition time in input j by the propagation delay, i.e.,

$$T_{tran}^i = \bigcup_{j \in \text{Inputs}} \{T_{tran}^j + \delta_j\}, \tag{6.8}$$

where δ_j is the propagation delay for the input j.

Once we have the T_{tran}^i, we can now compute all the parameters defined at each time instance in T_{tran}^i using the parameters in the previous time instances. For the computation of transition probabilities, just like in Equation 6.3, we multiply all the transition probabilities for the inputs in $I_i(t)$ but we multiply the static probabilities for the other inputs (since there is no transition at those nodes).

The static probabilities of a node at a particular time instance can be computed by the static probabilities at the most recent time instance. In the special case of the first time instance, we use the probabilities given by the zero-delay simulation as the steady state probabilities since $t = -\infty$.

The four transition probabilities at each time instant can then be evaluate using the equation 6.9.

$$P_i(t)(\theta) = \sum_{\sigma \in X_i(t)(\theta)} P_i(t)(\sigma), \text{ for } \theta \in \{00, 01, 10, 11\} \tag{6.9}$$

This equation now gives all the transition probabilities at the output node i at time t using the transition probabilities of the input nodes, and the output node at the previous time instances as wells as the logic function of the gate corresponding to the node i. Using the transition probabilities and the equations (6.6) and (6.7), we can compute the static probabilities at each node at each time instance t.

Again, like explained in the previous section, we can initialize all the static and transition probabilities at the inputs, assign gate delays to all the gates and then propagate the static and transition probabilities through the gates for each time instance t at each subsequent level till the outputs. This would provide us the the static and transition probabilities for all the nets in the circuit at all the possible time instances in just one simulation run.

6.4.3 Power Estimation

Computing the expected number of transitions at a node i for non-zero delay is summing up the expected value of transitions at each time instances (since expectation of sum is equivalent to sum of expectations), as given be the equation

$$\begin{aligned} E_i &= \sum_{t \in T^i_{tran}} 0 \times P_i(t)(00) + 1 \times P_i(t)(01) + 1 \times P_i(t)(10) + 0 \times P_i(t)(11) \\ &= \sum_{t \in T^i_{tran}} P_i(01) + P_i(10) \end{aligned}$$

As pointed out earlier, this expected value can be greater than 1 for a net in the circuit. The methodologies differ only until computing the static and transition probabilities of the nets. Once we find the expected values of the transition, the remainder of the procedure is the same. So, we can now use the node capacitances extracted at

each node and compute the average switching power consumption at each net in the circuit as well as the total average switching power consumption of the circuit using the Equation ,.

$$E_{switching} = \sum_{i \in \{\text{All Nets}\}} \frac{1}{2} \times E_i \times C_i \times V^2 \tag{6.10}$$

With this methodology, we have computed these transition probabilities for `c880` circuit. We have also done a simulation for 10000 inputs using ModelSim in order to compare the results. The expected number of transitions at each node of the circuit as given the algorithm and as seen from the simulation are presented in figure 6.3

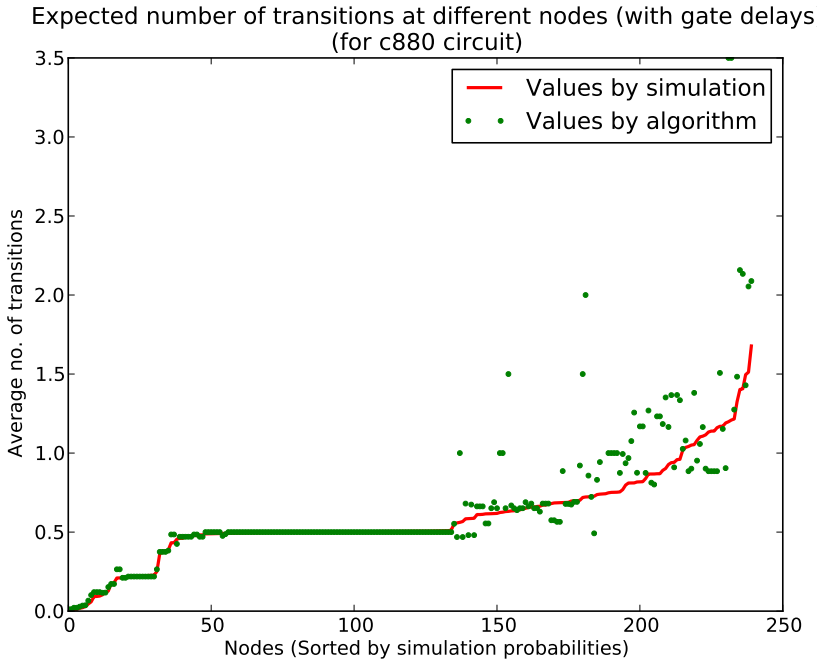


Figure 6.3: The expected number of transitions for all the nodes by the probability propagation algorithm as well as by Modelsim Simulation (for 10000 inputs) for a Booth Multiplier circuit for the case of non-zero delays

As we can see from figure 6.3, the estimated values for average number of transitions from the algorithm and the simulation are almost the same for the nets in the first few levels of the circuits, but differ as the number of glitches increase i.e. at the later levels of the circuit. This is because of the errors due to many assumptions that we have made in this approach that become increasingly significant with the increase in possible number of time instances at which transitions can occur at the net.

6.4.4 Limitations

Correlation of Inputs

Assuming that the inputs are uncorrelated can lead to significant errors in some particular cases, depending on the logic functions of the nodes in terms of primary inputs. While modeling correlation accurately might be too complex, it can be approximately modeled by tagging waveforms and equating correlation of two waveforms as correlation of their steady state values, similar to the approach by Tsui *et al.* (1993).

Inertial Delay Model

Our approach ignores Inertial Delay Model, while the simulations by ModelSim that we are using to validate the results assume Inertial Delay Model. Hence, the number of expected transitions shown by our algorithm can be much higher than the simulation value because of not ignoring the transitions that are spaced too closely.

Gate Delays

The gate delays for rise and fall transitions are averaged in our approach, while they are not in simulations. This might also lead to variations in number of transitions, but is less significant than the previous two limitations.

CHAPTER 7

GLITCH POWER REDUCTION

We have seen in earlier chapters that glitches contribute significantly to overall switching energy consumption of the circuit. The two approaches, *ambiguity propagation* and *probability propagation*, as discussed earlier, help us identify the set of nets in the circuit which have high value of average number of glitch transitions. However, glitches which occur at nets at an earlier level in the circuit propagate through the fan-out gates causing more number of transitions at further levels in the circuit. Hence, blockage of these glitches will not just reduce the switching power at that net, but also reduce the switching power caused by transitions propagated further in the circuit. The challenge now lies in identifying those nets in the circuit which cause the maximum power reduction if glitches at that node are blocked. In this chapter, we propose a methodology to identify these nodes and estimate the amount of power saved. We also validate the results of our approach by placing a latch at those nets to prevent glitches and estimating actual power reduction.

7.1 Sensitivity analysis

The blocking of glitches at certain nets in the design can dramatically reduce the switching activity of the circuit. If several stages of combinational logic are pipelined together, blockage of glitches at earlier levels of the design can significantly reduce the overall power consumption of the design. If there exists a mechanism to block glitches at a node without altering the functionality of the design, then this approach will help us identify the set of nodes which are the best choices to block glitches and thereby reduce power.

The power saved by blocking the glitch transitions at a node is given by equation (7.1).

$$P_{save}(i) = t_i \times (C_i + \sum_{j \in fan\text{-}out(i)} \alpha_{ij} \times C_j) \tag{7.1}$$

where t_i is average number of transitions at node i, C_i is the capacitance of node i, C_j is the capacitance of a fan-out node j and α_{ij} is the sensitivity of node j with respect to node i.

Equation (7.1) estimates the amount of power reduction at node i due to transitions at that node. A transition at a node can cause further transitions at its fan-out nodes. This effect is captured by the parameter α_{ij} (sensitivity factor). It accounts for the number of transitions at the fan-out nodes due to a transition at a node. It is theoretically difficult to estimate this factor. In this section, we propose a methodology to estimate this power reduction by simulation approach.

7.1.1 Procedure

Consider a node i in the circuit, where the power reduction due to blocking of glitches has to be estimated. We use the power extraction algorithm approach described in chapter 4 to estimate switching energy of the circuit in the presence of process variation and input variation. In order to estimate sensitivity of a node, we break the circuit at node i. We introduce a *multiplexer* at that node. One input of the multiplexer is connected to node i. At the other input, we induce a glitch transition by giving a pulse signal at the input. When the control signal of the multiplexer is set to 0, the normal mode of the circuit operation is simulated through the power extraction algorithm. The control signal is set to 1 once the steady state values at every node in the circuit after the normal mode of operation. We then give a pulse signal at the other input of the multiplexer. This induces a glitch transition at the other input of the multiplexer. This glitch propagates through the fan-out gates of the node i further in the circuit causing more number of transitions. The power extraction algorithm simulates the circuit in this mode to estimate the power dissipated in the circuit due to the induced glitch transition at the node i. This is better explained through an example below.

Consider the combinational circuit shown in figure 7.1. The node of interest represents the node where we need to estimate the reduction in switching energy if glitches at that node are blocked. This circuit is modified as shown in figure 7.2. We break

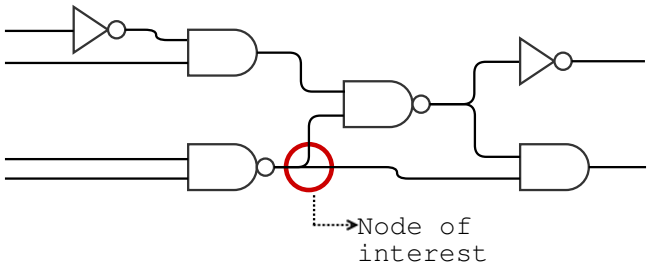


Figure 7.1: An example of a combinational circuit highlighting the node where power reduction due to blocking of glitches has to be estimated.

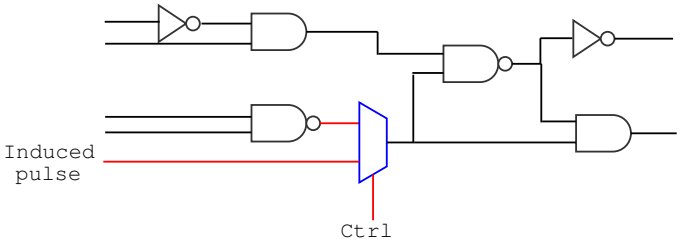


Figure 7.2: An example of a combinational circuit where a multiplexer has been inserted at the node where power reduction due to blocking of glitches has to be estimated.

the circuit at the node of interest and insert a multiplexer at that node as shown. The ctrl signal is set to 0 initially. It is set to 1 once the steady state values are set in the circuit due to applied input vector pair. The power extraction algorithm estimates the overall switching energy of the circuit. The ctrl signal is then made to 1 and a pulse signal is introduced at the other input of the multiplexer. The pulse propagates further in the circuit through the fan-out gates of the node of interest inducing more transitions. The power estimation algorithm now computes the switching energy in the circuit due to these transitions. This gives the estimate of power reduced in the circuit by a single glitch at that node. This estimate is multiplied with expected number of transitions at the node of interest to estimate the total power reduction in the circuit by blocking of glitches at that node. The expected number of transitions can be obtained by both the approaches, ambiguity propagation approach and probability propagation approach.

7.1.2 Results

An `ISCAS85` benchmark circuit `c880` is simulated for 100 input vectors by considering one standard `SDF` file generated by *Synopsys Design Compiler*. The circuit is simulated using the procedure described above considering each of the intermediary nets in the circuit as the node of interest.

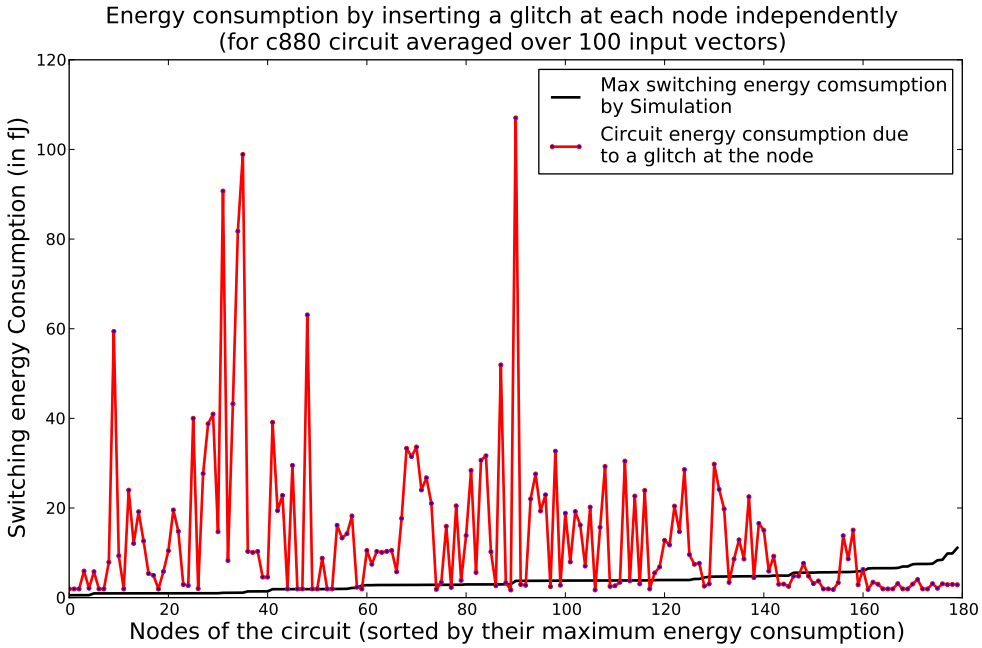


Figure 7.3: Average switching energy consumed by C880 circuit due to a glitch at the intermediary nodes. Maximum switching energy consumption at all the intermediary nets in the circuit by simulation using ModelSim and by ambiguity propagation approach are also shown. The circuit is simulated for 100 input vectors using one standard SDF file.

Figure 7.3 shows the average switching energy consumed by C880 circuit due to an induced glitch at all the intermediary nodes. The maximum power consumption at all those nets in the circuit by simulation of the circuit using ModelSim and the maximum power consumption at the nets estimated by ambiguity propagation approach are also indicated. From this figure, we observe that the nets with moderate power consumption values have more significant values for power consumption in the circuit due to an induced glitch at those nodes. This indicates that blocking of glitches at nets with a high value of expected number of transitions need not necessarily cause the highest reduction in power consumption of the circuit if glitches there are blocked. This is because those glitches might have been originated at an intermediary node and propagated through the circuit. Hence, blocking of glitches at the source node will cause more power reduction

in many cases. We also observe that the value of power consumption of the circuit due to an induced glitch at a node is higher than the other two values (power consumption at the net obtained by simulation and ambiguity interval propagation) in many cases. This is primarily because the value of power consumption due to induced glitch shows the power consumption in the entire circuit due to a glitch at the node. The other two values are the power consumption estimates for the specific net and not the whole circuit.

7.2 Glitch reduction using a latch

In the above section, we presented an approach to estimate the power consumption of a circuit due to a glitch at one of its intermediary nets. We validate this approach by introducing a latch at the net and estimating power consumption of the circuit. The latch is enabled when the steady state values are set at all the nets in the circuit due to applied input vector pair. Thus, the glitches that occur at the input of the latch are blocked from further propagation. Only the steady state transition at that net will be propagated further in the circuit once the latch is enabled. This is explained better with the help of an example given below.

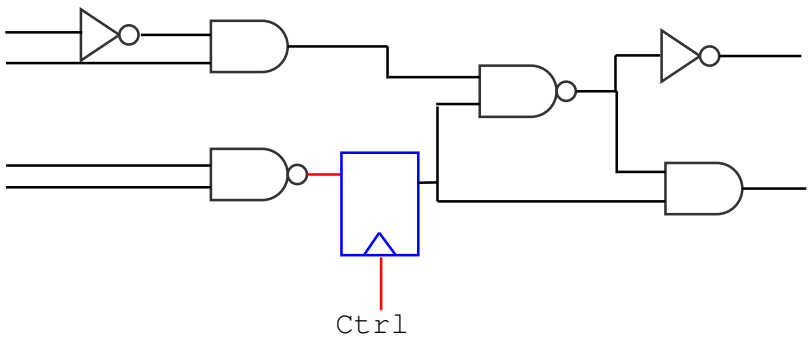


Figure 7.4: An example of a combinational circuit where a latch has been inserted at the node where power reduction due to blocking of glitches has to be estimated.

Consider the combinational circuit shown in figure 7.1. A latch is inserted at the node of interest as shown in figure 7.4. The latch is enabled through the `ctrl` signal, once the steady state values at all the nets in the circuit have been set. Thus, the glitches at the node connected to the input of the latch are blocked from further propagation in the circuit. This circuit is simulated using the power extraction algorithm to estimate

power consumption of the circuit when the glitches at an intermediary net are blocked.

7.2.1 Results

An `ISCAS85` benchmark circuit `c880` is simulated for 100 input vectors by considering one standard `SDF` file generated by *Synopsys Design Compiler*. The circuit is simulated using the procedure described above considering each of the intermediary nets in the circuit as the node where glitches have to be blocked.

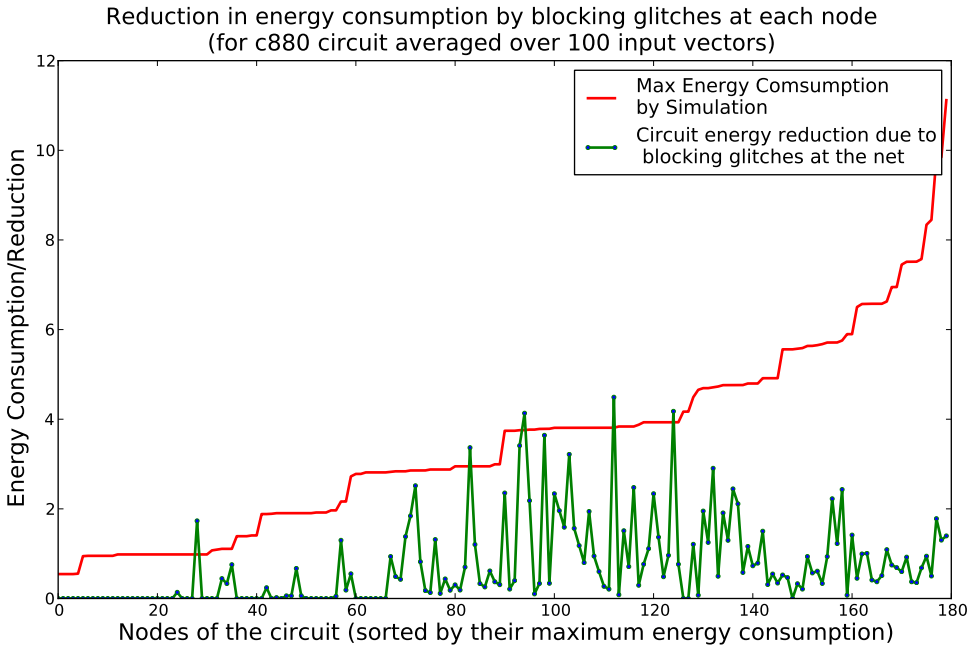


Figure 7.5: Average switching energy consumed by C880 circuit when glitches at an intermediary net is blocked by a latch. Maximum switching energy consumption at each net (by *ModelSim* Simulation) is also shown. The circuit is simulated for 100 input vectors using one standard `SDF` file.

In the Figure 7.8, we can observe that the nets where the glitches must be blocked are not necessarily the high-glitch nodes. Most of them have moderate glitching activity which is then propagating through the subsequent levels of gates leading to high glitches at the nets of later levels.

The power reduced in the circuit by blocking the glitches at a particular node should be highly correlated with the extra power consumed by circuit when a glitch is introduced at that node (as explained in the previous section). Thus, we now compare the values obtained by the two approaches for c880 circuit.

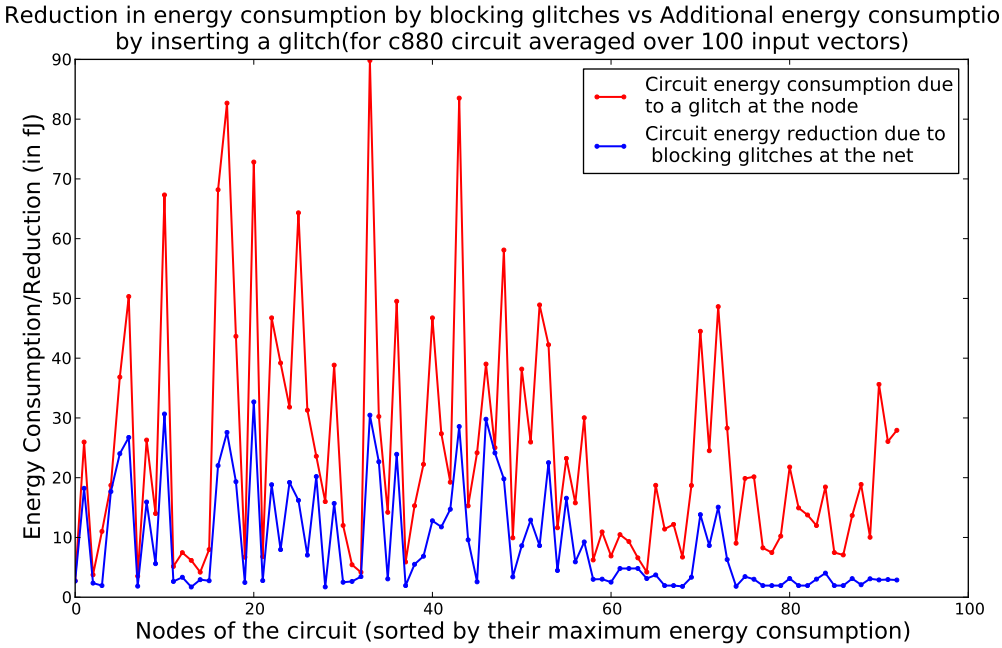


Figure 7.6: Average switching energy reduced by C880 circuit when glitches at an intermediary net is blocked by a latch. Average switching energy consumed by C880 circuit due to a glitch at the intermediary net as estimated by our approach is also shown. The circuit is simulated for 100 input vectors using one standard SDF file.

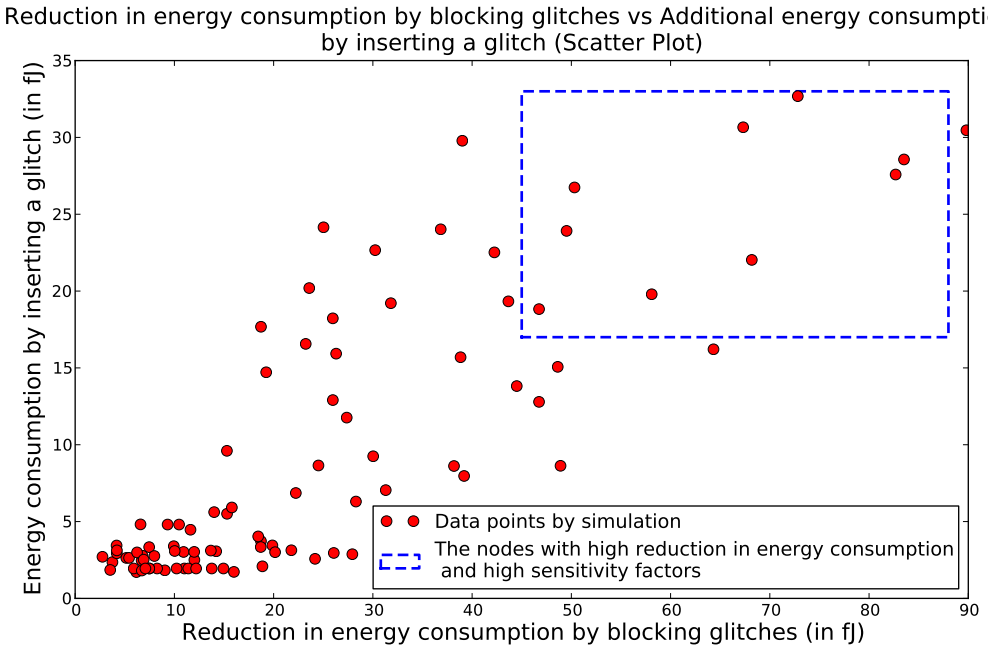


Figure 7.7: Scatter plot for average switching energy reduced by C880 circuit when glitches at an intermediary net is blocked by a latch versus average power consumed by C880 circuit due to a glitch at the intermediary net as estimated by our approach. The box shows the nodes that have high values as estimated by both the approaches.

Figures 7.6 and 7.7 compare the average power consumed by C880 circuit when glitches at an intermediary net is blocked by a latch and the average power estimates as given by our approach in the previous section. These values are plotted for all the nets with high values of power reduction on blocking the glitches. We observe a high correlation between the two values as seen in the figures.

7.3 Applications

In the previous two sections, we presented a methodology to identify the set of nets in the circuit which result in maximum power reduction if glitches at those nets are blocked. The results of this methodology can be used in several applications which aim at power reduction. Some of these applications are explained in subsections below.

7.3.1 Retiming for low power

As established in previous sections, reducing glitches in a circuit is essential to reduction of power dissipated in the circuit. Glitches arise due to unbalanced path delays in a circuit. So, one of the ways of reducing glitches is by trying to balance all the paths in the circuit using delay elements and buffers. This will minimize the effect of mismatch in signal arrival times in the circuit. However, this is difficult to achieve early in the design phase as layout process introduces some delay mismatch in the circuit. Another way of reducing glitches in a circuit is by retiming. Traditionally, retiming algorithms are aimed at optimizing timing constraints in a circuit. In this subsection, we elaborate on retiming and its application for low power.

Retiming

Retiming is the process of optimally distributing registers throughout a circuit to minimize clock period. It alters the clock period by insertion and deletion of registers in a circuit, without altering its intended functionality.

Consider the example circuit shown in figure 7.8 (a). In this circuit, there are two registers, shown in blue and green in the figure. Figures 7.8 (b) & (c) show the circuit

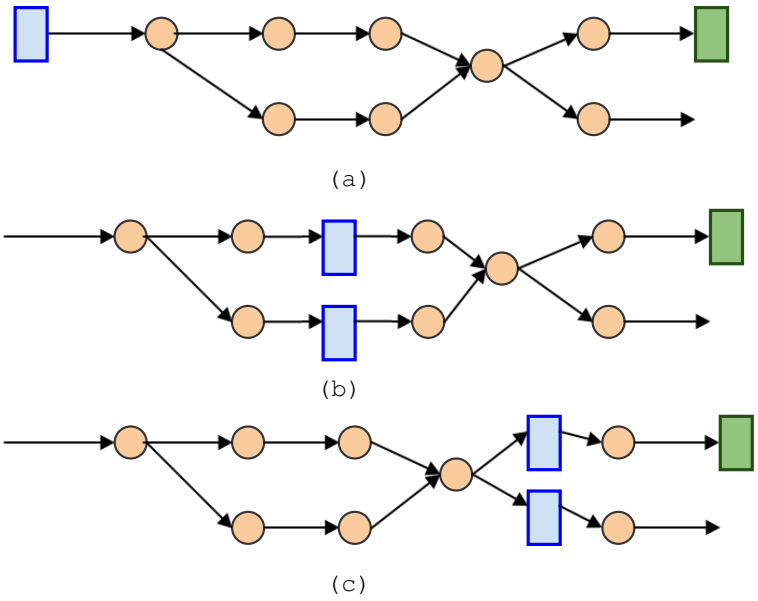


Figure 7.8: Example of retiming on a sample circuit. (a): Sample circuit (b)and (c): Retimed circuit

after performing retiming algorithm on the circuit. We see that the blue colored register has been moved to a different location in the circuit without altering the functionality of the circuit. This process is called retiming. Traditionally, retiming is looked from the perspective of timing of the circuit. Registers are optimally placed in the circuit such that the clock period is minimized. In the next subsection, we show how retiming can also be used for low power.

Retiming for low power

Registers placed at node do not allow glitches at their inputs propagate further in the circuit. So, if registers can be placed at alternate locations without affecting the intended functionality and timing constraints of the circuit, they can be placed at those nodes which cause maximum power reduction due to blocking of glitches. This is explained better by an example.

Consider the circuit shown in figure 7.9. In figures 7.9 (a) & (b), the register has been placed at two different locations in both the cases by applying the retiming algorithm. Let us assume that in both the cases, the timing constraints and functionality of the circuit are satisfied. If it is known that the node connecting combinational logic 1 and 2 causes maximum power reduction due to blocking of glitches in the circuit, plac-

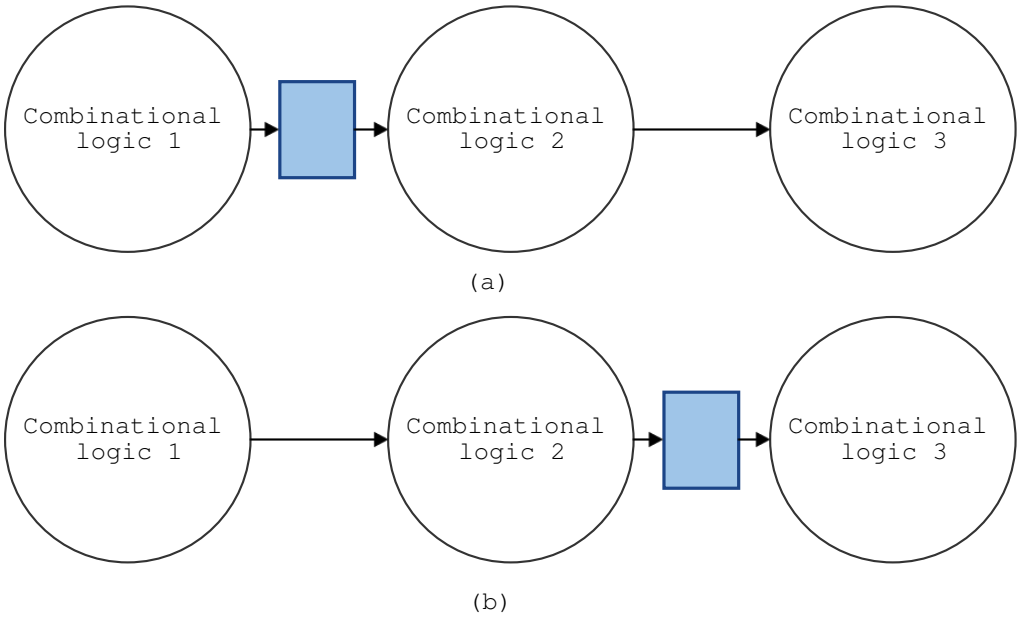


Figure 7.9: Placement of registers at two different locations by retiming algorithm on a sample circuit

ing a register at that node (as shown in figure 7.9 (a)) would significantly reduce the overall power consumption of the circuit. This is because the glitches at that node will not be permitted to propagate further causing additional transitions in the combinational logic block 2. The work done in this thesis helps us identify those nets in the circuit which cause a significant power reduction if glitches at those nets are blocked. These nets can be retimed for low power if the other constraints of the circuit are met.

7.3.2 Insertion of latches with delayed clocks

In the previous subsection, we saw how retiming can be used for low power applications. We moved the locations of existing registers in the circuit such that timing and functionality constraints are met and switching power consumption is minimized. In this subsection, we see another application of power reduction by blocking of glitches using latches with delayed clocks. Latches block glitches at a node if they are disabled by a clock signal for certain time interval till steady state values are set at the node.

Consider the circuit shown in figure 7.4. If the latch is disabled by the clock signal (shown in this figure as ctrl) for an initial time interval, it will block all the glitch transitions at its input till the latch is enabled. In order to meet the timing constraints

of the circuit, the latch needs to be enabled in such a way that the outputs of the combinational block reach their steady state values before the next clock edge occurs. Hence, the insertion of a latch at a net might reduce propagation of some glitches but need not necessarily block all the glitches from propagating further in the circuit. However, it can reduce the overall power consumption of the circuit by blocking glitches at the node for a certain time interval. This comes at the expense of increased area due to additional latches introduced in the design. There is a small additional amount of power consumption due to latches. However, the decrease in total switching energy of the circuit might more than compensate these overheads in certain cases when latches are introduced at those nets which cause maximum total power reduction in the circuit.

In order to implement this method, the first step might be to identify the potential nodes which result in maximal power reduction if latches are placed at those nets. For the same, the heuristic proposed in the first section of this chapter can be used to provide the potential nets in the circuit.

CHAPTER 8

CONCLUSIONS

8.1 Conclusions

- An AND gate was simulated at transistor level. The power dissipated by load capacitance as obtained by spice simulation was nearly equal to the theoretical value of $0.5 \times \mathrm{C}\,\mathrm{V}^2$. We also observed that power dissipated in the internal node capacitances of the gate was negligible in comparison to short circuit power.

- Monte Carlo analysis of circuits was done to identify the impact of process variation on switching energy consumption. We observed that mean and standard deviation of total switching energy of the circuits increased proportionately with an increase in standard deviation of process variation.

- We analyzed the individual and combined impact of process variation and variation in input stimuli on switching energy consumption. We then concluded that the impact of variation in input stimuli is more significant than the impact of process variation.

- We estimated the contribution of glitches to total switching energy consumption. We observed that glitches contribute $30 - 40\,\%$ to the total switching energy consumption.

- We implemented Ambiguity Interval propagation approach to estimate bounds on switching energy consumed at individual nets in the circuit. The maximum value of power dissipation as estimated by this approach was either equal to or larger than the maximum power consumption value obtained by simulation of the circuit. However, we also observed that most of the nets in the circuit which were identified to have maximum power consumption by ambiguity interval method also had maximum power consumption by simulation.

- Probability propagation algorithm was implement to estimate the number of transitions at every net in the circuit and thereby estimate switching energy. Expected number of transitions at all the nets in the circuit were nearly equal to the average number of transitions as estimated by simulation under zero propagation delay conditions for all the gates. On applying real propagation delay values for all the gates, we observed that the expected number of transitions from the algorithm and the simulation were the same for nets in the initial levels of the circuit. However, they differed considerably as the number of glitches increased at later levels of the circuit.

- We induced a glitch at an intermediary net in the circuit once the steady state values were set in all the nets and estimated total switching energy of the circuit. We observed that the nets with moderate power consumption values had more significant values for power consumption in the circuit due to an induced glitch at those

nets. Hence, blocking of glitches at source nets will prevent their propagation to later levels of the circuit and hence result in significant power reduction in many cases.

- On validation of the above approach, we identified a high correlation between the nets identified to have maximum total power consumption due to induced glitch and those nets which result in maximum total power saving by blocking of glitches at the nets through a latch.
- The work presented in this thesis helps identify those nets in the design which will result in maximum power saving if glitches at those nets are blocked. This can be used by several algorithms like retiming for low power which aims at reducing glitches in the circuit.

8.2 Future work

- The algorithms implemented for small combinational circuits can be scaled to large combinational blocks to characterize impact of glitches at higher levels of abstraction.
- We have assumed the inputs to be independent in most of our analysis. The glitch activity at inputs and correlation between inputs can be modeled to characterize combinational blocks for glitch activity.
- A retiming algorithm aimed at low power can be implemented using the results of the current work.
- The impact of process variation in the presence of correlation between inter-die as well as intra-die parameters can be identified by modeling at parametric level and implementing a quad-tree structure.
- Impact of partial glitches on switching energy consumption can be modeled with glitch width as parameter.

REFERENCES

1. **Alexander, J. D.** and **V. D. Agrawal** (2009). Algorithms for estimating number of glitches and dynamic power in cmos circuits with delay variations.

2. **Anderson, J. H.** and **F. N. Najm** (2004). Power estimation techniques for fpgas. *IEEE Trans. VLSI Syst.*, **12**, 1015–1027.

3. **Benini, L.**, **R. Hodgson**, and **P. Siegel**, System-level power estimation and optimization. ACM, 1998.

4. **Buyuksahin, K. M.** and **F. N. Najm** (2005). Early power estimation for vlsi circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, **24**, 1076–1088.

5. **Dhanwada, N. R.**, **D. J. Hathaway**, **J. Frenkil**, **W. R. Davis**, and **H. Demircioglu** (2012). Leakage power contributor modeling. *IEEE Design and Test of Computers*, **29**, 71–78.

6. **Dinh, Q.**, **D. Chen**, and **M. D. F. Wong** (). Dynamic power estimation for deep submicron circuits with process variation.

7. **Ghosh, A.**, **S. Devadas**, **K. Keutzer**, and **J. White**, Estimation of average switching activity in combinational and sequential circuits. *In In Proceedings of the 29 th Design Automation Conference*. 1992.

8. **Hao, Z.**, **R. Shen**, **S. X.-D. Tan**, **B. Liu**, **G. Shi**, and **Y. Cai**, Statistical full-chip dynamic power estimation considering spatial correlations. IEEE, 2011.

9. **Harish, B. P.**, **N. Bhat**, and **M. B. Patil**, Process variability-aware statistical hybrid modeling of dynamic power dissipation in 65 nm cmos designs. IEEE Computer Society, 2007.

10. **Mehra, R.** and **J. Rabaey**, Behavioral level power estimation and exploration. *In in Proc. Int. Wkshp. Low Power Design*. 1994.

11. **Meixner, M.** and **T. G. Noll**, Statistical modeling of glitching effects in estimation of dynamic power consumption. *In 2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*. 2014.

12. **Monteiro, J.**, **S. Devadas**, and **A. Ghosh**, Retiming sequential circuits for low power. *In In Proceedings of the Int'l Conference on Computer-Aided Design*. 1993.

13. **Najm, F. N.** (1994). A survey of power estimation techniques in vlsi circuits. *IEEE Transactions on VLSI Systems*, **2**, 446–455.

14. **Najm, F. N.**, Towards a high-level power estimation capability. ACM, 1995.

15. **Qu, G.**, **N. Kawabe**, **K. Usami**, and **M. Potkonjak** (2000). Function-level power estimation methodology for microprocessors.

16. **Raghunathan, A.**, **S. Dey**, and **N. K. Jha**, Glitch analysis and reduction in register transfer level power optimization. *In in Proc. Design Automation Conf*. IEEE Press, 1996.

17. **Tsui, C.-Y.**, **M. Pedram**, **A. M. Despain**, and **C. ying Tsui Massoud Pedram**, Efficient estimation of dynamic power consumption under a real delay model. *In IEEE Internations Conference on Computer-Aided Design*. 1993.