Copyright

by

Anand Ramalingam

2007

The Dissertation Committee for Anand Ramalingam
certifies that this is the approved version of the following dissertation:

Analysis Techniques for Nanometer Digital Integrated Circuits

Committee:

David Z. Pan, Supervisor

Anthony P. Ambler

David P. Morton

Sani R. Nassif

Michael Orshansky

Analysis Techniques for Nanometer Digital Integrated Circuits

by

Anand Ramalingam, B.E.; M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2007

To Amma and Appa

Acknowledgments

I would first like to thank my Advisor Dr. David Z. Pan for his advice throughout my doctoral studies. His unconditional support and unflappable temperament reassured me throughout my doctoral work. I would like to thank my committee members (in alphabetical order): Prof. Anthony P. Ambler, Prof. David P. Morton, Dr. Sani R. Nassif, and Prof. Michael Orshansky for their precious time and helpful comments.

I was very fortunate to have Dr. Sani R. Nassif as my mentor later in my graduate studies and he is a father figure to me. Early in my doctoral studies, I was lucky enough to work with Dr. Anirudh Devgan. I count myself extremely lucky to collaborate with Dr. Giri Devarayanadurg, Sreekumar V. Kodakara, Dr. Frank Liu, Dr. Gi-Joon Nam, Prof. Michael Orshansky, Ashish Kumar Singh, and Bin Zhang.

I would like to thank Ashish Kumar Singh who had a profound influence on me during his stay at Austin. His insistence on treating every problem in a formal and rigorous manner has hopefully made me a better researcher. I would like to thank Sreekumar V. Kodakara for being such a good friend. We have discussed many ideas and projects for over a decade but unfortunately have only one published paper to show for our efforts. Hopefully this situation will improve in the coming years.

My deep gratitude and appreciation to all my teachers who have made me what I am, particularly: Mr. Ranganathan, Mr. Somasundaram, Mrs. Punitha-

vathy, Mr. Louis Paul Vincent, Mr. Dhandapani, Prof. S. Jayaraman, Prof. Mark A. Horowitz, Prof. Brad Osgood, and Prof. Mendel Rosenblum.

I thank my childhood friends Arun Ananthapadmanabhan, S. Gurpreet Singh, and Pravin Prabhu Venkatesan. Arun introduced me to electronics when we were in our ninth grade and I have been hooked on to it ever since. I would like to thank Ms. Claire Ravi for her hospitality during my initial stay in America. I would like to thank my roommates and friends who were of tremendous help and supported me in every way they could: M. D. Venkatesh, Sriram M. Seshadri, Saranyan V. Rajagopalan, Rangaprabhu Parthasarathy, Manoj Ananthapadmanabhan, Visvanathan Subramanian, Kiran N. Rao, Rajesh Babu Narkedamilli, Venkatesan Santhirahasan, Balaji Gururao, Linson Zacharia, Jagannath G. Krishnan, Rehan Hameed, Sharad Yadav, Jitendra Mohan, Raman Kumar Jha, Karthik Veeravalli, Moinuddin Qureshi, Wajahat Qadeer, Sriram Rajagopal, Srihari Yamanoor, Venkata Anil Kiran Kottapalli, Rangaprasad Sampath, Perumal Karthik Chidambaranathan, Praveen Pasupathy, Vasu Kudravalli, Murari Mani, Sriram Sambamurthy, Sankaranarayanan Gurumurthy, Wei-Shen Wang, Bin Zhang, and Peng Yu. A special thanks to Y. M. Chetana, who died at a very young age. Her death shook me and helped me put my life into perspective. I am thankful for every new day in my life. My cousin Shankar Chidambaram and his family were a comforting presence in Austin for me. I would like to take this opportunity to thank my fellow UTDA group members especially: Minsik Cho, Tao Luo, Haoxing Ren, Ashutosh Chakraborty, Sean Xiaokang Shi and Kun Yuan. Prasad Hariharan helped me proofread the first draft of this thesis and found a whole host of mistakes and thus improving the readability of this dissertation. A big thanks to Chris Lilly, Melanie Gulick, Debi Prather, Melissa Campos, Shirley Watson, and Andrew Kieschnick who were prompt and helpful in administrative issues. They made my life

much easier in graduate school. I am grateful to many people for their kindness and I am sorry if I have missed acknowledging them here.

Finally, I thank my parents and my sister and her family for their unstinting encouragement and support through all these years of study and research. My sister with her quiet determination and enormous will power set standards for me from a young age which this kid brother always aspired to reach and hopefully met. I can say without losing any sense of objectivity that Amma has been my best teacher. Appa's discipline, a fraction of which filtered into me helped me immeasurably in finishing my graduate studies. Looking at my nieces, I can only imagine how much trouble I would have caused my parents. They put up with all my demands and supported me through trying times as only parents can. This thesis is dedicated to them.

Anand Ramalingam

The University of Texas at Austin
December 2007

Analysis Techniques for Nanometer Digital Integrated Circuits

Publication No. ____________________

Anand Ramalingam, Ph.D.
The University of Texas at Austin, 2007

Supervisor: David Z. Pan

As technology has scaled into nanometer regime, manufacturing variations have emerged as a major limiter of performance (timing) in VLSI circuits. Issues related to timing are addressed in the first part of the dissertation. Statistical Static Timing Analysis (SSTA) has been proposed to perform full-chip analysis of timing under uncertainty such as manufacturing variations. In this dissertation, we propose an efficient sparse-matrix framework for a path-based SSTA. In addition to an efficient framework for doing timing analysis, to improve the accuracy of the timing analysis one needs to address the accuracy of: waveform modeling, and gate delay modeling. We propose a technique based on Singular Value Decomposition (SVD) that accurately models the waveform in a timing analyzer. To improve the gate delay

modeling, we propose a closed form expression based on the centroid of power dissipation. This new metric is inspired by our key observation that the Sakurai-Newton (SN) delay metric can be viewed as the centroid of current. In addition to accurately analyzing the timing of a chip, improving timing is another major concern. One way to improve timing is to scale down the threshold voltage (V_{th}). But scaling down increases the subthreshold leakage current exponentially. Sleep transistors have been proposed to reduce leakage current while maintaining performance. We propose a path-based algorithm to size the sleep transistor to reduce leakage while maintaining the required performance.

In the second part of dissertation we address power grid and thermal issues that arise due to the scaling of integrated circuits. In the case of power grid simulation, we propose fast and efficient techniques to analyze the power grid with accurate modeling of the transistor network. The transistor is modeled as a switch in series with an RC and the switch itself is modeled behaviorally. This model allows more accurate prediction of voltage drop compared to the current source model. In the case of thermal simulation, we address the issue of ignoring the nonlinearity of thermal conductivity in silicon. We found that ignoring the nonlinearity of thermal conductivity may lead to a temperature profile that is off by 10° C.

Contents

Acknowledgments **v**

Abstract **viii**

List of Tables **xiv**

List of Figures **xvi**

Chapter 1 Introduction **1**

1.1 Overview of This Dissertation 4

1.2 Key Contributions 7

Chapter 2 An Accurate Sparse Matrix Based Framework for Statistical Static Timing Analysis **9**

2.1 A case for path based SSTA 13

2.2 Parameterized Gate Delay Modeling 15

2.3 Sparse Matrix based SSTA without Slope Propagation 18

2.3.1 Sparse-Matrix Based Static Timing Analysis (STA) 18

2.3.2 Sparse Matrix based Statistical Static Timing Analysis (SSTA) 20

2.3.3 Example 22

2.4 Sparse Matrix based SSTA with Slope Propagation 24

2.5 Experimental Results 28

2.5.1 Speedup Techniques 29

2.6 Comparative Studies 33

2.6.1 Comparison with Path based Monte-Carlo method 33

2.6.2 Implementation Details 35

2.6.3 Comparison with Block-based method 36

2.7 Summary 36

Chapter 3 Waveform Modeling Using Singular Value Decomposition 42

3.1 Data Based Model for a Gate 45

3.2 Analyzing Waveforms using SVD 46

3.3 Example of Moments Calculation 52

3.4 Example of SVD based Timing Model 53

3.5 Data Based Model for a Library 56

3.6 Waveform Propagation Across Gates in STA 58

3.7 Path Delay Evaluation in STA 60

3.8 Summary 61

Chapter 4 Robust Analytical Gate Delay Modeling 62

4.1 Sakurai-Newton Delay Approximation 64

4.2 Centroid of Current Based Delay 66

4.3 Centroid of Power Based Delay 71

4.4 Experimental Results 73

4.5 Summary 77

Chapter 5 Sleep Transistor Sizing **79**
5.1 Sizing the sleep transistor 82
5.2 Temporal Current Estimation 84
5.3 Timing Criticality Based Sizing 86
5.4 Results 89
5.5 Summary 91

Chapter 6 Power Grid Analysis using Behavioral Modeling of Transistors **92**
6.1 Power Grid Preliminaries 95
6.2 Transistor Network Modeling 97
6.3 Behavioral Switch Modeling 100
  6.3.1 Ideal switch 100
  6.3.2 ADC for an approximate switch model 101
  6.3.3 Current source based ADC for switch 102
  6.3.4 Conductance matrix is a $\mathcal{M}$-matrix 104
6.4 Speedup Techniques 106
  6.4.1 Local charge redistribution phase 107
  6.4.2 Global recovery phase 108
6.5 Overall Algorithm 109
6.6 Experimental Results 109
6.7 Summary 110

Chapter 7 Thermal Analysis Considering Nonlinear Thermal Conductivity **113**
7.1 Thermal Modeling and Temperature Simulation 115

7.2 A Fast Algorithm to solve $\mathbf{A}(\mathbf{x})\mathbf{x} = \mathbf{b}$ 117

7.2.1 Evaluating the reduced order Jacobian 119

7.3 Experimental Results 123

7.4 Summary 124

Chapter 8 Conclusion **125**

8.1 Future Work 127

Bibliography **128**

Vita **145**

List of Tables

2.1 Path-gate statistics of ISCAS89 benchmarks and runtime for $10,000$ simulations. 39

2.2 Runtime after logic depth based pruning for the three biggest benchmarks in ISCAS89. All paths whose logic depth was less than 90% of the maximum logic depth were pruned. The number of simulations were set to $10,000$. 40

2.3 Runtime comparison for the proposed matrix method versus repeated path tracing method. The number of simulations were set to $10,000$. 41

4.1 The correlation of HSPICE delay values with the delay metrics across different technologies and gates. The HSPICE delay of a gate is measured for its worst case input combination. 73

4.2 The percentage error between HSPICE delay values and the delay metrics across various technologies and gates. A *line* was fitted to the data points predicted by the delay metric. In this table the average $\min, \max$ estimation error percentage is shown. 74

5.1 Timing windows for 1-bit CLA (time unit is ps) 86

5.2 A worst case path in 1-bit CLA 88

5.3 Comparison of W_{sleep} obtained using module and cluster based design for 5% performance degradation. The unit is $\lambda = 0.1\mu m$. 90
5.4 Comparison of W_{sleep} obtained using *proposed* methods for 5% performance degradation. The unit is $\lambda = 0.1\mu m$. 91

6.1 Runtime over 10 cycles for random circuits. The drop predicted by both the proposed and the current source model are pessimistic. (Cycle-time = $750ps$) 111

List of Figures

1.1 As technology scales rapidly down to 45 nm the variation in process parameters increases correspondingly. The variation in y-axis is illustrated for one of the process parameters, threshold voltage (V_{th}). Please note that the x-axis must be read right-to-left as the technology scales down. 2

2.1 The number of paths versus the number of gates in ISCAS'89 benchmarks. By linear regression we get the following relationship: paths $\approx 0.04 \times \text{gates}^{1.8}$. 13

2.2 The number of paths versus the number of gates for one family of 10 industrial benchmarks. By linear regression we get the following relationship: paths $\approx 0.12 \times \text{gates}^{1.42}$. 14

2.3 The number of paths versus the number of gates for another family of 9 industrial benchmarks. By linear regression we get the following relationship: paths $\approx 0.43 \times \text{gates}^{1.17}$. 15

2.4 Scatterplot of inverter delay and the values predicted by the linear and our higher order model for various $(L, V_{\text{th}}, C_{\text{L}}, S_{\text{in}})$ tuples. The variation in delay is due to variations in process parameters (L, V_{th}) as well as variations in operating conditions $(C_{\text{L}}, S_{\text{in}})$. 16

2.5 Example circuit for illustrating the matrix formulation. The input pins are distinguished by the labels a and b. 18

2.6 Example circuit for illustrating how Monte-Carlo simulation is done in our framework. 22

2.7 Linear relationship between input slope and delay. 25

2.8 A simple circuit to illustrate SSTA with slope propagation. Here s_0 denotes the slope at the primary input. The output slope at gate g_1 in path 1 is denoted as s_{11} and in path 2 is denoted as s_{21}. 26

2.9 Histogram was obtained after doing Monte-Carlo simulation for $10,000$ runs. Note that the logic depth of paths with maximum delay in Figure 2.9(b) is always either 59 or 60. 30

2.10 Histogram of logic depth of paths in s38417. Note that there is a long tail and there are paths with logic depth of 47. Thus our pruning strategy of eliminating paths with logic depth less than 0.9×47 leaves us with just 280 critical paths. 32

2.11 Delay pdf of s38417 obtained after pruning paths plotted against the one obtained without pruning paths. Note that both the pdf's are virtually indistinguishable. 33

2.12 Delay pdf of s27 obtained using block-based method [1] (denoted as *blk*), path based method with *linear* delay models (denoted as *pl*) and path based method with *quadratic* models (denoted as *pq*). Results from path based method with *linear* models can be thought of how much error is introduced by an analytical max() and using worst case slope at the input of a gate. Results from path based method with *quadratic* models can be thought of how much error is introduced when we use linear delay models. Please note that the linear models are not adequate enough to model the tails of the distribution. . . . 37

3.1 A waveform is discretized and time points are recorded when a signal crosses a certain voltage threshold. Here time points are recorded when the waveform crosses the threshold of $\frac{1}{2}$ and 1. 46

3.2 Relative error in Frobenius norm when approximating using the first k moments. Note that using the first 2 moments the approximation is accurate within 1 % in Frobenius norm. The data is shown for an inverter and the number of time points is $n = 14$. 50

3.3 Plot of the first 4 right singular vectors (rsv) obtained on applying SVD to $\mathbf{T}$ in Eq. (3.1). The number of time points is $n = 14$. 51

3.4 Three stage inverter chain with a single capacitor modeling the load. 55

3.5 Gate characterization setup 55

3.6 Waveform comparison at the output of an inverter. We use a 2 moment approximation in addition to arrival time and it is clear from the figure that the waveforms are indistinguishable. 59

3.7 Test case for evaluating path delay. The units for resistance is Ohms (Ω) and the capacitors is femtoFarads(fF). Both the capacitors in the π-model have the same value. 60
3.8 Waveform at the output of the NAND gate in Figure 3.7. The proposed SVD method and Spice output have a very close match. . . . 61

4.1 The RC model of an inverter. Note that R is a nonlinear resistor modeling transistor and C_L is the load capacitance seen by the inverter. 66
4.2 Inverter waveforms when the output is discharging. The input v_{GS} is a step input. The output v_{DS} decreases linearly in the saturation region (till t_{sat}) and decays exponentially in the linear region (after t_{sat}). 68
4.3 RC model with discharging current as a controlled current source. . 69
4.4 HSPICE delay and the values predicted by the delay metrics for INV in 65nm technology under *nominal* supply voltages. The solid line is the HSPICE delay values and the dotted lines are the delays predicted by the various metrics. The V_{DD} was varied with load capacitance C_L = 20fF and threshold voltage $V_{T0} = 0.22V$. Note that *all* the delay metrics track under *nominal* supply voltages. 74
4.5 HSPICE delay and the values predicted by the delay metrics for INV in 65nm technology. The solid line is the HSPICE delay values and the dotted lines are the delays predicted by the various metrics. The V_{DD} was varied with load capacitance C_L = 20fF and threshold voltage $V_{T0} = 0.22V$. Note that only CPM can track the delay in the lower voltages while TN can track to quite an extent, the other two metrics SN and CP cannot track it. 75

4.6 Scatter plot of different delay metrics with the HSPICE delay for INV in 65nm technology. Since we have **not** multiplied by the constant of proportionality, no units are provided for the y-axis. 78

5.1 Power gating 80
5.2 Sleep transistor as a resistor 80
5.3 1-bit CLA 86
5.4 I_{exp} bounding the *falling* and *rising* timing windows of each gate (Table 5.1) in a 1-bit CLA. Refer to ESTIMATE-SWITCHING-CURRENT line 5 87
5.5 The estimated current discharge $I(t)$ of a 1-bit CLA got by summing up all the currents in Fig. 5.4. Refer to ESTIMATE-SWITCHING-CURRENT line 6. Also shown are the local maximum currents seen by the gates X_1 and X_2. Refer to Eq. (5.5). Note that by using local maximum instead of global maximum we reduce the size of the sleep transistor 88

6.1 The voltage drop at a node from SPICE simulation and the current source based approach. There is a difference of $0.025V$ in the voltage drop predicted. 94
6.2 Transistor is modeled a simple switch in series with a RC circuit. Note that if a transistor gets switched on to the grid node, some of the charge will come from transistors which are already *on* which is not captured in previous models. 97
6.3 Network having an open switch. The topology due to an open switch is different when compared to a closed switch in Figure 6.4. This leads to the conductance matrix $\mathbf{G}_{\text{open}}$. 98

6.4 Networks having a closed switch. The topology due to a closed switch is different when compared to an open switch in Figure 6.3. This leads to the conductance matrix $\mathbf{G}_{\text{close}}$. 99

6.5 Ideal switch. 100

6.6 Modeling a switch with a voltage source. By varying the value of the voltage source, we can simulate the *on* or *off* behavior of the switch. Since we are changing only the value of the voltage source, the conductance matrix remains the same irrespective of the state of the switch. 101

6.7 Associated Discrete Circuit (ADC) of an approximate switch model. The superscript $(n + 1)$ refers to the simulation step. 103

6.8 The figure is *not* drawn to scale. This is a stylistic depiction of the 2 phases that occur in power grid simulation. There are two phases, the first phase (dotted) is local charge redistribution and the second phase (dashed) is global recovery. 107

7.1 Electrical interpretation of the 3-d heat equation. Note that the thermal conductivities k_x, k_y and k_z are functions of temperature $T_{i,j,k}$ 116

7.2 Newton-Raphson iteration. Note that the tangent is evaluated during every iteration. This is equivalent to finding the inverse of a Jacobian matrix ($\mathbf{J} \in \mathbb{R}^{m \times m}$) for every iteration when solving the thermal circuit. 118

7.3 Constant Jacobian with speedup. Please observe that $x_{roj}^{(2)}$ is got by fitting a line through the previous two iterations. Also note that the $x_{roj}^{(2)}$ is closer to the root than $x_{cj}^{(2)}$ thus accelerating the constant Jacobian. 121

7.4 The difference in temperature profile in a silicon layer between having a constant thermal conductivity for silicon and incorporating nonlinear thermal conductivity for silicon. The chip dimension is 8 mm × 8 mm and it dissipates 100 W uniformly. The constant thermal conductivity evaluated at 27° C and used in the thermal simulation underestimates the peak temperature by 12%. This is ≈ 12° C in absolute value. 123

Chapter 1

Introduction

Moore's Law is the name given to Gordon Moore's prediction in 1965 that the number of transistors in a digital integrated circuit would roughly double every year [2]. The law was later revised to the number of transistors doubling every two years and then every 18 months. The semiconductor industry has been keeping up with Moore's Law for more than forty years. This pace is expected to be kept up for the next decade as well [3].

A major factor helping the semiconductor industry keep up with Moore's law is the ever decreasing minimum dimension of a transistor, also referred to as the feature size. The feature size is usually called *technology node* in the literature. The technology used in 2001 was 180 nm; since then it has scaled down to the present 65 nm (2007) [3].

This scaling of technology has been a simple way to improve the performance of integrated circuits. What we mean by performance here is the circuit timing, or in other words, the maximum delay of a signal traveling from input to output.

However, the scaling of technology into nanometer (sub-100 nm) regime has introduced two profound challenges. First, manufacturing variations pose the biggest challenge to scaling by being a major limiter of performance [4]. This is illustrated in Figure 1.1 which shows that the variation in a process parameter namely, threshold voltage (V_{th}) increases as the technology scales down to 45 nm [3]. Manufacturing variations exhibit themselves as systematic, spatial, and random changes in the parameters of transistors. Due to the random nature of such variations, the timing of a circuit is described by a probability density function (pdf). The challenge is to obtain an accurate pdf as quickly as possible with few assumptions on the type of probability distribution exhibited by the manufacturing variations.

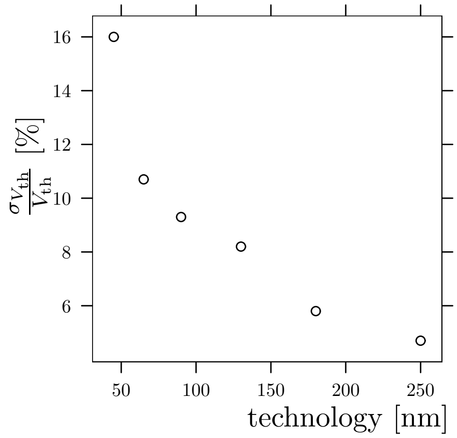


Figure 1.1: As technology scales rapidly down to 45 nm the variation in process parameters increases correspondingly. The variation in y-axis is illustrated for one of the process parameters, threshold voltage (V_{th}). Please note that the x-axis must be read right-to-left as the technology scales down.

Second, as technology has scaled aggressively, the rapid increase in transistor density has led to a corresponding increase in the chip's power density. Power consists of two components: leakage and dynamic power. The leakage power is directly proportional to the number of transistors on a chip. Limiting leakage power while maintaining the timing of a circuit is a key challenge. The dynamic power is directly proportional to the number of transistors switching at a given time. As transistors switch they consume power from the power supply grid, leading to fluctuation in supply voltage from its constant value. This voltage fluctuation, also called a voltage transient, decreases a circuit's performance, since lower the voltage, worse is the performance. An important challenge is knowing how to estimate the voltage transients in the power supply grid efficiently. This is a difficult problem since analyzing power grid also involves analyzing the transistor network which is coupled to it.

An increase in power consumption results in a rise in temperature. This in turn reduces the mobility of electrons, thereby worsening the performance of a circuit. Hence an accurate analysis of temperature of a chip is needed. But the thermal conductivity of silicon[1] is nonlinear. Hence there arises a need to solve a set of nonlinear equations simultaneously to obtain the temperature profile of circuit. An important challenge in thermal analysis is knowing how to efficiently solve this set of simultaneous nonlinear equations.

The above challenges have made analysis of timing and of factors affecting timing (such as temperature and power) among the most important problems for digital integrated circuits. In this dissertation, we address the challenges outlined above.

---

[1] As of 2007, a majority of digital integrated circuits are built using silicon.

1.1 Overview of This Dissertation

This dissertation studies six related research topics on timing, power and thermal analysis in digital integrated circuits. The first part of the dissertation consists of four topics and details techniques to analyze and improve timing of a circuit: (1) A framework for performing timing analysis under uncertainty, (2) A technique to model the shape of the waveforms accurately, (3) A closed form expression which computes the delay of a gate accurately, and (4) A technique to size sleep transistors in a way that reduces leakage power while maintaining the timing constraints of a circuit. The second part of the dissertation consists of two topics which detail techniques to analyze power and thermal profiles which directly affect the timing of a circuit: (1) A fast method to analyze the power grid of a circuit using a simple yet adequate model of a transistor, and (2) An algorithm to analyze a set of simultaneous nonlinear equations which arise when analyzing the thermal profile of a circuit.

The rest of this dissertation is organized as follows. In Chapter 2, we propose a sparse matrix framework for performing timing analysis under uncertainty in a fast and efficient manner. The proposed framework does not impose any restrictions on the probability density functions (pdf) of manufacturing variations. Apart from restricting the input slope to be linear, no limitation is placed on the delay model of gates in the circuit. Since it handles input slope accurately, the proposed framework is both the most general and the most accurate of all the timing analysis methods proposed so far in the literature.

Apart from providing an efficient framework for analyzing and improving the accuracy of timing analysis, one needs to improve the accuracy of: (1) Waveform modeling, and (2) Gate delay modeling.

Since the widely used ramp approximation of a waveform does not provide the required accuracy [5], in Chapter 3 we develop a technique that models the waveforms accurately. The technique is based on Singular Value Decomposition (SVD), and it naturally leads to a more general gate delay model which can be applied in any timing analysis engine with minor modifications. The proposed technique also allows a more flexible trade-off of accuracy versus computational and representational cost.

To improve the accuracy of the gate delay, in Chapter 4, we propose a closed form expression which has a high correlation with the delay observed in Spice, consistent across all major process technologies (90 nm, 65 nm, 45 nm). The closed form delay metric is based on the centroid of dissipated power. This new metric is inspired by our key observation and theoretic proof that the Sakurai-Newton (SN) delay metric [6] is indeed the Elmore delay, which can be viewed as the centroid of dissipated current.

These three topics deal with analyzing the timing of a circuit. Improving the timing of a circuit is another important concern. One way to improve timing is by scaling down the threshold voltage (V_{th}). But scaling down the threshold voltage (V_{th}) increases the subthreshold leakage current exponentially [4]. Leakage current is also referred to as leakage power since the leakage power is equal to leakage current multiplied by the supply voltage V_{DD}, which is a constant. One of the widely used techniques employed to reduce subthreshold power is power gating [7]. Power gating is a circuit technique in which the source nodes of the grounded gates in the functional block are connected to the drain of the NMOS sleep transistor. In the active mode, the sleep transistor is turned on to retain the functionality of the circuit. In the sleep mode, the sleep transistor is turned off, and the source nodes of the gates in the functional block float, thereby cutting off the leakage path. However, sizing the sleep transistor is a major challenge since over sizing results in silicon area being wasted, and under sizing results in the required performance not being achieved. In this dissertation, we propose a method to size the sleep transistor which reduces leakage power while maintaining the required performance. This is discussed in detail in Chapter 5.

In the second part of the dissertation, we present techniques to analyze the power and thermal profile of a circuit which directly affects the timing of a circuit. A technique to reduce leakage power while maintaining the timing requirements of a circuit was developed in Chapter 5. But in addition to the leakage component, the power dissipated by a circuit has a dynamic component. In Chapter 6, we develop an efficient technique to analyze the voltage transient drop in a power supply grid resulting from dynamic power consumption. This is an important problem from a timing perspective because a drop in the voltage supplied to a transistor leads to an increase in the delay thereby degrading the performance of the circuit.

The solution techniques available currently for power grid analysis rely on a model of representing the transistor as a current source. The advantage of such a model is that it decouples the transistor network from the power grid network and thus simplifies the analysis. The disadvantage, however, is that the drain capacitances of the PMOS transistors which are already *on* are not modeled correctly. The drain capacitances of the PMOS transistors which are *on*, act much like decoupling capacitances in the power grid. If the drain capacitance is modeled incorrectly, the voltage drop predicted turns out to be pessimistic. In our proposed model, we model the transistor as a simple switch in series with an RC circuit. The presence of switches leads to a non-constant conductance matrix during simulation. The switch is modeled behaviorally to make the conductance matrix a constant in the presence of switches, thus retaining the efficiency of the simulation.

The previous two chapters discussed techniques related to the power dissipated by a circuit. Power dissipation leads to an increase in temperature, which directly affects the timing of a circuit. This is because increase in temperature lowers the mobility of electrons thereby worsening the performance of the circuit. The solution techniques available currently for analyzing thermal profile assume thermal conductivity to be a constant in order to obtain a linear system of equations which can be solved efficiently. But thermal conductivity in reality is a *nonlinear* function of temperature and for silicon it varies by 22% over the range of $27 - 80^\circ$ C [8]. If the nonlinearity of thermal conductivity is ignored the thermal profile may be off by 10° C. Thus to obtain an accurate thermal profile, it is necessary to consider the nonlinear dependence of thermal conductivity on temperature. In this dissertation, the nonlinear system of equations arising from nonlinear thermal conductivity is solved efficiently using a variant of Newton-Raphson technique. This is discussed in detail in Chapter 7.

1.2 Key Contributions

Specifically, this dissertation develops,

1. In the context of timing analysis:
    - A sparse-matrix based framework for accurate path-based SSTA, which places no restrictions on process parameter distributions. It embeds an accurate polynomial-based delay model with takes into account slope

propagation naturally. It takes advantage of matrix sparsity and high performance linear algebra for an efficient implementation.

- A technique that accurately models the waveforms in a timing analyzer. The technique is based on Singular Value Decomposition (SVD) and it naturally leads to a more general gate delay model which can be applied to any timing analysis engine. This technique also allows a flexible trade-off of accuracy versus computational and representational cost.
- A closed form expression which models the delay of a gate. The closed form delay metric is based on the centroid of power dissipation. The delay metric exhibits a high correlation with the delay observed in Spice consistent across all major process technologies (90 nm, 65 nm, 45 nm).
- A methodology based on timing criticality and temporal currents to size the sleep transistor which reduces leakage power while maintaining the timing of the circuit. The timing criticality information and temporal current estimation are obtained using a static timing analyzer.

2. In the context of power and thermal analysis:
    - In the case of power grid simulation, we model the transistor as a simple switch in series with a RC circuit. The switch is modeled behaviorally to make the conductance matrix a constant in the presence of switches, thus retaining the efficiency of the simulation.
    - In the case of thermal simulation, we model thermal conductivity accurately as a *nonlinear* function of temperature. We efficiently solve the nonlinear system arising out of considering the thermal conductivity to be nonlinear by using a variant of Newton-Raphson.

Chapter 2

An Accurate Sparse Matrix Based Framework for Statistical Static Timing Analysis

As technology has scaled, manufacturing variations have emerged as a major limiter of performance [4]. These variations exhibit themselves as systematic, spatial and random changes in the parameters of active (transistor) and passive (interconnect) components. Furthermore, these variations increase with each new generation of technology. Statistical Static Timing Analysis (SSTA) has been proposed to perform full-chip analysis of timing under such types of uncertainty, and has been the subject of intense research recently [1, 9–25].

Statistical Static Timing Analysis (SSTA) predicts the parametric yield at a given target for a circuit design. It does so by finding the probability density function (pdf) of the circuit delay. SSTA algorithms can be classified into two major groups:

1. *Block-based* [1, 9–13] approaches use a breadth-first traversal of the circuit to compute circuit delay [9]. The delay pdf is propagated from the primary inputs to the primary outputs. The major difficulty in block-based approaches is the introduction of the max operator at each block, and the need to accurately estimate the maximum of two random variables in the same form in which those two variables are defined.

2. *Path-based* [14–18] approaches rely on enumeration of all or a large number of the most critical paths in the circuit [14]. Considering the case where all paths are enumerated, the max operator is deferred to the end of the analysis (i.e. taking the maximum of all the paths) and therefore does not introduce any inaccuracy in the computation. A major problem with path-based approaches is the *perception* that typical circuits have an exponential number of paths, making the computational requirement for such approaches impractical.

While there has been much work on algorithms for SSTA, there has been less work on issues relating to accuracy. Some of the sources of inaccuracy in SSTA are:

- The basic assumptions underlying static timing analysis, such as treating a gate as a node without considering the functionality which gives rise to false paths,
- The delay models used for gates and wires, and
- The model for process variations and their spatial and/or temporal distributions.

The *algorithmic* error introduced by SSTA algorithms can be traced back to the application of the max operator, an approximation to the behavior of true circuits. The max operation is further approximated in block-based SSTA algorithms [20–22]. While a direct assessment of that error is difficult, we propose that eliminating the max operation on parameterized form will aid in reducing the error. The algorithm we propose in this work applies max on a set of real numbers thus incurring no error.

The *model* error has been widely acknowledged and a number of researchers have made important contributions towards reducing the model error. The *parameterized delay form* expressed delays and arrival times as explicit linear functions of the process parameters [1]. The linear delay models was later expanded to quadratic delay models which improve the accuracy of delay estimates [19–22]. A related error, namely the modeling and handling of the slope of signals, has not received as much attention. In fact, current published approaches typically make a worst-case estimate of the slope or propagate the latest arriving slope [1] which can lead to significant error [26]. The polynomial models we propose in this work allow high accuracy by using higher order models, and naturally handle the slope and its propagation.

The *distribution* error, i.e. the error caused by lack of generality in the modeling of the statistical properties of process variations, has been the most difficult to deal with, due to the lack of published realistic manufacturing variability data. Earlier approaches assumed that process variables followed normal distributions [14], but recent work has shown how more general distributions can be handled, and how spatial and systematic correlation can be accommodated [25]. In this work, we make *no assumptions* about the character or distribution of any process parameter.

This work proposes a new approach to parameterized path-based SSTA. The proposed method starts with a preprocessing step of path enumeration and delay computation of all the paths in a parameterized form, which we then efficiently represent using a *sparse matrix*. We model the delay and slope of each component in the circuit using a general parameterized polynomial form which can include the influence of:

- Input waveforms and output loading,
- Manufacturing variations in parameters like threshold voltage and channel length,
- Operating environment variations in parameters like power supply voltage and temperature.

Next, the path delays in this same parameterized form are computed by a natural extension to the gate delay formulation. Given a sample of values from the distribution of manufacturing variations, this computation is shown to be simply a matrix/vector multiply that produces a vector of delays of each path in the circuit. Finally, the maximum circuit delay is obtained by applying the max operator to the path delays. The major attributes of this work are:

1. We show that the number of paths in practice is sub-quadratic in the number of gates, by evaluating the number of paths in the ISCAS89 benchmarks, as well as two different families of industrial circuits.
2. It can handle global, spatial and intra-die variations in one unified framework.
3. It can compute the delay based on an accurate propagation of slope along all paths.
4. It minimizes the impact of the error caused by approximating the max function commonly used in SSTA.

5. It is independent of the underlying distribution of the process parameters, and is not restricted to the usual Gaussian distribution.

2.1 A case for path based SSTA

The upper bound on the number of paths in an arbitrary network is exponential in the number of gates. A key observation in this work, however, is that, for the vast majority of practical circuits, the number of actual paths is far less than this theoretical upper bound, and is quite manageable. The theoretical upper bound is typically achieved by highly structured networks such as multipliers. With the easy availability of large amounts of memory in modern computers, storing and manipulating million of paths is not a problem.

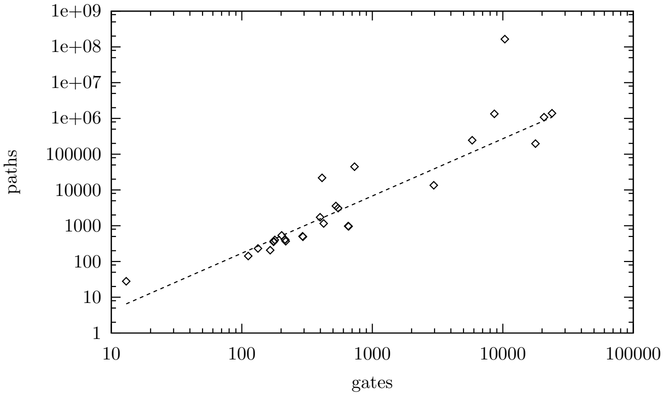


Figure 2.1: The number of paths versus the number of gates in ISCAS'89 benchmarks. By linear regression we get the following relationship: paths $\approx 0.04 \times \text{gates}^{1.8}$.

To test our conjecture, we enumerated *all* the latch to latch, primary input to latch, and latch to primary output paths in the ISCAS sequential circuit benchmarks [27] (see Figure 2.1), and found that the paths $\approx 0.04 \times$ gates$^{1.8}$. This is hardly the kind of explosive growth that might cause one to completely discount

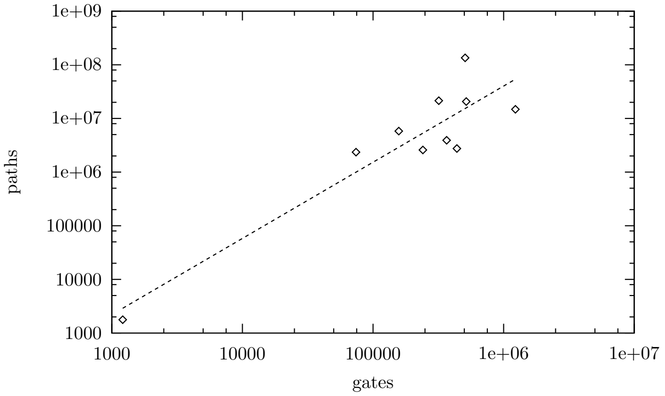


Figure 2.2: The number of paths versus the number of gates for one family of 10 industrial benchmarks. By linear regression we get the following relationship: paths $\approx 0.12 \times \text{gates}^{1.42}$.

a family of algorithms. But since the ISCAS benchmarks are small compared to modern designs, we extended our analysis to two different families of industrial benchmarks, one for large circuits (much larger than the ISCAS benchmarks), and another for moderately sized circuits (comparable to the ISCAS benchmarks).

We enumerated all paths for the circuits in these two families. For the first, larger family, shown in Figure 2.2, we observed that the number of paths $\approx 0.12 \times \text{gates}^{1.42}$. For the second, and smaller family, shown in Figure 2.3, we found paths $\approx 0.43 \times \text{gates}^{1.17}$.

While the above results do not suggest that a purely path-based SSTA algorithm is appropriate for all cases, they do demonstrate that such an algorithm can be practical for a significant number of cases. In practice, one can imagine pairing path-based and block-based algorithms with one being applied when the enumeration of paths results in a manageable number of paths, and the other being applied to circuits in which the number of paths exceeds some suitable threshold.

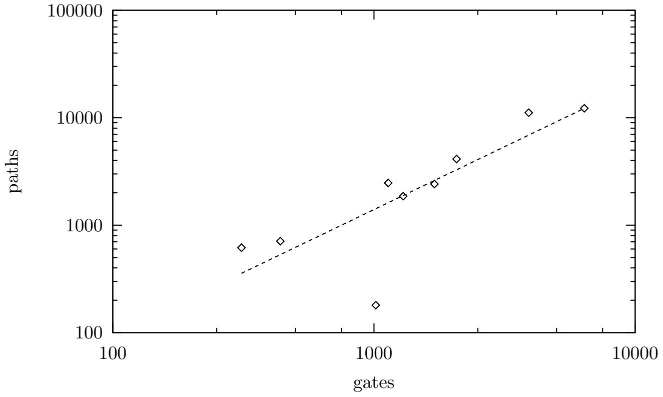


Figure 2.3: The number of paths versus the number of gates for another family of 9 industrial benchmarks. By linear regression we get the following relationship: paths $\approx 0.43 \times \text{gates}^{1.17}$.

2.2 Parameterized Gate Delay Modeling

The advantage of path-based SSTA is that it can naturally handle accurate nonlinear delay models. In this section, we present a parameterized gate delay model which explicitly takes slope propagation into account. In current published approaches, a worst-case estimate of the slope or the latest arriving slope is typically propagated [1] which can lead to significant error [26]. By modeling the input slope in the gate delay equation we avoid this modeling error.

It has been observed that a delay model linear in process variations has a large amount of error; while a quadratic model fits the gate delay quite accurately [19–21]. The need for a *higher order* delay model is illustrated in Figure 2.4, where we model delay as a function of gate length (L), threshold voltage (V_{th}), the output capacitance C_{L} and input slope S_{in}. The samples of L and V_{th} used to create Figure 2.4 were generated uniformly in the range $\mu \pm 3\sigma$ with $3\sigma = 0.2\mu$. The samples of S_{in} were generated in the range of 10 to 100 ps and samples of C_{L} were generated in the range of 1 to 10 fF.

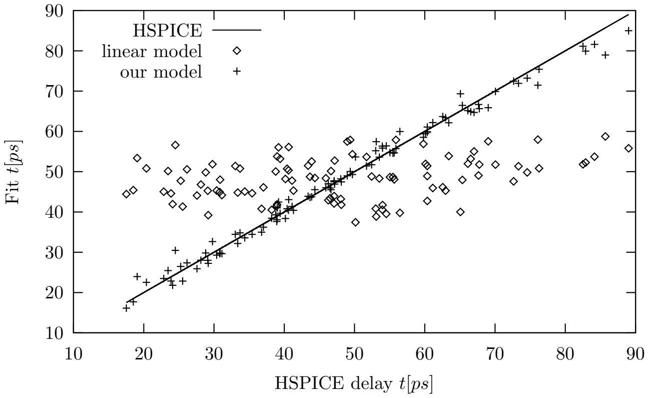


Figure 2.4: Scatterplot of inverter delay and the values predicted by the linear and our higher order model for various $(L, V_{\mathrm{th}}, C_{\mathrm{L}}, S_{\mathrm{in}})$ tuples. The variation in delay is due to variations in process parameters (L, V_{th}) as well as variations in operating conditions $(C_{\mathrm{L}}, S_{\mathrm{in}})$.

The x-axis shows the HSPICE delay for various (L, V_{th}, C_{L}, S_{in}) tuples. The y-axis shows the values predicted by the two delay models. It is clear that our model is a much better predictor of delay than the linear model. In order to generate the cell delay model for every gate in the library, we first simulate each gate by varying the process parameters, load capacitance C_{L} and input slope S_{in} *uniformly* as described above, and then fit to the delay equation given below:

$$\begin{multline}D = a_0^d + a_1^d L + a_2^d L^2 + a_3^d V_{\mathrm{th}} + a_4^d V_{\mathrm{th}}^2 + \\ C_{\mathrm{L}} \left(b_1^d L + b_2^d L^2 + b_3^d V_{\mathrm{th}} + b_4^d V_{\mathrm{th}}^2 \right) + \\ \alpha^d C_{\mathrm{L}} + \beta^d S_{\mathrm{in}} + \gamma^d S_{\mathrm{in}} C_{\mathrm{L}} \quad (2.1)\end{multline}$$

Similarly, the output slope was also fit to the same canonical form as the delay and is given below:

$$\begin{aligned} S_{\text{out}} = a_0^s + a_1^s L + a_2^s L^2 + a_3^s V_{\text{th}} + a_4^s V_{\text{th}}^2 + \\ C_{\text{L}} \left(b_1^s L + b_2^s L^2 + b_3^s V_{\text{th}} + b_4^s V_{\text{th}}^2 \right) + \\ \alpha^s C_{\text{L}} + \beta^s S_{\text{in}} + \gamma^s S_{\text{in}} C_{\text{L}} \end{aligned} \quad (2.2)$$

Note that both the output delay equation Eq. (2.1) and the output slope equation Eq. (2.2) are dependent explicitly on input slope S_{in}. The equations are valid only for a certain range of the parameters involved such as L. The canonical forms presented in Eq. (2.1) and Eq. (2.2) are equivalent to canonical forms presented in literature in the form of the deviations from the nominal values. If we replace L with

$$L = L_{\text{nominal}} + \Delta L$$

and V_{th} with

$$V_{\text{th}} = V_{\text{th,nominal}} + \Delta V_{\text{th}}$$

in Eq. (2.1) and Eq. (2.2) we can obtain the canonical forms presented in literature. Note that the constants a_0^d in Eq. (2.1) and a_0^s in Eq. (2.2) are intercepts obtained from linear regression. They should not be confused with nominal value of delay or output slope as they are usually denoted in literature [1, 28].

It should be noted that our formulation does not restrict the model order in any way, and higher order models are possible with no change to our methodology.

2.3 Sparse Matrix based SSTA without Slope Propagation

In the current section and the next, we present the sparse-matrix based SSTA formulation. First, we calculate the path delays without considering slope propagation and in the next section we take the slope into account. Let the delay of gate j from input a to the gate output be $d_{j_a} \in \mathbb{R}$. Later we will generalize the gate delay as a function of parameters $\mathbf{z}$, $d_{j_a} = f(\mathbf{z})$.

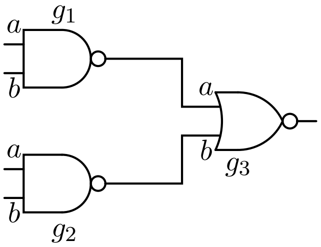


Figure 2.5: Example circuit for illustrating the matrix formulation. The input pins are distinguished by the labels a and b.

2.3.1 Sparse-Matrix Based Static Timing Analysis (STA)

Consider the circuit shown in Figure 2.5. This circuit has 4 paths and 3 gates. The gates have two inputs which we distinguish by labeling them a and b. We define an incidence matrix where each row represents a path, and each column represents a gate input. The columns are sorted by gate topological order. The *path-gate* incidence matrix for the example is given by:

$$\mathbf{A} = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{cccccc} g1_a & g1_b & g2_a & g2_b & g3_a & g3_b \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{2.3}$$

Since each path consists only of a small number of gates, matrix **A** is a very sparse. The delay of the gates can be written as *gate delay* vector:

$$\mathbf{d}_{\text{gate}} = \begin{bmatrix} d_{1_a} & d_{1_b} & d_{2_a} & d_{2_b} & d_{3_a} & d_{3_b} \end{bmatrix}^\top \tag{2.4}$$

where d_{1_b} is the delay from input pin b of gate 1 to its output. The delay of a path is given by the addition of gate delays along that path. Thus multiplying the path-gate incidence matrix with the gate delay vector gives the path delays:

$$\mathbf{d}_{\text{path}} = \mathbf{A}\mathbf{d}_{\text{gate}} \tag{2.5}$$

The overall circuit delay is given by the max of all path delays:

$$d_{\text{circuit}} = \max(\mathbf{d}_{\text{path}}) \tag{2.6}$$

Eq. (2.6) represents path based Static Timing Analysis (STA). We note that STA in this form is essentially a sparse matrix-vector multiplication, and that it requires only a single max operator to find the circuit delay. There are many data structures and algorithms developed for efficient sparse matrix manipulation which we can exploit [29]. We now turn our attention to the Statistical STA (SSTA).

2.3.2 Sparse Matrix based Statistical Static Timing Analysis (SSTA)

In this section, we drop the input specific delay for the sake of convenience. Let the delay of gate j be a function of r parameters $\mathbf{z}_j \in \mathbb{R}^r$. Thus $d_j = f(\mathbf{z})$ is a symbolic function of parameters instead of a real number.

$$d_j = \sum_{k=1}^{r} c_{jk} z_{jk} = \mathbf{c}_j^\top \mathbf{z}_j \tag{2.7}$$

Note that $\mathbf{z}$ need not consist of only linear parameters. For example, a possible second order gate delay model in channel length L and load C_L might be:

$$\mathbf{z}_j = \begin{bmatrix} 1 & L & L^2 & C_\mathrm{L} & C_\mathrm{L} L \end{bmatrix}^\top \tag{2.8}$$

The same formulation can trivially handle a mixed model such as:

$$\mathbf{z}_j = \begin{bmatrix} 1 & \sqrt{L} & L & C_\mathrm{L} & C_\mathrm{L} e^L \end{bmatrix}^\top \tag{2.9}$$

The gate delays of the circuit in Figure 2.5 can be written as,

$$\begin{aligned} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} &= \mathbf{diag}(\mathbf{c}_1^\top, \mathbf{c}_2^\top, \mathbf{c}_3^\top) \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3 \end{bmatrix} \\ \mathbf{d}_\mathrm{gate} &= \mathbf{C}^\top \mathbf{Z} \end{aligned} \tag{2.10}$$

where $\mathbf{diag}()$ denotes the diagonal of a matrix.

The path delays are obtained by multiplying the path-gate incidence matrix in Eq. (2.3) and the gate delay vector in Eq. (2.10)

$$\begin{aligned}\mathbf{d}_{\text{path}} &= \mathbf{A}\mathbf{d}_{\text{gate}} \\ &= \mathbf{A}\mathbf{C}^{\top}\mathbf{Z} \qquad (2.11)\end{aligned}$$

With Eq. (2.11) we have now extended the path delay calculation in Eq. (2.5) to include the dependence of delay on process parameters. Assuming that these process parameters are random variables with some well defined joint probability density function from which we can sample, our goal is to show how we can generalize this result to calculate the distribution of path delays, and by using the traditional max function, the distribution of overall circuit delay.

If the kth random sample is given by $\mathbf{Z}^{(k)}$ then the path delay vector corresponding to the kth random sample is given by

$$\mathbf{d}_{\text{path}}^{(k)} = \mathbf{A}\mathbf{C}^{\top}\mathbf{Z}^{(k)} \qquad (2.12)$$

Now if we take ℓ samples then Eq. (2.12) can be generalized as

$$\begin{bmatrix}\mathbf{d}_{\text{path}}^{(1)} & \dots & \mathbf{d}_{\text{path}}^{(\ell)}\end{bmatrix} = \mathbf{A}\mathbf{C}^{\top}\begin{bmatrix}\mathbf{Z}^{(1)} & \dots & \mathbf{Z}^{(\ell)}\end{bmatrix} \qquad (2.13)$$

To get the circuit delay distribution, we apply Eq. (2.6) to Eq. (2.13)

$$\begin{bmatrix}d_{\text{circuit}}^{(1)} & \dots & d_{\text{circuit}}^{(\ell)}\end{bmatrix} = \begin{bmatrix}\max(\mathbf{d}_{\text{path}}^{(1)}) & \dots & \max(\mathbf{d}_{\text{path}}^{(\ell)})\end{bmatrix} \qquad (2.14)$$

This is essentially a Monte Carlo simulation expressed in matrix form. A histogram of the circuit delay vector in Eq. (2.14) produces the circuit delay distribution. Thus

Eq. (2.14) represents path-based Statistical Static Timing Analysis (SSTA) ignoring slope. In this form, SSTA is a natural extension of STA as written in Eq. (2.6) and is in the form of a simple matrix-matrix multiplication. We make a few remarks about the matrices. It is important to note that $\mathbf{AC}^\top$ matrix is a sparse matrix, which allows for efficient storage as well as fast computation. The $\mathbf{Z}$ vector, though dense, depends only on the number of gates and not on the number of paths.

2.3.3 Example

We will present an illustrative example to show how matrices are constructed and how Monte-Carlo simulations are done in our framework.

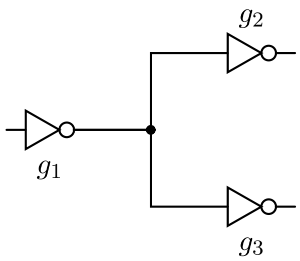


Figure 2.6: Example circuit for illustrating how Monte-Carlo simulation is done in our framework.

There are two paths in the circuit shown in Figure 2.6. This can be captured in the incidence matrix $\mathbf{A}$ of Eq. (2.11):

$$\mathbf{A} = \begin{array}{c} \\ p_1 \\ p_2 \end{array} \begin{array}{c} \begin{array}{ccc} g_1 & g_2 & g_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array} \tag{2.15}$$

For the sake of illustrative purposes, consider the delay to be function of just length L. Let $d_1 = 70L_1$, $d_2 = 50L_2$ and $d_3 = 60L_3$. Then the coefficient matrix ($\mathbf{C}^\top$) and parameter matrix $\mathbf{Z}$ in Eq. (2.11) is given by:

$$\mathbf{C}^\top = \begin{pmatrix} 70 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 60 \end{pmatrix} \tag{2.16}$$

$$\mathbf{Z} = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} \tag{2.17}$$

We get the path delays using Eq. (2.11):

$$\begin{aligned} \mathbf{d}_{\text{path}} &= \mathbf{A}\mathbf{C}^\top\mathbf{Z} \\ &= \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 70 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 60 \end{pmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} \\ &= \begin{bmatrix} 70 & 50 & 0 \\ 70 & 0 & 60 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = \begin{bmatrix} 70L_1 + 50L_2 \\ 70L_1 + 60L_3 \end{bmatrix} \end{aligned} \tag{2.18}$$

The Monte-Carlo simulation is done by sampling L_1, L_2 and L_3 from their respective distributions. For example, after sampling it turns out that $L_1 = 1.6$, $L_2 = 1.4$ and $L_3 = 1.5$. Then substituting these values in Eq. (2.18) leads to path delays for this instance:

$$\mathbf{d}_{\text{path}} = \begin{bmatrix} 182 \\ 202 \end{bmatrix} \text{ps} \tag{2.19}$$

This corresponds to one Monte-Carlo simulation as in Eq. (2.12). Repeating this ℓ times allows one to obtain a distribution for path delays and circuit delay. The number of samples ℓ depends on the confidence interval we seek in the variance of the distribution and it is usually set to $\ell = 10,000$ in the literature. We note that the extension to waveform (approximated by a ramp) propagation is straightforward and presented in more detail in Singh's dissertation [30]. Here we present a sketch of incorporating slope propagation in our SSTA framework.

2.4 Sparse Matrix based SSTA with Slope Propagation

We now extend our delay model to include slope propagation. It is important to note that the output slope of gate j cannot be specified unless we know which path it belongs to. For example, in Figure 2.5, gate g_3 at input pin a will have two different slopes namely:

1. Due to path 1 ($g_{1_a} \rightarrow g_{3_a}$), and
2. Due to path 4 ($g_{1_b} \rightarrow g_{3_a}$).

We use the same canonical form to express both delay and slope, but we restrict the dependence of delay and output slope on the input slope to be *linear*. This linearity is required in order to preserve the canonical form as delays are accumulated along a path. We also observed a linear relationship between delay and input slope for the cells in our library and we illustrate it in Figure 2.7 for one of the cells in the library. We use the superscripts d and s to distinguish among them. We delineate the input slope to a gate by the subscript in. The gate delay d_{ij} and the output slope s_{ij} of gate j in path i is given by:

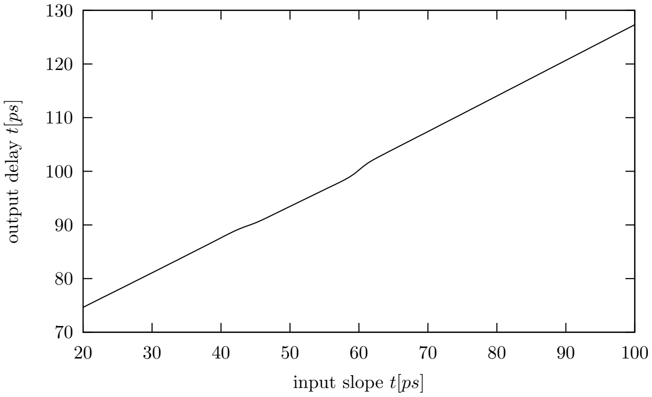


Figure 2.7: Linear relationship between input slope and delay.

$$d_{ij} = \lambda_j^d s_{\text{in}} + (\mathbf{c}_j^d)^\top \mathbf{z}_j \tag{2.20}$$

$$s_{ij} = \lambda_j^s s_{\text{in}} + (\mathbf{c}_j^s)^\top \mathbf{z}_j \tag{2.21}$$

From Eq. (2.20), one can see that the input slope at all the gates is required to calculate the gate and path delays. One way to solve for the input slope is to look at each path p separately and obtain the slope of each gate in an individual path. This method is illustrated using the Figure 2.8, and this simple circuit consists of inverters which allows us to conveniently drop the input-pin specific subscripts.

Let $\mathbf{s}_p$ be the column vector in which the values of slopes along path p are listed. Assume that the values are listed in the topological order of the gates along path p.

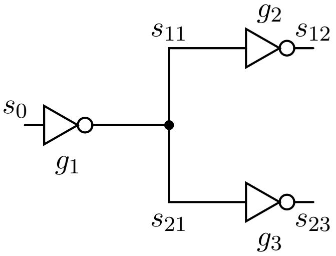


Figure 2.8: A simple circuit to illustrate SSTA with slope propagation. Here s_0 denotes the slope at the primary input. The output slope at gate g_1 in path 1 is denoted as s_{11} and in path 2 is denoted as s_{21}.

To illustrate, consider the path $p = 1$, through gates g_1 and g_2 in Figure 2.8. The column vector $\mathbf{s}_1$ is given by

$$\mathbf{s}_1 = \begin{bmatrix} s_0 \\ s_{11} \\ s_{12} \end{bmatrix} \tag{2.22}$$

and related by Eq.(2.21) as

$$\begin{aligned} \begin{bmatrix} s_0 \\ s_{11} \\ s_{12} \end{bmatrix} &= \begin{pmatrix} 0 & 0 & 0 \\ \lambda_1^s & 0 & 0 \\ 0 & \lambda_2^s & 0 \end{pmatrix} \begin{bmatrix} s_0 \\ s_{11} \\ s_{12} \end{bmatrix} + \begin{bmatrix} (\mathbf{c}_0^s)^\top \mathbf{z}_0 \\ (\mathbf{c}_1^s)^\top \mathbf{z}_1 \\ (\mathbf{c}_2^s)^\top \mathbf{z}_2 \end{bmatrix} \\ \mathbf{s}_1 &= \mathbf{\Lambda}_1^s \mathbf{s}_1 + \mathbf{diag}((\mathbf{c}_0^s)^\top, (\mathbf{c}_1^s)^\top, (\mathbf{c}_2^s)^\top)\, \mathbf{Z}_1 \\ &= \mathbf{\Lambda}_1^s \mathbf{s}_1 + (\mathbf{C}_1^s)^\top \mathbf{Z}_1 \end{aligned} \tag{2.23}$$

In general Eq. (2.23) is valid for any arbitrary path containing t gates. Thus, $\mathbf{s}_1 \in \mathbb{R}^{t+1}$, $\mathbf{\Lambda}_1^s \in \mathbb{R}^{(t+1)\times(t+1)}$ is *lower diagonal*, and $(\mathbf{C}_1^s)^\top \mathbf{Z}_1 \in \mathbb{R}^{t+1}$. If the circuit has p paths, then the Eq. (2.23) for all the p paths can be succinctly captured into one single equation shown below:

$$\begin{aligned}
\begin{bmatrix} \mathbf{s}_1 \\ \dots \\ \mathbf{s}_p \end{bmatrix} &= \mathbf{diag}(\mathbf{\Lambda}_1^s, \dots, \mathbf{\Lambda}_p^s) \begin{bmatrix} \mathbf{s}_1 \\ \dots \\ \mathbf{s}_p \end{bmatrix} + \begin{bmatrix} (\mathbf{C}_1^s)^\top \mathbf{Z}_1 \\ \dots \\ (\mathbf{C}_p^s)^\top \mathbf{Z}_p \end{bmatrix} \\
\mathbf{s} &= \mathbf{\Lambda}^s \mathbf{s} + \mathbf{diag}((\mathbf{C}_1^s)^\top, \dots, (\mathbf{C}_p^s)^\top)\,\mathbf{Z} \\
&= \mathbf{\Lambda}^s \mathbf{s} + (\mathbf{C}^s)^\top \mathbf{Z} \qquad (2.24)
\end{aligned}$$

From Eq. (2.24) we can solve for the slope $\mathbf{s}$ in the circuit

$$\mathbf{s} = (\mathbf{I} - \mathbf{\Lambda}^s)^{-1} (\mathbf{C}^s)^\top \qquad (2.25)$$

The equation for path delays is similar to Eq. (2.24) and can be calculated as,

$$\begin{aligned}
\mathbf{d}_{\text{path}} &= \mathbf{\Lambda}^d \mathbf{s} + (\mathbf{C}^d)^\top \mathbf{Z} \\
&= \mathbf{\Lambda}^d (\mathbf{I} - \mathbf{\Lambda}^s)^{-1} (\mathbf{C}^s)^\top \mathbf{Z} + (\mathbf{C}^d)^\top \mathbf{Z} \qquad \text{(using Eq. (2.25))} \\
&= \mathbf{\Lambda}^d (\mathbf{I} - \mathbf{\Lambda}^s)^{-1} (\mathbf{C}^s)^\top \mathbf{Z} + (\mathbf{C}^d)^\top \mathbf{Z} \\
&= \left(\mathbf{\Lambda}^d (\mathbf{I} - \mathbf{\Lambda}^s)^{-1} (\mathbf{C}^s)^\top + (\mathbf{C}^d)^\top \right) \mathbf{Z} \qquad (2.26) \\
&= \mathbf{D}^\top \mathbf{Z}
\end{aligned}$$

Since path delays are a function of process parameters, by taking a sample of all process parameters one can generate delay for all paths in the circuit as captured in the following equation:

$$\mathbf{d}_{\text{path}}^{(k)} = \mathbf{D}^\top \mathbf{Z}^{(k)} \qquad (2.27)$$

The circuit delay is given by the max of all path delays. Now if we take ℓ samples then we get ℓ samples of circuit delay captured in the following equation:

$$\begin{bmatrix} d_{\text{circuit}}^{(1)} & \dots & d_{\text{circuit}}^{(\ell)} \end{bmatrix} = \begin{bmatrix} \max(\mathbf{d}_{\text{path}}^{(1)}) & \dots & \max(\mathbf{d}_{\text{path}}^{(\ell)}) \end{bmatrix} \tag{2.28}$$

A histogram on these ℓ samples gives us the circuit delay distribution. Thus SSTA can be performed considering the slope and process variations.

2.5 Experimental Results

We implemented our algorithm using a combination of awk/perl scripts and C++. We report the results of experiments run on the ISCAS89 benchmarks using a 64-bit Linux machine with 16 GB RAM and running at 3.4 GHz. The delay models were generated using the 90 nm Berkeley Predictive Technology Model [31]. In the experiments only latch-to-latch paths were considered for timing. Thus in Table 2.1 only the latch-to-latch paths and the number of gates between the latches are listed. We modeled the effect of variations in channel length and threshold voltage, and assumed that the variance of these parameters was such that $3\sigma = 0.2\mu$. We modeled the impact of spatial correlation on parameter variations, and therefore required placement information for the circuits, which we obtained by placing the circuits using Dragon [32]. To properly account for random die-to-die (global) and within-die (intra) variations along with the spatial component mentioned above, we modeled each process parameter $z_{g,i}$ as:

$$z_{g,i} = \sqrt{0.5}\, z_{g,i}^{\text{global}} + \sqrt{0.25}\, z_{g,i}^{\text{intra}} + \sqrt{0.25}\, z_{g,i}^{\text{spatial}} \tag{2.29}$$

where $z_{g,i}^{\text{intra}}$ models the random variation and $z_{g,i}^{\text{spatial}}$ models the spatial variation by introducing new grid random variables [12].

We performed $10,000$ Monte Carlo simulations for each of the ISCAS benchmark circuits. The number of simulations performed in our experiment was set high in order to establish an accurate result. But a run with one tenth $(1,000)$ the number of samples would normally be sufficient to calculate the delay distribution to engineering accuracy. The results are summarized in Table 2.1. The table contains the number of gates and paths along with the runtime taken by the algorithm to compute the delay statistics of the circuit. Also shown is the breakdown of effort among

(a) Path enumeration (implemented in awk),

(b) Sparse matrix generation (implemented in perl), and

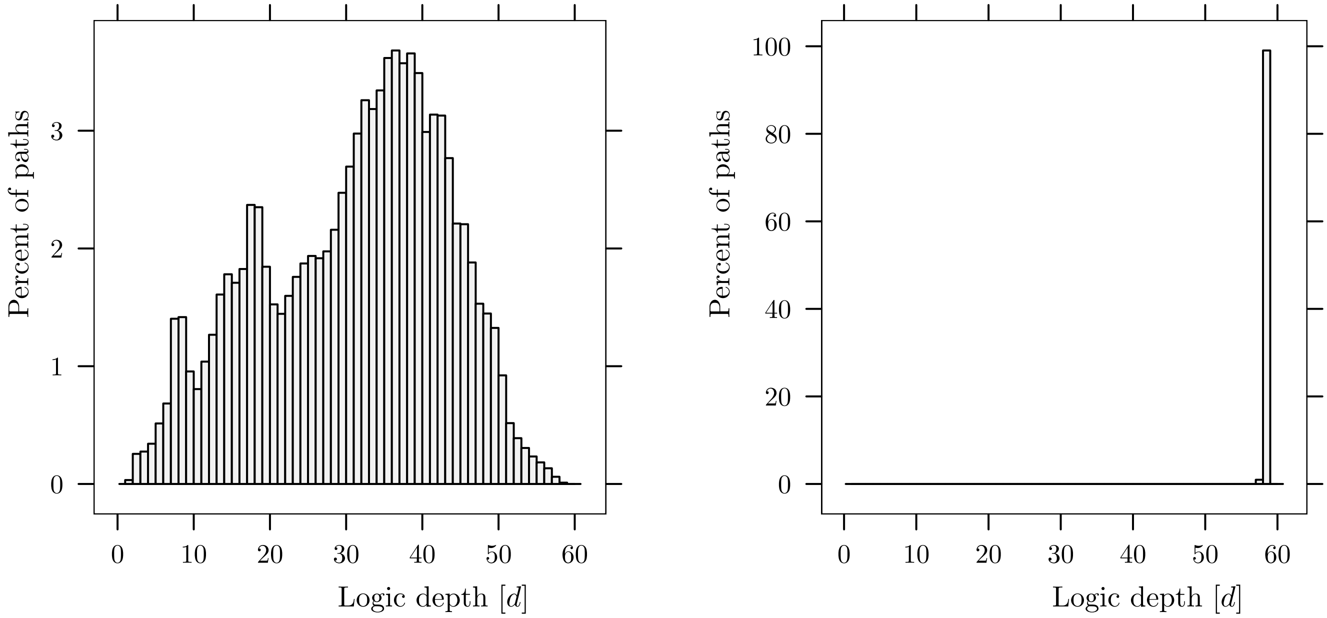
(c) Matrix multiplication (implemented in C++).

In Table 2.1, we have presented results for smaller benchmarks in ISCAS89. For the bigger benchmarks we implemented speedup techniques and they are discussed next.

2.5.1 Speedup Techniques

In this section, we describe techniques to speedup the Monte-Carlo simulation. The speedup technique is based on removing the non-critical paths to reduce the matrix size.

If we are concerned only about the circuit delay, the maximum of all path delays, then we can prune out non-critical paths based on their logic depth. To



(a) Histogram of logic depth of paths in s1423

(b) Histogram of logic depth of paths with the maximum delay in s1423

Figure 2.9: Histogram was obtained after doing Monte-Carlo simulation for $10,000$ runs. Note that the logic depth of paths with maximum delay in Figure 2.9(b) is always either 59 or 60.

ensure fairness, complex gate like XOR can be assigned an equivalent logic depth of 2 instead of being treated the same as a NAND gate.

The idea behind the pruning technique is illustrated using s1423. The histogram of logic depth of paths in s1423 is shown in Figure 2.9(a). In Figure 2.9(b), the histogram of the logic depth of the path with the maximum delay in a Monte-Carlo simulation is plotted. The histogram was obtained after doing Monte-Carlo simulation for $10,000$ runs. s1423 is representative of the ISCAS89 benchmarks and it is clear that the paths with higher logic depths tend to be ones which have maximum path delay. So we employ the strategy of eliminating paths with smaller logic depth when we are computing the circuit delay, the maximum of all path delays.

We have applied this pruning technique to the biggest benchmarks in the ISCAS89 benchmark suite. Our pruning strategy is to eliminate paths whose logic depth is less than 90% of the maximum logic depth of the circuit[1]. For example, if the circuit's maximum logic depth is 60 then all paths whose logic depth is less than $0.9 \times 60 = 54$ are eliminated.

An interesting question is what happens when the maximum logic depth is around 10 which is a typical number for a pipeline in a modern processor. The answer is, if the number of paths between the pipeline stages is around $100,000$ then there is no need for pruning. Otherwise an adaptive strategy to prune can be adopted. Here after every 100 Monte-Carlo runs one can prune out the paths which always end up in the bottom of decreasing sort of path delays in every Monte-Carlo run.

In Table 2.2, benchmark s38417 seems a little anomalous. This is because there is a long tail in the logic depth of s38417 as shown in Figure 2.10. This means that after pruning we are left with just 280 paths to analyze. To validate our approach we compare the pdf obtained from golden simulation (where no paths are eliminated) to the pdf obtained after pruning paths in Figure 2.11. Note that both the pdf's: one obtained after pruning paths, and the other obtained by considering all the paths, labeled exact, are nearly identical. The mean obtained from both methods were the same, while was the standard deviation obtained after pruning paths differed by less than 0.9% from the exact method.

---

[1] We found that retaining paths whose logic depth was greater than 90% of the maximum logic depth of the circuit provided the best tradeoff between accuracy and speed for our synthesized ISCAS89 benchmarks.

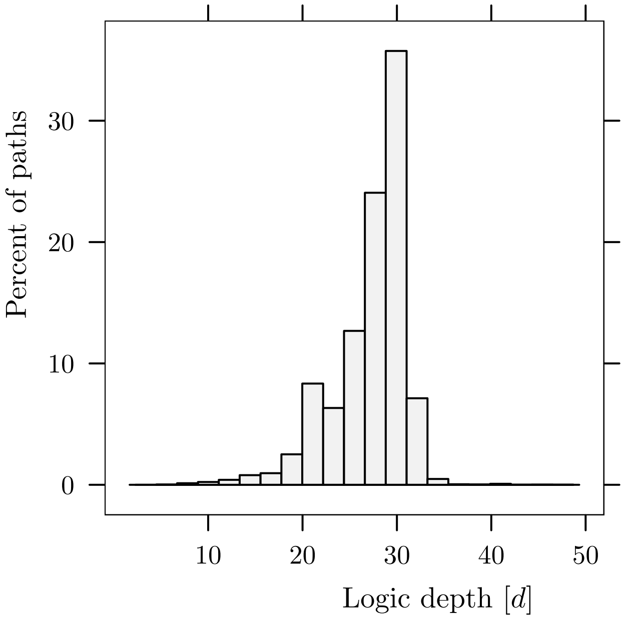


Figure 2.10: Histogram of logic depth of paths in s38417. Note that there is a long tail and there are paths with logic depth of 47. Thus our pruning strategy of eliminating paths with logic depth less than 0.9×47 leaves us with just 280 critical paths.

Note that the delay distribution of s38417 deviates from normality by having long tails in Figure 2.11. The deviation from normality is because of two reasons:

1. max operation;
2. The quadratic terms in delay equation in Eq. (2.1). Even if the parameters follow normal distribution, the quadratic terms make the distribution non-normal.

The results from the other two benchmarks s13207 and s38584 are similar to the results from s38417.

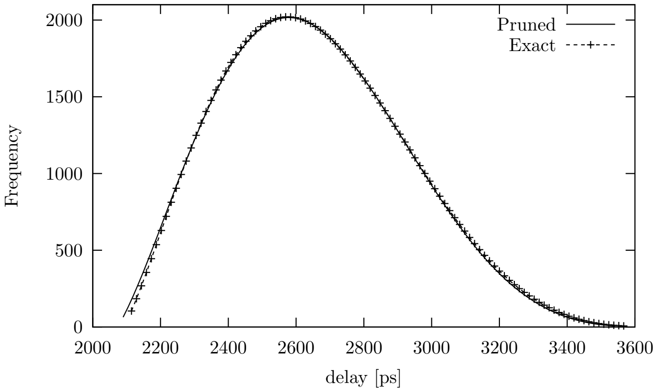


Figure 2.11: Delay pdf of s38417 obtained after pruning paths plotted against the one obtained without pruning paths. Note that both the pdf's are virtually indistinguishable.

2.6 Comparative Studies

In this section we compare our proposed method with a graph based Monte-Carlo method and a block-based method [1].

2.6.1 Comparison with Path based Monte-Carlo method

In this section, we compare the runtimes of the proposed method with a path-based Monte-Carlo (MC) since a block based MC cannot handle slope propagation accurately. By path-based MC simulation we mean evaluation of path delays without resorting to sparse matrix method. To understand the difference between the proposed sparse matrix method and the graph based MC method let us compare the algorithms used to implement them.

First we summarize the steps involved in a sparse-matrix based method as a pseudocode in Algorithm 1. Next we summarize the steps involved in a path-based MC as a pseudocode in Algorithm 2.

Algorithm 1 Sparse-Matrix-Based-SSTA

Input: Circuit description after it has been mapped to a library

Output: Timing distribution of the circuit

```
Enumerate *all* latch to latch paths in the circuit using Depth First Search (DFS) [33].
Calculate the parameterized delay for each of the paths and store it in sparse matrix. This process is captured in Eq. (2.26).
for $i = 1$ to $\ell$ do
    Generate a sample of process parameters, pre-multiply it by the sparse matrix to get a vector of path delays. This is captured in Eq. (2.27). Apply the max operator to get the circuit delay. This constitutes a single Monte Carlo simulation.
end for
The above for loop results in a vector of circuit delays of length $\ell$. This is captured in Eq. (2.28). From this one can generate circuit delay statistics and estimate the timing yield of the circuit.
```

Algorithm 2 Graph-Based-SSTA

Input: Circuit description after it has been mapped to a library

Output: Timing distribution of the circuit

```
Enumerate *all* latch to latch paths in the circuit using Depth First Search (DFS) [33].
for $i = 1$ to $\ell$ do
    Generate a sample of process parameters for every gate in the circuit.
    for $p = 1$ to npaths do
        Get the output slope and delay for every gate in the path based on the process parameters and input slope. Add the delays of all the gates in the path to get the path delay.
    end for
    Apply the max operator over all path delays to get the circuit delay. This constitutes a single Monte Carlo simulation.
end for
The outer for loop results in a vector of circuit delays of length $\ell$. From this one can generate circuit delay statistics and estimate the timing yield of the circuit.
```

The algorithms presented in Algorithm 1 and Algorithm 2 look deceptively similar. The major difference between the two lies in Step 2 of the Sparse matrix based method (Algorithm 1). We generate path delays in a parameterized form and store it as a sparse matrix. [2]. In the path based method, the delay model evaluation is done inside the `for()` loop as shown in Step 5 (Algorithm 2). Thus we need to do sample L and V_{th} for each simulation and then evaluate all the gates for their delays and output slope. Then gate delays along a path are added up to get the path delay. Our sparse matrix approach calculates delay and slope in a parameterized (symbolic) form and avoids this delay evaluation inside a `for()` loop. This is the reason behind the efficiency of our approach compared to a path-based Monte-Carlo method.

The runtimes for the two approaches are compared for smaller benchmarks in Table 2.3. From the table one can observe that the proposed approach has a runtime which is orders of magnitude faster than the path based method.

2.6.2 Implementation Details

We presented the construction of sparse matrix (**D**) in Eq. (2.27) as a series of operations on matrices as shown in Eq. (2.26). This was done to present our analysis in a mathematically rigorous as well as an elegant fashion. It should be noted that one can construct the sparse matrix (**D**) using graph traversal method. In fact the sparse matrix construction in our implementation was done using graph traversal method.

The sparse matrix can be thought of as an efficient data structure to hold the parameterized path delays. An alternative data structure such as array of hashes to

---

[2] It should be noted that there is no restriction on the gate delay model except the need to have input slope appear linearly. The linearity restriction helps us to preserve the canonical form of the delay and slope models.

hold parameterized path delays lead to slower runtimes. In array of hashes one can visualize array consisting of all possible paths; each element in the array points to a hash which consists of all the gates in that path. The runtime comparison is shown in Table 2.3. It is clear that having sparse matrix as a data structure is superior since it is results in a regular access from caches leading to a faster runtime.

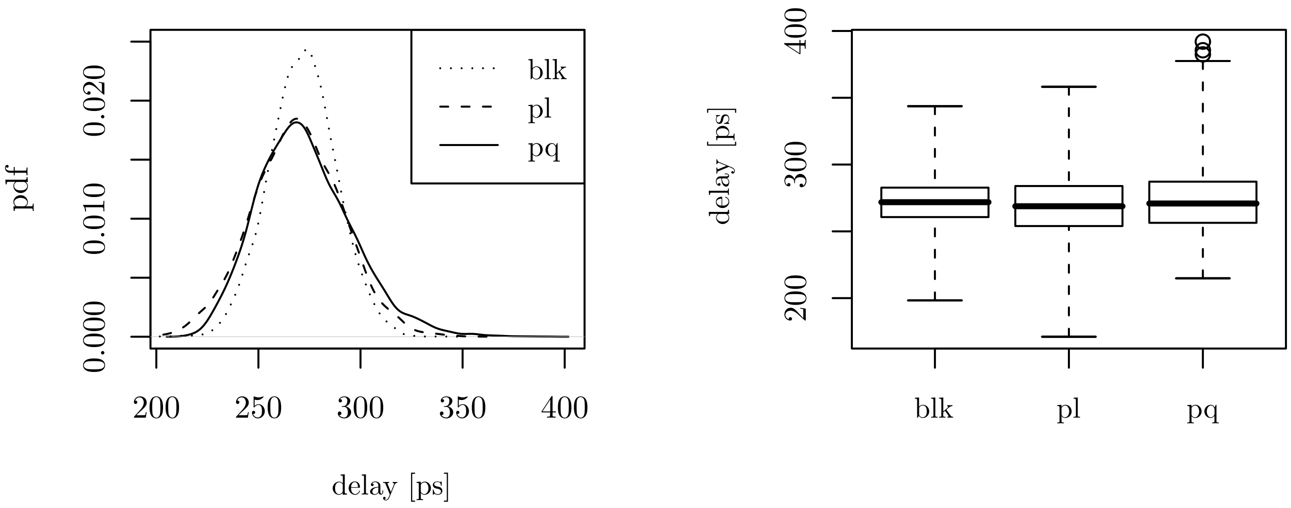
2.6.3 Comparison with Block-based method

A comparison of the proposed method with a block based method [1] is shown in Figure 2.12 for s27, clearly showing the proposed method's accuracy when compared with block-based method. The tails in the distribution are not captured by the block-based method since its delay model is linear. The block based method is also restricted to function only with Normal distributions. We note that the biggest benchmarks in ISCAS89 ran in a few seconds using the block-based method showing the runtime superiority of the block-based method.

2.7 Summary

This work demonstrates that it is possible and practical to perform path based statistical static timing analysis, and that such an analysis can be written compactly in matrix notation, allowing the use of standard highly optimized linear algebra techniques. The major advantage of this formulation is that it places no restrictions on process parameter distributions. It embeds accurate polynomial-based delay model which takes into account slope propagation naturally.

Data was presented to show that many practical circuits have a bounded number of paths, making such an analysis possible. It should be noted that this



(a) Density plot of circuit delay of s27

(b) Boxplot of circuit delay of s27

Figure 2.12: Delay pdf of s27 obtained using block-based method [1] (denoted as *blk*), path based method with *linear* delay models (denoted as *pl*) and path based method with *quadratic* models (denoted as *pq*). Results from path based method with *linear* models can be thought of how much error is introduced by an analytical max() and using worst case slope at the input of a gate. Results from path based method with *quadratic* models can be thought of how much error is introduced when we use linear delay models. Please note that the linear models are not adequate enough to model the tails of the distribution.

demonstration should not be taken as sufficient license to propose a purely path-based SSTA algorithm.

Table 2.1: Path-gate statistics of ISCAS89 benchmarks and runtime for $10,000$ simulations.

circuit	gates	paths	sps[s][%]	runtime [s]			
				generating[ap]		matrix[c] multiply	total
				paths	matrix		
s27	8	9	46.13	0.02	0.02	0.07	0.11
s1196	73	43	11.61	0.15	0.07	0.60	0.82
s1238	73	43	11.61	0.12	0.04	0.39	0.55
s208	50	72	10.48	0.07	0.04	0.49	0.60
s386	92	86	8.56	0.08	0.07	0.57	0.72
s820	187	207	3.42	0.15	0.13	1.14	1.42
s298	98	212	4.97	0.10	0.11	1.04	1.25
s832	188	219	3.41	0.15	0.12	1.07	1.34
s510	162	230	4.02	0.15	0.19	1.39	1.73
s641	237	238	12.82	0.41	2.14	4.80	7.35
s344	154	323	6.21	0.17	0.43	2.33	2.93
s349	155	333	6.11	0.21	0.40	1.70	2.31
s382	133	353	4.21	0.16	0.17	1.18	1.51
s1488	307	366	3.36	0.31	0.51	3.33	4.15
s1494	306	375	3.37	0.36	0.52	3.33	4.21
s526n	172	377	2.66	0.15	0.20	1.75	2.10
s526	171	379	2.67	0.15	0.12	1.76	2.03
s444	160	482	4.18	0.21	0.27	1.60	2.08
s953	328	723	2.54	0.40	0.76	3.93	5.09
s713	250	2650	17.54	5.13	36.47	50.42	92.02
s5378	1938	6858	0.66	4.44	9.62	20.28	34.34

[s] sps denotes sparsity, the percentage of nonzeros in the sparse matrix

[ap] We wrote awk/perl scripts to generate paths and build the sparse matrix

[c] We wrote C++ program to do matrix multiplication

Table 2.2: Runtime after logic depth based pruning for the three biggest benchmarks in ISCAS89. All paths whose logic depth was less than 90% of the maximum logic depth were pruned. The number of simulations were set to $10,000$.

circuit	pruned paths[p]	pruned runtime [s]			
		generating		matrix multiply	total
		paths	matrix		
s1423	334	22.20	4.83	19.31	46.34
s9234	33536	227.89	477.36	1691.61	2396.86
s35932	39168	91.70	150.53	1012.51	1254.74
s38584	35904	530.90	471.37	1732.91	2735.18
s13207	78082	715.66	1138.58	3946.16	5800.40
s38417	280	541.31	3.75	12.47	557.53

[p] This column shows the number of paths left after logic depth based pruning.

Table 2.3: Runtime comparison for the proposed matrix method versus repeated path tracing method. The number of simulations were set to $10,000$.

Circuit	Path MC[p][s]	Sparse Matrix[m][s]	Alt DS[d][s]
s27	220.03	0.11	1.12
s208	380.05	0.60	10.76
s1196	460.16	0.82	12.10
s298	880.09	1.25	29.68
s382	1360.13	1.51	56.00
s344	3060.17	2.93	81.05

[p] This column shows the runtime of a path based MC whose pseudo code was presented in Algorithm 2. Monte-Carlo simulation uses path-based approach since slope can not be accurately propagated in a block-based method.

[m] This column shows the runtime of the proposed sparse matrix method presented in Algorithm 1.

[d] This column shows the runtime of the proposed method with an alternative data structure, array of hashes. The algorithm is the same as the one presented in Algorithm 1 but the data structure changes from Sparse Matrix to Array of Hashes.

Chapter 3

Waveform Modeling Using Singular Value Decomposition

In the previous chapter, we proposed a path-based sparse matrix SSTA with waveform propagation. The waveform was approximated as a ramp propagated along the gates in a path. As technology scales into the nanometer regime, however, a ramp approximation is no longer sufficient [5]. Several studies in the literature have improved upon ramp approximation [34–40].

Current approaches can be broadly classified into one of the following three approaches:

- *Improved heuristic model* in which authors have proposed models other than ramp, e.g. Equivalent waveform model [34], Weibull distribution [36, 37]
- *Data based approaches* in which authors have proposed statistical techniques such as Principal Components Analysis (PCA) [35] or a heuristic approach [38].

- *Change of basis models* in which authors have proposed modeling current rather than voltage, e.g. CSM [39–42].

At the input side, an equivalent waveform modeling has been proposed which employs a weighted least squares fit [34]. The heuristic equivalent waveform consists of a ramp followed by an exponential. A heuristic for weight is given by $\frac{\partial v_{\text{out}}}{\partial v_{\text{in}}}$ which weighs the part of input waveform that affects the output waveform. This is an improvement over the ramp model, because the proposed equivalent waveform can model the non-linearity of real waveforms to some extent. An analytical technique for modeling the waveforms is to approximate the waveform with a cumulative probability distribution function (CDF). The Weibull distribution, a two parameter model, is used to model the waveforms [36]. The slope and shape of the waveform are approximately modeled, in contrast to a simple ramp model where there is only one parameter, slope. Since it also models the shape of the waveform, the Weibull model yields better results. The Weibull model has been extended to include crosstalk noise waveforms [37].

An alternative to the analytical approach is the data-based approach, which requires a set of all possible waveforms that would be encountered as a starting point. Such a set of *input* waveforms is generated by aggregating the waveforms at the output of different interconnect structures. The set of possible *output* waveforms is generated by collecting waveforms at the output of all gates in the library under process variations and different environmental conditions [38]. Generating and aggregating all possible waveforms is thus a time-consuming pre-processing step, but the process gets amortized over many runs of the timer. After all possible waveforms have been generated, a set of basis waveforms can be extracted which approximate all possible waveforms using an affine transformation [38]. The selection of basis waveforms from the set of all possible waveforms is done efficiently using unate covering heuristic. It has been shown that a few basis waveforms are sufficient for accurate waveform modeling.

Current source-models (CSM) based on transistor physics have been proposed to model waveforms [39, 40]. In CSM, the most important feature is the introduction of a current source to model the output drive. Thus it can capture the non-linearity of the driver accurately.

Another data-based approach is the Principal Components Analysis (PCA) based waveform modeling [35]. PCA based waveform modeling is a data-based approach in the sense that one generates all possible waveforms that one will encounter. The waveforms are discretized at n equal voltage intervals and recording the time at which certain voltage thresholds are crossed. If the crossing times are treated as random variables then if they are highly correlated, a dimension reduction technique like PCA can be applied to represent a waveform accurately in a reduced $r < n$ uncorrelated space [35].

Our approach builds on the PCA based waveform modeling [35]. The PCA method was described for a single gate and we generalize the method to a library of gates. We use Singular Value Decomposition (SVD) instead of PCA in this work. The reason is that we are interested in finding the orthogonal basis vectors of the waveforms and SVD provides a simple and direct way to do so. Each of these orthogonal basis vectors provide a linear combination of the n time points. Interestingly, as we will see later on, the first two of these orthogonal basis vectors (linear combinations) can be interpreted as 50% point and slope of the waveform respectively. This allows us to link this approach to current methodologies, and to have it gracefully degrade to a simple ramp approximation when needed.

Our goal in this work is to show that n time points are not needed to model the waveform since only a few orthogonal basis vectors (linear combinations) of these time points are sufficient to model the waveform accurately.

The contributions of this work are:

- We provide a rigorous mathematical analysis of waveforms using SVD leading to a generalized gate delay model.
- We link the proposed approach to ramp-based models and show it is a logical extension to current modeling and simulation methods.
- We show how the approach can provide a systematic method for trading off complexity vs. accuracy in the waveform models.
- We generalize and extend PCA approach [35] to a library of gates.

3.1 Data Based Model for a Gate

In this section we illustrate how to generate a SVD based timing model for a single gate. Later on, we extend our approach to the entire library. Since proposed timing model is based on data, assume that we have a diverse set of waveforms to work with to obtain our timing model. The method for generating a diverse set of waveforms is described later in Section 3.5.

A waveform is *discretized* by recording the time points t_k, when the voltages cross $\frac{k-1}{n-1}$, $1 \leq k \leq n$ as shown in Figure 3.1. In the case of falling waveforms the notation changes; the time points t_k denote the voltages crossing $\left(1 - \frac{k-1}{n-1}\right)$, $1 \leq k \leq n$. We also assume that the voltages are normalized ($V_{\text{DD}} = 1$).

The collection of output waveforms is discretized and the times at which the voltage thresholds are crossed are recorded. The discretized input and output

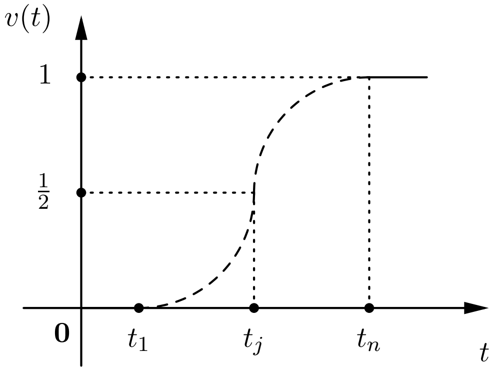


Figure 3.1: A waveform is discretized and time points are recorded when a signal crosses a certain voltage threshold. Here time points are recorded when the waveform crosses the threshold of $\frac{1}{2}$ and 1.

waveforms can be collected in one single matrix **T**:

$$\mathbf{T} = \begin{pmatrix} t_{1,1} & t_{1,2} & \dots & t_{1,n} \\ \dots & \dots & \dots & \\ t_{m,1} & t_{m,2} & \dots & t_{m,n} \end{pmatrix} \tag{3.1}$$

Note that each row in the matrix **T** contains one discretized waveform. This is similar to current procedures except that instead of taking each waveform and approximating it as a ramp, we take *all* the waveforms and develop a new model via the SVD process, which we outline next.

3.2 Analyzing Waveforms using SVD

The Singular Value Decomposition (SVD) is a fundamental theorem of Linear Algebra [43]. SVD is defined as [44]:

Definition 3.2.1 (Singular Value Decomposition). Let $m, n \in \mathbb{N}$ be arbitrary; we do not require $m \geq n$. Given $\mathbf{T} \in \mathbb{R}^{m \times n}$, not necessarily of full rank, a *singular value decomposition* (SVD) of $\mathbf{T}$ is a factorization

$$\mathbf{T} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^{\top} \tag{3.2}$$

where $\mathbf{U} \in \mathbb{R}^{m \times m}$ is orthonormal, $\mathbf{V} \in \mathbb{R}^{n \times n}$ is orthonormal, and $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is diagonal. In addition, $\mathbf{\Sigma}$ is assumed to have its diagonal entries σ_j nonnegative and in nonincreasing order; that is, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$ where $p = \min(m, n)$.

It can be proved that for any matrix SVD exists [44]. The matrix $\mathbf{U}$ is called the left singular matrix and its columns provide the *orthogonal basis for the columns* of $\mathbf{T}$. The matrix $\mathbf{V}$ is called the right singular matrix and its columns provide the *orthogonal basis for the rows* of $\mathbf{T}$. The diagonal elements of $\mathbf{\Sigma}$ are called the *singular values*.

As noted above, the columns of $\mathbf{V}$ provide the orthogonal basis for the *rows* of $\mathbf{T}$ and since each row contains a discretized waveform, the columns of $\mathbf{V}$ turn out to be the orthogonal basis vectors for the waveforms in matrix $\mathbf{T}$.

Now Eq. (3.2) can be rewritten by post-multiplying both sides by $\mathbf{V}$. Since $\mathbf{V}$ is orthonormal ($\mathbf{V}^{\top}\mathbf{V} = \mathbf{I}$) we get $\mathbf{T}\mathbf{V} = \mathbf{U}\mathbf{\Sigma}$. We denote the resultant product matrix as $\mathbf{M}$ called the *moments* matrix because this is another way to represent time points just like an equivalent representation of any function by its moments:

$$\mathbf{M} = \mathbf{T}\mathbf{V} = \mathbf{U}\mathbf{\Sigma} \tag{3.3}$$

The *moments* matrix defined here is a linear combination of time points weighed by the right singular vectors (rsv) $\mathbf{V}_{.j}$:

$$m_{ij} = \mathbf{T}_{i.}\mathbf{V}_{.j} = \sum_{k=1}^{n} t_{ik}v_{kj} \tag{3.4}$$

The right singular vectors transform a waveform from time domain $\mathbf{t} = (t_1, t_2, \ldots, t_n)$ to moments domain $\mathbf{m} = (m_1, m_2, \ldots, m_n)$ through $\mathbf{m} = \mathbf{tV}$ and vice-versa through $\mathbf{t} = \mathbf{mV}^\top$

This equivalent representation leads to an interesting possibility in the context of timing analysis. If a waveform can be represented accurately using a few moments, then by propagating these moments one can do an accurate waveform analysis instead of propagating all of the n time points. Suppose we represent a waveform by r moments, where $r < n$ then the last $n - r$ moments are set to zero. The process of setting the last $n-r$ moments to zero is equivalent to setting the last $n-r$ singular values to zero since zeroing a singular value will force the corresponding moment to zero. But zeroing out singular values is equivalent to approximating a matrix $\mathbf{T}$ with another matrix $\tilde{\mathbf{T}}$ having a smaller *rank* and this is proved in Theorem 2. The Frobenius norm is used to measure the goodness of approximation. This norm measures the goodness of fit using a root mean squares difference.

The above discussion can be summarized by noting the equivalence of the following statements:

- Approximating a waveform $\mathbf{T_{i.}}$ using the first r moments.
- Approximating a matrix considering the first r-singular values of matrix $\mathbf{T}$
- A rank-r approximation of matrix $\mathbf{T}$ in Frobenius norm.

Definition 3.2.2 (Frobenius Norm). Given $\mathbf{T} \in \mathbb{R}^{m \times n}$, the *Frobenius norm* of $\mathbf{T}$ is defined as

$$\|\mathbf{T}\|_F = \sqrt{\left(\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}^2\right)} \tag{3.5}$$

There is an equivalent way to compute the Frobenius norm of a matrix by using the singular values of a matrix which is stated next.

Theorem 1. *Given* $\mathbf{T} \in \mathbb{R}^{m \times n}$*, and its singular values after SVD is given by* $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$ *where* $p = \min(m, n)$*. Then the Frobenius norm of* $\mathbf{T}$ *can be computed by*

$$\|\mathbf{T}\|_F = \sqrt{\left(\sum_{i=1}^{p} \sigma_i^2\right)} \tag{3.6}$$

Now we are ready to state what the best approximation to a matrix in the context of Frobenius norm is.

Theorem 2. *Given* $\mathbf{T} \in \mathbb{R}^{m \times n}$*, with a singular value decomposition* $\mathbf{T} = \mathbf{U\Sigma V}^\top$*, the best approximation in Frobenius norm to* $\mathbf{T}$ *by a matrix of rank* $k \leq \min(m, n)$ *is given by*

$$\tilde{\mathbf{T}}_k = \mathbf{U}\,\mathbf{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)\,\mathbf{V}^\top$$

Thus the idea of having r singular values is equivalent to approximating a matrix $\mathbf{T}$ with another matrix $\tilde{\mathbf{T}}$ having rank r. As stated earlier, having r singular values is equivalent to having only the first r moments to represent the waveform while discarding the remaining $n - r$ moments. Thus we can define a metric in Frobenius norm to see how well r moments approximate a waveform.

Now define f_r as relative error incurred in Frobenius norm by approximating matrix $\mathbf{T} \in \mathbb{R}^{m \times n}$ by rank-r matrix $\tilde{\mathbf{T}}_r \in \mathbb{R}^{m \times r}$.

$$f_r = \frac{\|\mathbf{T} - \tilde{\mathbf{T}}_r\|}{\|\mathbf{T}\|} = \frac{\sqrt{\left(\sum_{i=k+1}^{p} \sigma_i^2\right)}}{\sqrt{\left(\sum_{i=1}^{p} \sigma_i^2\right)}} \tag{3.7}$$

where $p = \min(m, n)$. We plot the relative error in Frobenius norm f_r in Figure 3.2.

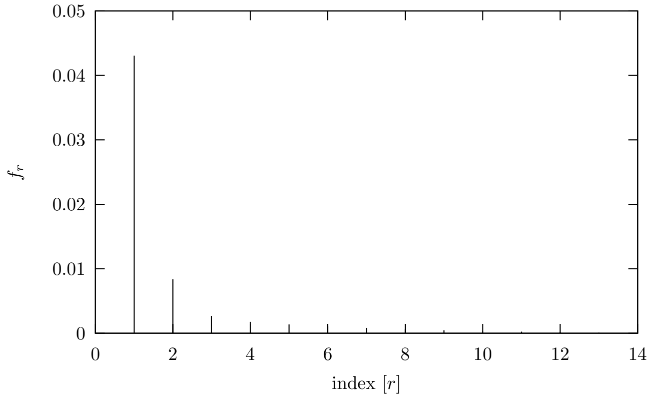


Figure 3.2: Relative error in Frobenius norm when approximating using the first k moments. Note that using the first 2 moments the approximation is accurate within 1 % in Frobenius norm. The data is shown for an inverter and the number of time points is $n = 14$.

In the figure, one can observe that the error measured by Frobenius norm quickly reduces to a very small quantity when we approximate using the first few moments. A more interesting comparison would be in terms of the absolute values involved. The error in approximating by r moments is given by:

$$\Delta\mathbf{T}_r = \mathbf{T} - \tilde{\mathbf{T}}_r \tag{3.8}$$

The entry $\Delta t_{r_{ij}}$, represents the error in approximating jth time point in the ith waveform with r moments. For a rank-3 approximation, we found that nearly 90%

of the entries in $\Delta\mathbf{T}_r$ had an absolute error which was less than 1 ps. Thus we need only a few moments to represent a waveform with a high degree of accuracy.

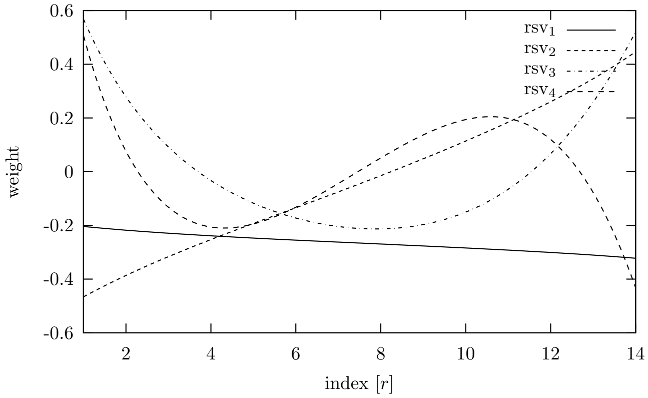


Figure 3.3: Plot of the first 4 right singular vectors (rsv) obtained on applying SVD to $\mathbf{T}$ in Eq. (3.1). The number of time points is $n = 14$.

Next we interpret what the moments mean. We will see that the moments are related to the more familiar notions of 50% time point and slope. Recall from Eq. (3.4) that moments of a waveform are obtained by the right singular vectors ($\mathbf{V}_{.j}$) weighting the linear combination of time points of a waveform.

The first right singular vector $\mathbf{V}_{.1}$ (rsv$_1$), weights the time points $\mathbf{t}$ nearly *equally* to produce the first moment m_1:

$$m_1 = \sum_{k=1}^{n} t_k v_{k1} \approx \frac{1}{n} \sum_{k=1}^{n} t_k$$

The first moment m_1 can be interpreted as the *average* of all the sample time points, and in that sense represents the 50% *time point* of the waveform.

The second right singular vector $\mathbf{V}_{.2}$ (rsv$_2$), weighs the time points $\mathbf{t}$ nearly *linearly* to produce the second moment m_2. This can be interpreted as a quantity proportional to the average slope of a linear ramp approximation to the waveform.

Note that the first and second moments are only *interpreted* to be the 50% time point and slope respectively and they are not the same in terms of numerical value. The third right singular vector weighs the time points *quadratically* to produce m_3; the fourth right singular vector weighs the time points *cubically* to produce m_4.

3.3 Example of Moments Calculation

We illustrate the calculation of moments by means of an example. After sampling a waveform at $n = 14$ time points, we get a time vector $\mathbf{w_{t_1}}$ and its corresponding voltage vector $\mathbf{w_v}$:

$$\mathbf{w_{t_1}}^\top = \begin{matrix}[599 & 632 & 656 & 673 & 690 & 705 & 716 \\ 728 & 740 & 752 & 764 & 776 & 788 & 802]\text{ps}\end{matrix} \tag{3.9}$$

$$\mathbf{w_v}^\top = \begin{bmatrix} 0 & \frac{1}{13} & \frac{2}{13} & \dots & \frac{11}{13} & \frac{12}{13} & 1 \end{bmatrix} \text{V} \tag{3.10}$$

Now collect m such waveforms, ($\mathbf{w_{t_i}}$, $i = 1, \dots, m$) in a matrix $\mathbf{T}$. On applying SVD Eq. (3.2) to $\mathbf{T}$, we obtain the right singular vectors $\mathbf{V}$. Note that once the SVD analysis is done, the right singular vectors are fixed. For example, the second right singular vector $\mathbf{V_{.2}}$ after the SVD analysis:

$$\mathbf{V_{.2}} = \begin{matrix}[-0.52 & -0.41 & -0.32 & -0.25 & -0.18 \\ -0.12 & -0.06 & -0.01 & 0.05 & 0.12 \\ 0.18 & 0.24 & 0.31 & 0.39 &]^\top\end{matrix} \tag{3.11}$$

The second moment (m_2) is given by the dot product of the waveform ($\mathbf{w_{t_1}}$) and the second right singular vector ($\mathbf{V_{.2}}$):

$$m_2 = \mathbf{w_{t_1}}^\top \mathbf{V_{.2}} = -202.31 \tag{3.12}$$

The second moment (m_2) was interpreted as quantity whose absolute value approximates the slope. If we define slope as the difference in time at which voltage crosses 0 V and 1 V then the slope of $\mathbf{w_{t_1}}$ in Eq. (3.9) turns out to be 203 ps which is approximately the same value as $|m_2| = 202.31$ ps.

It is important to note that the *right singular vectors are constant vectors*, for example Eq. (3.11) shows the second right singular vector. The right singular vectors just weight the linear combination of time points or in other words transform waveform from time domain to moments domain. Thus only the moments vary depending on the waveform time points and this is shown in Eq. (3.12), where depending on $\mathbf{w_{t_i}}$ we get different values of moments.

3.4 Example of SVD based Timing Model

To illustrate the proposed timing model we calculate the delay of the simple inverter chain in Figure 3.4 and compare it with the Weibull-based timing model [36]. The Weibull model has two parameters: slope, and shape. The waveforms are fitted to the Weibull model after their arrival time is normalized to 0. The arrival time of waveform is propagated across the gates separately from propagation of the Weibull parameters.

In our characterization, we follow the same strategy as proposed in the Weibull model. We normalize the arrival time of all input waveforms to 0 during characterization and keep the arrival time information separate from the waveform information. In other words, instead of doing SVD on (t_1, t_2, …, t_n) we do it on ($t_2 - t_1$, …, $t_n - t_1$), where t_1 is the arrival time. Due to this normalization we are now left with $n-1$ time points. This translates to the fact that we have $n-1$ moments for a given waveform instead of n moments. We found that the new first moment $\tilde{m}_1$ is equivalent to the old m_2 described in Eq. (3.12).

Another way to think about it is that we have forced the first moment to be the arrival time instead of the $t_{50\%}$ time point while keeping the rest of the moments. To ensure fairness in comparison, we use two moments in our proposed model in addition to the arrival time.

Now we intuitively introduce the idea of using *moments* to model the waveform propagation across the gates. In Section 3.2, we saw that the few moments are sufficient to characterize a waveform. The first and second moments are interpreted as quantities very similar to 50% time point and slope. Thus if we use only the first two moments, then the gate delay modeling is equivalent to the ramp based delay modeling. A simple ramp based delay equation is given by:

$$\text{delay} = a_0 + a_1 S_{\text{in}} + b_1 S_{\text{in}} C_{\text{out}} + c_1 C_{\text{out}} \tag{3.13}$$

The moments-based equation can be thought of as an generalization of Eq. (3.13):

$$t_{\text{arrival}} = a_0 + \sum_{i=1}^{2} a_i \tilde{m}_i^{\text{in}} + \sum_{i=1}^{2} b_i \tilde{m}_i^{\text{in}} C_{\text{out}} + c_0 C_{\text{out}} \tag{3.14}$$

$$\tilde{m}_j^{\text{out}} = a_{0j} + \sum_{i=1}^{2} a_{ij} \tilde{m}_i^{\text{in}} + \sum_{i=1}^{2} b_{ij} \tilde{m}_i^{\text{in}} C_{\text{out}} + c_0 C_{\text{out}} \tag{3.15}$$

where $j = 1, 2$. Note that we have *not* modeled the non-linear capacitance of the fanout gates. The equations were fitted using linear regression [45]. A more detailed discussion regarding moment modeling with complex load modeling and the error involved in fitting is described in Section 3.6.

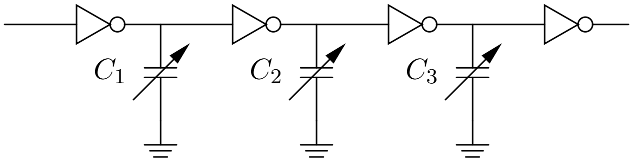
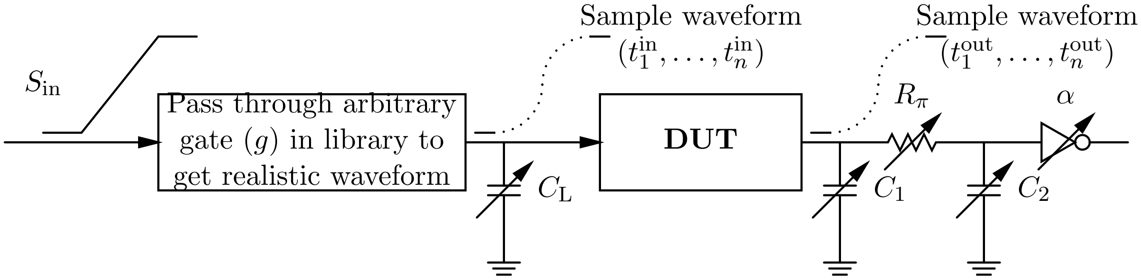


Figure 3.4: Three stage inverter chain with a single capacitor modeling the load.



S_{in}
Pass through arbitrary gate (g) in library to get realistic waveform
Sample waveform
$(t_1^{\text{in}}, \ldots, t_n^{\text{in}})$
DUT
Sample waveform
$(t_1^{\text{out}}, \ldots, t_n^{\text{out}})$
C_{L}
R_π
α
C_1
C_2

Figure 3.5: Gate characterization setup

We calculated delay for 1000 different values of the (C_1, C_2, C_3) tuple, where C_i, $i = 1, 2, 3$ was randomly sampled from 100 fF to 200 fF. The minimum error in delay using SVD model was 1.2% and the maximum error was 8.7% when compared to Spice. The corresponding statistics for the Weibull based model were 1.6% and 9.7%. and for a simple slope based model was 4.9% and 17.6%. It is clear that both Weibull and SVD based timing models are superior to the simple slope model, thus demonstrating the need for more complex delay modeling.

In the next section, we extend our SVD-based timing model to the entire library.

3.5 Data Based Model for a Library

In a timing analyzer we generally represent the circuit by a directed graph consisting of gates and the wires that connect them. In forming this representation, we make use of two main abstractions of a waveform:

- An abstraction of the switching waveform at the input and output of each gate in the circuit, and
- An abstraction for the model describing how these waveforms are changed when they go through gates or wires.

In current methodologies, the first abstraction is the ramp waveform model, and the second is the delay model for gates and wires. Since the waveform for a ramp model is represented by a tuple of delay and slope, we can represent a typical delay model as:

$$(D_{\text{out}}, S_{\text{out}}) = f((D_{\text{in}}, S_{\text{in}}), C_{\text{load}}, \ldots) \tag{3.16}$$

Note that while f is different for each gate and wire, the representation of the waveform is the *same* for all.

Thus the first step we must perform is the generation of a new *uniform* waveform model that would be valid for all components in a circuit. We do this by taking *all* the gates, which are collected in a gate *library*, and generating a large number of diverse waveforms from them. Mathematically this means that the matrix

T in Eq. (3.1) contains waveforms of *all* gates in the library instead of a single gate as described in Section 3.1. Both falling and rising waveforms are considered.

The SVD analysis described in Section 3.2 is now performed on the waveforms generated from all gates in the library. By building the model on all gates in the library, we insure that the representation of the waveform produced will be valid for all gates in the library, and thus can be used in an equation similar to Eq. (3.16).

An important thing to note is the *right singular vectors* **V**. The right singular vectors **V** obtained after SVD analysis of all the waveforms obtained from the library are the *same* for every gate in the library. Consider for the sake of argument that $\mathbf{V}_{.2}$ in Eq. (3.11) was obtained after doing SVD analysis on all the waveforms generated from the library. Then irrespective of whether the waveform is generated by an INV (inverter) or NAND, we use $\mathbf{V}_{.2}$ to generate the second moment for that waveform. This helps ensure a uniform model for all gates in the library.

In our delay models, we use a more complex load model, namely the π-model. Note that this load modeling is completely *orthogonal* to the waveform modeling which is the primary focus of the work. When the π-model is used to model the load in timing analysis, the accuracy of the simulation improves [46, 47]. Also, we model the non-linear capacitance of the driven gate marked as α in Figure 3.5.

The experimental setup to generate these waveforms for one gate is shown in Figure 3.5. By varying S_{in}, and C_L we vary the *input* waveform to the gate, and by varying C_1, R_π, C_2 and α we vary the *loading* on the gate. While it may appear that having a complex π-model will drastically increase the characterization time, we balance this addition by the use of experiment planning techniques such as Latin Hypercube Sampling (LHS) [48] in order to reduce the number of simulations required.

3.6 Waveform Propagation Across Gates in STA

In this section, we present the equations for propagating waveform through a gate with π-model load. The SVD-based timing model is a straightforward extension of Eq. (3.14) and Eq. (3.15):

$$\begin{aligned}\tilde{m}_j^{\text{out}} = a_0 + \sum_{i=1}^{r} a_i \tilde{m}_i^{\text{in}} + \sum_{j=1}^{2} b_j C_j + \sum_{i=1}^{r}\sum_{j=1}^{2} c_{ij} \tilde{m}_i^{\text{in}} C_j \\ + \sum_{j=1}^{2} d_j R_\pi C_j + e_0 R_\pi + f_0 \alpha,\ j = 1, \ldots, r \quad (3.17)\end{aligned}$$

where

- r is the number of moments used for characterization, usually set to 2 or 3.
- C_1, R_π, C_2 are the parameters of π-model interconnect. The interconnect values are assumed to be deterministic in this work.
- α is the width of the gate load being driven as shown in Figure 3.5.
- $\tilde{m}_i^{\text{in}}$ are the moments of the waveform whose arrival time has been normalized to 0 as described in Section 3.4.

The equations for characterizing arrival time t_{arrival} have the same form as in Eq. (3.17).

In the context of STA, the independent variables in Eq. (3.17) namely, $m_1^{\text{in}}, \ldots, m_r^{\text{in}}, C_1, R_\pi, C_2, \alpha$ are just numbers. Thus the Eq. (3.17), could have been any *arbitrary* function of the independent variables. But we have opted for a function which is linear in input moments ($\tilde{m}_i^{\text{in}}$). The linearity restriction becomes crucial in the context of Statistical STA (SSTA) [49].

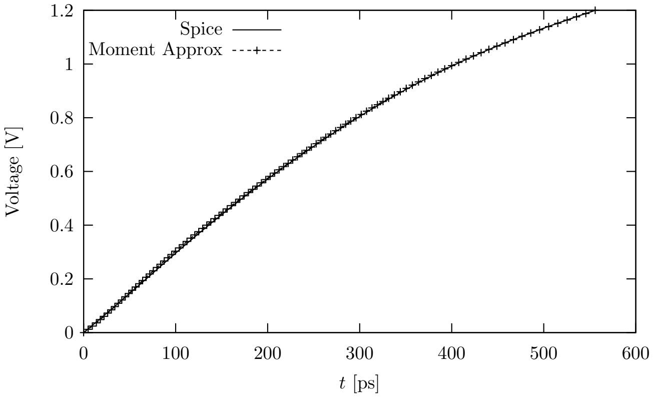


Figure 3.6: Waveform comparison at the output of an inverter. We use a 2 moment approximation in addition to arrival time and it is clear from the figure that the waveforms are indistinguishable.

The linearity restriction does not affect the accuracy of our modeling. To illustrate, consider Figure 3.6, where we plot the waveform obtained at the output of an inverter. An input waveform is fed to an inverter and simulated in Spice and the output is plotted with the legend 'Spice'. In the case of the legend 'Moment Approx' the output waveform was obtained as follows. The input waveform is transformed from timepoint representation to moment representation using $\tilde{\mathbf{m}} = \mathbf{t}\mathbf{V}$. We keep the first two moments alongside the arrival time and then using the fitting equations in Eq. (3.17), to find the moments of the output waveform. Then using the transformation from moments to time points $\mathbf{t} = \tilde{\mathbf{m}}\mathbf{V}^\top$ we obtain the time points of the output waveform which is plotted with the legend 'Moment Approx'. It is clear from the figure that the waveforms are indistinguishable thus illustrating the accuracy of the moments-based model.

Another way of comparing the output waveforms was to find the maximum relative difference between the time points predicted by the SVD-based model and Spice. In the case of INV, fitting over 1000 waveforms produced a maximum error of around 5%. The other gates in the library NOR and NAND had similar error statistics in the predicted output waveform.

We can now generalize and propagate moments across a path and recover the waveform at the end of the path.

3.7 Path Delay Evaluation in STA

In this section we demonstrate waveform propagation along a path by considering a simple stage consisting of an INV, NOR and NAND gate. with π-model for load as shown in Figure 3.7. The gate delay models are generated using 90 nm Berkeley Predictive Technology Model [31] and the interconnect parameters are obtained from ITRS roadmap [3]. We assume single-input switching while propagating waveform.

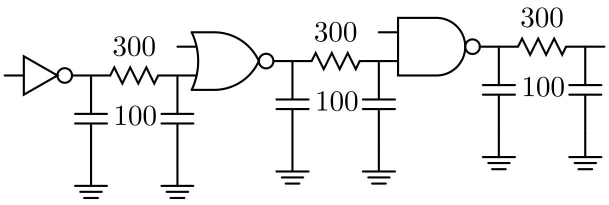


Figure 3.7: Test case for evaluating path delay. The units for resistance is Ohms (Ω) and the capacitors is femtoFarads(fF). Both the capacitors in the π-model have the same value.

We used the first $r = 2$ moments along with arrival time to propagate waveform. We compare our results with Spice and the waveforms at the output of NAND gate in Figure 3.7 is shown in Figure 3.8. The waveform predicted by the proposed method closely matches with the Spice result and the error in delay was less than 1%. The maximum error at any given time point was around 8%. and it is around the point at which the output waveform begins to rise ($t \approx 1600$ ps) in Figure 3.8.

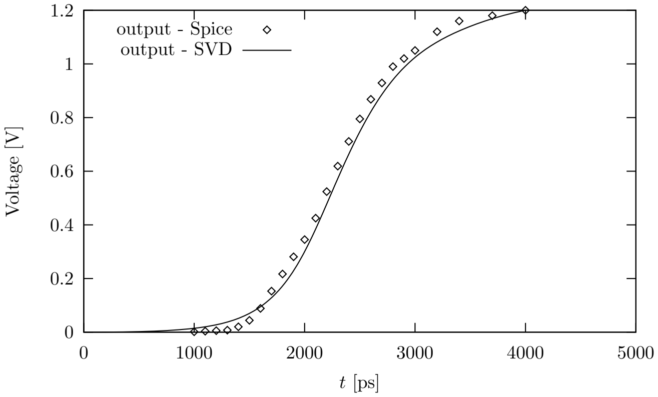


Figure 3.8: Waveform at the output of the NAND gate in Figure 3.7. The proposed SVD method and Spice output have a very close match.

3.8 Summary

This work presented a rigorous mathematical analysis of waveforms which led to a logical extension of current waveform modeling methods such as saturated-ramp model. The waveform models described in the work would find application in any situation where having a more detailed description of digital switching waveforms is useful. We demonstrated the application of waveform modeling to path-based STA demonstrating near Spice-like accuracy.

Chapter 4

Robust Analytical Gate Delay Modeling

In this chapter, we present an accurate closed form expression for gate delay. Apart from timing analysis, delay modeling is important to guide design optimization, such as transistor and gate sizing, interconnect optimization, placement, and routing. Closed form delay equations with sufficiently high accuracy are desirable since they are efficient and easy to implement. The alternative to closed form delay metrics are lookup tables. Though highly accurate, lookup tables less attractive since they are computationally expensive to use within an optimization loop [50]. The delay modeling consists of two distinct components, the gate and the interconnect delay modeling.

In the literature, significant attention has been devoted to *interconnect delay* characterization. The interconnects are often modeled as RC trees. The widely used Elmore delay is the first moment of the impulse response of the RC tree [51]. To improve the accuracy of the Elmore delay, models based on the higher order moment matching AWE [52] have been proposed. But AWE is expensive to use in optimization since it lacks closed-form expression. To improve the accuracy of Elmore delay and retain its simplicity, several works have proposed delay models that are functions of the higher moments of the impulse response of the RC tree [50, 53, 54]. Another fast approach is the matching of moments of the impulse response to a Probability Density Function (PDF) [55–58]. In the literature, the *gate delay* characterization has received less attention compared to interconnect delay characterization. The Sakurai-Newton (SN) delay approximation [6] is a widely used closed-form delay metric for the CMOS gates because of its simplicity and accuracy when gates operate in nominal voltages. The SN metric lacks accuracy when the CMOS gates operate at low supply voltages [59]. For the nanometer System-on-Chip (SoC) designs, due to the presence of multiple supply voltages the delay model needs to be robust across a wide range of operating scenarios.

In this paper, we propose a new, robust closed form gate delay metric based on the centroid of power dissipation. This new model is inspired by our key observation and theoretic proof that the SN metric can be viewed as the centroid of current dissipated by the gate. The proposed metric has a very high correlation coefficient (≥ 0.98) when correlated with the actual delays got from the HSPICE simulations. Such high correlation is consistent across all major process technologies. In comparison, the SN metric has a correlation coefficient between $(0.70, 0.90)$ depending upon the technology and the CMOS gate, and it is less accurate for lower supply voltages. Since our proposed metric has high fidelity across a wide range of supply voltages while retaining a simple closed form, it will be very useful to guide low voltage and low power designs.

To summarize, we make the following contributions:

- We show that the Elmore delay can be expressed as the centroid of dissipated current.
- We prove that the SN delay approximation is the exact Elmore delay of a CMOS gate.
- We propose a high fidelity closed form metric for the delay of a CMOS gate based on the centroid of the power dissipated by the gate.

4.1 Sakurai-Newton Delay Approximation

The Shockley model for MOSFET [60] fails in the short-channel region because it neglects velocity saturation effects. Sakurai and Newton proposed a model that takes into account the short-channel behavior while retaining the simplicity of the Shockley model [6,61]. They modified the quadratic dependence of the drain current on driving voltage to a α-power dependence, where $1 \leq \alpha \leq 2$ is the called the velocity saturation index.

The drain current i_D according to [6] is,

$$i_D = \begin{cases} \frac{k}{2}(v_{GS} - V_T)^{\alpha} & \text{saturation,} \\ \frac{k}{2}(v_{GS} - V_T)^{\alpha} \frac{v_{DS}}{V_{DS_{SAT}}} & \text{linear,} \\ 0 & \text{cutoff} \end{cases} \tag{4.1}$$

where

- $k = \left(\frac{W}{L}\right) \mu_n C_{ox}$, where μ_n is the mobility of electrons and C_{ox} is the oxide capacitance.

- $V_{DS_{SAT}}$ determines the boundary between linear and saturation regions when $v_{GS} = V_{DD}$.

For the delay *approximation* of the CMOS inverter, we assume a step input to the inverter. This enables us to extract the inherent delay of the gate ignoring the finite rise time of the input. The delay due to finite rise time can be incorporated into inherent delay due to step input using techniques such as PERI [62].

Since a step input is assumed, the drain current equation in (4.1) simplifies to,

$$i_D = \begin{cases} \frac{k}{2}(V_{DD} - V_T)^{\alpha} & V_{DD} - V_T < v_{DS} \leq V_{DD}, \\ \frac{k}{2}(V_{DD} - V_T)^{\alpha} \frac{v_{DS}}{V_{DD} - V_T} & v_{DS} \leq V_{DD} - V_T \end{cases} \tag{4.2}$$

where $(V_{DD} - V_T)$ is the boundary between linear and saturation regions under step input.

The main assumption in the delay approximation is that a constant saturation current I_{D0} discharges the output voltage from $v_{DS} = V_{DD}$ to $\frac{V_{DD}}{2}$.

$$t_{sn} = \frac{\Delta Q|_{\left(v_{DS} = V_{DD} \rightarrow \frac{V_{DD}}{2}\right)}}{I_{D0}} = \frac{C_L \frac{V_{DD}}{2}}{\frac{k}{2}(V_{DD} - V_T)^{\alpha}}$$

Thus the Sakurai-Newton (SN) delay metric is [6],

$$\boxed{t_{sn} \approx \frac{C_L V_{DD}}{k(V_{DD} - V_T)^{\alpha}}} \tag{4.3}$$

Note that this metric is an *approximation* to the delay since the transistor is assumed to be in *saturation* from $v_{DS} = V_{DD}$ to $\frac{V_{DD}}{2}$. The assumption is *weak*, since under the step input the transistor is in *saturation* region only from $v_{DS} = V_{DD}$ to $(V_{DD} - V_T)$. From $v_{DS} = (V_{DD} - V_T)$ to 0, the transistor is in *linear* region. In this paper,

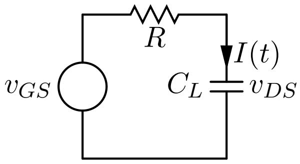


Figure 4.1: The RC model of an inverter. Note that R is a nonlinear resistor modeling transistor and C_L is the load capacitance seen by the inverter.

we model the transistor operating in saturation and linear regions as a nonlinear resistor R [59]. Thus the inverter can be modeled as an RC circuit [63] as shown in Figure 4.1. For an RC tree, the Elmore delay is an upper bound on the actual delay for any input waveform [64]. The theory behind the Elmore delay is discussed in the next section.

4.2 Centroid of Current Based Delay

In this section, we first show that the Elmore delay of a CMOS gate is the centroid of the current dissipated by it. Then we prove that the SN metric is the exact Elmore delay of the CMOS gate. This key observation will inspire us to propose a new delay metric in Section 4.3.

Lemma 1. *The Elmore delay of a CMOS gate is the centroid of the current dissipated by it when it is switching.*

Proof. The Elmore delay is defined as the centroid of the impulse response $h(t)$ of the system [65]. The centroid x_c of the function $f(x)$ is defined as,

$$x_c \quad = \quad \frac{\int_x x\, f(x)\, dx}{\int_x f(x)\, dx}$$

Thus the Elmore delay is given by,

$$t_{elmore} = \frac{\int_0^\infty t\,h(t)\,dt}{\int_0^\infty h(t)\,dt} \tag{4.4}$$

since $\int_0^\infty h(t)dt = 1$ for RC circuits with monotonic response [65] we can write (4.4) as,

$$t_{elmore} = \int_0^\infty t\,h(t)\,dt \tag{4.5}$$

Let $H(s)$ denote the Laplace transform of $h(t)$. The transfer function $H(s)$ is defined as the ratio of output to input voltages [66]. Since we assume a step input, the transfer function reduces to,

$$H(s) \quad = \quad \frac{V_{DS}(s)}{V_{GS}(s)} = \frac{V_{DS}(s)}{\frac{1}{s}} = sV_{DS}(s)$$

We apply the Inverse Laplace transform to get the impulse response, $h(t) = \frac{dv_{DS}}{dt}$. We know that the current discharged through the capacitor,

$$\begin{aligned} I(t) &= C_L \frac{dv_{DS}}{dt} \\ &= C_L h(t) \end{aligned}$$

Hence under the RC model with the assumption of step input,

$$I(t) \propto h(t) \tag{4.6}$$

$$t_{elmore} \quad = \quad \frac{\int_0^\infty t\,I(t)\,dt}{\int_0^\infty I(t)\,dt} \tag{4.7}$$

Thus the Elmore delay is shown as the centroid of the area under the current discharged through the load capacitor. □

We can now show the following result.

Theorem 3. *The Sakurai-Newton delay approximation is the exact Elmore delay of the CMOS gate under the following conditions:*

(i) A step input is applied;

(ii) The CMOS gate is modeled as an RC circuit.

Proof. We provide proof for the case when the gate is discharging. The proof for the case when the gate is charging is similar.

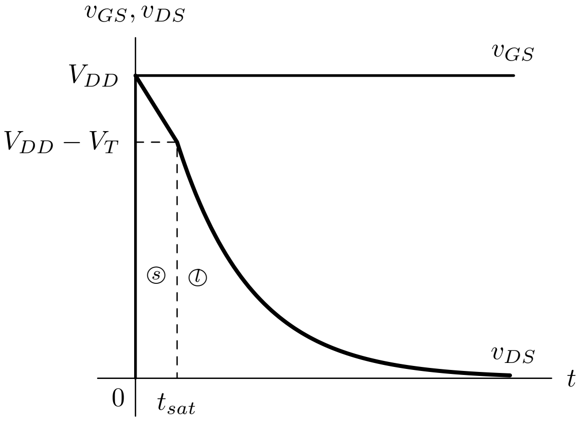


Figure 4.2: Inverter waveforms when the output is discharging. The input v_{GS} is a step input. The output v_{DS} decreases linearly in the saturation region (till t_{sat}) and decays exponentially in the linear region (after t_{sat}).

The input and output voltage waveforms associated with the discharging inverter are shown in Figure 4.2. When a rising step input ($v_{GS} = V_{DD}\, u(t)$) is applied to the inverter, the NMOS is on while the PMOS is off. The NMOS operates in the saturation region when the output discharges from $v_{DS} = V_{DD}$ to $(V_{DD} - V_T)$

and it operates in the linear region when the output discharges from $v_{DS} = (V_{DD} - V_T)$ to 0. The time taken by the output v_{DS} to reach $(V_{DD} - V_T)$ is denoted as t_{sat}, the time at which the NMOS transistor switches from saturation to linear region of operation.

The Elmore delay integral in (4.7) can be written as,

$$t_{elmore} \quad = \quad \frac{\int_0^{t_{sat}} t\, i_{D_{SAT}}\, dt + \int_{t_{sat}}^{\infty} t\, i_{D_{LIN}}\, dt}{\int_0^{t_{sat}} i_{D_{SAT}}\, dt + \int_{t_{sat}}^{\infty} i_{D_{LIN}}\, dt} \tag{4.8}$$

To evaluate (4.8), we need closed form expressions for $i_{D_{SAT}}$, $i_{D_{LIN}}$, and t_{sat}.

When the NMOS is saturated, the output voltage v_{DS} decreases linearly from V_{DD} to $(V_{DD} - V_T)$, shown as ⓢ in Figure 4.2. The decrease is linear because the current is a constant during that period which is given by,

$$i_{D_{SAT}} = \frac{k}{2}(V_{DD} - V_T)^{\alpha} \tag{4.9}$$

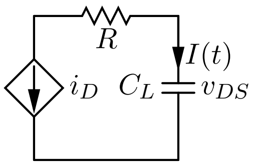


Figure 4.3: *RC* model with discharging current as a controlled current source.

When the output voltage v_{DS} goes below $(V_{DD} - V_T)$, the NMOS enters the linear region of operation, shown as ⓛ in Figure 4.2. The current in the linear region can be written as,

$$\begin{aligned} i_{D_{LIN}} &= k(V_{DD} - V_T)^{\alpha} \frac{v_{DS}}{V_{DD} - V_T} \\ &= \frac{v_{DS}}{R} \end{aligned}$$

where $\frac{1}{R} = k(V_{DD} - V_T)^{\alpha-1}$ is the resistance through which we discharge the load capacitor C_L as shown in Figure 4.3. We need an closed form expression for v_{DS} to evaluate $i_{D_{LIN}}$. The output voltage v_{DS} in the linear region is simply the voltage seen at the capacitor of a first order RC circuit under the step input. Thus the output voltage v_{DS} in the linear region can be written as,

$$v_{DS} = (V_{DD} - V_T)e^{\frac{-(t-t_{sat})}{RC_L}}u(t - t_{sat})$$

Thus the current during the linear region of operation can be written as,

$$i_{D_{LIN}} = k(V_{DD} - V_T)^{\alpha}e^{\frac{-(t-t_{sat})}{RC_L}}u(t - t_{sat}) \tag{4.10}$$

Finally we need t_{sat}, the time at which the NMOS switches from saturation to the linear region. Applying Kirchhoff current law to the output in Figure 4.3,

$$\begin{aligned} -C_L\frac{dv_{DS}}{dt} &= \frac{k}{2}(V_{DD} - V_T)^{\alpha} \\ -\int_{V_{DD}}^{V_{DD}-V_T} dv_{DS} &= \frac{\frac{k}{2}(V_{DD} - V_T)^{\alpha}}{C_L}\int_0^{t_{sat}} dt \end{aligned}$$

On integrating and simplifying we get,

$$t_{sat} = \frac{2C_LV_T}{k(V_{DD} - V_T)^{\alpha}} \tag{4.11}$$

Substituting the unknowns in (4.8), and evaluating the integrals we get,

$$t_{elmore} = \frac{\frac{C_L^2V_T^2}{k(V_{DD}-V_T)^{\alpha}} + \frac{C_L^2(V_{DD}^2-V_T^2)}{k(V_{DD}-V_T)^{\alpha}}}{C_LV_T + C_L(V_{DD} - V_T)}$$

$$\boxed{t_{elmore} = \frac{C_L V_{DD}}{k(V_{DD} - V_T)^{\alpha}}} \tag{4.12}$$

which is the same as (4.3). Thus the SN delay *approximation* is the *exact* Elmore delay of the CMOS gate. $\square$

In the nanometer technologies, the velocity saturation constant $\alpha \approx 1$. Thus (4.12) can be rewritten as,

$$t_{elmore} = \frac{C_L}{k\left(1 - \frac{V_T}{V_{DD}}\right)} \tag{4.13}$$

The SN metric (4.13) fails to track the delay when the supply voltages are low [59]. Taur and Ning [59] presented a simple curve fitting metric that works across a wide range of voltages. The Taur-Ning (TN) delay metric is given by,

$$\boxed{t_{tn} \propto \frac{C_L}{\left(0.7 - \frac{V_T}{V_{DD}}\right)}} \tag{4.14}$$

where 0.7 is a numerical fitting parameter. The TN metric suffers from the drawback of having higher absolute errors compared to the actual HSPICE delays. This is further discussed in Section 4.4. Another drawback is that it is applicable only when $\frac{V_T}{V_{DD}} \leq 0.5$ [59]. This means it may not be applied to very low V_{DD} designs.

4.3 Centroid of Power Based Delay

In this section, we derive a new metric based on the centroid of power (CP) which overcomes the drawbacks of the SN and TN delay metrics.

The SN metric can roughly be thought of as a charge-based delay since we integrate over current. The centroid of power can be thought of as an energy-based delay since we integrate over power. The delay obtained by taking the centroid of the power at the output can be written as,

$$t_{cp} = \frac{\int_0^{\infty} t\, v_{DS}\, i_D\, dt}{\int_0^{\infty} v_{DS}\, i_D\, dt} \tag{4.15}$$

Since the NMOS transistor is operating in two different regions, namely saturation and linear regions, (4.15) can be written as,

$$\begin{aligned} t_{cp} &= \frac{\int_0^{t_{sat}} t\, v_{DS_{SAT}}\, i_{D_{SAT}}\, dt + \int_{t_{sat}}^{\infty} t\, v_{DS_{LIN}}\, i_{D_{LIN}}\, dt}{\int_0^{t_{sat}} v_{DS}\, i_D\, dt + \int_{t_{sat}}^{\infty} v_{DS}\, i_D\, dt} \\ &= \frac{\frac{C_L^2(3V_{DD}-2V_T)V_T^2}{3k(V_{DD}-V_T)^{\alpha}} + \frac{C_L^2(V_{DD}+3V_T)(V_{DD}-V_T)^2}{4k(V_{DD}-V_T)^{\alpha}}}{\frac{1}{2}C_L(2V_{DD}-V_T)V_T + \frac{1}{2}C_L(V_{DD}-V_T)^2} \end{aligned}$$

which can be simplified to,

$$\boxed{t_{cp} = \frac{C_L(3V_{DD}^3 + 3V_{DD}^2V_T - 3V_{DD}V_T^2 + V_T^3)}{6kV_{DD}^2(V_{DD}-V_T)^{\alpha}}} \tag{4.16}$$

The correlation between the centroid of power (CP) delay metric and the HSPICE delay values is better than the correlation between the SN delay metric and the HSPICE delay values. The correlation attains near perfection with a modification in the Taur-Ning spirit [59].

We found out empirically that $\frac{1}{(V_{DD}-V_T)^2}$ tracks the delay better than $\frac{1}{V_{DD}^2}$. Substituting $(V_{DD}-V_T)^2$ for V_{DD}^2 in the denominator of (4.16), we get the modified centroid of power (CPM) metric,

$$\boxed{t_{cpm} \propto \frac{C_L(3V_{DD}^3 + 3V_{DD}^2V_T - 3V_{DD}V_T^2 + V_T^3)}{(V_{DD}-V_T)^2(V_{DD}-V_T)^{\alpha}}} \tag{4.17}$$

The correlation between the CPM delay metric and the HSPICE delay values is *almost perfect*. Also, the absolute error between the CPM metric and the HSPICE delay values is significantly lower when compared with the other metrics discussed in this paper. A possible reason for this near perfect tracking of delay is that the gate overdrive is proportional to $(V_{DD} - V_T)$ and not to V_{DD}. An alternative way to reason about this is the fact that $\frac{1}{(V_{DD}-V_T)^2}$ has a faster rate of change compared with $\frac{1}{V_{DD}^2}$ when V_{DD} varies.

4.4 Experimental Results

We used the Berkeley Predictive Technology Model [31] for our simulations. The simulations were run on the INV, NAND2, NOR2, XOR2 gates for their worst case input. The load capacitance C_L was varied from 20fF to 50fF. The supply voltage V_{DD} was varied from $2 \times V_{T0}$ to $6 \times V_{T0}$. The threshold voltage V_{T0} was varied within ±10% of its original value. The simulations were run on 45nm, 65nm, and 90nm technologies. Thus nearly 200 simulations were run on each gate for a given technology under its worst case input.

Table 4.1: The correlation of HSPICE delay values with the delay metrics across different technologies and gates. The HSPICE delay of a gate is measured for its worst case input combination.

Gate	45nm				65nm				90nm			
	SN	TN	CP	CPM	SN	TN	CP	CPM	SN	TN	CP	CPM
INV	0.76	0.97	0.81	0.99	0.76	0.95	0.82	0.99	0.90	0.99	0.94	0.98
NAND2	0.72	0.95	0.76	0.99	0.73	0.91	0.77	0.99	0.83	0.96	0.87	1.00
NOR2	0.73	0.96	0.78	0.99	0.75	0.92	0.80	0.99	0.90	0.99	0.93	0.99
XOR2	0.71	0.95	0.76	0.99	0.71	0.90	0.76	0.98	0.90	0.97	0.93	1.00

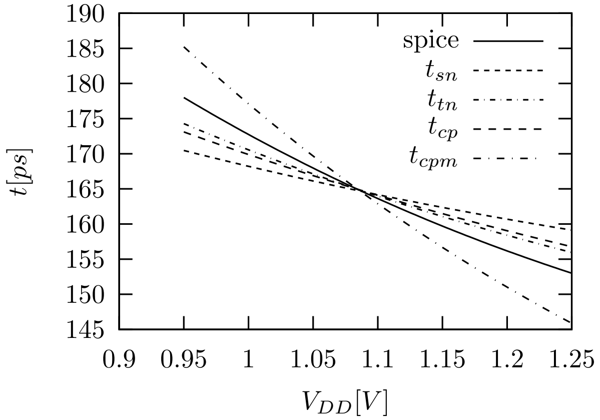


Figure 4.4: HSPICE delay and the values predicted by the delay metrics for INV in 65nm technology under *nominal* supply voltages. The solid line is the HSPICE delay values and the dotted lines are the delays predicted by the various metrics. The V_{DD} was varied with load capacitance $C_L = 20$fF and threshold voltage $V_{T0} = 0.22V$. Note that *all* the delay metrics track under *nominal* supply voltages.

The delay values predicted by the metrics were scaled by a constant value c. The constant c is obtained using linear regression. Suppose d_i is the delay obtained from HSPICE during the i th simulation and x_i is the delay predicted by the metric, c is obtained on minimizing $\sum_i (d_i - cx_i)^2$. Note that c changes as we take more samples of the parameters across a wider range. Thus a metric might be able to track the delay across small variations of supply voltage but it may not be able

Table 4.2: The percentage error between HSPICE delay values and the delay metrics across various technologies and gates. A *line* was fitted to the data points predicted by the delay metric. In this table the average min, max estimation error percentage is shown.

Gate	45nm (%)				65nm (%)			
	SN	TN	CP	CPM	SN	TN	CP	CPM
INV	$-161, 97$	$-41, 26$	$-139, 76$	$-14, 10$	$-94, 68$	$-37, 27$	$-78, 54$	$-9, 7$
NAND2	$-275, 137$	$-82, 45$	$-240, 112$	$-32, 14$	$-153, 91$	$-69, 43$	$-130, 76$	$-22, 11$
NOR2	$-209, 111$	$-59, 35$	$-181, 92$	$-19, 11$	$-112, 73$	$-50, 34$	$-96, 61$	$-13, 8$
XOR2	$-271, 141$	$-80, 47$	$-236, 115$	$-31, 15$	$-151, 94$	$-68, 45$	$-129, 79$	$-22, 13$

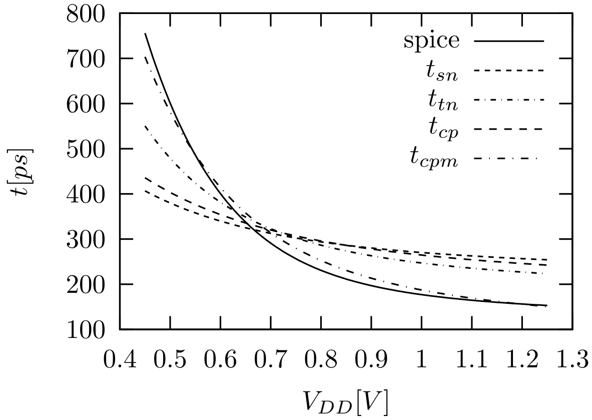


Figure 4.5: HSPICE delay and the values predicted by the delay metrics for INV in 65nm technology. The solid line is the HSPICE delay values and the dotted lines are the delays predicted by the various metrics. The V_{DD} was varied with load capacitance $C_L = 20$fF and threshold voltage $V_{T0} = 0.22V$. Note that only CPM can track the delay in the lower voltages while TN can track to quite an extent, the other two metrics SN and CP cannot track it.

to track delay under large variations of supply voltage. This is illustrated in the Figures 4.4 and 4.5.

In Figure 4.4, the CMOS gates operate under *nominal* supply voltages, $V_{DD} = 4 \times V_{T0}$ to $6 \times V_{T0}$ all the delay metrics correlate to HSPICE reasonably well. However, when the supply voltage drops below $V_{DD} = 4 \times V_{T0}$, only the CPM metric is able to track the delay well shown in Figure 4.5. The data is taken for an inverter in 65nm technology by varying the supply voltage V_{DD} from $2 \times V_{T0}$ to $6 \times V_{T0}$ and fixing the other circuit parameters.

The data obtained from other gates across various technologies and circuit parameters such V_{DD} and V_T yield similar results to Figure 4.5. There are two things to note in this figure:

1. The *correlation* measures the relative error. Intuitively, the relative error gives an estimate of how close the shape of the predicted delay curve is to the actual delay obtained from HSPICE simulations.
2. The *estimation error* gives the absolute difference between the predicted delay and the actual delay obtained from HSPICE simulations.

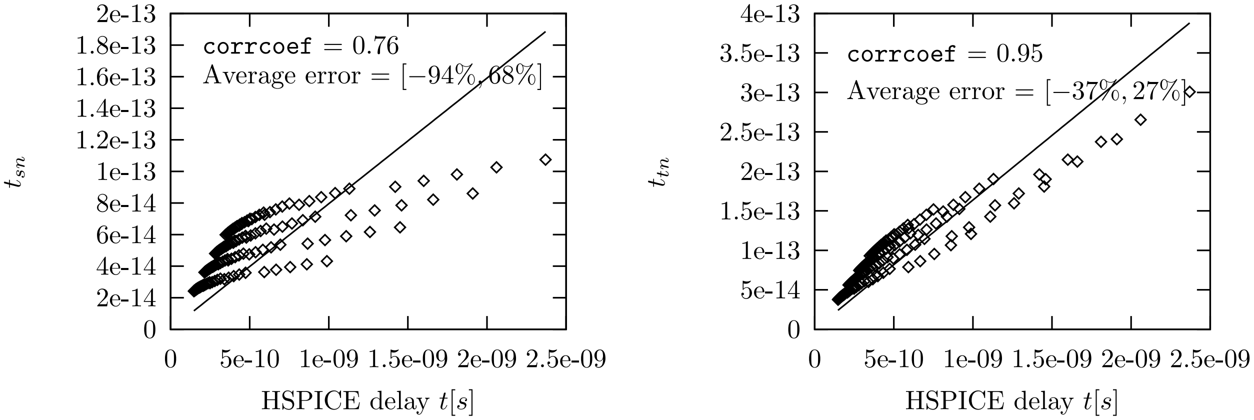
To visualize the performance of delay metrics with respect to the above two characteristics, we use a scatter plot. The scatter plot of different delay metrics versus the actual delay values for INV in 65nm technology is shown in Figure 4.6.

The data points are obtained by varying different circuit parameters. We fitted a line through the data points to obtain the constant of proportionality in the delay metrics. We then obtained the estimation error between the fitted line and the HSPICE delay values. The correlation is shown as `corrcoef` and the estimation error is shown as ‘Average error’ in the scatter plot. From the scatter plot it is clear that the CPM delay metric has the highest correlation and the lowest estimation error among all the delay metrics.

Table 4.1 summarizes the *correlation* coefficient of different delay metrics for various gates across the technologies. The correlation was taken between the actual HSPICE delays and the delay metric. From the table, we observe that the correlation coefficient of the CPM metric is consistently greater than 0.98, which is not exhibited by the other delay metrics. The estimation errors are tabulated in Table 4.2.

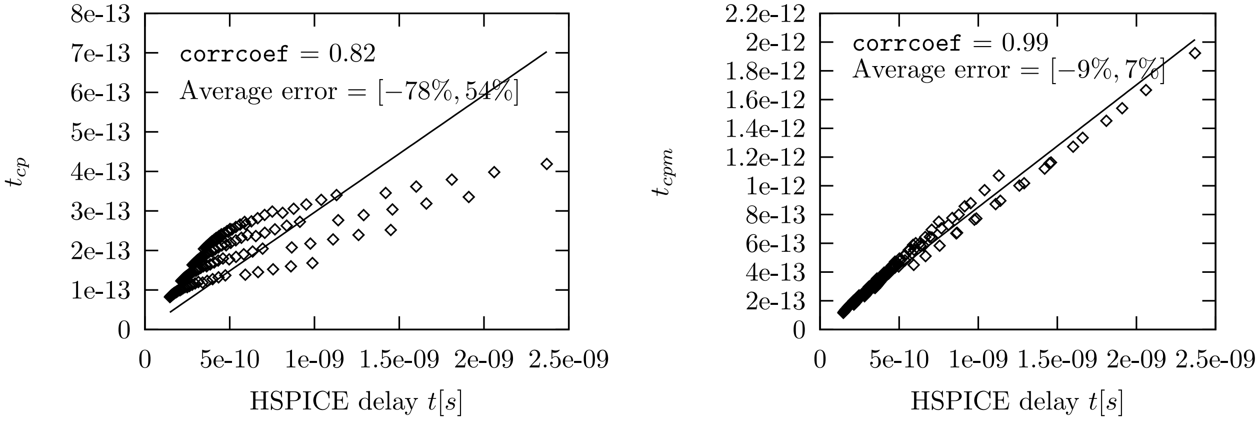
4.5 Summary

In this work, we proposed a new closed form delay metric based on the modified centroid of dissipated power. This new metric is inspired by our key observation that the SN delay can be viewed as the centroid of current. We also provide a theoretic proof that the SN delay is the Elmore delay of a CMOS gate when a gate is modeled as an RC circuit. The proposed metric has a very high correlation coefficient (≥ 0.98) when correlated with the actual delays got from the HSPICE simulations. Such high correlation is consistent across all major process technologies.



(a) Sakurai-Newton

(b) Taur-Ning



(c) Centroid of Power

(d) Modified Centroid of Power

Figure 4.6: Scatter plot of different delay metrics with the HSPICE delay for INV in 65nm technology. Since we have **not** multiplied by the constant of proportionality, no units are provided for the y-axis.

Chapter 5

Sleep Transistor Sizing

Till now we have focused on improving the accuracy in analyzing digital integrated circuits, with special emphasis on analyzing the timing of chip. Apart from accurately analyzing the timing of a chip, another major concern is improving its timing. One way to improve timing is to scale down the threshold voltage (V_{th}). But scaling down the threshold voltage (V_{th}) exponentially increases the subthreshold leakage current [4]. One of the techniques to reduce subthreshold leakage is power gating [7]. Power gating is a circuit technique in which the source nodes of the gates in the functional block which were grounded are connected to the drain of the NMOS sleep transistor. In the active mode, the sleep transistor is turned on to retain the functionality of the circuit. In the sleep mode, the sleep transistor is turned off, and the source nodes of the gates in the functional block float, thus cutting off the leakage path. Sleep transistor sizing is one of the major challenges in power gated circuits. If we overestimate the size we end up wasting silicon area and increasing the switching energy. If we underestimate the size, the required performance may not be achieved due to the increased resistance to the ground [7].

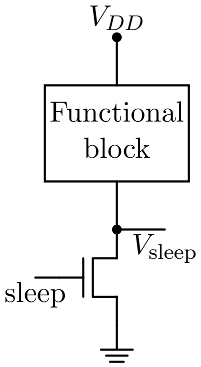


Figure 5.1: Power gating

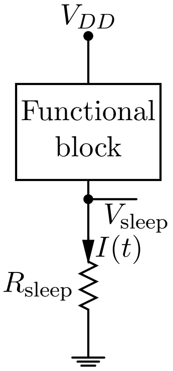


Figure 5.2: Sleep transistor as a resistor

In the literature, various methods have been proposed to size the sleep transistor. In [67], *module* based design was proposed where a single sleep transistor is used for the entire circuit. In [68], the circuit is partitioned into *clusters* to minimize the maximum simultaneous switching current. Each cluster has an individual sleep transistor. In [69], all the individual sleep transistors are wired together and the resulting mesh is called the *distributed* sleep transistor network (DSTN). The discharging current is shared by the sleep transistor network which reduces the size of the sleep transistor. The sizing in all the above methodologies is based on the *maximum* worst case switching current I_{peak} [69].

It was shown that the sleep transistor can be approximated by a linear resistor [7] that creates a finite voltage drop $V_{\text{sleep}} \approx R_{\text{sleep}} I(t)$, where $I(t)$ is the switching current through the sleep transistor as shown in Fig. 5.2. It is important to notice that different gates in a given path will see switching currents of *different magnitudes* through the sleep transistor. The delay of a gate is inversely proportional to the gate drive $V_{\text{GS}} = V_{\text{DD}} - V_{\text{sleep}} = V_{\text{DD}} - R_{\text{sleep}} I(t)$ [6]. To the first order, we can state that the penalty experienced by each gate due to the sleep transistor is proportional to the current $I(t)$ flowing through the sleep transistor when the gate is switching. Hence if we are able to estimate $I(t)$ efficiently, we can use $I(t)$ to size the sleep transistor instead of I_{peak} which penalizes different gates in a path uniformly.

In this work, we make the following contributions:

- An efficient method to estimate the temporal switching current $I(t)$ of the circuit.
- Sleep transistor sizing making use of timing criticality and temporal switching current $I(t)$ of the circuit, and

The results obtained indicate that our proposed technique results in area reduction of sleep transistors by 80% and 49% compared to module-based design and cluster-based design respectively.

5.1 Sizing the sleep transistor

The delay of a gate (τ_d) can be expressed as [6]:

$$\tau_d \propto \frac{C_\mathrm{L} V_\mathrm{DD}}{(V_\mathrm{DD} - V_{\mathrm{th}_L})^\alpha} \tag{5.1}$$

where C_L is the load capacitance at the gate output, V_{th_L} is the low threshold which is $0.7V$, $V_\mathrm{DD} = 3.3V$, and the velocity saturation index $\alpha \approx 1$ for $0.18\mu m$ CMOS technology.

The delay of a gate with the sleep transistor can be expressed as,

$$\tau_d^\mathrm{sleep} \propto \frac{C_L V_\mathrm{DD}}{((V_\mathrm{DD} - V_\mathrm{sleep}) - V_{\mathrm{th}_L})^\alpha} \tag{5.2}$$

where V_sleep is the potential of the virtual ground as shown in Fig. 5.1. Let $\tau_d^\mathrm{sleep} = (1+\Delta)\tau_d$, where $\Delta\tau_d$ is the penalty due to the sleep transistor. Applying Taylor series to the denominator and approximating the sleep transistor as a linear resistor R_sleep [7], the penalty can be written as,

$$\Delta\tau_d \propto \frac{V_\mathrm{sleep}}{V_\mathrm{DD} - V_{\mathrm{th}_L}}\tau_d = \frac{R_\mathrm{sleep} I(t)}{V_\mathrm{DD} - V_{\mathrm{th}_L}}\tau_d \tag{5.3}$$

where $I(t)$ is the switching current through the sleep transistor.

A path in a circuit consists of various gates and these gates experience discharging currents of different magnitudes. From Eq. (5.3), we find that the penalty for a gate due to the sleep transistor is proportional to $I(t)$. The delay penalty for a path ($\tau_\mathrm{penalty}^\mathrm{path}$) consisting of various gates can be written as,

$$\tau_\mathrm{penalty}^\mathrm{path} = \left(\frac{R_\mathrm{sleep}}{V_\mathrm{DD} - V_{\mathrm{th}_L}}\right) \sum_{\mathrm{gate} \in \mathrm{path}} I_\mathrm{local,max}\tau_d \tag{5.4}$$

where τ_d is the delay of the gate without the sleep transistor and $I_{\text{local,max}}$ is defined as,

$$I_{\text{local,max}} = \max_{[t_1,t_2]} I(t) \tag{5.5}$$

where $[t_1, t_2]$ is the time interval over which the gate switches. Notice that the $I_{\text{local,max}}$ is the maximum *local* temporal current over the discharging timing window of the gate. We *differ* from the previous methodologies in this respect since they use maximum *global* current I_{peak}. Rearranging Eq. (5.4),

$$R_{\text{sleep}} = \frac{(V_{\text{DD}} - V_{\text{th}_L})\, \tau_{\text{penalty}}^{\text{path}}}{\sum_{\text{gate}\in\text{path}} I_{\text{local,max}} \tau_d} \tag{5.6}$$

The current through the linearly-operating sleep transistor can be approximated as [68],

$$I_{\text{sleep}} \approx \mu_n C_{ox} \left(\frac{W}{L}\right)_{\text{sleep}} (V_{\text{DD}} - V_{\text{th}_L}) V_{\text{sleep}}$$

where μ_n is the mobility of electrons and C_{ox} is the oxide capacitance. Since the sleep transistor is operating in the linear region, then $R_{\text{sleep}} \approx \frac{V_{\text{sleep}}}{I_{\text{sleep}}}$. Then, the size of the sleep transistor can be written as,

$$\left(\frac{W}{L}\right)_{\text{sleep}} = \frac{1}{\mu_n C_{ox} (V_{\text{DD}} - V_{\text{th}_L}) R_{\text{sleep}}} \tag{5.7}$$

Thus if R_{sleep} is known, the W_{sleep} can be determined directly. To determine R_{sleep}, we need an estimate of the temporal current flowing through the sleep transistor. The temporal current estimation technique is described next.

5.2 Temporal Current Estimation

We present a technique to estimate the worst case current discharged by a circuit. The current estimation technique requires timing windows of each gate, and the current expected to be discharged by each gate. The timing windows are obtained using *PrimeTime* [70]. The expected discharge current (I_{exp}) of a gate is adapted from [68] and the pseudocode is shown below:

```
FIND-EXPECTED-CURRENT(gate)
Find $I_{\text{peak}}$ for each gate in the library using HSPICE
$I_{\text{exp}} = \alpha_s \times I_{\text{peak}}$                ▷ $\alpha_s$ is the switching factor
return $I_{\text{exp}}$
```

The switching factor α_s is defined as the probability of the output (Y) switching. Thus α_s for falling output is,

$$\alpha_s = P\{Y = 1 \rightarrow 0 | Y = 1\} \times P\{Y = 1\}$$

We illustrate I_{exp} calculation using OR2. The switching factor $\alpha_s = \frac{1}{4} \times \frac{3}{4} = \frac{3}{16}$. From HSPICE simulations, we find that the $I_{\text{peak}} = 0.72ma$. Thus the expected current for an OR2 gate in our library is, $I_{\text{exp}} = \alpha_s \times I_{\text{peak}} = \frac{3}{16} \times 0.72ma = 0.12ma$.

After calculating I_{exp} for all the gates in our library we can use it for estimating switching current of a circuit. The pseudocode is presented below:

```
ESTIMATE-SWITCHING-CURRENT(circuit)
Run PrimeTime on the circuit to get timing windows
$I(t) \leftarrow 0$
for every gate in the circuit
    do $I_{\mathrm{exp}} \leftarrow$ GET-EXPECTED-CURRENT(gate)
       ▷ Illustrated in Fig. 5.4
       $I_{gate}(t) \leftarrow$ timing windows bounded by $I_{\mathrm{exp}}$
       $I(t) \leftarrow I(t) + I_{gate}(t)$          ▷ Illustrated in Fig. 5.5
return $I(t)$
```

We bound both falling and rising timing windows by the falling I_{exp}. The assumption is safe, since for any gate, the worst case falling current through ground is always bigger than the short circuit current when the output rises. The switching factor α_s is the same for both falling and rising transitions.

To illustrate the current estimation procedure, consider the 1-bit carry lookahead adder (CLA) shown in Fig. 5.3. The timing analyzer *PrimeTime* is run on this circuit to obtain the timing windows shown in Table 5.1. Fig. 5.4 shows the currents associated with each timing window. To illustrate reading Fig. 5.4, consider the OR2 gate O_1 in Fig. 5.3. The falling window for O_1 from Table 5.1 is $[73.92, 260.11]ps$. The I_{exp} of O_1 is $0.12ma$. Thus we have bounding rectangle of current $0.12ma$ over $[73.92, 260.11]ps$ as shown in Fig. 5.4. Finally, the currents across all the timing windows are summed up to find the total discharging current of 1-bit CLA shown in Fig. 5.5. Once the temporal switching current $I(t)$ has been estimated we can use that current to size the sleep transistor.

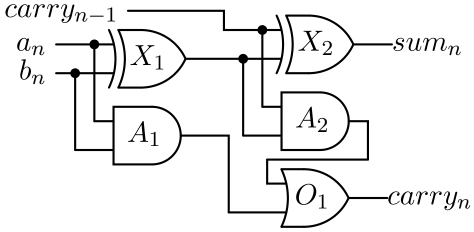


Figure 5.3: 1-bit CLA

Table 5.1: Timing windows for 1-bit CLA (time unit is ps)

Gate	$rise_{\min}$	$rise_{\max}$	$fall_{\min}$	$fall_{\max}$
X_1	98.90	107.52	104.24	183.28
A_1	51.52	56.33	42.82	43.01
X_2	83.22	290.10	132.75	277.42
A_2	58.31	168.80	46.43	232.62
O_1	123.60	234.09	73.92	260.11

5.3 Timing Criticality Based Sizing

When a sleep transistor is inserted in a circuit, the performance of the circuit is penalized by the reduction in driving voltage as evident in the Eq. (5.2). At a macroscopic level, this penalizes the paths. Thus if we can guarantee that the worst case path in the circuit, with sleep transistor switched on, satisfies the performance constraints, then we can guarantee the performance of all the paths in the circuit.

There are two potential problems in sizing the sleep transistor based on paths. First, the number of paths in a circuit is exponential in size. Second, the worst case path for CMOS need not be the worst case path in MTCMOS [67]. To overcome the above two problems, we use a heuristic from static timing analysis (STA). The path based STA uses the top K worst paths to generate the circuit delay, the maximum of all path delays [70]. This idea is adapted to sleep transistor sizing, and the pseudocode is shown below.

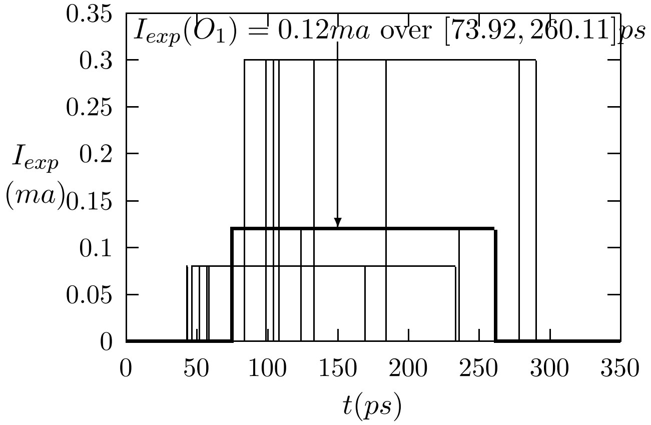


Figure 5.4: I_{exp} bounding the *falling* and *rising* timing windows of each gate (Table 5.1) in a 1-bit CLA. Refer to ESTIMATE-SWITCHING-CURRENT line 5

```
SIZE-SLEEP-TRANSISTOR($circuit, K$)
1  Run PrimeTime on the circuit to get critical paths
2  $R_{\text{sleep}} \leftarrow \infty$
3  for path ← 1 to $K$                    ▷ Size using top $K$ critical paths
4        do $R_{\text{path}} \leftarrow$ Size using Eq. (5.6)
5           $R_{\text{sleep}} \leftarrow \text{MIN}(R_{\text{sleep}}, R_{\text{path}})$
6  $\left(\frac{W}{L}\right)_{\text{sleep}} \leftarrow$ Size using $R_{\text{sleep}}$ in Eq. (5.7)
7  return $\left(\frac{W}{L}\right)_{\text{sleep}}$
```

To illustrate the sizing procedure, consider one of the worst case paths in the 1-bit CLA as shown in Table 5.2. τ_d in Table 5.2 is the delay experienced by each gate without the sleep transistor. $I_{\text{local,max}}$ in Eq. (5.6) differs for each gate in the path and it is obtained by looking up $I(t)$. For example, $I_{\text{local,max}}$ for X_2 is the maximum current discharged in the range $[183.28, 277.42]ps$. As shown in Fig. 5.5, the maximum current that flows in the above range is $I_{\text{local,max}}(X_2) = 0.92ma$.

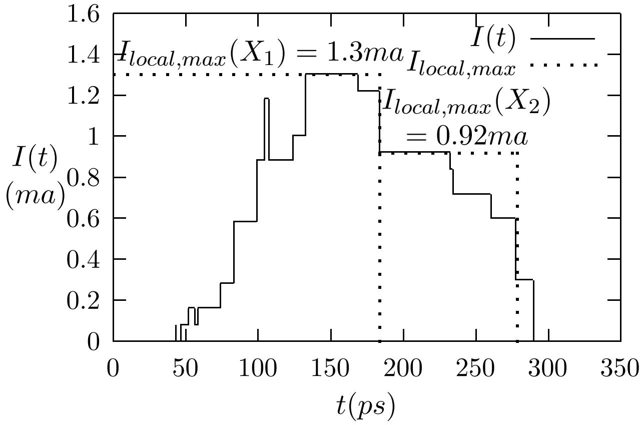


Figure 5.5: The estimated current discharge $I(t)$ of a 1-bit CLA got by summing up all the currents in Fig. 5.4. Refer to ESTIMATE-SWITCHING-CURRENT line 6. Also shown are the local maximum currents seen by the gates X_1 and X_2. Refer to Eq. (5.5). Note that by using local maximum instead of global maximum we reduce the size of the sleep transistor

Note that we are using a *local* maximum to bound the current instead of the global maximum that is used in previous methodologies. The above procedure is repeated for all gates in the path. Fig. 5.5 shows the local maximum currents seen by the gates X_1 and X_2 of the 1-bit CLA in Fig. 5.3.

Table 5.2: A worst case path in 1-bit CLA

Gate	$\tau_d(ps)$	$\tau_d^{\text{path}}(ps)$	fall/rise
X_1	183.28	183.28	fall
X_2	94.13	277.42	fall
$t_{arrival}$		277.42	

To illustrate the calculation of R_{path}, consider the path through X_1 and X_2. Let the penalty be 5% of the delay $(0.05 \times t_{arrival})$.

$$R_{\text{path}} = \frac{(3.3 - 0.7)\,(0.05 \times 277.42ps)}{183.28ps \times 1.3ma + 94.13ps \times 0.92ma} = 111\Omega$$

To illustrate the calculation of W_{sleep} we will assume the above R_{path} as the minimum resistance R_{sleep} obtained.

$$\left(\frac{W}{L}\right)_{\text{sleep}} = \frac{1}{1.25 \times 10^{-4}(3.3 - 0.7)111} = 27.77\lambda$$

Let $L_{\text{sleep}} = 2\lambda$ and we get $W_{\text{sleep}} = 55.55\lambda \approx 56\lambda$, where $\lambda = 0.1\mu m$.

An important observation is that only the *falling inverting* gates are affected by the NMOS sleep transistor. Thus we penalize only the falling inverting gates in a path. Since the non-inverting gates in our library are a series combination of the inverting gate and the inverter, only the *rising non-inverting* gates are penalized.

5.4 Results

The proposed sleep transistor sizing methodology has been implemented and its results are presented for various benchmark circuits. We use $0.18\mu m$ CMOS technology with $V_{DD} = 3.3V$, $V_{T_L} = 0.7V$, and $V_{T_H} = 0.9V$. L_{sleep} is set to $0.2\mu m$. The number of paths used to size the sleep transistor is set to $K = 100$ since $K > 100$ did not make any significant difference to the sizing.

In Table 5.3 under Module column, we compare our proposed module based sizing with module based sizing [67]. We obtain a sleep transistor area improvement of 80% on an average over [67], since the proposed methodology has a global objective of satisfying performance for every path of the circuit while module based

Table 5.3: Comparison of W_{sleep} obtained using module and cluster based design for 5% performance degradation. The unit is $\lambda = 0.1\mu m$.

Circuit	Module (λ)		Cluster (λ)	
	[67]	Proposed	[68]	Proposed
CLA4	825	125	204	127
Parity checker	960	235	369	284
Wallace tree	1365	427	1201	698
c432	3438	475	1272	385
c499	3840	1171	2094	1351

methodology has a restrictive local objective of satisfying performance for every gate. This coupled with the usage of $I(t)$ instead of I_{peak} leads to vast improvements in sizing.

In Table 5.3 under Cluster column, we compare our proposed cluster based sizing with cluster based sizing [68]. We cluster such that the critical path is entirely within a single cluster. To discuss the results for clustering, we need to define *slack*. The slack in cluster c_j (S_{c_j}) for 5% performance penalty is defined as, $S_{c_j} = 1.05 \times CP_{circuit} - CP_{c_j}$, where $CP_{circuit}$ is the critical path in the entire circuit and CP_{c_j} is the critical path in cluster c_j. Suppose the entire circuit is divided into two clusters c_1 and c_2. Let c_2 contain the critical path which implies c_1 has more slack. This slack can be exploited to size the sleep transistor even smaller in c_1. Since we also size based on $I(t)$ instead of I_{peak} we obtain an sleep transistor area improvement of of 49% on an average over [68].

In Table 5.4, we compare the sizes obtained using the proposed module and proposed cluster method. The circuits in Table 5.4 have gate counts numbering a few thousand, and have unbalanced paths. The presence of unbalanced paths is

Table 5.4: Comparison of W_{sleep} obtained using *proposed* methods for 5% performance degradation. The unit is $\lambda = 0.1\mu m$.

Circuit	Proposed (λ)	
	Module	Cluster
c880	638	509
c1908	479	457
c3540	1979	1933
c7552	12955	8325

ideal for clustering as discussed earlier with regard to slack. The results validate our intuition that clustering is better for bigger circuits.

The results were verified with HSPICE simulations using random input vectors and also using input vectors that exercise top K critical paths.

5.5 Summary

We have introduced a new path based methodology to size sleep transistors using temporal currents and timing windows. We have also proposed an efficient method to estimate the temporal switching current $I(t)$ of the circuit. The results obtained indicate that our proposed technique results in area reduction of sleep transistors by 80% and 49% compared to module based design and cluster based design respectively.

Chapter 6

Power Grid Analysis using Behavioral Modeling of Transistors

As technology is scaled aggressively the increase in transistor density leads to an increase in power density in a chip. Power consists of two components: leakage power and dynamic power. The dynamic power is directly proportional to the number of transistors switching at a given time. As transistors switch they consume power from the power supply grid, leading to fluctuation in supply voltage from its constant value. This voltage fluctuation, also called a voltage transient, decreases a circuit's performance, since lower the voltage, worse is the performance. An important challenge is knowing how to estimate the voltage transients in the power supply grid efficiently. In this chapter, we focus on analyzing these voltage transients and this is called the power grid *analysis*. The solution techniques currently available for power grid analysis rely on a model of representing the transistor network as a current source [71]. This simplification enables decoupling the transistor network from the power grid. The most significant advantage of this simplification is that the power grid problem mathematically reduces to solving a linear system of equations. Thus we can apply techniques from numerical linear algebra to solve the linear system of equations arising from the power grid [72–79]. Hierarchical analysis has also been used to analyze power grid [80, 81]. In hierarchical analysis, the power grid is partitioned and a macromodel is created for each partition. The macromodel makes the problem of analyzing large power grids tractable. Since the deterministic techniques mentioned above solve the entire system, they are not suitable for incremental analysis. The need for incremental analysis gave rise to stochastic techniques such as random walk [82–84]. Stochastic techniques have also been applied to study the effect of process variations on the power grid [85–88]. When a transistor switches on and connects to the grid, the charge that is supplied to the transistor comes from the capacitors nearby. This *locality* effect has been exploited to design fast algorithms [89, 90]. Methods have also been proposed for power grid analysis in the context of floorplanning [91, 92].

In the literature reviewed so far, the nonlinear transistor network is modeled as a current source which results in a linear system of equations. But this modeling might lead to pessimism in the voltage drop prediction. In Figure 6.1, we illustrate a voltage drop at a node in a power grid. In the current source model, we replace the transistor network with time-varying current sources which model the switching current drawn from the grid. The current source model does not accurately model the decoupling capacitances provided by the PMOS transistors which are already on and currently not switching. This leads to pessimism in the voltage drop predicted.

There is a difference of $0.025V$ in the voltage drop predicted in one of our benchmark circuits.

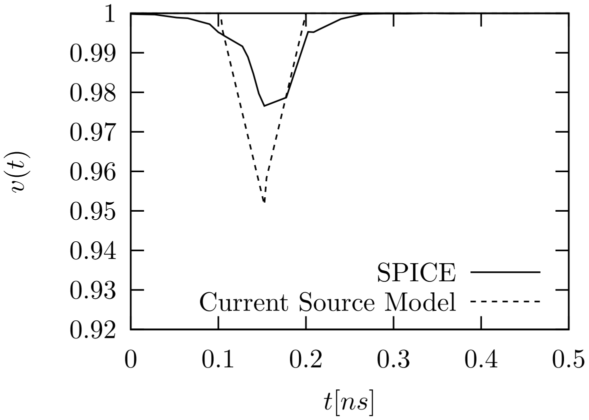


Figure 6.1: The voltage drop at a node from SPICE simulation and the current source based approach. There is a difference of $0.025V$ in the voltage drop predicted.

We summarize the disadvantages in modeling the transistor network as a current source:

1. When a PMOS transistor switches and connects to the power grid, some of the charge is supplied by the PMOS transistors that are already *on*. These PMOS transistors which are already *on*, act much like a “decoupling capacitance” in the power grid. By ignoring or incorrectly modeling this *local charge sharing* effect, the designer is likely to overestimate the amount of decoupling capacitance needed. This leads to wastage of power and silicon area. The *local charge sharing* effect is not correctly captured in the current source model.

2. The number of transistors that get switched *on* differs from cycle to cycle. This implies that the amount of capacitance seen by the power grid also varies from cycle of cycle. The *time-varying capacitance* is not captured in the current source model.

The *modeling of transistor network* with respect to the power grid has received little attention. In this work, *we focus on the modeling aspect* of the transistor network which results in an accurate power grid analysis.

The contributions of this work are summarized below:

- We analyze the power grid by modeling the transistor network accurately instead of replacing the transistor network with a time-varying current source.
- The transistor is modeled as a simple switch in series with a RC circuit. The switch is modeled *behaviorally* as a Norton current source model. The behavioral modeling of the switch is the key element in making the proposed simulation efficient. Note that modeling the switch as a PWL resistor leads to convergence problems associated with abrupt non-linearities [93].
- The proposed model offers the middle ground between the accuracy of SPICE simulation and the speed of the current source model.

It should be noted that we have adapted techniques from the literature but the overall flow is original.

6.1 Power Grid Preliminaries

We adapt the power grid modeling described in [71] where the power grid is modeled as a passive Linear Time Invariant (LTI) network consisting of resistors, inductors, and capacitors. Since the ground grid is symmetrical we restrict our analysis to the power grid alone [94]. The power grid can be described using the Modified Nodal Analysis (MNA) formulation [95]:

$$\mathbf{G}V(t) + \mathbf{C}\frac{dV(t)}{dt} = I(t) \tag{6.1}$$

where

- $\mathbf{G} \in \mathbb{R}^{m\times m}$ is the conductance matrix which depends on the topology of the circuit. Also, m denotes the number of nodes in the power grid and the transistor network.
- $\mathbf{C} \in \mathbb{R}^{m\times m}$ is the admittance matrix resulting from capacitive and inductive elements.
- $V(t) \in \mathbb{R}^{m}$ is a time-varying vector of voltages at the nodes.
- $I(t) \in \mathbb{R}^{m}$ has two kinds of rows [74]:
  1. Rows with positive V_{DD} value corresponding to the nodes connected to voltage sources;
  2. Rows with 0, correspond to all other nodes.

Since we restrict our attention to the voltages, we ignore the KCL equations around the voltage sources. If all the voltage sources are grounded, this results in the conductance matrix $\mathbf{G}$ which is positive definite and it can be shown to be a $\mathcal{M}$-matrix [74]. This gives rise to efficient methods for solving the linear system [29,96].

The ordinary differential equation in Eq. (6.1) can be solved in time domain using Backward Euler technique [95]. We use the Norton current source as the associated discrete circuit (ADC) model for both capacitor and inductor. This is because Thevenin's voltage source is not suited for MNA [95] since for every voltage source we need an extra row in the conductance matrix.

On applying ADC to the energy-storage elements, we get:

$$\left(\mathbf{G} + \frac{\mathbf{C}}{h}\right) V(t+h) = I(t+h) + \frac{\mathbf{C}}{h} V(t) \tag{6.2}$$

where h is the time-step taken in the transient simulation.

If we fix the time-step h to be a constant, then the matrix $\left(\mathbf{G} + \frac{\mathbf{C}}{h}\right)$ turns out to be a constant for the entire duration of simulation. This leads to greater efficiency in the transient solve since we need just one LU factorization of $\left(\mathbf{G} + \frac{\mathbf{C}}{h}\right)$ for the entire transient simulation and the cost can be amortized over many runs of the transient simulation. The nodal voltages at each time point in the transient is got by a Forward-Backward Solve (FBS) which is $O(m^2)$ compared to $O(m^3)$ for a direct solve [33].

6.2 Transistor Network Modeling

In this section, we describe our transistor network modeling. We differ from the literature by modeling the transistor not as time-varying current source but as a simple RC circuit [97]. The transistor is connected to the power grid through a switch as shown in Figure 6.2.

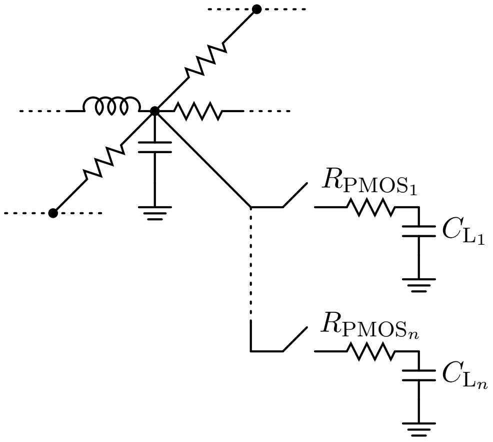


Figure 6.2: Transistor is modeled a simple switch in series with a RC circuit. Note that if a transistor gets switched on to the grid node, some of the charge will come from transistors which are already *on* which is not captured in previous models.

The advantage of this modeling is that it can accurately capture the self-loading effects of the transistor and the charge-sharing among the switching transistors. But the major disadvantage of this modeling is that based on whether the switch is on or off, the topology changes, leading to a different $\left(\mathbf{G}+\frac{\mathbf{C}}{h}\right)$ matrix during every switching instant. This makes the conductance matrix non-constant and it will make the transient simulation inefficient. To make the matters worse, if we have k transistors modeled as switches then potentially we have 2^k different topologies [98]. This fact is illustrated by building conductance matrices for the circuits in Figure 6.3 and Figure 6.4 which differ only in the state of the switch.

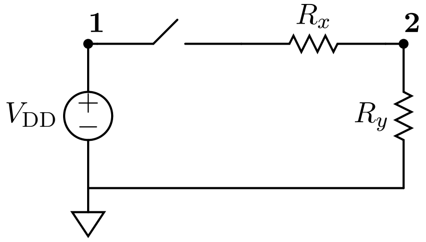


Figure 6.3: Network having an open switch. The topology due to an open switch is different when compared to a closed switch in Figure 6.4. This leads to the conductance matrix $\mathbf{G}_{\text{open}}$.

The MNA matrix equations corresponding to the circuit in Figure 6.3 containing an open switch is given by

$$\begin{array}{c c} & \begin{array}{cc} \mathbf{1} & \mathbf{2} \end{array} \\ \begin{array}{c} \mathbf{1} \\ \mathbf{2} \end{array} & \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{R_y} \end{pmatrix} \end{array} \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} V_{\text{DD}} \\ 0 \end{pmatrix}$$

$$\mathbf{G}_{\text{open}} V = I \tag{6.3}$$

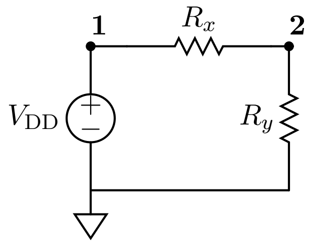


Figure 6.4: Networks having a closed switch. The topology due to a closed switch is different when compared to an open switch in Figure 6.3. This leads to the conductance matrix $\mathbf{G}_{\text{close}}$.

The MNA matrix equations corresponding to the circuit in Figure 6.4 containing a closed switch are given by

$$\begin{array}{cc} & \begin{array}{cc} \mathbf{1} & \mathbf{2} \end{array} \\ \begin{array}{c} \mathbf{1} \\ \mathbf{2} \end{array} & \begin{pmatrix} 1 & 0 \\ -\frac{1}{R_x} & \frac{1}{R_x} + \frac{1}{R_y} \end{pmatrix} \end{array} \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} V_{\mathrm{DD}} \\ 0 \end{pmatrix}$$

$$\mathbf{G}_{\text{close}} V = I \tag{6.4}$$

Note that we drop the KCL equations around the voltage node **1** since we are interested only in the voltage at any given node. It is clear from Eq. (6.3) and Eq. (6.4), depending on whether the switch is closed or open we get a different conductance matrix **G**.

In the next section, we describe a technique to keep the conductance matrix $\left(\mathbf{G} + \frac{\mathbf{C}}{h}\right)$ in the MNA formulation in Eq. (6.2) a *constant*, irrespective of the state of the switches and thus get back the efficiency achieved by having a constant conductance matrix.

6.3 Behavioral Switch Modeling

In this section, we describe a discrete-time approximation to the ideal switch which models the switch behaviorally. This renders the conductance matrix $\left(\mathbf{G} + \frac{\mathbf{C}}{h}\right)$ in the MNA formulation in Eq. (6.2) a *constant*, irrespective of the state of the switches.

6.3.1 Ideal switch

The ideal switch shown in Figure 6.5 has zero resistance when *on* and infinite resistance when *off*. Also the ideal switch can move from one to state to another instantaneously. This change in resistance causes a change in topology and hence we get different conductance matrices.

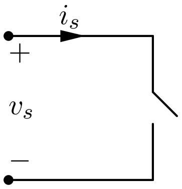


Figure 6.5: Ideal switch.

But from the view of simulation, the behavior of the ideal switch (s) can be captured by the following equations:

$$s = \text{open} \Leftrightarrow i_s = 0 \tag{6.5}$$

$$s = \text{short} \Leftrightarrow v_s = 0 \tag{6.6}$$

This *behavioral modeling of switch* is a key in achieving efficiency in power grid simulation.

6.3.2 ADC for an approximate switch model

The industry standard circuit simulators like SPICE use a two-valued resistor for modeling the transistor switch. But this leads to convergence problems and long execution times for stiff networks [99]. For an efficient simulation with switches, we just need to model the behavior of the switch as captured in Eq. (6.5) and Eq. (6.6).

We now describe Approximate Discrete Circuit (ADC) for an approximate model of switch. This was independently developed by Hui and Morrall [100], and by Pejović and Maksimović [101] in 1994, motivated by the switching power system simulations.

Before describing the ADC formally, it will be instructive to understand what makes the conductance matrix a constant irrespective of the state of the switches. To illustrate, consider the circuits in Figure 6.3 and Figure 6.4. We would like to have the same conductance matrix for both the circuits since they differ only in the state of the switch.

Let the switch be modeled by a voltage source (v_s) in series with a finite resistance (r_s) as shown in Figure 6.6.

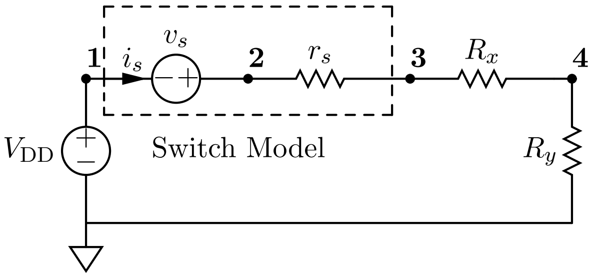


Figure 6.6: Modeling a switch with a voltage source. By varying the value of the voltage source, we can simulate the *on* or *off* behavior of the switch. Since we are changing only the value of the voltage source, the conductance matrix remains the same irrespective of the state of the switch.

Applying KVL to the circuit in Figure 6.6, we get,

$$-V_{\mathrm{DD}} - v_s + i_s(r_s + R_x + R_y) = 0 \tag{6.7}$$

To simulate the switch being *open* ($s = 0$) we need $i_s = 0$ as in Eq. (6.5). This can be easily achieved by setting $v_s = -V_{\mathrm{DD}}$. Similarly, to simulate the switch being *short* ($s = 1$), we set $v_s = 0$. Thus by changing the value of the voltage source we can simulate the behavior of a switch without changing the topology of the circuit. This is the intuition behind the switch modeling.

The above intuition is formalized by writing out the MNA equations for the circuit in Figure 6.6.

$$\begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \end{array}\!\!\begin{array}{c} \begin{array}{cccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -\frac{1}{r_s} & \frac{1}{r_s} + \frac{1}{R_x} & -\frac{1}{R_x} \\ 0 & 0 & -\frac{1}{R_x} & \frac{1}{R_x} + \frac{1}{R_y} \end{pmatrix} \end{array} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix} = \begin{pmatrix} V_{\mathrm{DD}} \\ v_s \\ 0 \\ 0 \end{pmatrix} \tag{6.8}$$

By setting v_s to 0 or $-V_{\mathrm{DD}}$ in the RHS of Eq. (6.8) we can simulate the switch?s being on or off. Note that the conductance matrix on LHS remains a constant irrespective of the state of the switches.

6.3.3 Current source based ADC for switch

Since MNA lends itself more naturally to a current source compared to a voltage source [95], we use a current source instead of a voltage source to simulate the state of a switch. The ADC for an approximate switch model is shown in Figure 6.7. The

ADC of an approximate switch can be thought of as a linearized, discrete equivalent circuit of a nonlinear resistor [101].

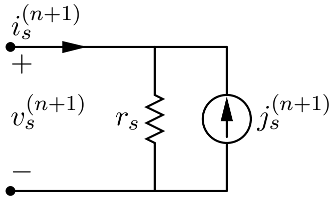


Figure 6.7: Associated Discrete Circuit (ADC) of an approximate switch model. The superscript $(n + 1)$ refers to the simulation step.

The state of the switch is captured by changing the value of the current source $j_s^{(n+1)}$. This is similar to changing the value of voltage captured in Eq. (6.7). In our simulations, we modeled the switch as a current source as shown in Figure 6.7. The behavioral model of the switch is given by [101]:

$$j_s^{(n+1)} = \begin{cases} -i_s^{(n)} & \text{if } s^{(n+1)} = 0, \\ \frac{v_s^{(n)}}{r_s} & \text{if } s^{(n+1)} = 1. \end{cases} \tag{6.9}$$

Note that we use $\frac{v_s^{(n)}}{r_s}$ when $s^{(n+1)} = 1$ instead of 0 as we had discussed in the example above. But when the switch is closed there is almost no voltage drop across the switch. Hence $v_s^{(n)} \approx 0$ and thus consistent with our intuition. We adopt this notation to ensure consistency with the power electronics literature. The *efficiency of our proposed algorithm* compared to SPICE depends on this behavioral modeling of the switch. We show that the algorithms previously presented in the literature can be applied to our new model without significant change. This is done by proving that the resultant conductance matrix $\mathbf{G}$ is a $\mathcal{M}$-matrix.

6.3.4 Conductance matrix is a $\mathcal{M}$-matrix

Definition 6.3.1 ($\mathcal{M}$-matrix). A matrix $\mathbf{A} \in \mathbb{R}^{m\times m}$ is an $\mathcal{M}$-matrix if it satisfies the following conditions: (1) $a_{ii} > 0 \quad \forall i$; (2) $a_{ij} \leq 0 \quad \forall i \neq j$; (3) $a_{ii} \geq \sum_{j\neq i} |a_{ij}| \quad \forall i$; (4) $a_{ii} > \sum_{j\neq i} |a_{ij}|$ for at least one i;

Theorem 4. *The conductance matrix $\mathbf{A} = (\mathbf{G} + \frac{\mathbf{C}}{h})$ obtained by modeling switch by its ADC is an $\mathcal{M}$-matrix.*

Proof. We need to demonstrate that the conductance matrix $\mathbf{A}$ satisfies all the four conditions in Definition 6.3.1.

The first condition is $a_{ii} > 0 \quad \forall i$. This is satisfied due to the MNA stamping [95]. The diagonal entries of $\mathbf{A}$ is given by

$$a_{ii} = \sum_{j \in \text{nodes}} g_{ij} + \frac{c_i}{h} \tag{6.10}$$

By definition, a node is a junction of at least two elements. This implies the RHS of Eq. (6.10) has at least two distinct entries. Since the conductances and capacitances are positive values, we can conclude that the first condition $a_{ii} > 0 \quad \forall i$ holds for our conductance matrix $\mathbf{A}$.

The second condition is $a_{ij} \leq 0 \quad \forall i \neq j$. This is again satisfied due to the MNA stamping?in other words, it is correct by construction. The only exception comes when there is a voltage source connected to the node. KCL needs to account for the current through the voltage source (i_{v_s}). In MNA stamping, it turns out that the coefficient of i_{v_s} is +1 causing the second condition to fail. Since we are interested only in the voltages at a given node, this situation is avoided by ignoring the KCL around the node connected to the voltage source.

The third condition is $a_{ii} \geq \sum_{j \neq i} |a_{ij}| \quad \forall i$. This follows from Eq. (6.10).

$$\begin{aligned}
a_{ii} &= \sum_{j \in \text{nodes}} g_{ij} + \frac{c_i}{h} & \\
&= \sum_{j \in \text{nodes}} |g_{ij}| + \frac{c_i}{h} & \text{(Since } g_{ij} \text{ is positive, } g_{ij} = |g_{ij}|) \\
&\geq \sum_{j \in \text{nodes}} |g_{ij}| & \geq \sum_{j \neq i} |g_{ij}| \\
&= \sum_{j \neq i} |a_{ij}| & \text{(Since } a_{ij} = g_{ij} \quad \forall i \neq j) \qquad (6.11)
\end{aligned}$$

The fourth condition is $a_{ii} > \sum_{j \neq i} |a_{ij}|$ for at least one i. We need at least one voltage source for our power grid. In our MNA construction, the KCL equations around the voltage source nodes are dropped. We can state the MNA construction corresponding to the voltage source mathematically as follows. If there is a voltage source at node i then $a_{ii} = 1, \quad a_{ij} = 0 \quad \forall i \neq j$. The RHS corresponding to this row is set to V_{DD}. Thus the fourth condition directly follows from this construction. ☐

It should be noted that it is straightforward to make **A** a symmetric matrix. The matrix is not symmetric due to the fact that we drop the KCL around the voltage source node. The technique to make **A** symmetric is illustrated by an example [74]. Consider the circuit shown in Figure 6.4. The MNA matrix equations corresponding to the circuit in Figure 6.4 are reproduced below for convenience.

$$\begin{array}{c c} & \begin{array}{cc} \mathbf{1} & \mathbf{2} \end{array} \\ \begin{array}{c} \mathbf{1} \\ \mathbf{2} \end{array} & \begin{pmatrix} 1 & 0 \\ -\frac{1}{R_x} & \frac{1}{R_x} + \frac{1}{R_y} \end{pmatrix} \end{array} \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} V_{\text{DD}} \\ 0 \end{pmatrix}$$

Since $V_1 = V_{\mathrm{DD}}$, we can rewrite our system of equations as:

$$\begin{array}{c} \\ \mathbf{1} \\ \\ \mathbf{2} \end{array} \begin{array}{c} \begin{array}{cc} \quad\mathbf{1} & \qquad\mathbf{2} \end{array} \\ \begin{pmatrix} 1 & 0 \\ & \\ 0 & \frac{1}{R_x} + \frac{1}{R_y} \end{pmatrix} \end{array} \begin{array}{c} \\ \begin{pmatrix} V_1 \\ \\ V_2 \end{pmatrix} \end{array} \begin{array}{c} \\ = \end{array} \begin{array}{c} \\ \begin{pmatrix} V_{\mathrm{DD}} \\ \\ \frac{V_{\mathrm{DD}}}{R_x} \end{pmatrix} \end{array}$$

Now, the conductance matrix turns out to be a symmetric matrix. Note that the conductance matrix is still an $\mathcal{M}$-matrix even after its transformation to a symmetric matrix.

6.4 Speedup Techniques

In this section, we discuss speedup techniques to improve the runtime of the proposed modeling. To get an intuition behind these techniques, it is instructive to look into the phases that occur in power grid simulation. There are two phases which occur in power grid simulation and they are shown in Figure 6.8 which depicts the phases stylistically. Please note that the graph is not drawn to scale.

1. *Local charge redistribution.* This happens when the PMOS transistor gets switched on to the power grid. Most of the charge is supplied by the local capacitors.

2. *Global recovery phase.* This happens when the power supply starts supplying charge to the capacitors. It brings back all the capacitors to their original state of being fully charged.

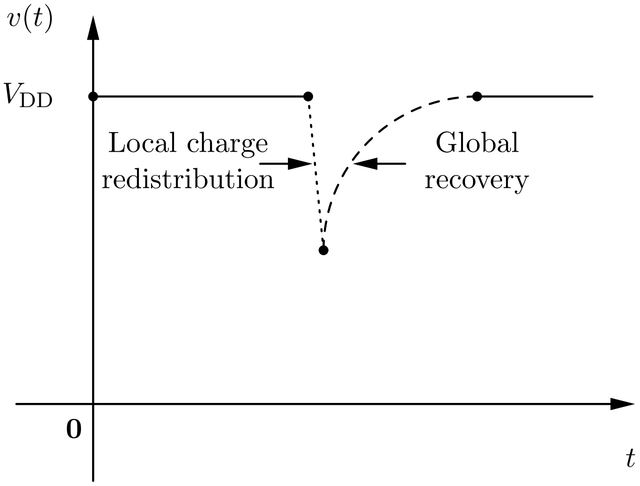


Figure 6.8: The figure is *not* drawn to scale. This is a stylistic depiction of the 2 phases that occur in power grid simulation. There are two phases, the first phase (dotted) is local charge redistribution and the second phase (dashed) is global recovery.

6.4.1 Local charge redistribution phase

The major drawback of this proposed modeling is that the time-step (h) in Eq. (6.2) is decided by the fast transients during the switching of transistors which cause a voltage drop in the grid almost instantaneously (*local charge redistribution*). Since we need to track these transients accurately, the time-step (h) is set to tens of picoseconds. When the power grid recovers after this fast transient voltage drop, we can track the voltages well with a time-step in the range of hundreds of picoseconds. Thus if we can calculate this voltage drop using a fast approximation without doing a detailed simulation then we can use a bigger time-step for the rest of the simulation. This implies a faster runtime for the proposed power grid modeling.

The researchers in the power electronics community have addressed this issue of charge redistribution [93]. But they assume instantaneous charge redistribution by ignoring the resistances. This is not a good approximation in the power grid problem. We need an approximation which also takes into account the resistances of the power grid.

The idea behind the approximate method to calculate the voltage drop is that when a transistor switches on at a grid, most of the charge comes from the capacitors nearby (*local charge redistribution*). The recovery of the capacitors nearby to their original fully charged state is due to charge being supplied from the nearby voltage sources (*global recovery phase*).

The voltage drop is due to *local charge redistribution.* The idea of *locality* is exploited to devise a fast method to calculate the drop. It has been shown that using the first two shortest paths (in terms of impedance) from the node to the voltage source to calculate the drop gives an approximation within 10% of the SPICE results [102]. Increasing the number of shortest paths will lead to a better approximation of the voltage drop.

But the drawback of this method is that it works only if the switching events are isolated. Though we do not define *isolated switching event* formally in this work, it can be stated informally as follows. A switching event is said to be isolated if there are no switching events in its locality. Since our proposed modeling results in a non-linear system, we cannot use superposition when switching events occur simultaneously in nearby power grid nodes. We defer the voltage drop calculation during simultaneous switching to future work.

6.4.2 Global recovery phase

A simplifying assumption in scheduling gate switching helps speedup the global recovery phase. We assume that all gates that can switch in a given cycle, switch during the positive edge of the cycle. For example, consider an inverter chain (inv1 − inv4 − inv16) hooked to the same power grid node. If inv1 is switched on during a cycle, inv16 also gets switched on but after some finite delay. But in our model, we switch on both inv1 and inv16 at the same time. This makes our voltage drop to be near the positive edge of the clock. Since our goal is to observe the dynamics at power grid nodes over several cycles, we are not concerned about the exact time at which the voltage drop occurs.

Thus in *global recovery phase*, observe that once grid recovers back to the supply voltage in a given cycle it is going to stay at the supply voltage. This is because all the gates that were scheduled to be switched during a given cycle get switched during the start of the cycle. Thus the power grid simulation can be fast-forwarded to the start of the next cycle once all the nodes recover back to the supply voltage.

6.5 Overall Algorithm

The proposed power grid simulation with the behavioral modeling of transistor is given in Algorithm 3.

6.6 Experimental Results

We implemented our model and algorithm using a combination of Awk/Perl/Matlab scripts since demonstrating the concept is our primary intent. If implemented in C/C++ with well tuned data structures and code, the speedup obtained compared to SPICE will be much greater than the current implementation. We report the results of experiments run on random benchmarks [103]. The experiments were done using a 32-bit Linux machine with 4 GB RAM and running at 3.4 GHz. The delay models were generated using the $90nm$ Berkeley Predictive Technology Model [31].

The results are presented in Table 6.1. The error in our proposed model is very small compared to the current source based model. Note that the error in voltage drop predicted by current source model for ckt9 compared to ckt1 is bigger. This is because in ckt9, there are more transistors hooked to the power grid node compared to ckt1. Hence the decoupling capacitance provided by the PMOS transistors which are on, is much bigger in ckt9 compared to ckt1. Since the current source model does not properly model the drain capacitance provided by the PMOS transistors which are on, there is a higher error in voltage drop predicted for ckt9 compared to ckt1.

While solving $\mathbf{Ax} = \mathbf{b}$ we employed simple LU factorizations rather than the specialized algorithms for solving $\mathcal{M}$-matrix presented in the literature. Thus an implementation using compiled language and specialized algorithms for solving $\mathbf{Ax} = \mathbf{b}$, would lead to a greater speedup.

The speedup technique discussed in Section 6.4.2 was also implemented. This technique improved runtime by nearly an order of magnitude. If the switching is isolated, then we can use the speedup technique discussed in Section 6.4.1 which will improve the runtime.

6.7 Summary

We have proposed a new model for power grid simulation while retaining the features that make power grid simulation amenable to linear algebraic methods presented in the literature. The proposed model can be seamlessly integrated into the existing power grid models. It is more accurate compared to the current source model and

Table 6.1: Runtime over 10 cycles for random circuits. The drop predicted by both the proposed and the current source model are pessimistic. (Cycle-time = $750ps$)

ckt	Nodes	Runtime [s]		Speedup	Error^{e}[%]	
		Prop^{p}	SPICE		CS^{c}	Prop^{p}
1	3658	4.86	153.41	31.57×	8.5	2.2
2	3958	5.70	165.19	28.98×	8.6	0.5
3	4258	6.00	179.26	29.88×	9.7	1.1
4	4558	6.56	188.70	28.77×	10.9	0.55
5	4858	7.13	199.07	27.92×	13.1	1.2
6	5158	7.80	205.73	26.38×	12.4	0.2
7	5458	8.37	215.99	25.81×	13.6	0.15
8	5758	8.95	234.18	26.17×	14.8	0.25
9	6058	9.57	252.39	26.37×	15.4	1.1

[e] Error [%] is given by $\frac{V_{\text{predicted}} - V_{\text{SPICE}}}{V_{\text{SPICE}}} \times 100$

[p] Proposed Model

[c] Current Source Model

it also retains the efficiency of the current source model by employing a constant conductance matrix. The proposed model offers the middle ground between the accuracy of SPICE simulation and the speed of the current source model.

Algorithm 3 Power-Grid-Simulation

Input: Transistor representation of the blocks
Input: Input switching patterns at the primary inputs of the block
Input: *RC* representation of the power grid
Input: h = time-step, set by the fastest transient in *local charge redistribution*
Output: Power grid voltage at various user specified points on the power grid

```
// (Section 6.2)
Model transistor as a switch connected to a $RC$ circuit.
// (Section 6.3)
Model switch by its ADC in Eq. (6.9).
Generate the conductance matrix for the power grid model as in Eq. (6.2).
while time $<$ time-stop do
  if positive edge of clock then
    // Conductance matrix in LHS of Eq. (6.2) is
    // constant irrespective of the state of switches
    Update RHS in Eq. (6.2) based on the state of switches. Please note that
    the state of switch denotes whether the PMOS transistor is on or off.
    // Speedup technique for local charge redistribution
    // (Section 6.4.1)
    if switching events are isolated then
      Use approximate methods to calculate the voltage drop due to the local
      charge redistribution.
      Update time.
    end if
  end if
  // use any efficient $\mathcal{M}$-matrix solver
  Solve for unknown voltages in Eq. (6.2).
  Update ADC of energy-storage elements.
  time = time + $h$.
  // Speedup technique for global recovery phase
  // (Section 6.4.2)
  if all the nodes have recovered to supply voltage then
    Fast forward time to the start of next cycle.
  end if
end while
```

Chapter 7

Thermal Analysis Considering Nonlinear Thermal Conductivity

In this chapter we focus on analyzing the temperature profile of a chip accurately. The motivation comes from the fact that as technology is scaled aggressively the packing density as well as power density in a chip increases. The increase in power density leads to increase in temperature on the chip. The increase in temperature in turn creates many challenges,

1. At higher temperatures electron mobility decreases due to the phonon scattering effect. The decrease in electron mobility leads to increased gate delay and hence a lower clock frequency.

2. Reliability of transistors is exponentially dependent on the operating temperature of the junction. A difference of $10 - 15°$ C can result in nearly 2× difference in the lifespan of the devices [104].

3. The leakage has super-linear dependency on the temperature. A difference of $30°$ C will affect the leakage by 30% [105].

Thus the temperature directly affects performance, reliability, and power. Hence it is important to accurately simulate the thermal profile of a chip. Although the electrothermal simulation for analog circuits [106] is a well studied problem, the techniques are not applicable for simulating the thermal profile of a VLSI chip. This is because these electrothermal analyses are done in a SPICE-like manner, and hence do not scale well making them unsuitable for the thermal simulation of a VLSI chip. Therefore, new methods have been proposed in the literature for the thermal analysis of VLSI circuits.

The simulation of the thermal profile essentially involves solving the heat equation. The heat equation is a partial differential equation (PDE) with boundary conditions which on discretization leads to a system of equations. The system of equations is *linear* if the thermal conductivity of the chip layers is assumed to be a constant. The early thermal simulators for VLSI chips solved the linear system directly [107]. With the increase in packing density, the number of variables to solve in the linear system has increased. Model order reduction was used to reduce the system of equations to solve and thereby increase computational efficiency [108]. To increase the efficiency of the transient solve, the ADI method was used [109]. The linear system resulting from discretizing the heat equation is a symmetric positive definite matrix, called the M-matrix. Multigrid techniques can be used to accelerate solving the M-matrix and have been used for efficient full-chip thermal analysis [110].

Thermal simulators have provided the technology that spawned many thermal based applications. Applications have been proposed in the areas of leakage analysis [105], reliability [111] among many others.

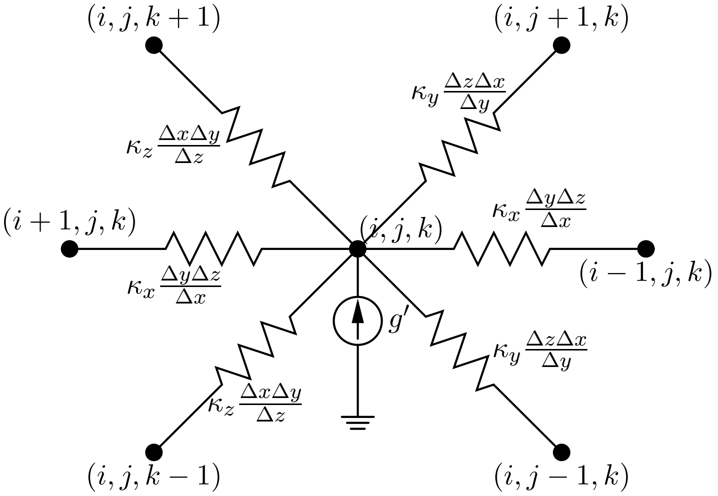
The *efficiency* of the solutions proposed in literature depends on the assumption that thermal conductivity is a constant, which leads to a linear system of equations. But thermal conductivity is a *nonlinear* function of temperature, and for silicon it varies by 22% over the range $27-80^\circ$ C [8]. Thus, ignoring the nonlinearity of the thermal conductivity might lead to a difference of temperature by 10° C as shown in Section 7.3. To get an accurate thermal profile, it is therefore important to consider the nonlinear dependence of the thermal conductivity on temperature. This work addresses the challenge of solving the nonlinear system of equations efficiently. A fast algorithm which is a variant of the Newton-Raphson technique is proposed to solve the nonlinear system of equations.

7.1 Thermal Modeling and Temperature Simulation

A general 3D thermal analysis involves solving the heat conduction equation which is discussed in detail in [105]. On discretizing the heat condition equation using finite differences and assuming steady-state conditions, the heat equation can be written as

$$\begin{aligned}\kappa_x(T_{i,j,k})\frac{\Delta y\Delta z}{\Delta x}&\left(2T_{i,j,k}-T_{i-1,j,k}-T_{i+1,j,k}\right)\\&+\kappa_y(T_{i,j,k})\frac{\Delta z\Delta x}{\Delta y}\left(2T_{i,j,k}-T_{i,j-1,k}-T_{i,j+1,k}\right)\\&+\kappa_z(T_{i,j,k})\frac{\Delta x\Delta y}{\Delta z}\left(2T_{i,j,k}-T_{i,j,k-1}-T_{i,j,k+1}\right)\\&=\Delta x\Delta y\Delta z\times g(x,y,z)\end{aligned}\tag{7.1}$$

where $\kappa_x(T_{i,j,k})\frac{\Delta y \Delta z}{\Delta x}$ is interpreted as the thermal conductance in the x direction. The electrical interpretation of Eq. (7.1) is shown in Figure 7.1.


$(i, j, k+1)$
$(i, j+1, k)$
$\kappa_y \frac{\Delta z \Delta x}{\Delta y}$
$\kappa_z \frac{\Delta x \Delta y}{\Delta z}$
$(i+1, j, k)$
(i, j, k)
$\kappa_x \frac{\Delta y \Delta z}{\Delta x}$
$(i-1, j, k)$
$\kappa_x \frac{\Delta y \Delta z}{\Delta x}$
g'
$\kappa_y \frac{\Delta z \Delta x}{\Delta y}$
$\kappa_z \frac{\Delta x \Delta y}{\Delta z}$
$(i, j, k-1)$
$(i, j-1, k)$

$$g' = \Delta x \Delta y \Delta z \times g(x, y, z)$$

Figure 7.1: Electrical interpretation of the 3-d heat equation. Note that the thermal conductivities k_x, k_y and k_z are functions of temperature $T_{i,j,k}$

Generalizing Eq. (7.1) leads to a matrix formulation

$$\mathbf{K}(\mathbf{T})\mathbf{T} = \mathbf{g} \tag{7.2}$$

similar to the circuit relation $\mathbf{GV} = \mathbf{i}$, thus capturing the analogy of voltage with temperature and current with heat sources. The challenge lies in solving this nonlinear system of equations efficiently. A fast algorithm to solve this nonlinear system is proposed in the next section.

7.2 A Fast Algorithm to solve $\mathbf{A}(\mathbf{x})\mathbf{x} = \mathbf{b}$

The nonlinear system of equations, $\mathbf{A}(\mathbf{x})\mathbf{x} = \mathbf{b}$, needs to be solved efficiently to make the problem of nonlinear thermal conductivity practical.

The widely used iterative solver for the nonlinear system of equations is the Newton-Raphson method. The Newton-Raphson iteration can be expressed as

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - [\mathbf{J}(\mathbf{x}^{(k-1)})]^{-1}\mathbf{f}(\mathbf{x}^{(k-1)}) \tag{7.3}$$

where

$$\begin{aligned} \mathbf{J}(\mathbf{x}) &= \frac{\partial(\mathbf{A}(\mathbf{x})\mathbf{x} - \mathbf{b})}{\partial \mathbf{x}} \\ \mathbf{f}(\mathbf{x}) &= \mathbf{A}_{\mathbf{i}.}(\mathbf{x})\mathbf{x} - \mathbf{b} \end{aligned}$$

and $\mathbf{A}_{\mathbf{i}.}(\mathbf{x})$ denotes the row i of the matrix $\mathbf{A}$.

In each iteration of the Newton-Raphson in Eq. (7.3), the Jacobian needs to be inverted. In Figure 7.2, note that the tangent is evaluated during every iteration. This tangent evaluation is equivalent to finding the inverse of a Jacobian matrix ($\mathbf{J} \in \mathbb{R}^{m \times m}$) in each iteration when solving the thermal circuit.

Evaluating a matrix inverse has an asymptotic complexity $O(n^3)$. Considering that the sizes of matrices in thermal simulation are in the order of tens of thousands, this is a time consuming operation. This motivates the need for a variant of the Newton-Raphson algorithm. Since the evaluation of the Jacobian inverse is the bottleneck in Newton-Raphson, the Jacobian inverse is evaluated once during the first iteration and it is used in every iteration thereof. The Newton-Raphson

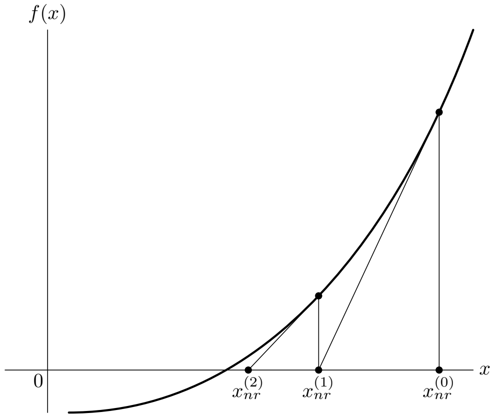


Figure 7.2: Newton-Raphson iteration. Note that the tangent is evaluated during every iteration. This is equivalent to finding the inverse of a Jacobian matrix ($\mathbf{J} \in \mathbb{R}^{m\times m}$) for every iteration when solving the thermal circuit.

iteration with a constant Jacobian inverse can be expressed as

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - [\mathbf{J}(\mathbf{x}^{(0)})]^{-1}\mathbf{f}(\mathbf{x}^{(k-1)}) \tag{7.4}$$

where $\mathbf{J}(\mathbf{x}^{(0)})$ is the Jacobian evaluated during the initial guess. The iteration in Eq. (7.4) is called the *constant Jacobian.* This will work if the initial guess is close to the final solution. If the initial guess is random, the solver may output nonphysical temperatures, or worse, it may not converge. Since the temperatures on the chip cannot go lower than the room temperature and are usually in the range $[300, 400]$ K, the initial guess is set to the room temperature 300 K. In constant Jacobian, the slope of the tangent is the same for every iteration which leads to slower convergence. To accelerate the constant Jacobian, an approximation to the Jacobian named *reduced order* Jacobian is proposed and its algorithmic details are described next.

7.2.1 Evaluating the reduced order Jacobian

Let $\mathbf{x}^{(0)}, \ldots , \mathbf{x}^{(k-1)}$ be the first k iterations when solving the equation $\mathbf{A}(\mathbf{x})\mathbf{x} = \mathbf{b}$. Then $\mathbf{x}^{(k)}$ can be approximately calculated by fitting a plane through the vectors $\mathbf{x}^{(0)}, \ldots , \mathbf{x}^{(k-1)}$. The plane fitting is described next.

Let the vector $\mathbf{x}^{(j)}$ be divided into p partitions $\mathbf{x}_i^{(j)}, i = 0, \ldots , p - 1$, where $k = p + 1$. This procedure is repeated for all the vectors $\mathbf{x}^{(j)}$, $j = 0, \ldots , k - 1$. Let the norm error be defined as $\epsilon = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|$. The error $\epsilon_i^{(j)}$ can be thought of error in the partition i during the iteration j. The idea is to fit a plane through the vectors $\mathbf{x}^{(0)}, \ldots , \mathbf{x}^{(k-1)}$, such that the error in the $k + 1$ iteration $\epsilon^{(k)}$ goes to 0. Thus the following system of linear equations needs to be solved

$$\begin{pmatrix} \epsilon_0^{(0)} & \cdots & \epsilon_0^{(k-1)} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots \\ \epsilon_{p-1}^{(0)} & \cdots & \epsilon_{p-1}^{(k-1)} \end{pmatrix} \begin{pmatrix} \alpha^{(0)} \\ \ldots\ldots \\ \alpha^{(k-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ . \\ 0 \end{pmatrix} \tag{7.5}$$

At first, this looks like a trivial solution. But

$$\sum_{j=0}^{k-1} \alpha^{(j)} = 1 \tag{7.6}$$

This implies that the RHS is non-zero and a non-trivial solution exists. Once $\alpha^{(j)}$ are determined, the $\mathbf{x}^{(k)}$ can be found using

$$\mathbf{x}^{(k)} = \alpha^{(0)}\mathbf{x}^{(0)} + \ldots + \alpha^{(k-1)}\mathbf{x}^{(k-1)} \tag{7.7}$$

The above equation can be thought of as a *reduced order* Jacobian and can be rewritten as a recursion

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \underbrace{\begin{bmatrix} \mathbf{x}^{(k-1)} - \mathbf{x}^{(0)} \\ \cdots \\ \mathbf{x}^{(k-1)} - \mathbf{x}^{(k-2)} \end{bmatrix}^{\top} \begin{pmatrix} \epsilon_0^{(k-1)} - \epsilon_0^{(0)} & \cdots & \epsilon_0^{(k-1)} - \epsilon_0^{(k-2)} \\ \multicolumn{3}{c}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots} \\ \epsilon_{p-1}^{(k-1)} - \epsilon_{p-1}^{(0)} & \cdots & \epsilon_{p-1}^{(k-1)} - \epsilon_{p-1}^{(k-2)} \end{pmatrix}^{-1}}_{\textit{reduced order } \text{Jacobian}} \times \begin{pmatrix} \epsilon_0^{(k-1)} \\ \cdots\cdots \\ \epsilon_{p-1}^{(k-1)} \end{pmatrix} \tag{7.8}$$

The intuition for the *reduced order* Jacobian can be got by considering the number of partitions, $p = 1$. Thus at least $k = p + 1 = 2$ iterations are needed before *reduced order* Jacobian algorithm can be applied. Recall that the norm error for iteration j is defined as $\epsilon^{(j)} = \|\mathbf{A}\mathbf{x}^{(j)} - \mathbf{b}\|$. The system of equations in Eq. (7.5) reduces to

$$\begin{aligned} \alpha^{(0)}\epsilon_0^{(0)} + \alpha^{(1)}\epsilon_0^{(1)} &= 0 \\ \alpha^{(0)}\epsilon_0^{(0)} + (1 - \alpha^{(0)})\epsilon_0^{(1)} &= 0 \qquad \text{using Eq. (7.6)} \\ \alpha^{(0)} &= \frac{-\epsilon_0^{(1)}}{\epsilon_0^{(0)} - \epsilon_0^{(1)}} \end{aligned}$$

Thus $\mathbf{x}^{(2)}$ can be predicted as,

$$
\begin{aligned}
\mathbf{x}^{(2)} &= \alpha^{(0)}\mathbf{x}^{(0)} + \alpha^{(1)}\mathbf{x}^{(1)} \\
&= \alpha^{(0)}\mathbf{x}^{(0)} + (1 - \alpha^{(0)})\mathbf{x}^{(1)}
\end{aligned}
$$

The graphical illustration of the algorithm is shown next by considering a 1-dimensional ($m = 1$) root finding. Since the number of equations to be solved is one, the number of partitions is trivially one ($p = 1$).

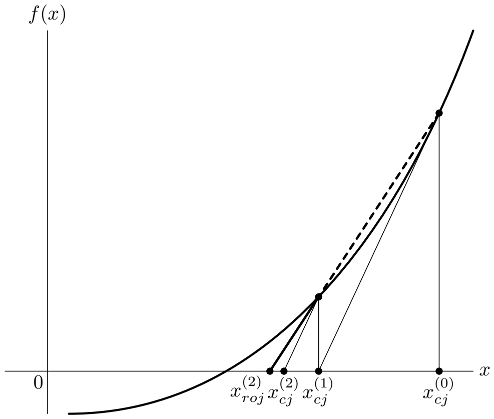


Figure 7.3: Constant Jacobian with speedup. Please observe that $x_{roj}^{(2)}$ is got by fitting a line through the previous two iterations. Also note that the $x_{roj}^{(2)}$ is closer to the root than $x_{cj}^{(2)}$ thus accelerating the constant Jacobian.

The speedup of the constant Jacobian is illustrated in Figure 7.3. Note that $x_{roj}^{(2)}$ is got by fitting a line through the previous two iterations. Also note that the $x_{roj}^{(2)}$ is closer to the root than $x_{cj}^{(2)}$ thus accelerating the constant Jacobian.

The *reduced order* Jacobian is applied every q iterations to accelerate the constant Jacobian and q is an integer usually between 2 to 5. The complete pseudocode is shown in Algorithm 4.

Algorithm 4 Accelerated-Constant-Jacobian

Input: A system $\mathbf{F}(\mathbf{x}) \triangleq \mathbf{A}(\mathbf{x})\mathbf{x} - \mathbf{b}$ consisting of m nonlinear equations in m unknowns:

$$f_i(x_0, x_1, \ldots, x_{m-1}) = 0, \quad i = 0, 1, \ldots, m-1$$

Input: An initial guess $\mathbf{x}^{(0)} = (x_0^{(0)}, x_1^{(0)}, \ldots, x_{m-1}^{(0)})$ of the solution.

Input: p, the number of partitions $\mathbf{A}(\mathbf{x})$ is divided into.

Output: $\mathbf{x}^* = (x_0^*, \ldots, x_{m-1}^*)$ solving m nonlinear equations simultaneously.

```
$k \leftarrow 0$
repeat
   // Accelerate using reduced Jacobian
   // by using it every $q$th iteration
   // after the first $k = p + 1$ iterations
   if $((k > p)$ and $!(k\%q))$ then
      Find $\mathbf{x}^{(k+1)}$ from the last $k (= p + 1)$ iterations using Eq. (7.8)
   else
      // do constant Jacobian
      $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - [\mathbf{J}(\mathbf{x}^{(0)})]^{-1}\mathbf{f}(\mathbf{x}^{(k)})$
   end if
   $k \leftarrow k + 1$
until ( $\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| \leq \epsilon$ and $\|\mathbf{F}(\mathbf{x}^{(k)})\| \leq \epsilon$ )
```

7.3 Experimental Results

The effect of nonlinear thermal conductivity was tested by considering the silicon layer of the chip. The simplified version has the silicon layer's boundaries at the room temperature of 27° C. The dimension of the chip is 8 mm × 8 mm. For simplicity assume that the entire silicon layer dissipates 100 W uniformly. For comparison, the difference in temperatures obtained by assuming constant thermal conductivity and nonlinear thermal conductivity is studied.

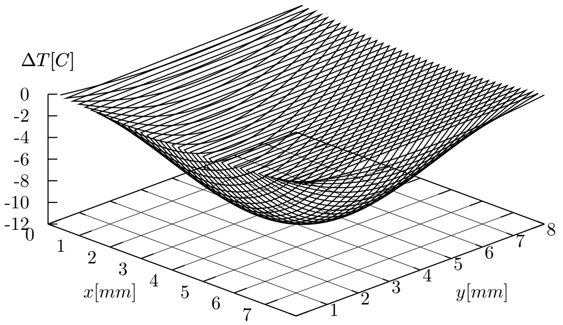


Figure 7.4: The difference in temperature profile in a silicon layer between having a constant thermal conductivity for silicon and incorporating nonlinear thermal conductivity for silicon. The chip dimension is 8 mm × 8 mm and it dissipates 100 W uniformly. The constant thermal conductivity evaluated at 27° C and used in the thermal simulation underestimates the peak temperature by 12%. This is ≈ 12° C in absolute value.

In Figure 7.4, the constant thermal conductivity evaluated at 27° C, underestimates the peak temperature by 12% when compared to the thermal profile obtained by considering the nonlinear thermal conductivity. Similarly, if the constant thermal conductivity is evaluated at 127° C, the peak temperature is overestimated by 13%. This inaccuracy in the thermal profile demonstrates the need for considering the nonlinearity of the thermal conductivity while doing the thermal simulation.

7.4 Summary

In this work, we studied the effects of ignoring the nonlinearity of thermal conductivity. We have shown that ignoring the thermal conductivity may result in a temperature profile that is off by 10° C and can cause inaccurate results in reliability analysis.

Chapter 8

Conclusion

This dissertation explored techniques in analyzing timing, temperature and power grid issues which arise due to scaling in digital integrated circuits. In this chapter we conclude and point out directions for future work based on this dissertation.

We demonstrated that it is possible and practical to perform path based statistical static timing analysis, and that such an analysis can be written compactly in matrix notation, allowing the use of standard highly optimized linear algebra techniques. The major advantage of this formulation is that it places no restrictions on process parameter distributions. It embeds an accurate polynomial-based delay model which takes into account slope propagation naturally. Data was presented to show that many practical circuits have a bounded number of paths, making such an analysis possible. It should be noted that this demonstration should not be taken as sufficient license to propose a purely path-based SSTA algorithm. (Chapter 2)

We presented a rigorous mathematical analysis of waveforms which led to a logical extension of present waveform modeling methods such as saturated-ramp model. The waveform models described in the work would find application in any situation where having a more detailed description of digital switching waveforms is useful. We demonstrated the application of waveform modeling to path-based STA demonstrating near-Spice like accuracy. (Chapter 3)

A new closed form delay metric based on the modified centroid of dissipated power was also presented. This new metric was inspired by our key observation that the Sakurai-Newton (SN) delay can be viewed as the centroid of current. We also provided a theoretic proof that the SN delay is the Elmore delay of a CMOS gate when a gate is modeled as an RC circuit. (Chapter 4)

We introduced a new path based methodology to size sleep transistors using temporal currents and timing windows. We have also proposed an efficient method to estimate the temporal switching current of the circuit. The results obtained indicate that our proposed technique results in area reduction of sleep transistors compared to the methods presented in the literature. (Chapter 5)

We also studied the effects of ignoring the nonlinearity of thermal conductivity while simulating the temperature profile of a chip. Ignoring thermal conductivity may lead to a temperature profile that is off by 10° C and can cause inaccurate results in reliability analysis. (Chapter 7)

A new model for power grid simulation was also presented, which retains the features that make power grid simulation amenable to linear algebraic methods presented in the literature. The proposed model can be seamlessly integrated into the existing power grid models. It is more accurate compared to the current source model and it also retains the efficiency of the current source model by including a constant conductance matrix. The proposed model offers a middle ground between the accuracy of SPICE simulation and the speed of the current source model. (Chapter 6)

8.1 Future Work

A possible future work is to extend the sparse matrix formulation for SSTA to handle wires, and show how incremental computation may be done in the framework. Potential application of this formulation lies in gate sizing. The sparse matrix contains the parameterized delays of every path in the circuit. Thus one can easily look up the sensitivity of a particular path to a particular variation and also which paths a given gate affects. This might lead to a formulation which globally optimizes the gate sizes rather than the sensitivity based local optimization which is widely used now. In the context of waveform modeling using SVD, the model can be extended to include interconnects and process variations. In the SSTA framework, one can incorporate the SVD based waveform models in the path-based timer using sparse-matrix framework. In the context of power grid analysis, one can investigate how size of decoupling capacitance reduces when the drain capacitance of the transistors is taken into account. Another possible topic is investigating approximation methods to find the voltage drop in the local charge distribution faster. This will help to improve time-step in the simulation and hence the runtime.

Bibliography

[1] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 331–336, 2004.

[2] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), April 1965.

[3] Semiconductor Industry Association. *International Technology Roadmap for Semiconductors*, 2005.

[4] Shekhar Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4):23–29, 1999.

[5] Larry McMurchie and Carl Sechen. WTA: waveform-based timing analysis for deep submicron circuits. In *ICCAD '02: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pages 625–631, 2002.

[6] T. Sakurai and A. Richard Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *IEEE Journal of Solid-State Circuits*, 25(2):584–594, 1990.

[7] James Kao, Anantha Chandrakasan, and Dimitri Antoniadis. Transistor sizing issues and tool for multi-threshold CMOS technology. In *DAC '97: Proceedings of the 34th annual conference on Design automation*, pages 409–414, 1997.

[8] Angela D. McConnell, Srinivasan Uma, and Kenneth E. Goodson. Thermal conductivity of doped polysilicon layers. *Journal of Microelectromechanical Systems*, 10(3):360–369, September 2001.

[9] Jing-Jia Liou, Kwang-Ting Cheng, Sandip Kundu, and Angela Krstic. Fast statistical timing analysis by probabilistic event propagation. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 661–666, 2001.

[10] Aseem Agarwal, David Blaauw, Vladimir Zolotov, and Sarma Vrudhula. Computation and refinement of statistical bounds on circuit delay. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 348–353, 2003.

[11] Anirudh Devgan and Chandramouli Kashyap. Block-based static timing analysis with uncertainty. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, pages 607–614, 2003.

[12] Hongliang Chang and Sachin S. Sapatnekar. Statistical timing analysis under spatial correlations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(9):1467–1482, 2005.

[13] Jiayong Le, Xin Li, and Lawrence T. Pileggi. STAC: statistical timing analysis with correlation. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 343–348, 2004.

[14] Anne E. Gattiker, Sani R. Nassif, Rashmi Dinakar, and Chris Long. Timing yield estimation from static timing analysis. In *ISQED '01: 2nd International Symposium on Quality of Electronic Design*, pages 437–442, 2001.

[15] Jing-Jia Liou, Angela Krstic, Li-C. Wang, and Kwang-Ting Cheng. False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation. In *DAC '02: Proceedings of the 39th conference on Design automation*, pages 566–569, 2002.

[16] Aseem Agarwal, David Blaauw, Vladimir Zolotov, Savithiri Sundareswaran, Min Zhao, Kaushik Gala, and Rajendran Panda. Path-based statistical timing analysis considering inter and intra-die correlations. In *ACM/IEEE International Workshop on Timing Issues*, 2002.

[17] J. A. G. Jess, K. Kalafala, S. R. Naidu, R. H. J. M. Otten, and C. Visweswariah. Statistical timing for parametric yield prediction of digital integrated circuits. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 932–937, 2003.

[18] Michael Orshansky and Arnab Bandyopadhyay. Fast statistical timing analysis handling arbitrary delay correlations. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 337–342, 2004.

[19] Yaping Zhan, Andrzej J. Strojwas, Xin Li, Lawrence T. Pileggi, David Newmark, and Mahesh Sharma. Correlation-aware statistical timing analysis with non-gaussian delay distributions. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 77–82, 2005.

[20] Lizheng Zhang, Weijen Chen, Yuhen Hu, John A. Gubner, and Charlie Chung-Ping Chen. Correlation-preserved non-gaussian statistical timing analysis with

quadratic timing model. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 83–88, 2005.

[21] Vishal Khandelwal and Ankur Srivastava. A general framework for accurate statistical timing analysis considering correlations. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 89–94, 2005.

[22] Hongliang Chang, Vladimir Zolotov, Sambasivan Narayan, and Chandu Visweswariah. Parameterized block-based statistical timing analysis with non-gaussian parameters, nonlinear delay functions. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 71–76, 2005.

[23] Khaled R. Heloue and Farid N. Najm. Statistical timing analysis with two-sided constraints. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM international conference on Computer-aided design*, pages 829–836, 2005.

[24] Debjit Sinha and Hai Zhou. A unified framework for statistical timing analysis with coupling and multiple input switching. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM international conference on Computer-aided design*, pages 837–843, 2005.

[25] Jaskirat Singh and Sachin S. Sapatnekar. Statistical timing analysis with correlated non-gaussian parameters using independent component analysis. In *ACM/IEEE International Workshop on Timing Issues*, 2006.

[26] David Blaauw, Vladimir Zolotov, and Savithri Sundareswaran. Slope propagation in static timing analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(10):1180–1192, 2002.

[27] Franc Brglez, David Bryan, and Krzysztof Koźmiński. Combinational profiles of sequential benchmark circuits. In *Proc. of International Symposium on Circuits and Systems*, pages 1929–1934, 1989.

[28] Hongliang Chang and Sachin S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 621, 2003.

[29] Yousef Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[30] Ashish Kumar Singh. *Statistical Algorithms for Circuit Synthesis under Process Variations and High Defect Density.* PhD thesis, The University of Texas, Austin, 2007.

[31] Yu Cao, Takashi Sato, Michael Orshansky, Dennis Sylvester, and Chenming Hu. New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation. In *Proceedings of Custom Integrated Circuits Conference*, pages 201–204, 2000.

[32] Maogang Wang, Xiaojian Yang, and Majid Sarrafzadeh. Dragon2000: standard-cell placement tool for large industry circuits. In *ICCAD '00: Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pages 260–263, 2000.

[33] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* McGraw-Hill Higher Education, 2001.

[34] Masanori Hashimoto, Yuji Yamada, and Hidetoshi Onodera. Equivalent waveform propagation for static timing analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(4):498–508, 2004.

[35] Sani R. Nassif and Emrah Acar. Advanced waveform models for the nanometer regime. In *ACM/IEEE International Workshop on Timing Issues*, 2004.

[36] Chirayu S. Amin, Florentin Dartu, and Yehea I. Ismail. Weibull-based analytical waveform model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(8):1156–1168, 2005.

[37] Alireza Kasnavi, Joddy W. Wang, Mahmoud Shahram, and Jindrich Zejda. Analytical modeling of crosstalk noise waveforms using weibull function. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 141–146, 2004.

[38] Amit Jain, David Blaauw, and Vladimir Zolotov. Accurate delay computation for noisy waveform shapes. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM international conference on Computer-aided design*, pages 946–952, 2005.

[39] John F. Croix and D. F. Wong. Blade and razor: cell and interconnect delay analysis using current-based models. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 386–389, 2003.

[40] Igor Keller, Ken Tseng, and Nisath Verghese. A robust cell-level crosstalk delay change analysis. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 147–154, 2004.

[41] Chirayu Amin, Chandramouli Kashyap, Noel Menezes, Kip Killpack, and Eli Chiprout. A multi-port current source model for multiple-input switching effects in CMOS library cells. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 247–252, 2006.

[42] Hanif Fatemi, Shahin Nazarian, and Massoud Pedram. Statistical logic cell delay analysis using a current-based model. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 253–256, 2006.

[43] Gilbert Strang. The fundamental theorem of linear algebra. *American Mathematical Monthly*, 100(9):848–855, November 1993.

[44] Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

[45] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006.

[46] Peter R. O'Brien and Thomas L. Savarino. Modeling the driving point characteristic of resistive interconnect for accurate delay estimation. In *ICCAD '89: Proceedings of the 1989 IEEE/ACM international conference on Computer-aided design*, pages 512–515, 1989.

[47] Dmitry Messerman, Alex Gershtein, Sergey Goldenberg, and Vladi Tsipenyuk. Advanced modeling techniques for accurate transistor-level timing analysis. In *ACM/IEEE International Workshop on Timing Issues*, 2007.

[48] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, May 1979.

[49] Anand Ramalingam, Ashish Kumar Singh, Sani R. Nassif, Gi-Joon Nam, Michael Orshansky, and David Z. Pan. An accurate sparse matrix based framework for statistical static timing analysis. In *ICCAD '06: Proceedings of the 2006 ACM/IEEE international conference on Computer-aided design*, pages 231–236, 2006.

[50] Charles J. Alpert, Anirudh Devgan, and Chandramouli V. Kashyap. RC delay metrics for performance optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(5):571–582, May 2001.

[51] Jorge Rubinstein, Paul Penfield, and Mark A. Horowitz. Signal delay in RC tree networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2(3):202–211, July 1983.

[52] Lawrence T. Pillage and Ronald A. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(4):352–366, April 1990.

[53] Bogdan Tutuianu, Florentin Dartu, and Lawrence Pileggi. An explicit RC-circuit delay approximation based on the first three moments of the impulse response. In *DAC '96: Proceedings of the 33rd annual conference on Design automation*, pages 611–616, 1996.

[54] Andrew B. Kahng and S. Muddu. An analytical delay model for RLC interconnects. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(12):1507–1514, December 1997.

[55] Rony Kay and Lawrence Pileggi. PRIMO: Probability interpretation of moments for delay calculation. In *DAC '98: Proceedings of the 35th annual conference on Design automation*, pages 463–468, 1998.

[56] Tao Lin, Emrah Acar, and Lawrence Pileggi. h-gamma: an RC delay metric based on a gamma distribution approximation of the homogeneous response. In *ICCAD '98: Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pages 19–25, 1998.

[57] Frank Liu, Chandramouli Kashyap, and Charles J. Alpert. A delay metric for RC circuits based on the weibull distribution. In *ICCAD '02: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pages 620–624, 2002.

[58] Charles J. Alpert, Frank Liu, Chandramouli Kashyap, and Anirudh Devgan. Delay and slew metrics using the lognormal distribution. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 382–385, 2003.

[59] Yuan Taur and Tak H. Ning. *Fundamentals of Modern VLSI Devices.* Cambridge University Press, 1998.

[60] William Shockley. A unipolar 'field-effect' transistor. In *Proceedings of Institute of Radio Engineers*, pages 1365–1376, 1952.

[61] T. Sakurai and A. Richard Newton. A simple MOSFET model for circuit analysis. *IEEE Transactions on Electron Devices*, 38(4):887–894, April 1991.

[62] Chandramouli V. Kashyap, Charles J. Alpert, Frank (Ying) Liu, and Anirudh Devgan. Closed-form expressions for extending step delay and slew metrics to ramp inputs for RC trees. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(4):509–516, April 2004.

[63] David Hodges, Horace Jackson, and Resve Saleh. *Analysis and Design of Digital Integrated Circuits: In Deep Submicron Technology.* McGraw-Hill, 2003.

[64] Rohini Gupta, Bogdan Tutuianu, and Lawrence Pileggi. The elmore delay as a bound for RC trees with generalized input signals. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(1):95–104, January 1997.

[65] W.C. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, 19(1):55–63, January 1948.

[66] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. *Signals and Systems.* Prentice Hall, 1996.

[67] James Kao, Siva Narendra, and Anantha Chandrakasan. MTCMOS hierarchical sizing based on mutual exclusive discharge patterns. In *DAC '98: Proceedings of the 35th annual conference on Design automation*, pages 495–500, 1998.

[68] Mohab Anis, Shawki Areibi, and Mohamed Elmasry. Design and optimization of multithreshold CMOS (MTCMOS) circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(10):1324–1342, 2003.

[69] Changbo Long and Lei He. Distributed sleep transistor network for power reduction. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 181–186, 2003.

[70] Himanshu Bhatnagar. *Advanced ASIC Chip Synthesis: Using Synopsys Design Compiler, Physical Compiler and PrimeTime.* Kluwer Academic Publishers, 1999.

[71] Howard H. Chen and J. Scott Neely. Interconnect and circuit modeling techniques for full-chip power supply noise analysis. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging*, 21(3):209–215, August 1998.

[72] Tsung-Hao Chen and Charlie Chung-Ping Chen. Efficient large-scale power grid analysis based on preconditioned krylov-subspace iterative methods. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 559–562, 2001.

[73] Yu Min Lee and Charlie Chung-Ping Chen. Power grid transient simulation in linear time based on transmission-line-modeling alternating-direction-implicit method. In *ICCAD '01: Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 75–80, 2001.

[74] Joseph N. Kozhaya, Sani R. Nassif, and Farid N. Najm. A Multigrid-like technique for power grid analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(10):1148–1160, October 2002.

[75] Haihua Su, Emrah Acar, and Sani R. Nassif. Power grid reduction based on algebraic multigrid principles. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 109–112, 2003.

[76] Yu Zhong and Martin D. F. Wong. Fast algorithms for IR drop analysis in large power grid. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM international conference on Computer-aided design*, pages 351–357, 2005.

[77] Quming Zhou, Kai Sun, Kartik Mohanram, and Danny C. Sorensen. Large power grid analysis using domain decomposition. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 27–32, 2006.

[78] Jin Shi, Yici Cai, Sheldon X.-D. Tan, and Xianlong Hong. High accurate pattern based precondition method for extremely large power/ground grid analysis. In *ISPD '06: Proceedings of the 2006 international symposium on Physical design*, pages 108–113, 2006.

[79] Hao Yu, Yiyu Shi, and Lei He. Fast analysis of structured power grid by triangularization based structure preserving model order reduction. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 205–210, 2006.

[80] Min Zhao, Rajendran Panda, Sachin S. Sapatnekar, and David Blaauw. Hierarchical analysis of power distribution networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(2):159–168, February 2002.

[81] Sachin S. Sapatnekar and Haihua Su. Analysis and optimization of power grids. *IEEE Design & Test of Computers*, 20(3):7–15, 2003.

[82] Weikun Guo, Sheldon X.-D. Tan, Zuying Luo, and Xianglong Hang. Partial random walks for transient analysis of large power distribution networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer*, E87-A(12):3265–3272, December 2004.

[83] Haifeng Qian, Sani R. Nassif, and Sachin S. Sapatnekar. Early-stage power grid analysis for uncertain working modes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(5):676–682, May 2005.

[84] Haifeng Qian, Sani R. Nassif, and Sachin S. Sapatnekar. Power grid analysis using random walks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(8):1204–1224, August 2005.

[85] Imad A. Ferzli and Farid N. Najm. Statistical verification of power grids considering process-induced leakage current variations. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, pages 770–777, 2003.

[86] Sanjay Pant, David Blaauw, Vladimir Zolotov, Savithri Sundareswaran, and Rajendran Panda. A stochastic approach to power grid analysis. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 171–176, 2004.

[87] Peng Li. Variational analysis of large power grids by exploring statistical sampling sharing and spatial locality. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM international conference on Computer-aided design*, pages 645–651, 2005.

[88] Praveen Ghanta, Sarma Vrudhula, Sarvesh Bhardwaj, and Rajendran Panda. Stochastic variational analysis of large power grids considering intra-die correlations. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 211–216, 2006.

[89] Eli Chiprout. Fast flip-chip power grid analysis via locality and grid shells. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 485–488, 2004.

[90] Sanjay Pant and Eli Chiprout. Power grid physics and implications for CAD. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 199–204, 2006.

[91] Su-Wei Wu and Yao-Wen Chang. Efficient power/ground network analysis for power integrity-driven design methodology. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 177–180, 2004.

[92] Chen-Wei Liu and Yao-Wen Chang. Floorplan and power/ground network co-synthesis for fast design convergence. In *ISPD '06: Proceedings of the 2006 international symposium on Physical design*, pages 86–93, 2006.

[93] Jiří Vlach and Kishore Singhal. *Computer Methods for Circuit Analysis and Design.* Van Nostrand Reinhold, New York, NY, USA, 1993.

[94] Atsushi Muramatsu, Masanori Hashimoto, and Hidetoshi Onodera. Effects of on-chip inductance on power distribution grid. In *ISPD '05: Proceedings of the 2005 international symposium on Physical design*, pages 63–69, 2005.

[95] Leon O. Chua and Pen-Min Lin. *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques.* Prentice Hall Professional Technical Reference, Eaglewood Cliffs, NJ, USA, 1975.

[96] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A multigrid tutorial: second edition.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[97] Mark Alan Horowitz. *Timing Models for MOS Circuits.* PhD thesis, Stanford University, January 1984.

[98] Yoshishige Murakami. A Method for the Formulation and Solution of Circuits Composed of Switches and Linear *RLC* Elements. *IEEE Transactions on Circuits and Systems*, 34(5):496–509, May 1987.

[99] Huang-Jin Wu and Wu-Shiung Feng. Efficient simulation of switched networks using reduced unification matrix. *IEEE Transactions on Power Electronics*, 14(3):481–494, May 1999.

[100] Shu-Yuen Ron Hui and S. Morrall. Generalised associated discrete circuit model for switching devices. *IEE Proceedings on Science, Measurement and Technology*, 141(1):57–64, January 1994.

[101] Predrag Pejović and Dragan Maksimović. A method for fast time-domain simulation of networks with switches. *IEEE Transactions on Power Electronics*, 9(4):449–456, July 1994.

[102] Shiyou Zhao, Kaushik Roy, and Cheng-Kok Koh. Decoupling capacitance allocation and its application to power-supply noise-aware floorplanning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(1):81–92, 2002.

[103] Maha Nizam, Farid N. Najm, and Anirudh Devgan. Power grid voltage integrity verification. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 239–244, 2005.

[104] Ram Viswanath, Vijay Wakharkar, Abhay Watwe, and Vassou Lebonheur. Thermal performance challenges from silicon to systems. *Intel Technology Journal*, 4(Q3), August 2000.

[105] Haihua Su, Frank Liu, Anirudh Devgan, Emrah Acar, and Sani Nassif. Full chip leakage estimation considering power supply and temperature variations. In *ISLPED '03: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 78–83, 2003.

[106] Sang-Soo Lee and David J. Allstot. Electrothermal simulation of integrated circuits. *IEEE Journal of Solid-State Circuits*, pages 1283–1293, 1993.

[107] Yi-Kan Cheng, Prasun Raha, Chin-Chi Teng, Elyse Rosenbaum, and Sung-Mo Kang. ILLIADS-T: An electrothermal timing simulator for temperature-sensitive reliability diagnosis of CMOS VLSI chips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 668–681, 1998.

[108] Ching-Han Tsai and Sung-Mo Kang. Fast temperature calculation for transient electrothermal simulation by mixed frequency/time domain thermal model reduction. In *DAC '00: Proceedings of the 37th conference on Design automation*, pages 750–755, 2000.

[109] Ting-Yuan Wang and Charlie Chung-Ping Chen. Thermal ADI: A linear-time chip level dynamic thermal-simulation algorithm based on alternating-direction-implicit (ADI) method. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 691–700, 2003.

[110] Peng Li, Lawrence T. Pileggi, Mehdi Asheghi, and Rajit Chandra. Efficient full-chip thermal modeling and analysis. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 319–326, 2004.

[111] Danqing Chen, Erhong Li, Elyse Rosenbaum, and Sung-Mo Kang. Interconnect thermal modeling for accurate simulation of circuit timing and reliability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 197–205, 2000.

Vita

Anand Ramalingam was born in Kanyakumari (Cape Comorin), India on 21 May 1979. He joined the P.S.G. College of Technology, Coimbatore in 1996 where he received the Bachelor of Engineering degree in Electronics and Communication Engineering in April 2000. He worked in Honeywell India as a software engineer for more than a year. In September 2001, he joined the graduate program in the Department of Electrical Engineering at the Stanford University where he received the Master of Science degree in Electrical Engineering in June 2003. In August 2003, he joined the graduate program in the Department of Electrical and Computer Engineering at the University of Texas at Austin. Currently he is working under the supervision of Prof. David Z. Pan in the area of timing analysis and circuit simulation.

Permanent Address: 45, Sivasakthi Nagar,
Thaneer Pandal Road,
Coimbatore, Tamil Nadu,
India 641 004

This dissertation was typeset with $\LaTeX\,2_\varepsilon$[1] by the author.

---

[1] $\LaTeX\,2_\varepsilon$ is an extension of $\LaTeX$. $\LaTeX$ is a collection of macros for $\TeX$. $\TeX$ is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.