Copyright
by
John Francis Croix
2002

The Dissertation Committee for John Francis Croix
certifies that this is the approved version of the following dissertation:

Cell and Interconnect Timing Analysis Using Waveforms

Committee:

Martin Wong, Supervisor

Jacob Abraham

Donald Fussell

Farid Najm

Baxter Womack

Cell and Interconnect Timing Analysis Using Waveforms

by

John Francis Croix, B.S., 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

August 2002

UMI Number: 3099442

UMI Microform 3099442

Copyright 2003 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against

unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
PO Box 1346
Ann Arbor, MI 48106-1346

Dedicated to those people in my life whose sacrifice and support made this possible.

Acknowledgments

This dissertation would not be possible were it not for the support and encouragement of my research advisor, Professor Martin Wong. Even during those inevitable lulls in which progress was measured in weeks or months, Professor Wong's unwavering confidence in my ability to complete this dissertation successfully gave me the incentive to continue.

Professor Farid Najm of the University of Toronto deserves special recognition. While I knew what I wanted to relate in this dissertation, I was not always able to do so in a concise and clear manner. Professor Najm provided invaluable guidance to make sure that my ideas were expressed completely and articulately and contributed greatly to the quality of this dissertation.

Next I would like to thank Professor Larry Pileggi. Professor Pileggi was my advisor at UT prior to his appointment at Carnegie Mellon University. I originally met Professor Pileggi while I was working at Texas Instruments and decided to pursue my Ph.D. at UT as a direct result of our interaction and his encouragement.

Curtis Ratzlaff and Guru Rao have been a constant source of support, friendship, and encouragement. They represent that rare breed of person that tends to blur the line between friend and family.

My mother, sister, aunts, and uncles continue to be a source of support and inspiration for me. My children, Nicholas, Joshua, and Brandon, constantly remind me not to take myself too seriously. This was especially true during those periods when I was deep in “dissertation mode” and had little ability to focus on anything other than “just getting that next result down on paper.”

Friends and colleagues at Silicon Metrics Corporation deserve special mention. Notably I wish to thank Bill Wood, Stephen Strauss, Jim Clardy, Callan Carpenter, Vess Johnson, Scott Yore, Frank Childers, Robert Gonzalez, Tamara Cryar, and Brian Allgyer.

Finally I would like to thank Hannah Gourgey. Having completed her own Ph.D. at UT recently, she encouraged me to work past obstacles of my own creation toward a successful conclusion.

Cell and Interconnect Timing Analysis Using Waveforms

Publication No. ____________________

John Francis Croix, Ph.D.
The University of Texas at Austin, 2002

Supervisor: Martin Wong

Advancements in the fabrication of integrated circuits have led to the dominance of interconnect delay in overall path delay – the total delay between two sequential elements in a circuit. As a result, great improvements in high-accuracy, rapid interconnect analysis algorithms have been made. While interconnect can be modeled efficiently at near-SPICE accuracy now, archaic standard cell models remain a limiting factor in timing analysis as out-dated cell modeling technology has not kept pace with nanometer effects exhibited in digital ICs today.

This dissertation presents improvements in the accuracy of existing cell models for the front end of the IC design flow through a process called over sampling and data reduction. For the back end, where accuracy with respect to silicon is critical, Blade, a new cell model and accompanying runtime engine, is presented. The Blade environment brings SPICE accuracy to timing verification flows at speeds tens of thousands of times faster than SPICE. Finally, a rapid interconnect timing system, RAZOR, is presented which works in conjunction with BLADE to obtain SPICE-accurate stage delay analysis capability.

Together BLADE and RAZOR provide sophisticated timing analysis not previously available outside the domain of transistor-level timing analysis engines: production and consumption of arbitrary voltage waveforms, near-SPICE accuracy, and unparalleled runtimes. When used in conjunction with front-end models created through over sampling and data reduction, model consistency throughout the entire design flow is achieved.

Table of Contents

Acknowledgments **v**

Abstract **vii**

List of Tables **xii**

List of Figures **xiii**

Chapter 1. Introduction **1**

Chapter 2. Literature Review **5**

- 2.1 Cell Characterization and Modeling 9
  - 2.1.1 Overview 10
  - 2.1.2 Effective Capacitance 10
  - 2.1.3 Empirical Modeling 14
    - 2.1.3.1 Characterization Techniques 14
    - 2.1.3.2 Cell Modeling 15
    - 2.1.3.3 The Synopsys Model 18
    - 2.1.3.4 Other Characterization Models 19
  - 2.1.4 Analytic Modeling 21
    - 2.1.4.1 Transistor Modeling 21
    - 2.1.4.2 Cell Modeling 23
    - 2.1.4.3 Path Sensitization 24
- 2.2 Interconnect Delay Analysis 25
  - 2.2.1 Early Analysis 26
  - 2.2.2 Advances in Moment Matching 27
  - 2.2.3 Timing Analysis 28
- 2.3 Circuit Simulators 30
- 2.4 Conclusion 35

Chapter 3. Model Development **37**
- 3.1 Cell Characterization and Modeling 39
  - 3.1.1 Auto-Ranging 40
    - 3.1.1.1 Capacitive Limits 41
    - 3.1.1.2 Input Transition Time Limits 42
    - 3.1.1.3 Input Voltage Waveforms 42
    - 3.1.1.4 Putting It All Together 44
  - 3.1.2 Over Sampling and Data Reduction 45
    - 3.1.2.1 Error Calculation 47
    - 3.1.2.2 Reduction of Characterization Tables 51
  - 3.1.3 Conclusion 52
- 3.2 Blade 54
  - 3.2.1 The Blade Current Model 56
  - 3.2.2 The Blade Runtime Engine 57
    - 3.2.2.1 The Nonlinear Solver 58
    - 3.2.2.2 Problem Formulation 60
  - 3.2.3 Accounting for Internal Parasitics 62
    - 3.2.3.1 Internal Capacitance 63
    - 3.2.3.2 Internal Cell Interconnect 66
    - 3.2.3.3 Model Calibration 66
    - 3.2.3.4 Conclusion 68
  - 3.2.4 Compound Cells 68
  - 3.2.5 Noisy Inputs 72
  - 3.2.6 Conclusion 74
- 3.3 Razor: Accurate Stage Delay 75

Chapter 4. Results **80**
- 4.1 Over Sampling and Data Reduction 80
  - 4.1.1 Dependent Variable Analysis 80
  - 4.1.2 Independent Variable Analysis 82
  - 4.1.3 Path Errors 83
- 4.2 Blade 84
  - 4.2.1 Calibration 86
  - 4.2.2 Using the Models 88

4.2.3 Performance 91
4.2.4 The Noise Immunity Filter 92
4.3 RAZOR 96
4.3.1 Performance Overview 97
4.3.2 Accuracy 98
4.4 Final Remarks 101

Chapter 5. Conclusion **104**

Bibliography **106**

Vita **114**

List of Tables

4.1 Calibration Parameters for Sample Cells Paths 88
4.2 Blade CPU Runtime Results 94
4.3 Razor CPU Runtime Results 98

List of Figures

2.1 Synchronous digital logic data flow 5
2.2 Cell and interconnect delay in STA 7
2.3 Cell timing measurements 11
2.4 Voltage waveform tail 13
2.5 Characterization table interpolation 19
2.6 Thevenin equivalent gate model 20
2.7 Recursive convolution 30
2.8 Newton-Raphson iteration 32

3.1 A driver cell used to create realistic inputs for characterization . . 44
3.2 Characterization table error calculation 50
3.3 Blade characterization 56
3.4 Simple Blade evaluation 58
3.5 Secant Iteration 60
3.6 Companion models in the Blade runtime engine 61
3.7 Trapezoidal companion model of a capacitor 62
3.8 Modeling internal capacitances 64
3.9 Blade current model creation for an OR gate 70
3.10 Blade OR model calibration 71
3.11 Noise immunity filter 73
3.12 Stage delay 76
3.13 Recursive convolution using fixed timesteps 77

4.1 Errors in traditional characterization 81
4.2 Optimized table errors 82
4.3 Results of data reduction 83
4.4 Validating path timing 85
4.5 I–V response surface 86
4.6 Blade calibration 87
4.7 Blade compound cell analysis 89

4.8 BLADE noise analysis 90
4.9 BLADE modeling with tails 91
4.10 Noise immunity on small AOI 93
4.11 Large AOI noise immunity filter 95
4.12 RAZOR comparison to HSPICE 99
4.13 RAZOR noise comparison to HSPICE 100

List of Algorithms

3.1 Data reduction error table generation 48

3.2 Data reduction result generation 53

Chapter 1

Introduction

In 1965, Gordon Moore, co-founder of Intel, made the observation that the number of transistors per square inch on integrated circuits (ICs) doubled every year since the invention of the integrated circuit. His subsequent prediction, that this doubling would continue to occur at approximately 18-month intervals, has become known as *Moore's Law*.

While Moore's Law typically is referred to in discussions concerning manufacturing capability and integrated circuit performance, it has broad economic implications, too. In order to create these circuit designs, the design teams also had to double in size each 18 months. Such design team growth, however, was not economically feasible, and, thus, the computer-aided design (CAD) industry was born. Instead of growing the design team to increase the number of transistors on a circuit, automation was employed to make each designer more productive.

Engineers today have a vast array of CAD applications at their disposal to handle the complexities associated with nanometer technology requirements. These applications are combined into an overall design flow to provide automation from design concept through physical layout, verification, and mask generation. Virtually all of these flows, either implicitly or explicitly, rely on the concept of successive approximation refinement. That is to say, early in the design flow, when the functionality of the circuit is still being defined, the data used to model this functional view provides only a rough approximation of actual silicon performance. As the designer progresses further into the flow, the closer the estimated performance should be to actual silicon – at least in theory.

In older IC design flows consistent results were often more important than accurate results when predicting silicon performance. In other words, as long as all of the applications within the design flow produced results that were both consistent with one another and "close enough" to silicon, a functional integrated circuit could be produced. Guardband was added to the result at each stage of the flow to mitigate the inaccuracy inherent in the data. As a result, CAD application development focused on better and faster algorithms, leaving the application user with the responsibility of creating data for those applications.

With the advent of transistor feature sizes of 0.25μm and below, the process of design guardbanding became too unwieldy. If the design were guardbanded in the same manner that produced previous generations of integrated circuits, the design constraints would be so overwhelming as to make it impossible to meet the design goals. The only way to create these new designs was to have a flow that was both consistent and accurate, thus reducing the amount of required guardband.

Interconnect delay has been the primary area of focus for accuracy improvements in the last 13 years because it has become the dominant component of overall path delay (the delay between two sequential elements). Consequently, interconnect delay models have advanced significantly while cell delay models, the other component of path delay, have advanced very little. Interconnect delay approximations can be made today with near-SPICE accuracy at speeds thousands of times faster than SPICE. Unfortunately, this increase in accuracy provides little additional benefit without similar advances in cell models due to the nonlinear effects that are not accurately represented.

There are two ways in which interconnect model accuracy is lost with respect to cell models. First, the vast majority of cell models still provide a simple two-point (linear) ramp as input to the interconnect. In reality the output voltage of a cell can be highly nonlinear, varying greatly across input voltage waveforms and output loads. Second, the typical input to a cell model (the “output” of the previous interconnect) is a two-point ramp approximation of the voltage waveform produced during interconnect modeling. Thus, even though interconnect analysis can be performed using nonlinear waveform inputs and producing nonlinear outputs, the typical cell model is not sophisticated enough to either consume or produce such data and, therefore, is unable to benefit fully from accurate interconnect modeling.

To make matters worse, cell characterization, the process of creating a cell model, is a difficult, tedious, and time-consuming process. Attempts to create more accurate cell models have, in the past, placed an even larger burden on cell characterization. With typical cell library characterization times measured in CPU months, engineers in charge of this process have been resistant to any increase in CPU overhead. The idea that increased accuracy comes only at the expense of increased overhead has so dominated the engineer’s mindset that many companies have resigned themselves to the thought that their cell libraries would never fully meet their accuracy requirements, leaving no alternative but targeted, selective guardbanding.

This dissertation shatters these long-held preconceptions concerning cell modeling and accuracy. Recognizing that simple, fast models have their place early in the design flow, a methodology called *over sampling and data reduction*[1] is introduced to address the characterization and modeling of front-end cell models. This process maximizes the accuracy of the model without increasing CPU or engineering requirements. Unfortunately, given the current state of cell modeling, it is these simple models that are used throughout the entire flow, not just in the early stages of the design process. Fundamental limitations of the existing models substantially reduce their usefulness in the later portions of the design flow where a high correlation to silicon is required.

To address the nonlinear effects inherent in today's nanometer designs, a new modeling paradigm is necessary. The second portion of this dissertation introduces a new cell model and evaluation engine (Blade) that is able to both produce and consume nonlinear waveforms with near-SPICE accuracy at speeds tens of thousands of times faster than SPICE. Generation of the Blade cell model requires fewer computing resources than traditional cell characterization methodologies require today. Additionally, the Blade engine is a lightweight, nonlinear solver than can be embedded within CAD applications easily.

Finally, as cell models and interconnect models must be able to interact with one another, a stage delay model (unified cell and interconnect delay model) is presented that marries these two technologies. This stage delay model, Razor, maintains SPICE accuracy while exhibiting runtimes between five and six orders of magnitude faster than SPICE.

---

[1] Over sampling and data reduction is a single process, not two independent processes. For that reason, this term is used with singular verbs throughout the dissertation.

Chapter 2

Literature Review

In modern synchronous, digital integrated circuit designs, data signals traverse from one sequential element to another within a time period. That time period is defined by the clock signal(s) gating the two sequential elements as illustrated in Figure 2.1.

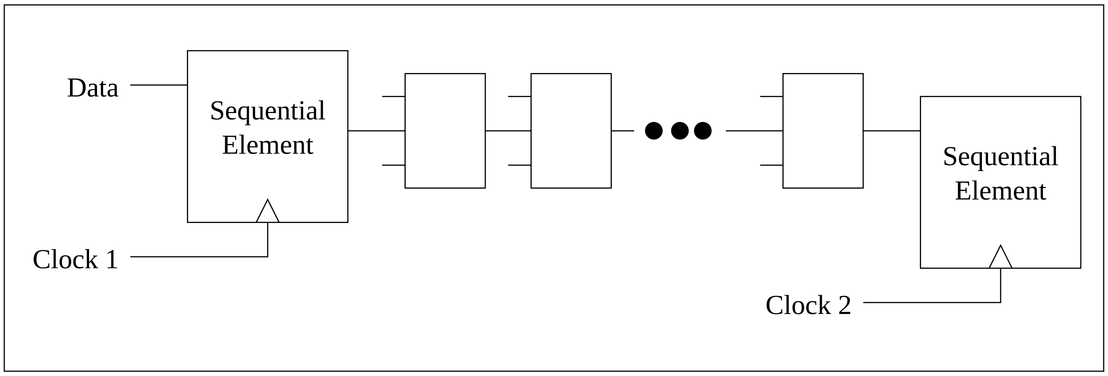


Figure 2.1: Signals traverse from one sequential element, through the cells and interconnect implementing the logic function, to another sequential element. This must be accomplished between arrival of the clock signal at the first sequential element and the arrival of the clock signal at the second.

Prior to the advent of static timing analysis (STA) [22], timing verification of such digital logic often involved the creation of test patterns that were used to drive simulation programs [12]. Some programs, such as SPICE [40], performed analysis at the transistor level to obtain maximum accuracy when compared to silicon. Other types of simulators, such as cycle-based simulators [39], ignored timing and propagated only logic values between sequential elements to perform functional verification. These simulations required a substantial amount of CPU time and produced results that were valid only for the given test pattern. To obtain a high degree of confidence that a design met its timing and functional requirements, many such simulations using different patterns had to be made in an attempt to exercise a statistically sufficient set of logic paths.

This dissertation does not address the problem of functional verification. Once a design has met its functional requirements, timing verification still must be performed to guarantee that the specific logic implementation used meets the timing constraints. While the exact implementation does directly impact the timing, verification of functionality and timing correctness can be performed independently.

With the use of static timing analysis, timing correctness can be determined by independently analyzing the timing through individual logic functions and interconnect. The worst delays and signal slews are propagated from one cell to another across the interconnect. The worst-case signal arrival times at a sequential element are compared against the timing constraints and a determination is made as to whether the circuit meets its timing requirements. Cell models are used to rapidly calculate both the timing from the input pin of a logic function to its output pin and to determine the rate that the output signal transitions (low-to-high or high-to-low). Interconnect analysis algorithms use the output transition times of the cells to determine the delay to the next cell as well as the input transition time to that cell.

In modern static timing, two-point linear approximations of the waveform

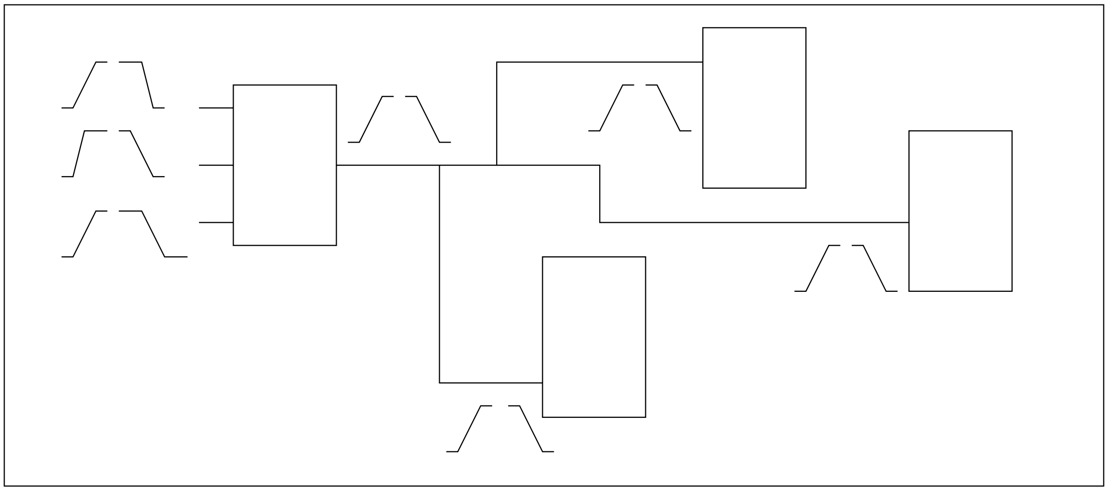


Figure 2.2: Rising and falling waveforms at the inputs of each cell are used to calculate intrinsic delays and output transitions times. Only two transition times are propagated through the interconnect to the next cell: one rising and one falling transition. Waveforms are approximated by linear ramps.

driving a cell's input pins are used in conjunction with the interconnect load to calculate intrinsic delay and output transition time. Often the interconnect load is simply represented by a single capacitor, though more sophisticated models have been proposed. The transition times of signals driving the interconnect are also modeled as two-point linear ramps. In worst-case analysis, illustrated in Figure 2.2, only the slowest rising and falling output transition times are propagated through the interconnect to the cell inputs of the next level.

Characterization is the process of empirically gathering electrical information about a cell, such as timing, and creating an abstract model to represent that data for a variety of expected operating conditions. Characterization is usually performed by simulating the cell with a highly accurate simulator, such as SPICE. Such data gathering processes are highly CPU intensive, taking days, weeks, or even months of CPU time for a large cell library. However, the repeated evaluation of the resultant model during timing verification occurs rapidly, thus justifying the one-time cost of accurate cell characterization.

To obtain accurate cell models, characterization must occur over the expected operating range of the cell. Differences between the expected environment used during characterization and the actual environment that the cell is used within can be a source of large discrepancies between the predicted behavior and the actual behavior and may result in operational failures of an integrated circuit.

Timing models produced through characterization can be highly useful in the early portion of the design flow as model accuracy is often less important than raw evaluation speed. A 10% to 20% estimation error in cell and interconnect delay is not as critical early in the design process as in the later stages. During the early portion of the design flow, many different implementations are constantly being evaluated in an attempt to meet functionality requirements. Once the functionality has stabilized, timing verification and accuracy become more important. It is during these latter stages of design that the simplifying assumptions made during cell characterization are often violated and more accurate models required.

This is not to say that the accuracy of front-end cell models is not important. These same cell models often are used by various programs during the back end of the design flow, when accuracy is critical. Even when different models are used, consistency between the front-end and back-end models remains important. If a program in the front end of the design flow specifies an implementation using models that are inconsistent with the models in the back end, the specification defined by the front-end programs may not be realizable in the back end. This type of situation leaves the designer in a quandary as he/she cannot iterate back through the front-end programs because those programs were the very ones to specify the implementation. When these inconsistencies arise, the only solution is to tighten constraints on the system, potentially introducing new implementation difficulties in portions of the logic that, in reality, are not critical. By maintaining accuracy in front-end models, consistency with back-end models is guaranteed as a byproduct, reducing unnecessary iterations and improving designer productivity.

This chapter reviews the development of cell and interconnect models used in the integrated circuit design flow. It traces the evolution of these models and algorithms from their early development through to the present day. Only by understanding both the design flow and the strengths and weaknesses of existing cell and interconnect models can the power of the Blade and Razor models be fully appreciated.

Timing systems are also reviewed. In order to be effective, the Blade and Razor models require an accompanying application program that can evaluate them. Existing systems consume cell and interconnect models to produce stage and path delay approximations. A similar system was developed for the Blade and Razor models so that accurate stage delay approximation can occur.

2.1 Cell Characterization and Modeling

Cell models fall into one of two categories: those created from empirical data and those that are analytically created [57]. Analytic models predict a cell's electrical performance as a function of the fabrication technology and the size and topology of the transistors that compose the cell. Empirical models, on the other hand, are created by simulating the cell, using some electrical simulator such as SPICE, measuring the results, and fitting the measured data to the model format. Characterization, by definition, is the process of empirically gathering data to create a model.

2.1.1 Overview

The timing characterization of a cell is performed by driving the input port of a cell with a rising or falling voltage waveform, placing a load on the output port (usually a capacitive load, C_l), and setting secondary pins to V_{DD} or V_{SS} to obtain a rising or falling response on the output pin. Measurements are then taken of the propagation delay (T_d) between the voltage waveforms at the input pin and the output pin, the input transition time (T_{in}), and the output transition time (T_{out}). T_d is usually measured at the 50% point of V_{DD} between the two pins. T_{in} and T_{out} are measured along the more linear region of the voltage transition, usually between the 10% and 90% points or the 20% and 80% points of V_{DD}. This is depicted in Figure 2.3.

To obtain comprehensive characterization data, a range of input transition times and capacitive loads is applied to each input/output pin pair. This process occurs for every input/output pin pair combination of the cell using both rising and falling input transitions.

2.1.2 Effective Capacitance

Cell characterization using simple input transition times and capacitive loads requires a large investment of CPU time. While purely capacitive loads are poor approximations for the highly resistive and tightly coupled interconnect

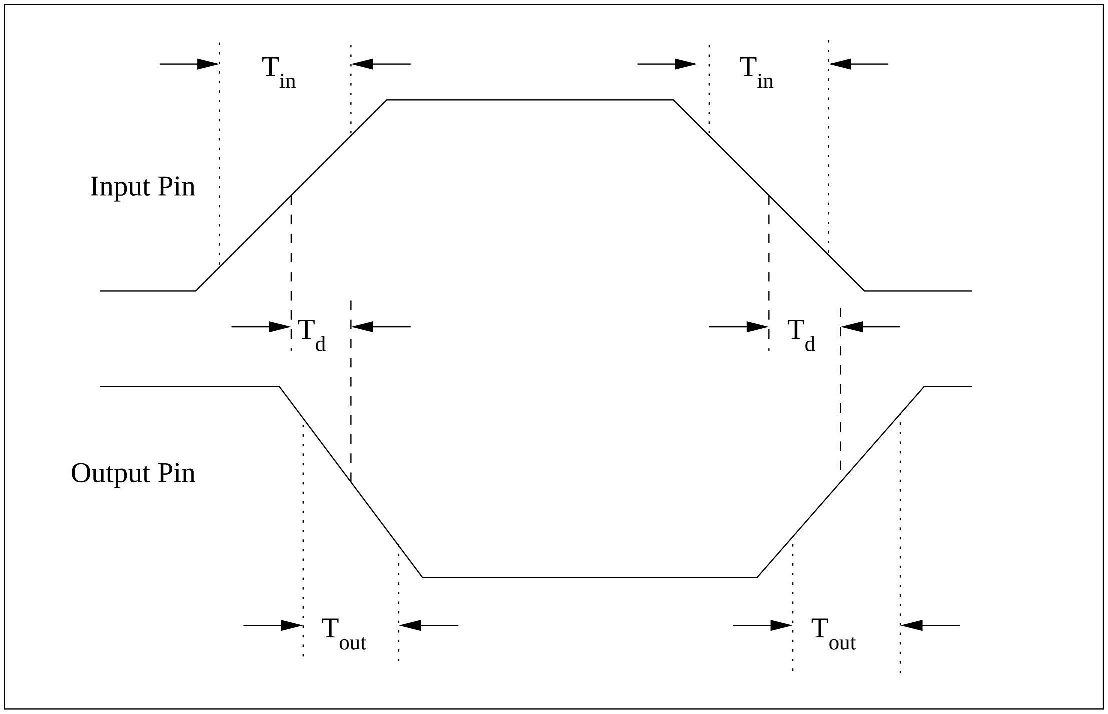


Figure 2.3: Measurements that are made on the rising and falling inputs and outputs of a cell are shown: intrinsic delay (T_d), input transition time (T_{in}), and output transition time (T_{out}).

seen in today's designs, increasing the complexity of the interconnect model to something as simple as a π-model[1] would result in a huge increase in these CPU requirements.

In order to more accurately predict a cell's behavior while maintaining the simplicity of a capacitive load, the concept of an effective capacitance, C_{eff}, was introduced in [51] and refined in [48] and [13]. The resistive shielding of the interconnect prevents the current from quickly reaching the furthest capacitances.

---

[1] The term π-model is used to describe a capacitor–resistor–capacitor series. The graphical representation of this series looks like the letter π.

This results in a more rapid rise in the voltage at the output pin than would be seen by a single capacitor with a value that is the total capacitance of the interconnect, C_{tot}. Calculation of C_{eff} is done by equating the mean current into the interconnect with the mean current into a single capacitor over the transition period.

Most timing verification systems that use empirical models require the calculation of C_{eff} at model evaluation time as the cell's characterized performance is often measured when the cell is simulated using only a simple capacitive load. Analytic models may also utilize C_{eff}, but this is a function of the form and runtime requirements placed on the model. Due to the widespread use of empirical cell models, C_{eff} has become a fixture in timing verification systems.

Unfortunately, the utility of C_{eff} is rapidly diminishing in today's nanometer designs. Because C_{eff} is based on mean current flow, the cell's intrinsic delay (often measured at the 50% point of the output voltage waveform) can be roughly approximated using a capacitive load of C_{eff}. However, as the output voltage waveform seen when modeling the cell and interconnect in SPICE may exhibit a distinct "tail" due to resistive shielding (see Figure 2.4), the output transition times cannot be accurately approximated with a simple capacitive load.

More accurate models of the interconnect can be created for use in characterization [42, 45]. These models can be used later in the design flow to perform in situ characterization for each specific cell instance. However, in the front end of the design flow, C_{eff} will continue to be used.

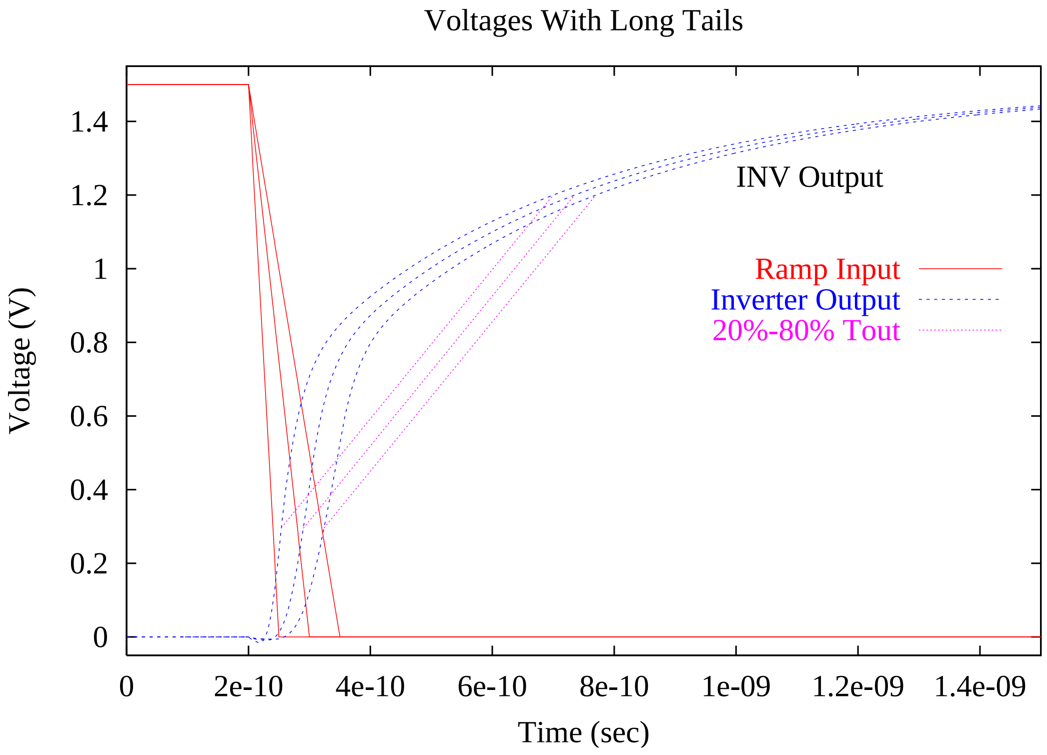


Figure 2.4: The voltage output waveform of an inverter exhibits a distinct tail when driving a load with high resistive shielding. A linear approximation of the output transition time can lead to significant errors during timing verification. The 20% to 80% T_{out} approximations are shown as straight lines across the tails.

2.1.3 Empirical Modeling

Cell characterization is the process of empirically measuring the electrical performance of a cell. Cell modeling is the process of abstracting this data into a model for rapid evaluation by an application in the design and/or verification flow [36]. Most papers and books that discuss cells and their models omit the process of characterization. However, without accurate characterization, the empirical data used to derive the model may not approximate the cell's behavior in a circuit with sufficient accuracy or consistency.

2.1.3.1 Characterization Techniques

Little literature exists concerning the actual process of characterization. While various researchers have written papers concerning the form and benefits of different cell models, the process of empirically gathering the data to create the model has largely been left as an exercise for the reader. What little information exists can be found by searching the U.S. patent database and by examining the characterization methodologies used by commercial products as well as internal tools in integrated circuit companies.

Early characterization systems were developed internally by large integrated circuit design companies. These companies had their own fabrication facility with proprietary cell models and transistor models tailored to their design process. Characterization occurred internally and the resultant proprietary models were delivered to the IC designers (both internal design groups and external customers). Many of these characterization systems simply used input provided by the characterization engineer (also known as the librarian) to determine which data points should be extracted. This raw measured data was then used to perform a curve fit to create the cell's electrical model.

Later systems automated the process of specifying the environment under which data should be gathered. Instead of exactly specifying every capacitive load and input transition time to be simulated, the librarian specified the minimum and maximum bounds, and the characterization system distributed its runs across these bounds.

Eventually, as design features continued to shrink, the response surface[2] of these measured values began to exhibit distinct nonlinear regions. Predicting where these nonlinear regions occurred prior to simulation was nearly impossible in large cell libraries. Thus, in some characterization systems, Monte Carlo simulation was employed to verify the models created from these measurements against the SPICE measurements under those same conditions. Regions exhibiting large interpolation errors could then be identified and the points used for data extraction in subsequent runs could be selected to reduce or eliminate these errors. This was successfully employed in [19] to automatically determine where additional points should be measured by recharacterizing in the areas that exhibited errors greater than some user-defined tolerance.

2.1.3.2 Cell Modeling

Characterization data, once empirically measured, is used to create the cell model. Empirical models used in production IC design flows tend to fall

---

[2] The term *response surface* refers to measured intrinsic delays or output transition times. When plotted against the input transition time and output load, a 3-D surface is formed as shown in Figure 4.3.

into one of two categories: characteristic equations and characterization tables. A characteristic equation is a monolithic equation used to represent the cell's behavior across one or more variables. The characterization table uses a multitude of equations, stored in a table, only one of which is applicable for a specific range of input variables.

Characteristic equations are determined by fitting the empirical data to the equation. Typically they are quite complex and contain square and logarithmic terms in order to model the inherent nonlinearities exhibited by the measured intrinsic delay and output transition time response surfaces. Equation 2.1 is an example 21-term characteristic equation, used in the early 1990s, to calculate a cell's response as a function of voltage, temperature, fabrication process, input transition time, and capacitive load. Some characteristic equations in use today for nanometer designs contain over 100 terms.

$$\begin{aligned}
f(V,T,P,t_{in},C_l) \quad = \quad & a_1V + a_2T + a_3P + a_4t_{in} + a_5C_l + a_6VT + a_7VP + \\
& a_8Vt_{in} + a_9VC_l + a_{10}TP + a_{11}Tt_{in} + a_{12}TC_l + \\
& a_{13}Pt_{in} + a_{14}PC_l + a_{15}t_{in}C_l + a_{16}\ln(V) + a_{17}\ln(T) + \\
& a_{18}\ln(P) + a_{19}\ln(t_{in}) + a_{20}\ln(C_l) + a_{21} \qquad (2.1)
\end{aligned}$$

Characterization tables have become more popular in recent years as it has become more and more difficult to create a single equation that adequately represents all measured values. Just as a complex curve can be approximated using a series of connected line segments, a cell's response surface can approximated using an array of simple characteristic equations. As the equations used in characterization tables tend to be nearly linear, the selection of the boundary points (the points that define the region for which a given equation is valid) becomes critical. The selection of two points on opposite sides of a very nonlinear region will result in large interpolation errors when the model is evaluated.

The creation of characterization tables is complicated by the fact that the actual response surface being approximated is unknown. The data points used to create the characterization tables might not be good representatives of the overall response surface. It is not possible to know whether they are indeed good representatives a priori. While programs such as SPICE can provide the response of a circuit at a very specific input transition time and output capacitive load, the true response surface is not known in advance over all input transitions and output loads. Thus simply picking a random sampling of input transition times and output loads to measure at and store in a characterization table does not guarantee accuracy when interpolation is used to determine the response at points not actually measured.

Measuring at a greater number of data points and storing larger tables to represent the cell can minimize large interpolation errors. However, for a cell library of hundreds or thousands of cells and tens of thousands of tables, the memory consumption becomes prohibitive. Furthermore, larger tables require more CPU time to search through and interpolate across when randomly accessed. The combination of the two makes smaller tables much more desirable than larger tables.

In [29] a new model is proposed that reduces runtime characterization data memory requirements and CPU time for evaluation by treating input to output paths through a cell as an edge in a graph. These edges are combined to produce a substantially smaller graph while maintaining the delay accuracy. Unfortunately, this model, which the authors refer to as the concise delay network, represents delay as a single value instead of an equation or table and, thus, does not provide the accuracy required for nanometer designs.

2.1.3.3 The Synopsys Model

Consider the characteristic equation used by Synopsys [56] in its characterization tables of the following form:

$$r = f(T_{in}, C_l) = a_1 * T_{in} + a_2 * C_l + a_3 * T_{in} * C_l + a_4 \tag{2.2}$$

The circuit response, r (intrinsic delay or output transition time), is a function of the input transition time, T_{in}, and the capacitive load on the output port, C_l. To determine the response of the circuit at a specific input transition time and capacitive load, the two transition time indices of the table, t_r^- and t_r^+, that tightly bound the input transition time of the response to be solved for are located. The same operation is performed for the capacitive load (C_r^- and C_r^+).

$$C_r^- \leq C_l \leq C_r^+ \tag{2.3}$$

$$t_r^- \leq T_{in} \leq t_r^+ \tag{2.4}$$

These four bounding values determine four entries in the characterization table.

$$\begin{aligned}
r^{--} &= table(t_r^-, C_r^-) \\
r^{+-} &= table(t_r^+, C_r^-) \\
r^{-+} &= table(t_r^-, C_r^+) \\
r^{++} &= table(t_r^+, C_r^+)
\end{aligned}$$

Using the table values, the coefficients a_1, a_2, a_3, and a_4 can be determined for the characteristic equation that spans these four points [47].

$$\begin{bmatrix} t_r^- & C_r^- & t_r^- * C_r^- & 1 \\ t_r^+ & C_r^- & t_r^+ * C_r^- & 1 \\ t_r^- & C_r^+ & t_r^- * C_r^+ & 1 \\ t_r^+ & C_r^+ & t_r^+ * C_r^+ & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} r^{--} \\ r^{+-} \\ r^{-+} \\ r^{++} \end{bmatrix} \tag{2.5}$$

Once the characteristic equation has been determined, the interpolated response at the desired input transition time and capacitive load can be calculated. In Figure 2.5, "X" indicates the interpolated value of the response surface defined by these bounding points.

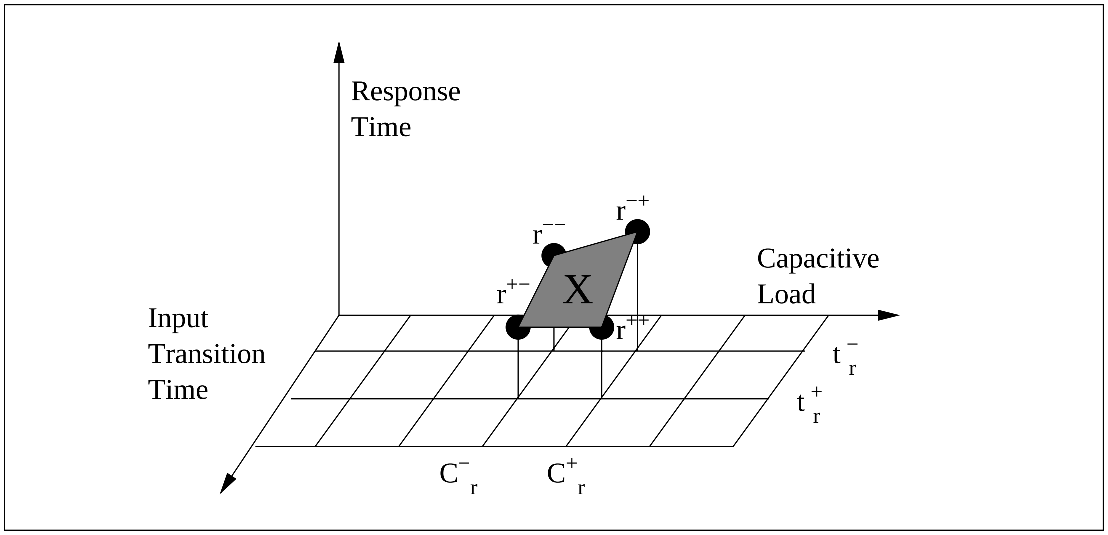


Figure 2.5: The response at point **X** is interpolated using the equation that spans the points in the table that bound it.

2.1.3.4 Other Characterization Models

While characterization tables and characteristic equations represent the bulk of the gate models used in IC design flows, they are by no means the only models or the most accurate for arbitrary loads. In [13] the authors present a simulation model that is evaluated using transient analysis to create a waveform from which intrinsic delay and output transition time can be determined. This model consists of a linear time-varying voltage source in series with a fixed resistance to model a gate. Figure 2.6(a) shows the model. The characteristics for the voltage supply (t_0 and Δt) and driver resistance (R_d) are obtained through characterization as described previously using various input transition times and capacitive loads. This model was later refined in [3], as shown in 2.6(b), to account for nonlinear effects, such as voltage tails, seen in today's nanometer designs.

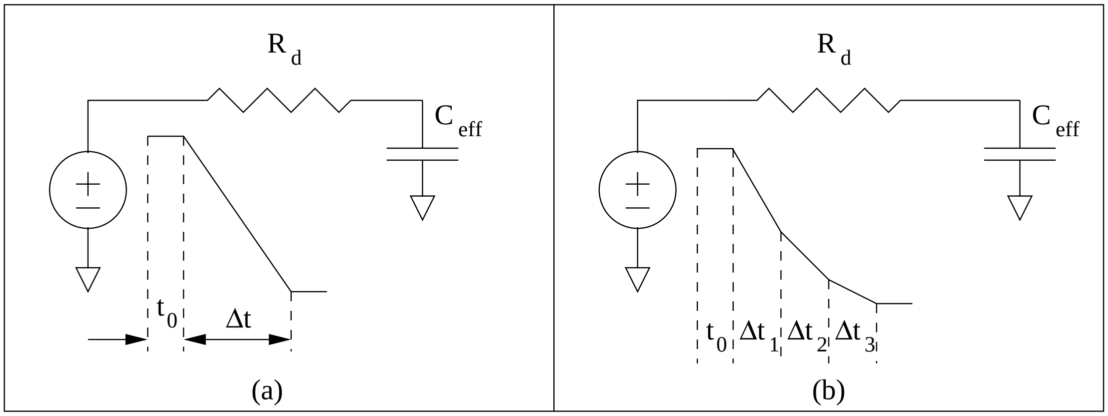


Figure 2.6: The original Thevenin model presented in [13] used a fixed slew rate for the voltage supply is shown in (a). In [3] the idea was extended to a multi-point voltage supply, shown in (b), to obtain better accuracy with respect to SPICE.

The Thevenin equivalent model of [13] is further extended in [15]. The 20%, 50%, and 90% points of the output waveform are measured with respect to the midpoint of the input waveform. The purpose of breaking the single model into two models is to handle situations in which the output voltage exhibits nonlinear tails that could not otherwise be measured. This model attempts to match the output voltage at three points but is unable to predict the waveform shape between these points, making the calculation of output transition time and voltage thresholds difficult.

In [8] the authors develop a new cell model for use when multiple input pins are switching simultaneously. A more extensive characterization system is required that varies transition times and arrival times of signals at multiple input pins simultaneously. This data is then fit to a characteristic equation governing the behavior of the cell. A new static timing analysis application is required to take advantage of this model because cell delay and output transition time cannot be computed prior to all signals arriving at the input pins of the cell. It should be noted that the authors do not address the problem associated with the additional CPU resources required to perform this more exhaustive characterization.

2.1.4 Analytic Modeling

Analytic models for cells fall into one of two categories. In the first, analytic models of the transistors forming the cell are created and evaluated at runtime. In the second, a cell model is created directly. Both models are described below.

2.1.4.1 Transistor Modeling

Circuit simulation programs such as SPICE use detailed transistor models that can be globally applied against transistors of all sizes and characteristics. While this degree of flexibility is required for a general-purpose analysis program, simplified models can be generated once the exact transistor characteristics have been determined for a specific logic function implementation.

Consider the case of a two-input NAND gate. Once a physical implementation of the NAND gate has been created, certain portions of the transistor model that would be used to evaluate a transistor in that gate can be replaced with constants, thus improving circuit simulation time in specialized simulators. Additionally, certain simplifying assumptions can be made once the operating conditions of a design are known. Of course there is usually a tradeoff between the degree of simplification and the attendant loss of accuracy.

In [25] the authors propose an interpolation scheme that can be used to replace explicit evaluations required for traditional BSIM3 transistor models in use today [9]. This interpolation mechanism results in a unified expression across all regions of the transistor. BSIM3 models used in SPICE have multiple equations to represent the behavior of the transistor. The relevant equation used is a function of the voltages at the nodes of the transistor. Application of this interpolating transistor model in a nonlinear SPICE-like application results in near-SPICE accuracy while improving overall throughput.

Because most timing analysis applications assume that only a single input will switch at a time, simple single-stage logic (for example, NANDs, NORs, and AOIs) can be treated as inverters. This simplification is used in [21] to create two transistor models, one NMOS and one PMOS transistor model, which form an inverter to represent the cell. These transistor models are used in conjunction with a simplified π-model representation of the interconnect to simulate the cell's response to input waveforms. The authors report error margins of up to 11% when compared to SPICE. These errors tend to increase for compound cell logic (such as OR and AND gates).

A similar approach is taken in [5]. NAND and NOR gates are replaced by equivalent inverters with transistors that are modeled analytically. Complex gates are reduced to NAND/NOR combinations that are then further reduced into inverters. The proposed model exhibits errors of up to 6% while improving runtimes 12× to 15× over SPICE.

In [49] PMOS and NMOS transistors are replaced with current-source models. The values associated with the models are analytically determined. Four different current-source models are created for a transistor, one for each region of activity based on the gate voltage. The current used to drive the interconnect is the difference between the current flowing in the PMOS and NMOS transistors. The motivation for this methodology was to easily drive AWE-based [45] interconnect models to obtain accurate interconnect delay information. The authors tested their model using an inverter and obtained results showing speedups of 100× over SPICE.

2.1.4.2 Cell Modeling

Unlike transistor models, analytic cell models treat the cell as a single entity. While some characterization may take place to create the analytic model, it is not the traditional characterization methodology used to create empirical models. Empirical models measure delay values directly while analytic models measure electrical parameters (typically current) from which timing information can be calculated.

In [20], the authors present a current model for a CMOS inverter driving a π-model for the interconnect. The current supply approximates the inverter for the time in which the transistors are in saturation. The π-model is used to compute a single effective capacitive load against which the current model is run to calculate cell delay. This model exhibited errors of up to 8% when compared to HSPICE, but it was able to run two to three orders of magnitude faster. The model was not extended by the authors to handle circuits more complex than an inverter.

Yet another current-source model is described in [31]. A single current source replaces the cell and is used to drive a π-model representing the interconnect. The parameters for the current source are analytically determined. The authors report that their models do not accurately model the tails exhibited when driving interconnect with high resistive shielding. Results run typically within 5% to 11% of HSPICE for the more linear regions of the output voltage waveform. The authors only model an inverter, so it is unclear as to the general applicability of the equations governing a cell's behavior.

2.1.4.3 Path Sensitization

For cells with a large number of input pins, exhaustive cell characterization is impractical due to the number of different state-dependent delays possible. For example, characterizing all input pin to output pin delays in an eight-bit adder is combinatorially explosive due to the number of different pin settings possible on the secondary input pins not being characterized. Path sensitization is in itself a large problem with a separate body of research surrounding it. This section discusses a few representative essays.

In [38] the authors describe a symbolic solution to the problem of pin sensitization that uses Elmore delay estimates to weight the delay values associated with transistor switching times. Using this approach, specific paths through the cell can be uniformly described as exhibiting approximately the same delay and a representative path can be characterized using a more accurate simulator to obtain the eventual timing model.

A second approach is presented in [55]. In this methodology the authors attempt to find both the longest and shortest data-dependent delays between two pins through a combination of topological sorting and delay approximation through individual gates that the larger circuit is composed of. Once the paths have been identified, more traditional characterization can occur on those paths.

2.2 Interconnect Delay Analysis

As feature sizes decrease, the gate delay decreases while the effective contribution of interconnect delay increases. Interconnect delay has become the dominant factor in determining circuit performance. The point at which interconnect began to dominate overall delay occurred at approximately 0.8 μm [27].

As interconnect delay importance grew, focus shifted away from cell delay models to interconnect models. Many of the early interconnect analysis approaches used step or linear input transitions to calculate delay and transition times through the interconnect. With today's nanometer designs exhibiting highly nonlinear waveforms at the cells' output pins, these simplifications are no longer valid.

This section discusses interconnect delay analysis and its use in timing verification systems. RAZOR, presented in the next chapter, builds upon existing interconnect analysis methods to produce a fast, highly accurate approach that models any arbitrary waveform used to drive the interconnect.

2.2.1 Early Analysis

In 1948, Elmore [18] created a simple delay model for wideband amplifiers. For a (linearized) amplifier circuit that has a monotonic step response, its impulse response, or transfer function, $h(t)$, is non-negative and can be treated as a probability distribution function.

Given a distribution $h(t)$, the n^{th} moment of the distribution is as follows:

$$\int_{-\infty}^{\infty} t^n h(t) dt \tag{2.6}$$

If $h(t)$ has an area of 1, equation 2.7 defines moment 0. The first moment is then the mean of $h(t)$.

$$\int_{-\infty}^{\infty} h(t) dt = 1 \tag{2.7}$$

To approximate the 50% delay response, T_d, of the interconnect, the median can be calculated. The 50% delay due to a unit step voltage into this system corresponds to the median point of $h(t)$:

$$\int_{0}^{T_d} h(t) dt = \int_{T_d}^{\infty} h(t) dt = \frac{1}{2} \tag{2.8}$$

Elmore proposed that the median of $h(t)$ could be approximated by the mean of $h(t)$ if $h(t)$ is nearly symmetric.

In [52] the Elmore delay, as it was called, was used to analyze a certain class of RC[3] circuits that approximated digital gate and interconnect delays. The

---

[3] The term RC refers to circuits composed only of resistors and capacitors.

authors proved that the impulse response of a tree or mesh of resistors and capacitors is non-negative, thus validating that $h(t)$ could be treated as a probability distribution function.

2.2.2 Advances in Moment Matching

Asymptotic Waveform Evaluation, or AWE [45], uses the moments of the step response to approximate the dominant poles of a linear interconnect circuit. In order to create a q^{th}-order approximation of a circuit, $2q$ moments are computed where q is less than the actual order of the circuit. The calculation of poles and residues from moments relies upon a form of Padé approximation. Once the poles and residues are known, the response of the circuit can be computed as a sum of exponentials.

The development of AWE represents a milestone in interconnect delay analysis as only a few moments are required to approximate signal propagation through interconnect consisting of linear elements. AWE can be used to calculate both driving-point impedance models as well as the response to a voltage input at a node within the interconnect network, even for large networks that challenge the capacity of simulation engines such as SPICE. Furthermore, the creation and evaluation of these models can occur thousands of times faster than SPICE analysis as evidenced by the RICE software described in [50].

While AWE can produce single-input/single-output low-order model approximations for an interconnect network, other moment-matching techniques [43, 54] have since been created to produce multiple-input/multiple-output low-order model approximations. In [43] the authors introduce Passive Reduced-Order Interconnect Macromodeling Algorithm, or PRIMA, to create stable and passive[4] models of the interconnect. Its advantage over the block Arnoldi approach of [54] is the guarantee of passivity.

2.2.3 Timing Analysis

Once a low-order model of the interconnect has been created, it must be used in a system to analyze timing through the interconnect. Most existing empirical models use a linear ramp to approximate the output voltage of a cell. This saturated ramp[5] can be applied to the low-order interconnect model to quickly obtain time/voltage values for nodes within the interconnect network.

A linear approximation for the voltage waveform at a cell's output pin is only valid if the ramp closely approximates the actual voltage through the important regions of interest. Output transition times, in the late 1980's, were measured between the 10% and 90% points of V_{DD}. As feature sizes shrank, so, too, did the linear portion of the output voltage waveform. By the early 1990's, the output transition time was typically measured between the 20% and 80% points. Today it is not uncommon to see these measurements taken at the 30% and 70% points because of the nonlinearities associated with nanometer design.

Approximating the output transition time by this continuously narrowing linear voltage region is both impractical and inaccurate. The shape of the waveform outside of this window can be very important to the performance of a gate being driven. With the greater emergence of nonlinear waveforms, a solution must

---

[4] A passive system is one that cannot generate energy.

[5] A normalized input voltage that swings from zero to one with some slope and then remains constant afterwards is referred to as a saturated ramp.

be found that both comprehends and propagates a better waveform specification.

Little literature exists that describes the simulation of an arbitrary voltage waveform through interconnect. It can be argued that such efficient simulation techniques have not previously been required due to the lack of cell models that can produce or consume such waveforms. The approximation of nonlinear waveforms with a linear ramp has recently become a large source of error, however, and it is believed that this source of error must be addressed for accurate timing verification in nanometer designs.

Recursive convolution [4] was introduced in 1992. In this approach, a voltage waveform can be described using a series of line segments as shown in Figure 2.7. The slopes of these ramps can be used as multipliers to solve the generalized system. The overall response can be determined by summing the multiple zero-state ramp responses of each segment up to the point in time targeted for analysis[6]. Recursive convolution, as described in [4] is highly accurate but, for a large number of line segments, also can be highly complex requiring a large number of floating-point calculations at each time point.

FTD [34] is another approach to the propagation of piecewise linear representations of a voltage waveform through interconnect. It directly solves the exact solution at a time point t_k by using the solution at time t_{k-1} and assuming a linear response for a reasonably small timestep Δt.

---

[6] Recursive convolution will be described in more detail in the next chapter with the introduction of RAZOR.

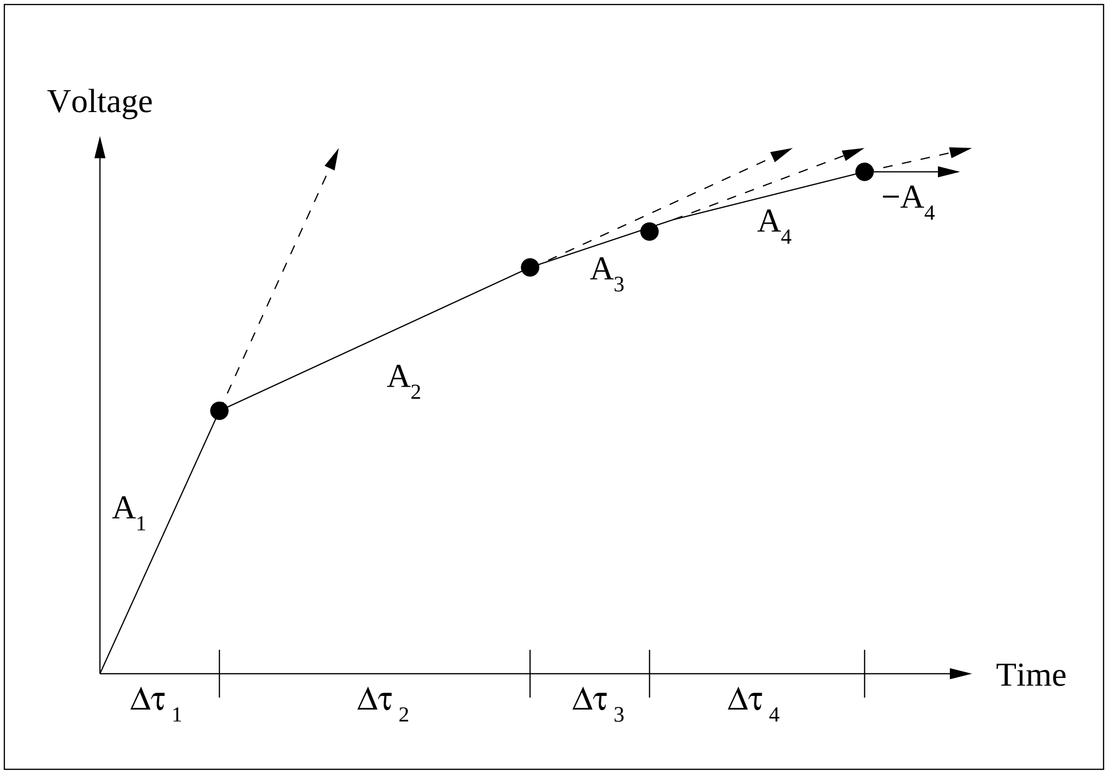


Figure 2.7: A series of ramps and their respective time offsets can be used to fully describe a voltage waveform.

2.3 Circuit Simulators

This section describes some of the better-known circuit simulators. These simulators are designed to evaluate entire blocks of logic including gates and interconnect. This list is not meant to be exhaustive but is meant to provide an overview of the technology used to simulate large digital systems as cell and interconnect models are only as good as the systems that consume them.

SPICE [40] is the reference platform against which all other simulators compare themselves. Since its introduction into the IC design flow, SPICE has become the final arbiter with respect to the electrical behavior of a design. Entire software industries have developed around SPICE engines as well as the creation of models used by SPICE during simulation.

SPICE reads a netlist[7] of the circuit and builds a matrix representing the electrical connections and components within that system. Elements like transistors and diodes are replaced by simplified models (companion models) that, themselves, may in turn be further simplified with their companion models (like the capacitor). Complex differential equations govern the behavior and values associated with these companion models. In order to guarantee system stability, implicit integration techniques such as trapezoidal integration are used [37].

At each time point, the matrix representing the circuit is built and values stored within it. Newton-Raphson iteration (Figure 2.8) is used to converge upon a set of nodal voltages and element currents. These voltages and currents are stored and analysis begins anew at the next time point using the previous solution as the initial value for the next simulation.

Though SPICE is highly accurate, the generation and inversion of matrices during each Newton-Raphson iteration greatly impacts its execution time and memory consumption. Additionally, the complexities of the equations governing element behavior noticeably impact the simulation time. For example, there are often many different models for a NMOS or PMOS transistor, the selection of which is determined by characteristics specified in the netlist for that transistor instance. Furthermore, there may be well over 100 different variables governing the behavior of each transistor model. Many of the high speed and/or high capacity

---

[7] A netlist is a description of electrical components, such as resistors and capacitors, and the way that they connect to one another to form a circuit.

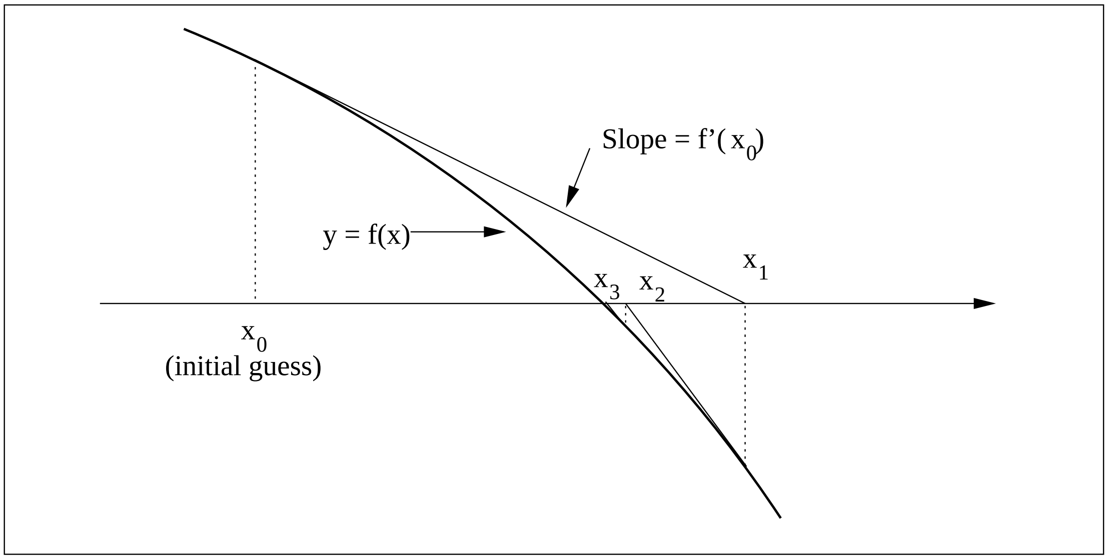


Figure 2.8: Newton-Raphson iteration is used to calculate the solution of $f(x) = 0$. The X intercept of the tangent of $f(x_n)$ is used to determine the next guess, x_{n+1}.

simulators achieve their improvements over SPICE by simplifying the models, the numerical integration process, matrix partitioning, or some combination of these [30].

The matrices that SPICE builds to represent the circuit tend to be highly sparse. In general matrix inversion is an $O(n^3)$ operation [10], and, while the original matrices are sparse, their inverses are not. In [32] sparse matrix techniques are described that take advantage of the characteristics of these matrices to greatly accelerate the operations performed on them during circuit simulation. These algorithms were incorporated into Berkeley's SPICE3 circuit simulator.

In [41] the authors describe a method for increasing the convergence rate of the Newton-Raphson iterations by altering the method by which the next guess is calculated. Using a methodology called "error-function-curvature driven Newton update correction," the number of iterations required for Newton-Raphson convergence can be dramatically reduced and the importance of the initial guess is minimized[8]. Their technique is generally applicable across a variety of different nonlinear simulators.

Crystal [44] is a timing analysis program that replaces detailed transistor-level analysis with fast lookup-table timing approximations. Furthermore, instead of responding only to explicit stimulus as input from the user, Crystal performs an analysis of the netlist to determine the worst-case timing path through the logic. While worst-case delay values are also propagated in STA, Crystal does so only if the path it is searching is active within the circuit. Distinguishing between true and false paths is not normally a function of static timing analysis. Instead, the user typically creates a list of false paths to use during a post-processing stage to remove those paths from any list of critical paths that may have been found.

Forward Euler integration is not used in SPICE during circuit simulation because the timestep required to guarantee system stability could be so small as to make the total simulation time unreasonable [46]. In Adaptively Controlled Explicit Simulation, or ACES [17], the authors use an approximation for Forward Euler and adaptively control the timestep used during simulation to guarantee stability. The system also exploits spatial latency by partitioning the system (reducing matrix sizes) and omitting those portions of the system that do not change between timesteps.

Motis [6], a timing simulator developed by AT&T Bell Laboratories, was

---

[8] The convergence rate for Newton-Raphson iteration is highly dependent upon the initial guess used.

one of the first simulators to use multiple simplifications to speed transient analysis including the use of simplified models for MOS transistors and event-driven simulation. Unlike SPICE, which used differential equations to model NMOS and PMOS transistors, Motis used simpler table-lookup models that could be rapidly evaluated. These transistor models were derived through precharacterization of specific gate types. Once the individual transistor models were calculated, the entire gate was replaced with a model consisting of two current sources: one for the PMOS section and one for the NMOS section. Recomputation of the gate equivalent only took place when event-driven simulation indicated a significant change in the gate environment.

ILLIADS [53] is a circuit simulation program designed to evaluate circuits that are too large for traditional simulators such as SPICE. In addition to increasing capacity, ILLIADS exhibits dramatic runtime improvements over SPICE through the use of analytic solutions for differential equations in the innermost analysis loops. A new MOS circuit primitive is also used to significantly improve ILLIADS performance without sacrificing significant accuracy.

TETA [1, 14], or Transistor-Level Engine for Timing Analysis, is a SPICE-like simulator targeted for interconnect analysis. In modern designs, interconnect delays account for both a large portion of the overall path delay as well as a large portion of the memory and CPU time required during circuit simulation. TETA is designed to efficiently analyze general circuits with dominant interconnect characteristics by replacing the gates with models that are more efficient for interconnect analysis. Individual transistors are precharacterized to create current models that replace the transistors in the netlists. The interconnect is replaced by an N-port representation derived through moment-matching techniques. Simulation occurs in two stages. In the first stage, the interconnect is replaced by a voltage supply and the cell state is solved for using the simplified transistor models. In the second stage the gate is replaced by a current supply and the interconnect is modeled using the N-port representation. These two stages iterate until the system stabilizes at which point the next timestep is made.

In [58] a timing simulator is proposed that uses functional information about the circuit to propagate individual gate delays. Control inputs are provided by the user and the data-dependent delays are propagated with respect to the functional modes determined by these control inputs. While relatively high error margins of 25% or more are possible using this technique, this algorithm can be used as a first approximation to determine the subset of paths that must be analyzed further using a more accurate timing verification methodology.

2.4 Conclusion

A large body of research has developed around cell and interconnect modeling as well as circuit simulation. SPICE is the reference simulator against which all models and accelerated simulators are measured. While SPICE produces highly accurate electrical measurements, long simulation times and vector-dependent results make its use on all but the most timing-critical paths impractical.

To accelerate timing verification, cell and interconnect models and simulation environments have been developed that attempt to address the capacity and CPU time limitations of SPICE. In general, most approaches to date have sacrificed accuracy for speed or capacity.

In the next chapter a new method for cell characterization is presented, called over sampling and data reduction, that improves the accuracy of characterization tables used in timing verification systems today. Recognizing that a better model is required in the back end of the design flow that can handle today's nanometer effects accurately and quickly, a new cell and stage delay modeling system is proposed: BLADE and RAZOR. The system is composed of both the models and the runtime environment within which the models are used.

Chapter 3

Model Development

Cell timing models used in production IC design flows have not appreciably changed in the last 15 years. While the formats vary somewhat, ranging from characteristic equations to table models, they are fundamentally the same in terms of their inputs and outputs. Inputs to the cell model typically include a single capacitive load value and a single value to approximate the slope of the input voltage. The models produce a single value for the intrinsic delay and another for the approximate slope of the output voltage across the capacitor the cell is loaded with.

When these models were originally developed, they provided both the evaluation speed and the accuracy required by integrated circuit designers. Unfortunately, while these models still evaluate rapidly, they are fundamentally unable to accurately account for nanometer effects that must be considered today. The interconnect delay, for example, is a much larger component of overall delay and, with the presence of high resistive shielding, the voltage waveform profiles of a cell are no longer linear. Cell characterization methodologies that use a simple capacitor to gather empirical data are not able to capture these nonlinear effects.

Cell models are used throughout the entire IC design flow. Their inability to accurately model today's silicon makes their use in the latter portion of the flow problematic as the designer must add sufficient guardband to the design to account for these deficiencies. Clearly a new cell model is required that exhibits both accuracy and evaluation speed. As interconnect delay accuracy cannot be sacrificed, such a model must also be able to produce results that can be consumed by a highly accurate interconnect model. Existing cell models are still useful in the early portion of the design flow, where evaluation speed is crucial, but they must also be as accurate as possible so that both front-end and back-end models produce consistent results throughout the entire flow.

This chapter is divided into three primary sections. The first section introduces a novel cell characterization technique, over sampling and data reduction, which maximizes the potential accuracy of table-lookup cell models. Specifically, only by determining the response surface of a cell can accurate characterization tables be constructed that best approximate the surface with very little loss of accuracy. Table-lookup models are used today throughout most IC design flows.

The second section describes BLADE, a new cell model and evaluation engine. Characterization for the BLADE model is faster and simpler than characterization for table-lookup models or characteristic equations. BLADE models are also able to produce and consume nonlinear waveforms and drive arbitrary loads at or near SPICE accuracy with runtimes that are tens of thousands of times faster than SPICE. The performance and accuracy of the BLADE model is such that it could easily replace the traditional table-lookup model or characteristic equation model in the back end of the integrated circuit design flow when accuracy is more crucial than raw model evaluation speed.

The third and final section details the interface between the BLADE model and the interconnect model. Traditionally this interface has been the source of accuracy loss. As Blade produces nonlinear waveforms, the interconnect model needs to efficiently consume this data to produce equally accurate waveforms for the cells being driven. This section describes an interface, Razor, based on poles and residues, that consumes arbitrary waveforms produced by Blade without sacrificing accuracy, speed, or memory. Razor also produces voltage waveforms that the Blade model can, in turn, directly consume without loss of accuracy. This combination of cell and interconnect delay calculation is called stage delay calculation.

3.1 Cell Characterization and Modeling

The accuracy of a cell characterization table is a function of the points stored within the table from which interpolated values are calculated. This section addresses the topic of accuracy by focusing on the answers to two questions: Over what range of values should characterization occur? Within that range, which and how many points should be selected for measurement?

To answer these two questions, the concept of over sampling and data reduction is introduced. Over sampling is used to create a detailed view of a cell's response surface by measuring a cell's electrical behavior at many different points beyond just those stored within the final model. Data reduction is a dynamic programming solution to reduce this detailed response surface to a manageable size for use by CAD applications. Two different approaches to data reduction are presented. The first approach is CPU intensive but produces highly accurate results when compared to the original data. In the event that CPU time is not plentiful, a second approach is presented which increases error margins only slightly over the first but executes substantially faster.

3.1.1 Auto-Ranging

A cell library is often composed of only a few dozen different logic functions. However, each logic function may be represented many times with different physical implementations. For example, there can be as many as 20 different inverters in a typical cell library.

The physical implementation of a cell determines its electrical, temporal, and spatial characteristics. For example, a buffer composed of very small transistors may not be able to source or sink enough current within a given time period to make it a suitable choice as a clock buffer driving a large interconnect network. However, such a buffer might be ideal to drive a small network with very few cell loads. The ability of a cell to drive a load is referred to as a cell's *drive strength*. Cells built with larger transistors tend to have faster output transition times and smaller intrinsic delays than those built with smaller transistors for a given load. On the other hand, larger transistors require more area on the integrated circuit and consume more power.

A library has a variety of different implementations for a given logic function so that the smallest implementation that meets the design goals can be used. Using a cell that has a larger drive strength than required to meet timing constraints leads to higher power consumption and chip area. These qualities, in turn, affect consumers in terms of battery life for their cellular phones and/or cost of their electronic products, for example.

The job of the engineering team designing a robust cell library is to provide a large enough selection of drive strengths that can, when used in the integrated circuit, meet the electrical and economic design goals. Knowing that a given cell implementation is designed to function optimally under a certain set of conditions, the characterization engineer, or librarian, need only create a model within that target operational range.

Auto-ranging is the process of automatically determining what conditions a cell is designed to operate under. Once these limits have been determined, characterization can occur within these established parameters to create the model. For cell models that are functions of the input transition time and the capacitive load on the output, four limits must be determined.

3.1.1.1 Capacitive Limits

To determine the upper-bound capacitive load that a cell must be able to drive, the timing goals for the integrated circuit can be used. Consider the case of a microprocessor operating at 1GHz. The clock period for such a processor is 1 nanosecond. There may be eight or more levels of logic, between two sequential elements, that must be evaluated within this clock period.

Design parameters can be used by the librarian to limit the maximum capacitive load that a cell is characterized against. In this fictional microprocessor, for example, an output transition time of 1 nanosecond is unreasonable as the signal would be unusable in a 1-nanosecond clock period. Armed with specific design goals, the librarian can make an informed determination of the maximum output transition time that a cell should produce in the integrated circuit. This, in turn, can be used to automatically drive simulations that determine the maximum capacitive load placed on the output of the cell to obtain this maximum output transition time. Because the output transition time is also a function of input transition time, the cell should be simulated using the fastest reasonable input transition time to determine the maximum capacitive load that the cell can drive.

The minimum capacitive load can be determined through inspection of the cell library. The minimum load a cell might drive would be the active load that occurs when a cell adjoins the cell it drives. Thus, the lower capacitive limit for a cell would be the effective input pin capacitance of the smallest cell it could reasonably be expected to drive in this fashion[1].

3.1.1.2 Input Transition Time Limits

In the previous section the slowest output transition time was determined by the librarian. A good approximation for the slowest input transition time would be this output transition time. While it is true that transition times degrade over interconnect, too much of a degradation would cause the design to fail to meet its timing constraints.

The fastest input transition time can be determined through simulation. The fastest input transition times exhibited by a cell occur when the cell is very lightly loaded and is driven by a large cell through abutment.

3.1.1.3 Input Voltage Waveforms

For an empirical model to be accurate, the conditions under which the measured data is gathered must closely approximate the conditions under which

---

[1] Often this value is so small that a value of 0 picofarads can be used as well.

the cell is expected to operate within the IC. In older generations of IC fabrication technology, saturated linear ramps could closely approximate the actual voltage waveforms created by a cell. However, the decrease in transistor feature sizes and increase in interconnect load has resulted in output transitions much less linear in nature. Therefore, in order to accurately capture electrical data about a cell, the input transitions to the cell during characterization must reasonably approximate these nonlinear waveforms.

The simplest way to create input waveforms that approximate those expected during normal operation is to use a cell from the cell library as a driver cell to create these waveforms. However, directly coupling the output of the driver cell to the cell being characterized may not yield predictable input transition times. When the cell being characterized is loaded, coupling back to the input pin will result in an alteration of the input waveform. In other words, two different input waveforms can be created by the driving cell by simply changing the capacitive load on the cell being characterized while all other conditions remain unchanged.

The solution to this dilemma is shown in Figure 3.1. By separating the driving cell that creates the input waveform from the cell being characterized, C_l coupling cannot alter the waveform shape because the voltage-controlled voltage source (VCVS) is an ideal source generating a forced response. This driver cell is precharacterized for a specific (fixed) input transition time and a variety of capacitive loads. During the characterization of a cell, a value for $C_{rise/fall}$ is chosen such that a waveform with the desired slew rate is produced by the driver.

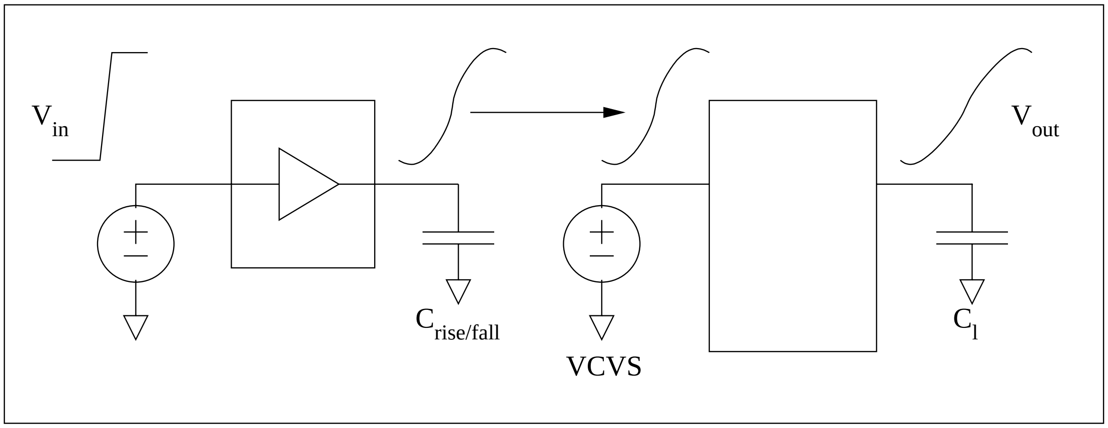


Figure 3.1: A buffer is used as a driver cell to create a realistic voltage waveform for use in characterization. By altering $C_{rise/fall}$ the input transition time to the cell being characterized can be altered. A VCVS guarantees that C_l coupling to the input pin will not alter the shape of V_{out}.

3.1.1.4 Putting It All Together

Using just a few pieces of information from the user, the cell's upper characterization limits can be automatically determined. This operation can be performed for each unique characterization table being generated.

To determine the fastest input transition time, the fastest buffer in the cell library should be used to directly drive the input pin being characterized. In the event that a buffer is not the highest drive-strength cell in the library, a buffering function can usually be created by tying the secondary inputs of the largest drive-strength cell to V_{DD} or V_{SS} as appropriate and/or by creating a compound cell by prepending an inverter to the input. The output pin should be loaded with the minimum capacitive load that might be driven by that cell in the design.

Using this same netlist configuration, another simulation can be performed to determine the capacitive load for which the output transition time reaches the predefined limit. Many simulators perform this type of optimization function directly.

The other two values, the minimum capacitive load and the slowest input transition time, can be specified by the user directly. Unlike the other two parameters, these may be more globally applicable and need not change from cell to cell.

3.1.2 Over Sampling and Data Reduction

Once a cell's characterization limits have been determined, the selection of points to measure within those limits must still be made. Some cells may have a nearly linear response surface for intrinsic delay and output transition time while others may not. Without actually knowing the characteristics of the response surface, it is not possible to determine beforehand where or how many measurements should be made to obtain the best possible representation in the characterization table.

One rather obvious solution to the problem would be to create a very large characterization table of sufficient granularity to ensure that interpolation errors are minimized. Modeling the surface with a smaller table introduces a greater chance of interpolation error than with a larger table. However, program memory and runtime constraints place practical limits on the size of the tables. Obviously, smaller tables are less of a drain on system resources than larger tables. Random table lookups also take longer for bigger tables than for smaller tables.

Essentially, there are only two methods available to improve overall characterization table accuracy while maintaining reasonable table sizes. The first method is an iterative one. A set of points can be selected and the cell's response measured at these points. Then, using Monte Carlo simulation, a second set of points can be selected and measured. These measured points can be compared to the interpolated value from the characterization table and the errors then determined. If the error exceeds some predetermined margin, the selection of points can be refined to attempt to reduce the error, and the process repeated.

The second method, called over sampling and data reduction, draws its origins from the first. In the first method, a number of points were measured, some of which were used within the characterization table and others used for error analysis. Instead of iterating in this fashion to converge upon a solution, over sampling and data reduction is the process of performing many more measurements of the cell's response to obtain a highly accurate representation of the overall response surface. Once this data has been gathered, entire rows or columns of data that can be interpolated without significant error can be removed from the table. Error calculation becomes part of the process in determining which measured points are no longer needed, thus removing the need for separate error analysis.

The difficulty with over sampling and data reduction is that the selection of any row or column for removal affects any subsequent selection. Thus, a greedy selection of rows or columns for removal will not necessarily yield the smallest tables while maintaining desired error margins.

3.1.2.1 Error Calculation

In trying to determine which indices of a large table to use in the final, reduced table, the error must be calculated for those indices used in the reduced table and compared against any other potential indices. Thus, any point $R_{i,j}$ in the response table, R, represents two potential indices (i for the transition time and j for the capacitive load) in the reduced table. Stated another way, any point $R_{i,j}$ may be the lower-left corner of a rectangle that defines the reduced table area coverage. Therefore, the first step is to determine the error associated with choosing point $R_{i,j}$ and any point above and to the right as the bounds of a reduced table entry. This is done by calculating the coefficients describing the corners of this rectangle, interpolating for any points in the rectangle, and summing the difference between the interpolated values and the measured values from SPICE as shown in Algorithm 3.1.

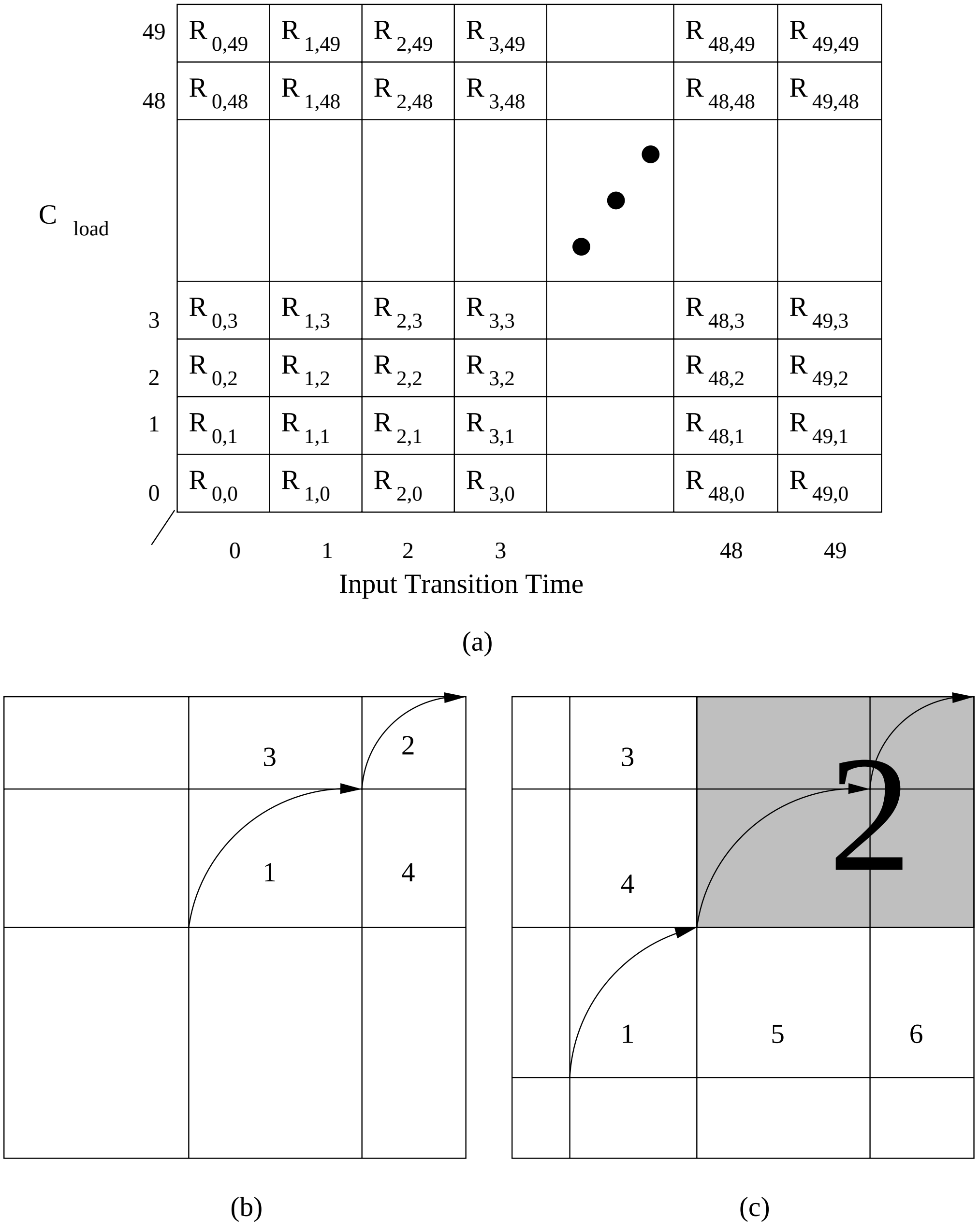
As an example, consider points $R_{1,0}$ and $R_{3,2}$ in Figure 3.2(a). The coefficients of the characterization equation are calculated using points $R_{1,0}$, $R_{3,0}$, $R_{1,2}$, and $R_{3,2}$, the four corners of the rectangle defined by $R_{1,0}$ and $R_{3,2}$. Interpolation error for this region is determined by comparing the SPICE measured data within this region to the interpolated values. For the region defined by $R_{1,0}$ and $R_{3,2}$, interpolated values are calculated for indices that correspond to points $R_{1,0}$, $R_{2,0}$, $R_{3,0}$, $R_{1,1}$, $R_{2,1}$, $R_{3,1}$, $R_{1,2}$, $R_{2,2}$, and $R_{3,2}$. The differences between each of these values and the SPICE measured values in the large table are summed as the error associated with this region.

A complete error table can be calculated for the response surface using this procedure for every point in the table to every point above and to the right. This

Algorithm 3.1 Data reduction relies upon the creation of a table that stores interpolation errors. Any two points in the original measured data can define a rectangular region which may be part of the reduced response surface. The error associated with this region is the sum of the differences between the interpolated values in this region and the SPICE measured values.

```
Require: Measured SPICE data in array $Data$
  $ErrorTable \Leftarrow \emptyset$
  for $I = 0$ to $NumLoads - 2$ do
    for $J = 0$ to $NumInputSlews - 2$ do
      $FromIndex \Leftarrow I \times NumInputSlews + J$
      for $K = I + 1$ to $NumLoads - 1$ do
        for $L = J + 1$ to $NumInputSlews - 1$ do
          /*
           * Calculate coefficients of the characteristic equation
           * for the region defined by the four points
           * $Data[I][J]$, $Data[I][L]$, $Data[K][J]$, and $Data[K][L]$
           */
          $ToIndex \Leftarrow K \times NumInputSlews + L$
          $CalculateCoefs(Data, I, J, K, L, a_1, a_2, a_3, a_4)$
          /*
           * Calculate interpolation error and store as the
           * error of this region
           */
          $Error \Leftarrow 0$
          for $M = I$ to $K$ do
            for $N = J$ to $L$ do
              $Error \Leftarrow Error + InterpError(Data[M][N], a_1, a_2, a_3, a_4)$
            end for
          end for
          $ErrorTable[FromIndex][ToIndex] \Leftarrow Error$
        end for
      end for
    end for
  end for
```

table is used in the next phase of data reduction to obtain the final cell model.



C_{load}							
49	$R_{0,49}$	$R_{1,49}$	$R_{2,49}$	$R_{3,49}$		$R_{48,49}$	$R_{49,49}$
48	$R_{0,48}$	$R_{1,48}$	$R_{2,48}$	$R_{3,48}$		$R_{48,48}$	$R_{49,48}$
					● ● ●		
3	$R_{0,3}$	$R_{1,3}$	$R_{2,3}$	$R_{3,3}$		$R_{48,3}$	$R_{49,3}$
2	$R_{0,2}$	$R_{1,2}$	$R_{2,2}$	$R_{3,2}$		$R_{48,2}$	$R_{49,2}$
1	$R_{0,1}$	$R_{1,1}$	$R_{2,1}$	$R_{3,1}$		$R_{48,1}$	$R_{49,1}$
0	$R_{0,0}$	$R_{1,0}$	$R_{2,0}$	$R_{3,0}$		$R_{48,0}$	$R_{49,0}$
	0	1	2	3		48	49

Input Transition Time

(a)

(b)

(c)

Figure 3.2: Traversal of the characterization table. (a) Resource table. (b) Two steps from intermediate point. (c) Three steps from intermediate point.

3.1.2.2 Reduction of Characterization Tables

In order to reduce a large characterization table into a smaller characterization table, a dynamic programming solution is used [24]. Consider, for example, the reduction from a 50×50 table to a 10×10 representation. In the previous section the error was calculated from any point to another above and to the right in one step. It is now possible to estimate the error from any point to $R_{49,49}$ in two steps, as shown in Figure 3.2(b). The approximate error is the error between that point and any intermediate point above and to the right plus the error between the intermediate point and $R_{49,49}$ (regions one and two in Figure 3.2(b)). Additionally, the selection of the intermediate point implies that its two indices would also divide the entire table, so the error associated with regions three and four must also be included.

This method of searching is performed for every point in the table for any two steps. For any point in the table, the algorithm finds the intermediate point that it can step to such that the accumulated error is minimized for all regions above and to the right of it. This value and a pointer to the intermediate point is now stored on that point as the path from that point to $R_{49,49}$ in two steps.

This process is repeated for three steps, as shown in Figure 3.2(c). To go from any point to $R_{49,49}$ in three steps, the error can be approximated by the error from that point to an intermediate point and the error from that point to $R_{49,49}$ in two steps. Because all two-step data has been calculated and stored, this value need not be recalculated. The selection of an intermediate point implies the acceptance of two intermediate table indices, the error for which must also be included. This corresponds to regions three, four, five, and six in Figure 3.2(c).

This entire process is repeated until the data for nine steps is calculated for every point as shown in Algorithm 3.2. It is now a simple matter to trace the path from point $R_{0,0}$ to $R_{49,49}$ in nine steps to arrive at the ten indices to use for the table. The error associated with nine steps from point $R_{0,0}$ is the error associated with using these indices with respect to the large 50×50 table.

An alternative approach is to select the indices for the input transition time in one pass and the capacitive load indices in a second. This method causes some loss in accuracy but requires significantly fewer computing resources both in terms of compute time and memory.

In this algorithm, dynamic programming is used to determine the indices. However, instead of calculating the characteristic equation between every point pair in the table, the equation is only generated for points that span entire rows or columns of the table. Thus, every point in the bottom row of the table is matched with every point on the top row and to the right of it. The characteristic equation is generated and errors measured. Dynamic programming is used to determine the boundaries between these equations in the manner previously described. The same approach is used to determine column indices independently.

3.1.3 Conclusion

Creating characterization tables without knowledge of the response surface being approximated can lead to large errors when comparing an interpolated response against the exact response as calculated by a reference program such as SPICE. While Monte Carlo simulation can be used to refine the data gathered in this manner, it cannot provide the guarantee of a certain margin of error.

Algorithm 3.2 A reduced response table is derived from the measured data by applying dynamic programming. After completion, the reduced table indices can be found by tracing the *Next* fields of *StepTable*.

```
/*
 * Initialize the table with precalculated error values
 */
$StepTable \Leftarrow \emptyset$
$LastEntry \Leftarrow NumLoads \times NumInputSlews - 1$
for $I = 0$ to $NumLoads - 1$ do
  for $J = 0$ to $NumInputSlews - 1$ do
    $StateIndex \Leftarrow I \times NumInputSlews + J$
    $StepTable[StateIndex][0].Err \Leftarrow ErrorTable[StatIndex][LastEntry]$
    $StepTable[StateIndex][0].Next \Leftarrow LastEntry$
  end for
end for
/*
 * Now calculate errors from any point in the table to
 * the corner using previously calculated error data.
 */
for $S = 1$ to $NumSteps$ do
  for $I = 0$ to $NumLoads - (S + 2)$ do
    for $J = 0$ to $NumInputSlews - (S + 2)$ do
      $FromPt \Leftarrow I \times NumLoads + J$
      $StepTable[FromPt][S] \Leftarrow \emptyset$
      for $K = I + 1$ to $NumSteps - (S + 1)$ do
        for $L = J + 1$ to $NumInputSlews - (S + 1)$ do
          $NextPt \Leftarrow K \times NumLoads + L$
          $Err \Leftarrow ErrorTable[FromPt][NextPt]$
          $NextErr \Leftarrow StepTable[NextPt][S - 1].Err$
          $SideErr \Leftarrow SideError(ErrorTable, FromPt, NextPt, StepTable)$
          $TotalErr \Leftarrow Err + NextErr + SideErr$
          if $StepTable[FromPt][S].Err > TotalErr$ then
            $StepTable[FromPt][S].Err \Leftarrow TotalErr$
            $StepTable[FromPt][S].Next \Leftarrow NextPt$
          end if
        end for
      end for
    end for
  end for
end for
```

By performing a much more exhaustive set of simulations to obtain a complete view of the response surface, a better selection of representative points can be made for the characterization table that more accurately represents the entire surface. Even with the surface known, however, the selection of points can be shown to be NP-hard (the selection of any single point in the table has an impact on the selection of all points in the table). By using a dynamic programming solution to identify these indices, a near-optimal solution can be determined in polynomial time with very small error margins when compared to the original data.

3.2 BLADE

Over sampling and data reduction can provide a very accurate view of the response of a cell when compared to the SPICE data used to create the characterization table. However, that data is empirically gathered using a certain profile for the input voltage and for the capacitive load. When the model is used in an environment for which this profile is no longer valid, the interpolated response produced by the model may exhibit large errors when compared to silicon performance. For those portions of the IC design flow that need a greater guarantee of accuracy, a model that is independent of these constraints and that also evaluates rapidly is required.

Research for advanced models that meet both accuracy and runtime requirements has traditionally followed one of two paths: enhanced empirical models and SPICE-like evaluation engines. Until now, neither has sufficiently met both requirements well enough to displace the traditional characteristic equation or characterization table from the back end of the IC design flow.

While there have been several empirical models proposed, most are extensions of the existing models. A voltage waveform is used as an input to the cell, some load is placed on the output of the cell, and the electrical characteristics of the cell are measured. Unfortunately, despite increasing sophistication, these types of empirical models still tend to produce inaccurate results when they are used in environments in which the input voltage or load characteristics do not match those used during model creation. For example, noisy input voltage waveforms can dramatically impact the actual performance of a cell, but cells are not normally characterized using such inputs.

SPICE-like evaluation engines tend to produce better results over a broad range of usage models and environments, but only at the expense of CPU runtime and memory. While engines exist that run many times faster than SPICE, these engines often achieve their speed enhancements through the use of simplified transistor models. Simplified transistor models, by their very nature, tend to be less accurate than those models used by SPICE. These small errors can be compounded during transient analysis as there are multiple transistors in a gate and multiple timesteps over a single transient run. Additionally, even with speed improvements of 100× over SPICE, these engines are still too slow to apply against large portions of the integrated circuit design.

The Blade model follows a hybrid approach. A detailed I–V model[2] of the cell is empirically created by performing DC sweeps in SPICE. Because a DC analysis does not capture the effect of internal capacitance or internal interconnect

---

[2] An I–V model is a current (I) and voltage (V) model.

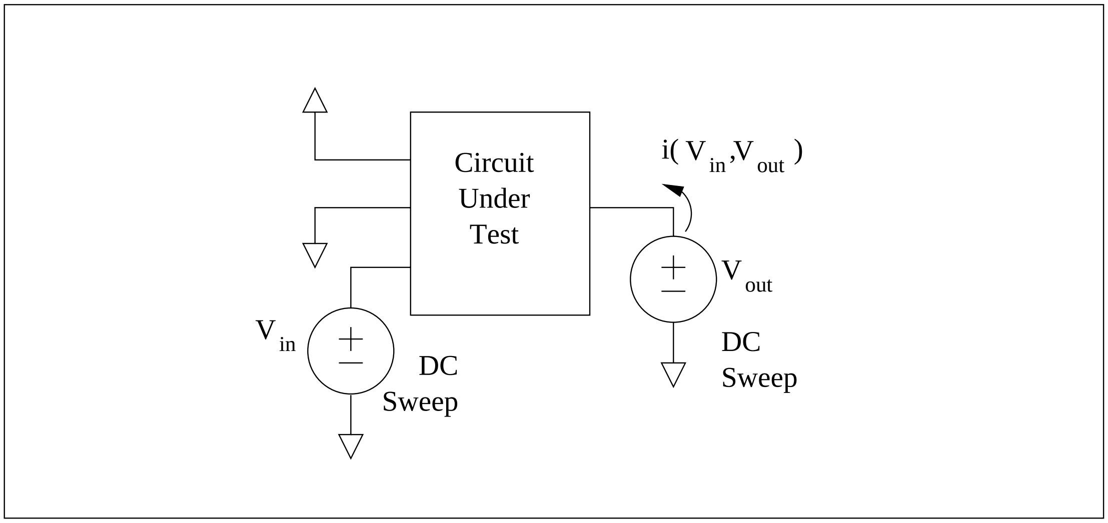


Figure 3.3: To create the Blade I–V table, voltage supplies are attached to the input and output pins and the current flow is measured.

delay, a subsequent transient analysis is made to gather data from which internal parasitics and delay values can be modeled. Once all data has been empirically gathered, the model is evaluated in the Blade runtime engine.

3.2.1 The Blade Current Model

The Blade model is a current-based model. Instead of measuring a voltage waveform output by the cell in response to a voltage input and a capacitive load, the ability of the cell to source or sink current in response to a voltage level on the input and a voltage level on the output is measured. In this manner a two-dimensional table can be constructed. One index is the voltage level on the input and the second is the voltage level on the output. The process is illustrated in Figure 3.3.

The two voltage supplies attached to the cell are independently swept from

V_{SS} to V_{DD}. The other pins of the cell are set such that a transition at the input pin would cause a transition at the output pin[3]. To obtain maximum accuracy, the voltage stepsize must be sufficiently small so as to fully capture the current response surface. Unlike transient analysis, performing this type of DC sweep is extremely fast in SPICE. Thus, a large amount of data can be gathered rapidly so as to fully capture the I–V characteristics of the cell.

In the Blade model the data structure into which the current is stored is identical to the characterization table described in the previous section. Thus, to conserve space, the technique of over sampling and data reduction can be performed on this I–V table, if so desired.

The end result of this phase of the Blade characterization process is a voltage-controlled current-source model of a pin-to-pin path through the cell, controlled by the input and output voltages. To completely model a cell, multiple Blade pin-to-pin models are required, one for each path that exists within the cell.

3.2.2 The Blade Runtime Engine

Before developing the Blade model further, it is important to understand how this model can be evaluated. There are two important aspects to the Blade runtime engine. The first is the nonlinear solver that is used to calculate the Blade model's response. The second is the formulation of the problem on which the nonlinear solver operates. Both are described in the following sections.

---

[3] In the event that there are several such settings, state-dependent models can be created, if so desired.

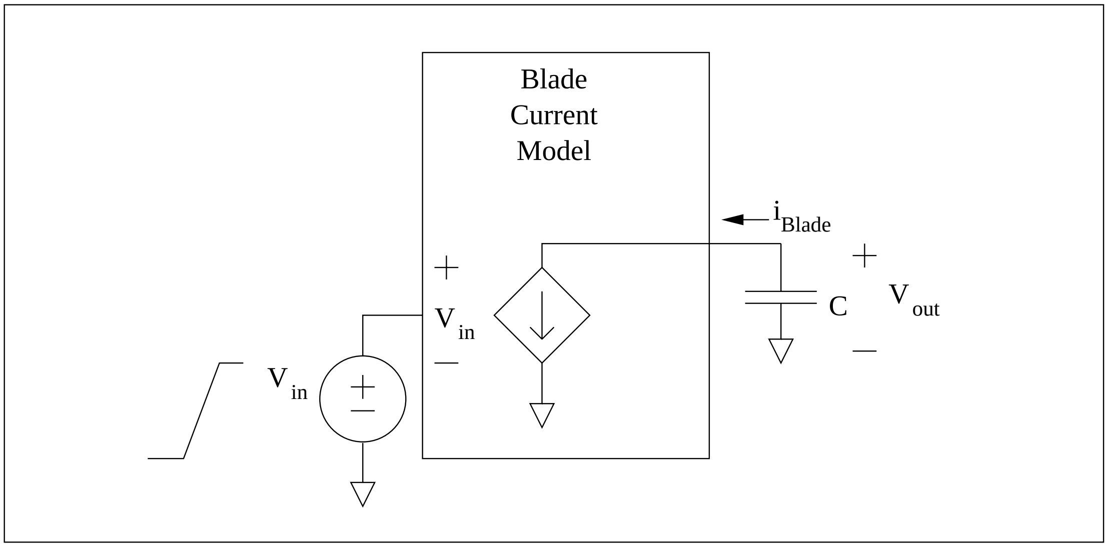


Figure 3.4: A simple example of the Blade model loaded with a single capacitor and driven by a saturated voltage ramp input.

3.2.2.1 The Nonlinear Solver

The Blade nonlinear solver is used to determine the transient response of a cell driving a load. Consider the very simple case of the Blade model loaded with a capacitor and being driven by some input voltage waveform as shown in Figure 3.4. Given a time-indexed array of the input voltage, V_{in}, and an initial voltage across the capacitor, the Blade nonlinear solver determines the output voltage at each time point.

In the Blade model, the current is a function of both V_{in} and V_{out}. The current charging the capacitor is a function of the capacitance and the change in the voltage across the capacitor over time.

$$i_{\text{BLADE}} = f_{\text{BLADE}}(V_{in}, V_{out}) \tag{3.1}$$

$$i_{cap} = C\frac{dV_{out}}{dt} \tag{3.2}$$

Recognizing that all current sourced by the BLADE model must flow into the capacitor, an objective function can be derived.

$$f_{obj}(V_{out}) = C\frac{dV_{out}}{dt} - f_{\text{BLADE}}(V_{in}, V_{out}) = 0 \tag{3.3}$$

The function $f_{obj}(V_{out})$ is a nonlinear first-order differential equation. By replacing the single capacitor with its companion model (covered in the next section), a simple nonlinear equation can be formulated for $f_{obj}(V_{out})$.

Newton-Raphson (NR) iteration is used within most SPICE engines to solve for roots of nonlinear equations as it converges upon an answer quadratically. In the NR method, an iterative approach is used to find an answer. Given a function of the form

$$f(x) = 0 \tag{3.4}$$

with a previous guess of x_k, a new estimate for x, x_{k+1}, can be derived:

$$x_{k+1} = x_k + \Delta x_k \qquad \text{where } \Delta x_k = \frac{-f(x_k)}{f'(x_k)} \tag{3.5}$$

Newton-Raphson iteration is illustrated in Figure 2.8.

As stated previously, the NR method converges upon an answer quadratically. In other words,

$$\frac{(x_{k+1} - x_k)}{(x_k - x_{k-1})^2} \approx C \qquad \text{where } C = \text{ constant} \tag{3.6}$$

In order for Newton-Raphson to function properly, the first derivatives must exist and be continuous. Because the BLADE current model is a table-lookup model and, as such, has no first derivative, Newton-Raphson iteration cannot be applied.

Secant iteration converges almost as rapidly as Newton-Raphson iteration (1.67 as opposed to 2.0 for Newton-Raphson) and does not place a restriction upon

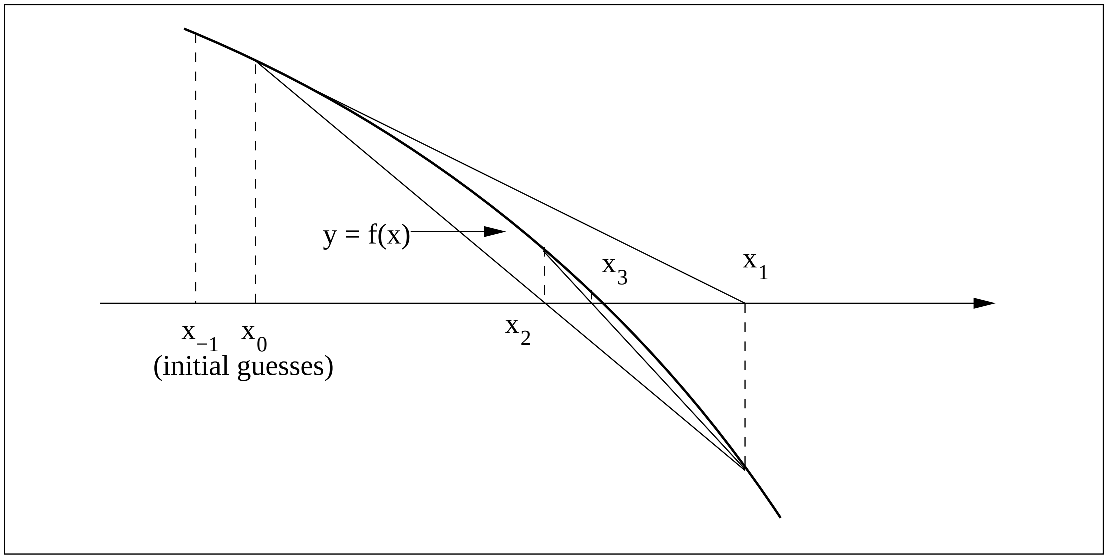


Figure 3.5: Secant iteration converges to a solution for $f(x) = 0$ by using the previous two estimates to solve for the next.

the data content [35]. Instead of using the derivative of the function to calculate the next guess, secant uses only the function itself. The slope is determined by using the last two points.

$$x_{k+1} = x_k + \frac{-y_k \Delta x_{k-1}}{y_k - y_{k-1}} \tag{3.7}$$

This approach requires two initial guesses instead of one: x_{-1} and x_0.

Given the nature of the Blade model and the need to obtain maximum runtime speeds, the secant approach is the method used within the Blade runtime engine.

3.2.2.2 Problem Formulation

In SPICE a cell model is composed of numerous transistors, diodes, capacitors, resistors, and/or inductors. The companion model for the active components

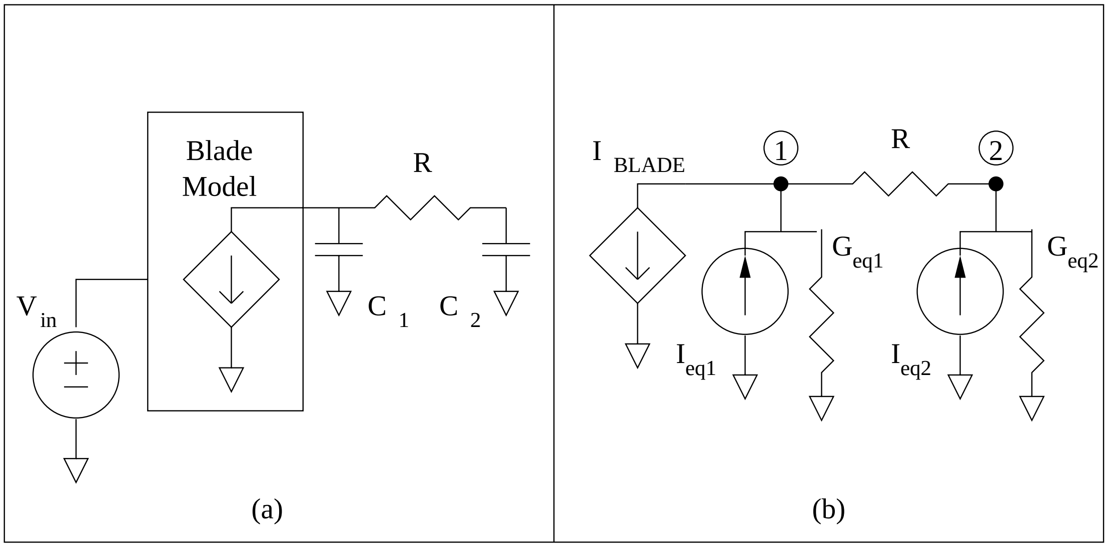


Figure 3.6: A Blade cell model is shown driving a π-model in (a). In (b) each capacitor is replaced by its companion model.

may end up changing for each Newton-Raphson iteration. This process, in turn, results in a change to the modified nodal admittance (MNA) matrix [23] used to represent the circuit and a recalculation of that matrix's inverse. Because the Blade model is a voltage-controlled current source, the MNA matrix is not modified at any time during the nonlinear iterations. Additionally, where a cell might need a 300- or 400-element matrix to represent the resistors, capacitors, diodes, and transistors that compose it, plus additional room to represent the load being driven, the Blade model only needs a matrix to represent the load.

Consider the case of a Blade cell model driving a π-model as shown in Figure 3.6. That same figure also shows the companion model representation that is used to populate the MNA matrix.

The trapezoidal companion model for a capacitor is shown in Figure 3.7.

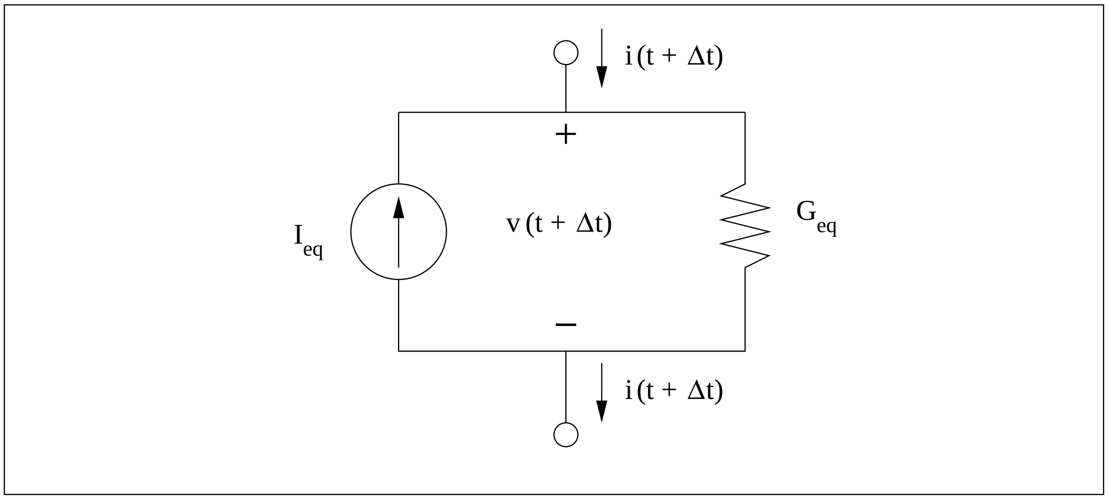


Figure 3.7: The Norton equivalent trapezoidal companion model for a capacitor consists of linear elements which can be solved for using numerical integration.

The relationship between the capacitor and its companion model is given by the following equations [46]:

$$I_{eq} = i(t) + \frac{2C}{\Delta t}v(t) \tag{3.8}$$

$$G_{eq} = \frac{2C}{\Delta t} \tag{3.9}$$

Using this companion model, the MNA for the Blade model driving a π-model can be developed.

$$\begin{bmatrix} \frac{1}{R} + G_{eq1} & -\frac{1}{R} \\ -\frac{1}{R} & \frac{1}{R} + G_{eq2} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} I_{eq1} - I_{\text{Blade}} \\ I_{eq2} \end{bmatrix} \tag{3.10}$$

As the MNA matrix never changes unless the timestep, Δt, changes, the Blade model is able to rapidly analyze the circuit.

3.2.3 Accounting for Internal Parasitics

At this point the Blade model consists solely of a voltage-controlled current source operating on a lookup table. Because this table was created by DC (steady-state) simulation, the effects of any internal parasitics were lost. Unless these parasitics are included within the model, the BLADE model will always be optimistic when compared to SPICE. The transient effects of these parasitics must be captured.

During the development of the BLADE model, two different parasitic types were targeted: internal capacitance and internal interconnect.

3.2.3.1 Internal Capacitance

The internal capacitance includes the capacitance between the four nodes of a transistor: C_{GS}, C_{GD}, C_{BS}, and C_{BD}. When the transistors are organized in such a way as to create a simple logic function implementation, it is possible to make simplifying assumptions concerning these capacitances.

To better explain the basis for the simplifying assumptions, consider the case of a two-input NOR gate as shown in Figure 3.8. In Figure 3.8(a) the two-input NOR is shown with these internal capacitors. One of the input pins is tied to V_{SS} as indicated. The remaining pin is the input pin of interest.

In Figure 3.8(b), the PMOS transistor tied to V_{SS} is replaced with a short while the NMOS transistor tied to V_{SS} is replaced with an open. Also, the capacitors that are tied to V_{SS} or V_{DD} are independently split out for the purposes of illustration. The dark black line in this figure represents a short in place of the PMOS transistor with its gate tied to V_{SS}.

Next, as shown in Figure 3.8(c), capacitors that do not affect the signal delay (those with both ends tied to one of V_{SS} or V_{DD}) are removed. Internal capacitors in parallel are combined.

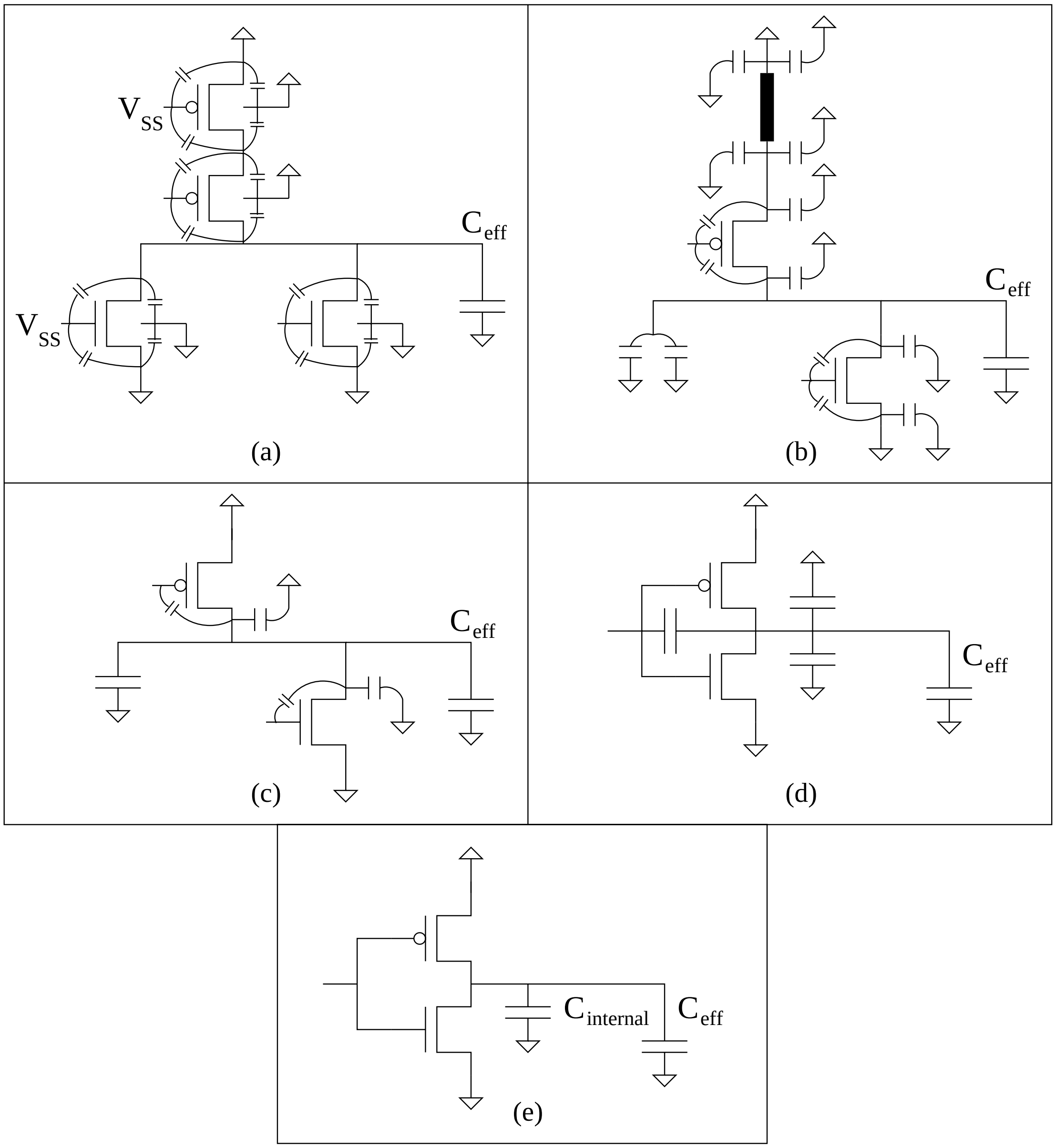


Figure 3.8: Simplifying assumptions concerning internal capacitance and the Blade model. A two-input NOR is used to illustrate the simplifications when one input is being characterized while the other is held to a constant value.

In Figure 3.8(d), the schematic is redrawn to show that the pin driving the PMOS transistor and the pin driving the NMOS transistor are the same pin. Capacitors in parallel are combined.

The final representation, shown in Figure 3.8(e), represents a single inverter driving an internal capacitor, representing various internal capacitances, and the original C_{eff} load. The capacitor tied between the input pin and the output pin is combined with the other internal capacitors as the input is driven by an ideal source. Thus, tying the capacitor to the input pin has no effect on the input signal. When modeled as a simple linear capacitor, it can be combined with the other capacitors in parallel.

This type of simplification is valid for applications, such as static timing analysis, in which analysis is performed under the assumption of only a single input pin switching at any given point in time. For many logic functions, the effect of holding all inputs but one constant is that the input waveform is inverted at the cell's output pin, as shown with the two-input NOR. For many compound cells, the result is a buffer[4].

The purpose of this illustration is simply to provide an understanding behind the use of a single capacitor to represent internal capacitances within the BLADE model. Experimental results, presented in the next chapter, show that the addition of a single capacitive value to represent internal capacitances provides an effective approximation such that accurate waveforms can be generated for combinatorial cells with the BLADE model.

---

[4] Compound cells are covered in a later section.

3.2.3.2 Internal Cell Interconnect

Waveforms used to drive interconnect models degrade at the far end of the interconnect. For long pieces of interconnect that connect a cell to those cells it drives, this degradation can be severe.

Interconnect is also present inside a cell between the point that the input pin is identified to exist and the transistor gate(s) that the input pin drives. It is also present between the transistor source and drain nodes and the output pin location. Because standard cells are typically very small, the amount of waveform degradation across this interconnect is almost immeasurable. Also, waveform degradation in the BLADE model is already being accounted for in the internal capacitance adjustments.

The other effect of interconnect is to delay the propagation of voltage waveforms from the input or output pin location to or from the transistors. This delay must be accounted for. In the BLADE model, this delay is a single constant that is used to offset any voltage waveform created by the evaluation of the model.

3.2.3.3 Model Calibration

A calibration program is used to determine the internal capacitance adjustment and the time adjustment for internal interconnect. In this process, SPICE transient analysis is performed on the cell of interest. The cell is loaded with a simple capacitor (approximating the average load such a cell might drive in a real design) and is driven by some generally representative input waveform. Unlike traditional cell characterization, a saturated linear ramp is sufficient to obtain accurate results for the purpose of calibration[5].

Using the results of this SPICE transient analysis, the BLADE model can be calibrated. The process begins by reading the SPICE results and creating an environment under which the DC-based BLADE model will operate including duplication of the exact load used in the SPICE analysis. In addition to the load used in the SPICE run, the calibration program uses an additional capacitor, $C_{internal}$, tied to the output of the model, to mimic the effect of internal capacitance. Once the model is ready to be evaluated, the same input used to drive the SPICE model is used to drive the BLADE model.

After the model environment is created, many different BLADE evaluations are made. Each run differs from the previous by incrementally adding to $C_{internal}$. After each simulation, the waveform created by the BLADE model and the waveform created by SPICE are compared. The Root Mean Squared, or RMS[6] value, of the time differences between the two waveforms is calculated for key points along the waveform. The $C_{internal}$ value for which the RMS is minimum is the capacitance chosen to represent internal capacitances and waveform degradation effects of the internal interconnect[7].

Once the internal capacitance is known, the same data can be used to ac-

---

[5] The BLADE model is designed to determine the response of a cell to arbitrary input waveforms and output loads. Thus, calibration can occur using any input waveform and load. The use of a saturated linear ramp and single capacitor is merely for simplicity.

[6] $RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}$

[7] This brute force method to finding $C_{internal}$ can be replaced with a more elegant approach that finds the calibration parameters faster. However, the BLADE model evaluates so rapidly that the slight decrease in CPU time associated with a more sophisticated algorithm does not justify the additional effort required to enhance the calibration program.

count for internal interconnect delay. By examining the time difference between the Blade and SPICE waveforms, an overall time delay can be calculated. Specifically, the time difference between the 50% points of the two waveforms is used as the second component of the cell calibration: time shift.

Once calculated, the internal capacitance adjustment and the time offset are used for all Blade evaluations between the input pin and output pin, irrespective of the load (capacitor, π-model, tree, or mesh). These values are also independent of the direction of the voltage swing on the input pin or output pin as demonstrated in the next chapter. The calibration parameters are unique, however, to the specific input-to-output pin path being modeled.

3.2.3.4 Conclusion

Instead of pursuing an analytic approach to determine the effect of internal parasitics on the cell, an empirical approach was chosen. Given the speed of the Blade model (covered in the next chapter), the process of calibrating the model was quickly accomplished. Using empirical measurements simplified the process as the model generator did not need to consider custom SPICE transistor models or attempt to mimic the calculations performed by SPICE once a model was selected.

3.2.4 Compound Cells

Compound cells are those standard cells that are composed of two or more primitive logic functions. In a primitive cell, the gates of all transistors are driven by inputs from outside the cell. Compound cells, on the other hand, use the output of one or more primitive gates to drive one or more primitive gates, all within the compound cell. Examples of compound cells include simple noninverting logic cells like AND gates and OR gates.

Characterization of compound cells is more complex than characterization of primitive cells. While the method used for primitive gates can be used to derive a current model for the gate, the calibration process will not yield a single time offset value that can be used across a variety of input waveforms. To accurately characterize a compound cell, the primitive cells each must be characterized independently. For the purposes of discussion, the characterization of an OR gate will be presented.

The first step in characterizing the primitives that compose the compound cell is the extraction of the primitive netlists from the compound cell SPICE netlist. In the case of the OR gate, the node that identifies the output of the NOR gate is located and the single SPICE netlist of the OR gate is broken into two subcircuit SPICE netlists as shown in Figure 3.9: one for the NOR gate and one for the INV gate it drives.

Once the SPICE netlists for the two primitive gates have been created, the I–V characteristics for each primitive gate are extracted. This process is identical to the process carried out for primitive standard cell logic described previously and is illustrated in Figure 3.9.

After the current profile has been determined, the model must be calibrated. Calibration of the OR gate must occur in two stages. Both the NOR gate and the INV gate must be calibrated individually.

Unlike a primitive logic gate, the output of the NOR gate never will be loaded with anything other than the INV gate it drives. Thus, calibration of the

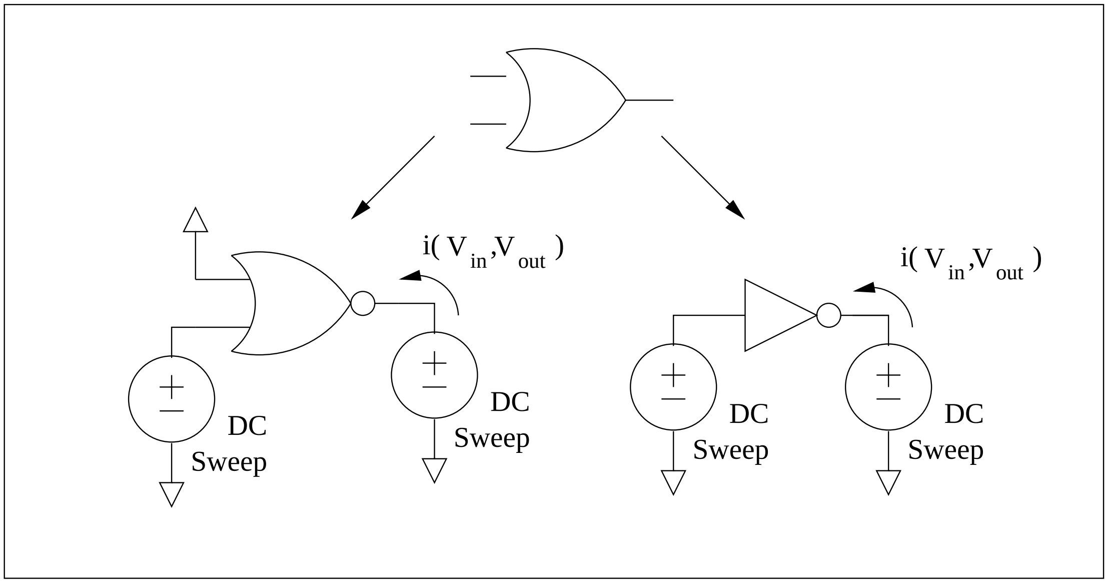


Figure 3.9: For compound cells like an OR gate, current models are extracted for both the NOR component and the INV component.

NOR gate simply involves the determination of a $C_{internal}$ value to represent both the internal capacitance of the NOR gate as well as the effective capacitance of the input pin of the INV gate. Once calculated, this fixed value is used in all subsequent evaluations of the Blade OR gate model.

In the case of the NOR gate, a SPICE deck very much like that of a primitive gate is created. There are only two major differences. First, instead of placing a capacitor on the output of the NOR gate, the gate is simply loaded with the INV that it drives in the OR gate. Second, the output pin of the INV gate is loaded with some reasonable capacitive load. A voltage waveform drives the pin of interest and the voltage at the output of the NOR gate is measured. From these two voltage waveforms the value of $C_{internal}$ for the NOR gate can be calculated as described for primitive logic gates.

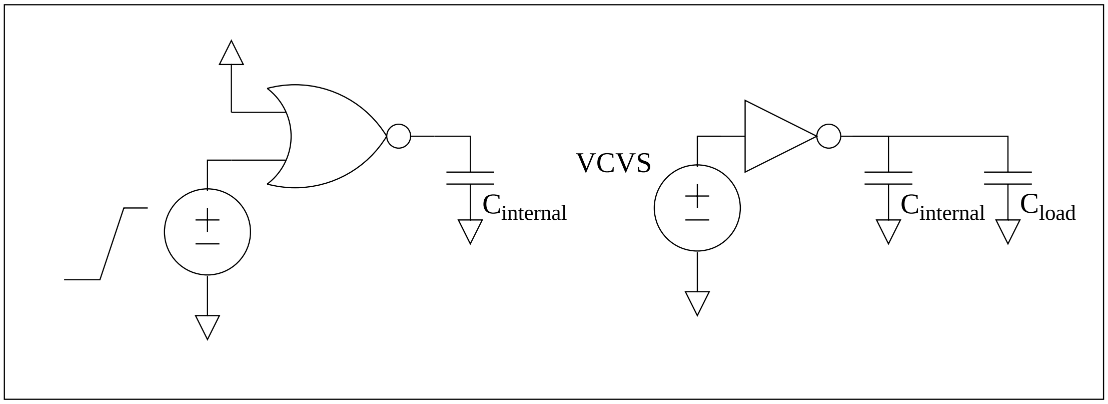


Figure 3.10: Calibration for the OR compound cell occurs in two stages. The first determines a $C_{internal}$ value on the NOR gate while driving the INV gate. The second uses that waveform to calibrate the INV gate. The runtime engine transparently treats the two cells as one (as shown) during the evaluation process.

Unlike the NOR gate, the INV portion of the OR gate is calibrated just like any primitive gate. A voltage waveform drives the input, the output is loaded with some capacitive load, and the input and output voltage waveforms are used to determine the $C_{internal}$ value for the INV gate.

The final stage of calibration puts these two BLADE models together to determine the time shift between the OR gate and the two BLADE components. This is easily done by driving the OR gate with a voltage waveform, placing a load on the output, measuring the output voltage waveform, and calibrating to this waveform using the NOR and INV values previously determined.

The NOR and INV models are unified to produce the BLADE model of the OR gate. During model evaluation, the results produced by evaluating the NOR gate are used to drive the INV gate to generate the output waveform of the compound OR gate, as shown in Figure 3.10.

3.2.5 Noisy Inputs

Cells differ from one another in the way that they respond to voltage waveforms that include distortions due to noise. Some cells are relatively noise tolerant while others exhibit dramatic variations in the shape of the output voltage waveforms. The response of a cell to noise imposed on its input signal is largely dependent upon the intrinsic noise immunity of the cell. This immunity can be attributed to the number, placement, and size of transistors, the amount of interconnect in the cell, and the number of contacts in the cell, among other things.

The standard formulation of the Blade model, consisting of a voltage-controlled current source and an internal capacitor, unintentionally disregards the noise immunity characteristics of a cell. For cells with little to no noise immunity, this formulation accurately represents the cell's response. However, for larger or more complex cells, the omission of this characteristic from the model may result in the incorrect transfer of a noise response at the model's output.

To alleviate the problem of incorrect noise modeling, an enhancement to the Blade model, called the noise immunity filter, has been developed. This noise rejection filter conditions the input signal to correct for the lost noise immunity in the model. Specifically, a single-pole low-pass filter is used, thereby reducing the undesirable high-frequency transfer characteristic of the original Blade model. This filter can be thought of as a simple RC segment applied to the input signal as shown in Figure 3.11.

The noise immunity filter is characterized after the other components of the Blade model have been calculated. The process starts by establishing a cell's

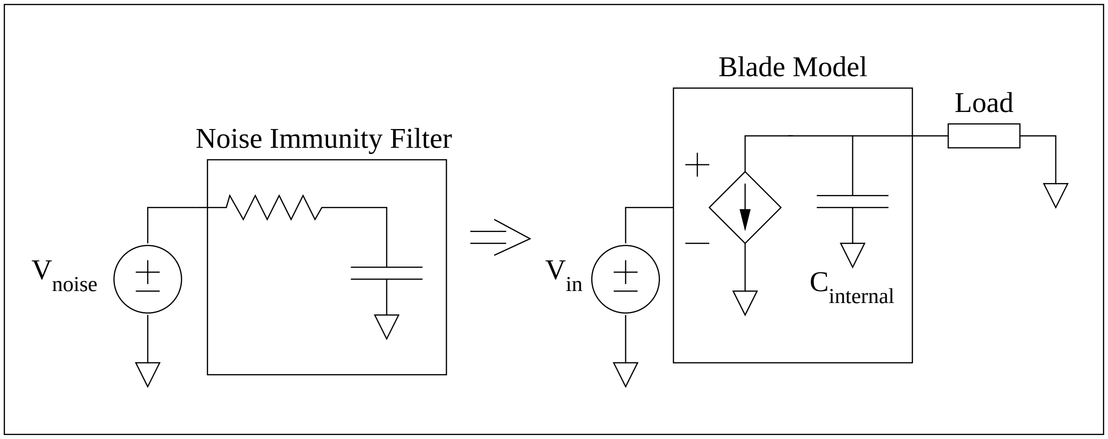


Figure 3.11: The BLADE model for noise differs slightly from the standard model. A low-pass filter provides protection from variations in the input voltage for cells that exhibit some degree of noise immunity.

sensitivity to noise. Using SPICE, a cell's response to a noisy input is determined. As each cell will respond differently, there is no single waveform that is guaranteed to provide this data. For example, the use of an underdamped waveform will result in the creation of a filter that does not accurately model a cell's true response.

The presence of the noise immunity filter has no effect on the value of $C_{internal}$ for the BLADE model. However, it does impact the time offset value. Thus, characterization involves the determination of a pole such that the BLADE- and SPICE-generated waveforms agree, as determined by the RMS value in the calibration program. This is done by simply simulating the BLADE model with an input waveform that has first been processed through the noise immunity filter. After the pole has been calculated, a new time offset is determined by measuring the time difference between the 50% point of the two output waveforms. It is important to note that this value may be a negative number in the resultant model.

The noise immunity filter can be used irregardless of the input waveform shape. For example, the filter can be used to process a saturated linear ramp without effecting the quality of the model's output. However, because it introduces an additional step in both the model characterization process as well as the model evaluation process, it's use should be restricted to cells located in environments in which noise will be present.

3.2.6 Conclusion

The Blade model and runtime engine combine an empirically derived current model with a nonlinear engine to achieve both speed and accuracy. By focusing on the ability of a cell to source or sink current, the Blade system is designed to respond to arbitrary voltage waveform inputs including noisy waveforms and those with long tails. The Blade model is also designed to handle arbitrary loads including lumped-C, π-models, trees, and meshes.

Maximum speed is achieved through high-performance lookup tables like those used as characterization tables. The nonlinear solver uses secant iteration to converge almost as rapidly as Newton-Raphson iteration.

Whereas traditional cell characterization can require hundreds of SPICE transient analysis runs to measure accurate pin-to-pin data under very specific conditions, characterization for a Blade model is much simpler. A single SPICE DC sweep and a single transient analysis run are required for primitive logic. For compound cells, two runs are required. The dramatic reduction in the number of transient analysis runs results in huge speed improvements over traditional characterization.

3.3 RAZOR: Accurate Stage Delay

Stage delay is the measured time between a point of the voltage waveform used to drive the input of a cell to a point of the voltage waveform at the next cell being driven as shown in Figure 3.12. Stage delay accuracy is a function of the accuracy of the cell model, the accuracy of the interconnect model, and the ability to propagate waveforms created by the cell model through the interconnect model. In the past, stage delay accuracy has been hindered by the limitations of the cell model and by the inability to propagate arbitrary waveforms through the interconnect model efficiently.

The output of a BLADE transient analysis is a time-indexed voltage array in which the timesteps are uniform. Thus, for a 1 nanosecond transient analysis period with a 1 picosecond timestep, a 1000-segment piecewise linear (PWL) description is computed. This PWL represents the voltage waveform that must propagate through the interconnect accurately and efficiently to determine the voltage waveform at each cell being driven. By taking advantage of the fixed stepsize, a simple and efficient calculation of the voltage at a node in the interconnect can be calculated.

Let A_i represent the slope of the input voltage segment between time $(i-1)\Delta t$ and $i\Delta t$ where Δt is the fixed timestep. The input voltage to the interconnect can be described by the sum of a series of infinite ramps shifted in time. Figure 3.13 illustrates this concept. It can be described mathematically by equation 3.11.

$$v_{in}(t) = v_{in}(i\Delta t) = A_1 i\Delta t + (A_2 - A_1)(i-1)\Delta t + \ldots + (A_i - A_{i-1})\Delta t \quad (3.11)$$

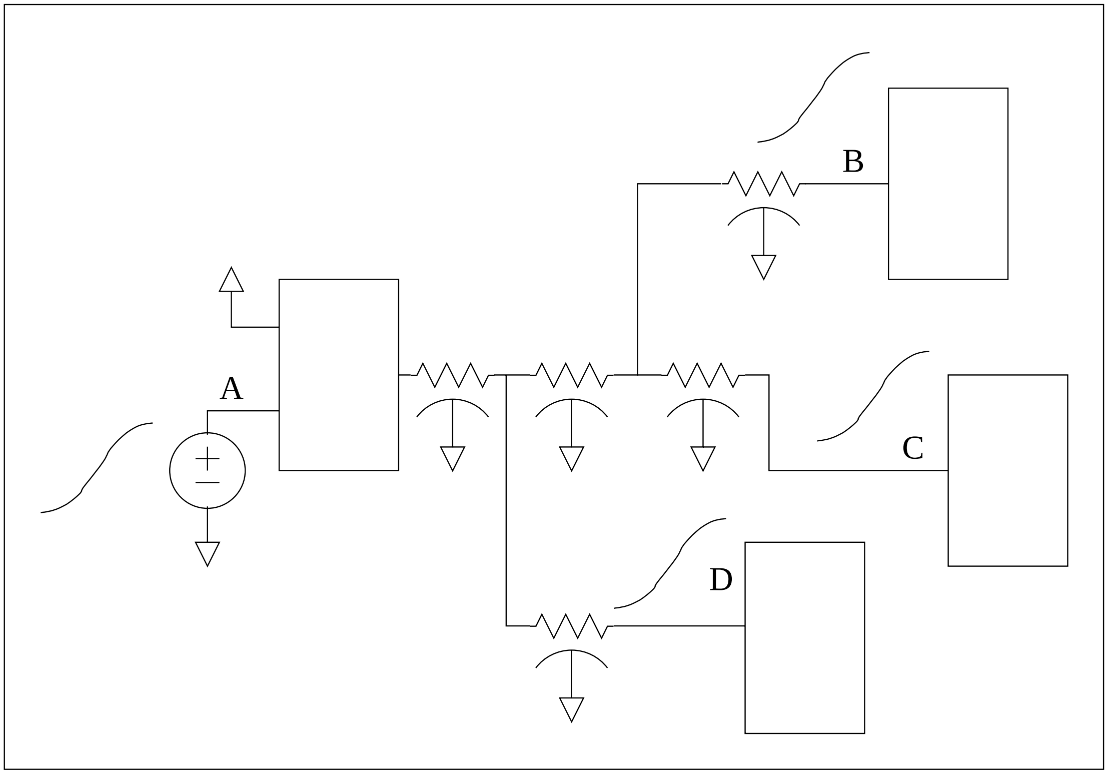


Figure 3.12: Stage delay is the delay between a point on the voltage waveform of a cell's input to a point on the voltage waveform of a cell it drives, through the interconnect. This figure shows three such paths: A to B; A to C; and, A to D.

Using a moment-matching technique such as AWE [45], n^{th}-order reduced interconnect models can be constructed for each interconnect sink (the point at which the interconnect and cell being driven are connected) driven by $v_{in}(t)$. Given n poles and residues of a sink node (p and k, respectively) driven by an infinite ramp input voltage of slope A, the voltage at the sink at time t, $v(t)$, can be described by a reduced-order model of the form given in equation 3.12.

$$v(t) = A[t - \sum_{i=1}^{n} \frac{k_i}{p_i}(1 - e^{p_i t})] \tag{3.12}$$

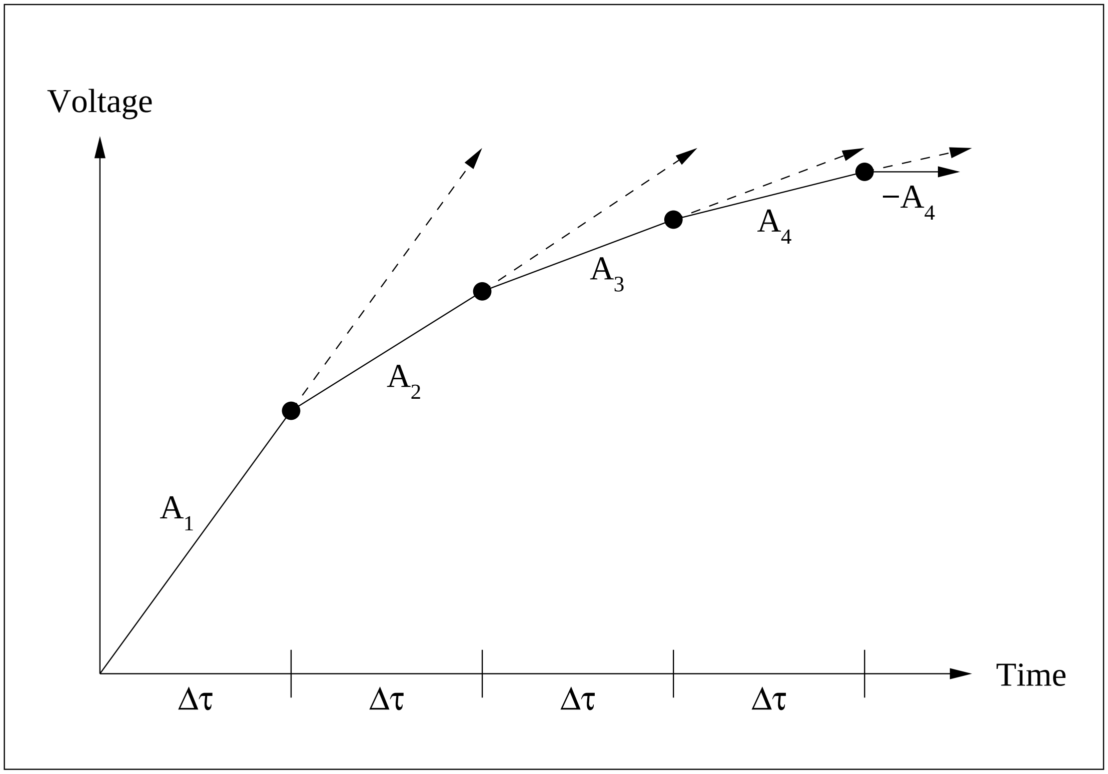


Figure 3.13: A voltage waveform can be described by the sum of a series of infinite ramps, each shifted in time by fixed timestep of Δt.

The response for a saturated ramp input can be described by equation 3.13 where τ is the point in time after which the voltage remains constant.

$$v(t) = \begin{cases} A[t - \sum_{i=1}^{n} \frac{k_i}{p_i}(1 - e^{p_i t})] & \text{if } t < \tau \\ A[t - \sum_{i=1}^{n} \frac{k_i}{p_i}(1 - e^{p_i t})] - A[(t-\tau) - \sum_{i=1}^{n} \frac{k_i}{p_i}(1 - e^{p_i (t-\tau)})] & \text{if } t \geq \tau \end{cases} \tag{3.13}$$

This approach can be extended to accommodate multiple input ramps.

This process, called recursive convolution [4], has been shown to produce an exact solution to the problem of interfacing PWL voltage representations to reduced-order frequency-domain representations of the interconnect.

Razor is a novel implementation of recursive convolution that takes advantage of the fixed timestep size used in Blade to efficiently and accurately calculate the voltage waveforms at each interconnect sink node. Razor is an $O(np)$ algorithm where n is the number of timesteps and p is the number of poles.

Although Razor was developed for Blade, it can be applied efficiently to the general PWL problem in which timesteps are not uniform. An arbitrary PWL can be converted into a new PWL representation with uniform timesteps with little to no loss of accuracy by selecting a timestep of sufficient resolution. This uniform timestep PWL then can be used as input to Razor.

Razor achieves its speed through the calculation of partial sums of the fully-factored saturated ramp equation given in equation 3.13. Consider a two-pole representation of a RC interconnect sink with an infinite ramp of slope A driving it. For a fixed timestep size of Δt, $v(t) = v(i\Delta t)$. The contribution of PWL segment j after time $i\Delta t$ (where $i \geq j$) is given by 3.14.

$$A_j[(i-j)\Delta t - (\frac{k_1}{p_1} - \frac{k_1}{p_1}e^{p_1(i-j)\Delta t} + \frac{k_2}{p_2} - \frac{k_2}{p_2}e^{p_2(i-j)\Delta t})] \qquad (3.14)$$

Rearranging terms yields 3.15.

$$A_j[(i-j)\Delta t - (\frac{k_1}{p_1} + \frac{k_2}{p_2}) + \frac{k_1}{p_1}e^{p_1(i-j)\Delta t} + \frac{k_2}{p_2}e^{p_2(i-j)\Delta t}] \qquad (3.15)$$

Each of these components can be independently calculated and the partial sums added.

$$v(t) = v(i\Delta t) = \sum_{m=1}^{4} S_m(i\Delta t) \tag{3.16}$$

$$S_1(i\Delta t) = \sum_{j=1}^{i} A_j \Delta t = A_i \Delta t + S_1((i-1)\Delta t) \tag{3.17}$$

$$S_2(i\Delta t) = -A_i(\frac{k_1}{p_1} + \frac{k_2}{p_2}) \tag{3.18}$$

$$S_3(i\Delta t) = (A_i - A_{i-1})\frac{k_1}{p_1}e^{p_1\Delta t} + e^{p_1\Delta t}S_3((i-1)\Delta t) \tag{3.19}$$

$$S_4(i\Delta t) = (A_i - A_{i-1})\frac{k_2}{p_2}e^{p_2\Delta t} + e^{p_2\Delta t}S_4((i-1)\Delta t) \tag{3.20}$$

Because the poles, residues, and timesteps are all constant, the fractions and exponents can be precomputed. These precomputed values are then propagated through the calculation of the various S_i terms. A similar derivation can be made for RLC[8] circuits (with imaginary pole and residue components) as well as for higher order representations.

---

[8] The term RLC refers to a circuit consisting only of resistors (R), capacitors (C), and inductors (L).

Chapter 4

Results

This chapter presents experimental results achieved when using the characterization models and the Blade and Razor runtime environments. It is divided into three sections, each detailing the results obtained by each model.

4.1 Over Sampling and Data Reduction

Implementation of the over sampling and data reduction characterization methodology was originally performed under Linux using G++. Avant! HSPICE was used to extract empirical cell data. This methodology was compared to a cell characterization methodology in which data was extracted and Monte Carlo simulation was performed to refine the data selection.

Over sampling and data reduction have been implemented in a production characterization system. Typical results are shown in this section. A 50×50 characterization response surface was extracted for various cell paths. This large data set was used both to derive a reduced table as well as to compare against to determine total error margins.

4.1.1 Dependent Variable Analysis

Figure 4.1 shows the error margins associated with a typical cell in a library used for a high-performance microprocessor design. The largest errors were seen in the regions of fastest input transition time and, most notably, light capacitive load. This also happens to be the region favored by synthesis programs when generating a cell netlist.

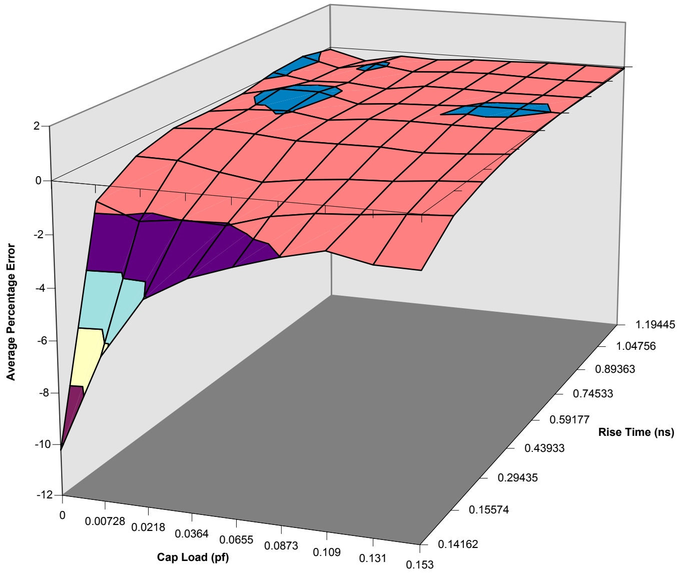


Figure 4.1: A 3-D surface of the error percentages between a Monte Carlo-based characterization system table and a 50 × 50 table created by over sampling.

This same data was used to create a 6 × 6 reduced table in which both input transition time and capacitive load were optimized simultaneously. The error margins between this reduced table and the original HSPICE data are shown in Figure 4.2. In general, the errors tended to fall within ±0.2% providing a highly accurate interpolation estimate of the actual response to a given input transition time and output capacitive load. Figure 4.3 shows an overlay of the measured response surface and the reduced form of the response surface for a four-input AND gate.

When the over sampling and data reduction methodology was originally

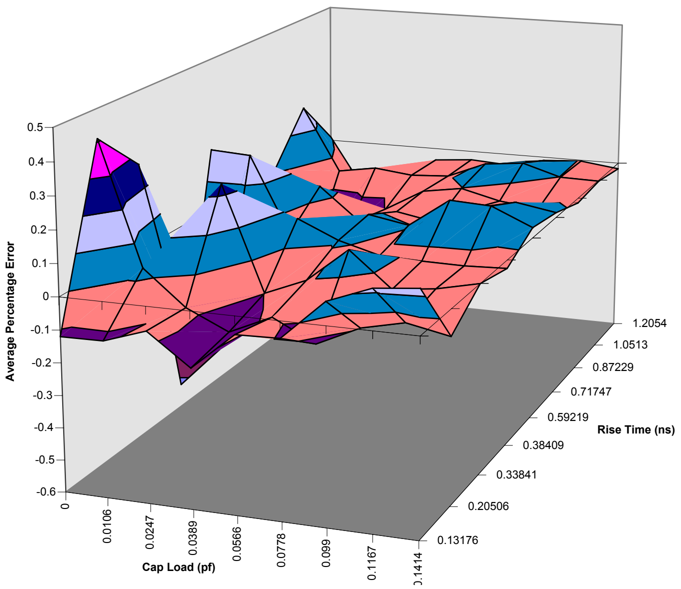


Figure 4.2: A 3-D surface of the error percentages between the reduced table and the 50 × 50 table from which it was derived.

implemented, benchmarks were performed on a 200MHZ AMD K6 CPU with 64Mb of physical memory. Generation of a 10×10 reduced table from the original 50×50 took approximately 17 CPU minutes, 16 of which were used to calculate the errors in the initial stage. The primary factor determining CPU runtime was the number of steps required to create the reduced table. A 10×10 table took twice as long to create as a 7×7 table.

4.1.2 Independent Variable Analysis

The assumption of independence between input transition time and capacitive load resulted in a minor increase in error rates, but required only five CPU seconds to complete a 6×6 table generation. Error margins typically were between $+0.5\%$ and -0.8%. These error margins were still substantially lower

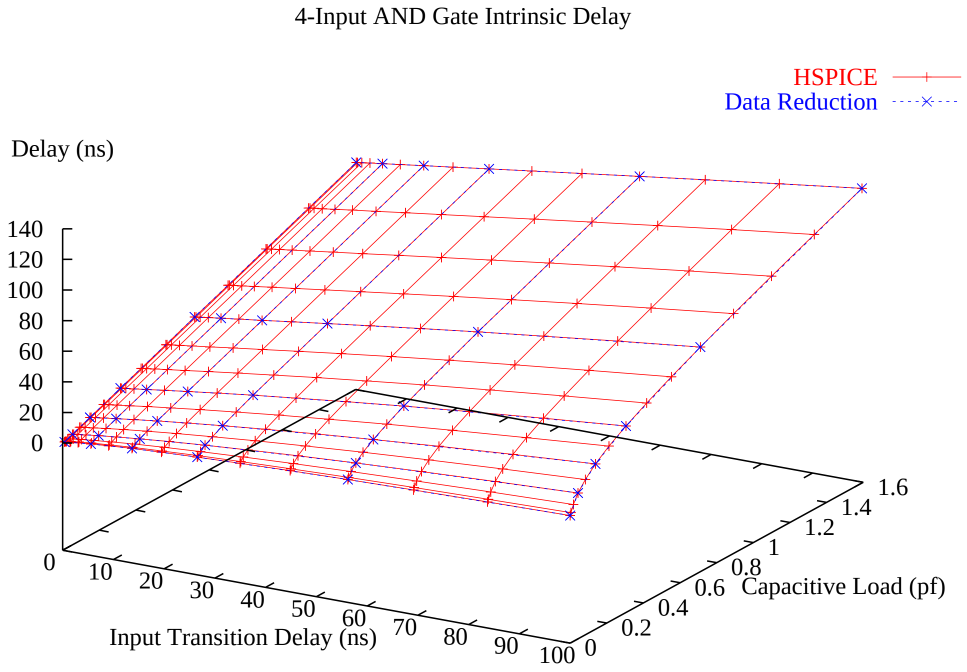


Figure 4.3: A 15 × 15 response surface measured with HSPICE. The reduced response surface is overlaid. The nonlinear regions for low cell loads are accurately represented by the reduced response surface.

than those exhibited by the Monte Carlo-based characterization system.

4.1.3 Path Errors

Over sampling and data reduction were used to create the characterized cell library used to in the AMD Athlon microprocessor. In this type of timing sensitive application, it was not enough to simply say that a reduced table approximated a much larger table with small errors. The true test of accuracy was how well those cells functioned in timing paths when compared to SPICE.

In order to verify the path accuracy of the characterization data, a Tcl program was written to selectively prune a path out of the Athlon design, create a SPICE netlist that accurately represented the path, and compare SPICE measured data to the results of the static timing analysis program used by the designer. An example of a pruned path is shown in Figure 4.4. The program extracted all components of the selected timing path along with any secondary components. The secondary cells were also loaded with a capacitor that was matched to the effective load that they drove (tertiary loading). All pins not part of the timing path were biased to V_{DD} or V_{SS} such that they propagated the signal from input to output. For those cells that were in the timing path, the secondary pins were biased such that the direction of the output transitions (rising or falling) matched those specified by static timing analysis. The interconnect model was also extracted from static timing analysis and duplicated within the SPICE deck. The entire circuit was then simulated using SPICE.

Over 200 different timing paths were run through this Tcl program. While the exact numbers remain confidential, it can be stated that over 98% of the paths had cumulative path delays with error margins of 2% to 4% (pessimistic) when the static timing analysis results for the path were compared to HSPICE. Error margins using the Monte Carlo methodology were between 10% and 30% of HSPICE.

4.2 BLADE

The BLADE modeling system and runtime engine was written in C++ and compiled under RedHat Linux 7.1 using G++ 2.95.3. Avant! HSPICE version

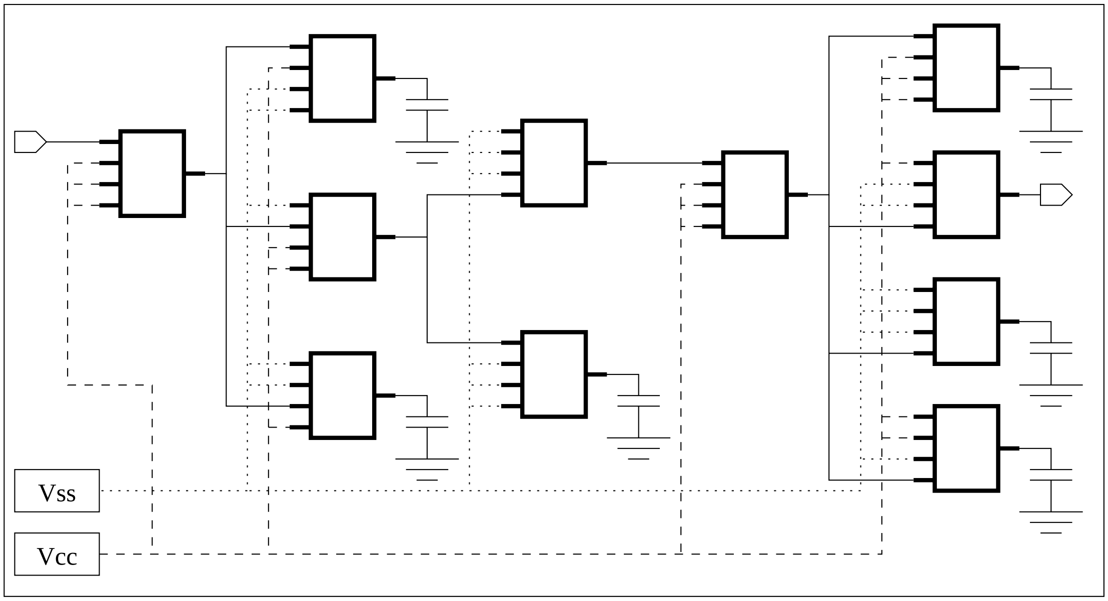


Figure 4.4: By biasing secondary pins to V_{DD} or V_{SS} and applying tertiary capacitive load to branching cells, a path can be created for use in a SPICE run which matches the signal transitions reported by static timing analysis. A comparison of the SPICE results can be made to the timing values reported by static timing analysis to determine the accuracy of the underlying cell library.

2001.4 for RedHat Linux 7.1 was used to perform the DC current extraction, the model calibration, and CPU runtime comparisons. All runs, both BLADE and HSPICE, were made on a 1800+ AMD Athlon (1.533GHz clock speed) with 512Mb of memory.

BLADE models were created for cells in a 0.13 μm, 1.5V production cell library. All cell netlists were created by parasitically extracting the cell's layout. Cell netlists consisted of various combinations of transistors, diodes, resistors, and capacitors.

BLADE models were created using the process outlined in the previous chapter. The input and output voltages were swept from 0V to 1.5V in 0.05V increments, yielding a 31×31 I–V table. Figure 4.5 shows the I–V curve extracted for one path through a four-input AOI cell. HSPICE DC sweep CPU runtimes never exceeded one CPU second for any cell in the library.

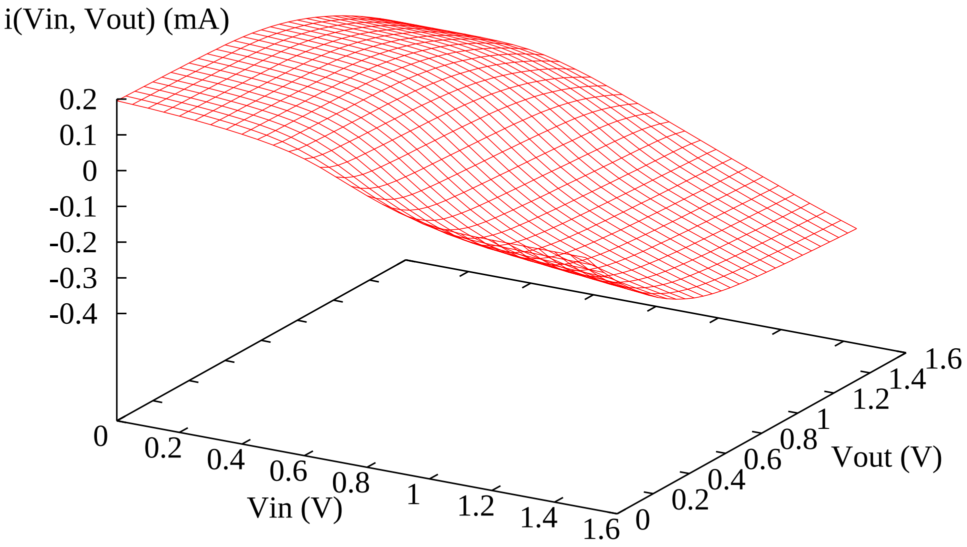


Figure 4.5: The I–V curve for a four-input AOI cell from input pin A1 to output pin ZN.

4.2.1 Calibration

Once the DC current flow was measured, the process of calibration began. Figure 4.6 shows the waveforms created by the BLADE model of a four-input NAND gate at various points in the calibration process. The final model required an additional internal capacitance addition of 0.02 picofarads and a 0.010

nanosecond offset for internal parasitics.

The process of calibration occurred rapidly. Given the speed of the Blade model and runtime engine, the calibration program was able to investigate 3000 different values to account for internal capacitances. $C_{internal}$ values from 0 picofarads to 0.3 picofarads in steps of 0.1 femtofarads were examined in the current implementation. Calibration runtimes were typically under one CPU second.

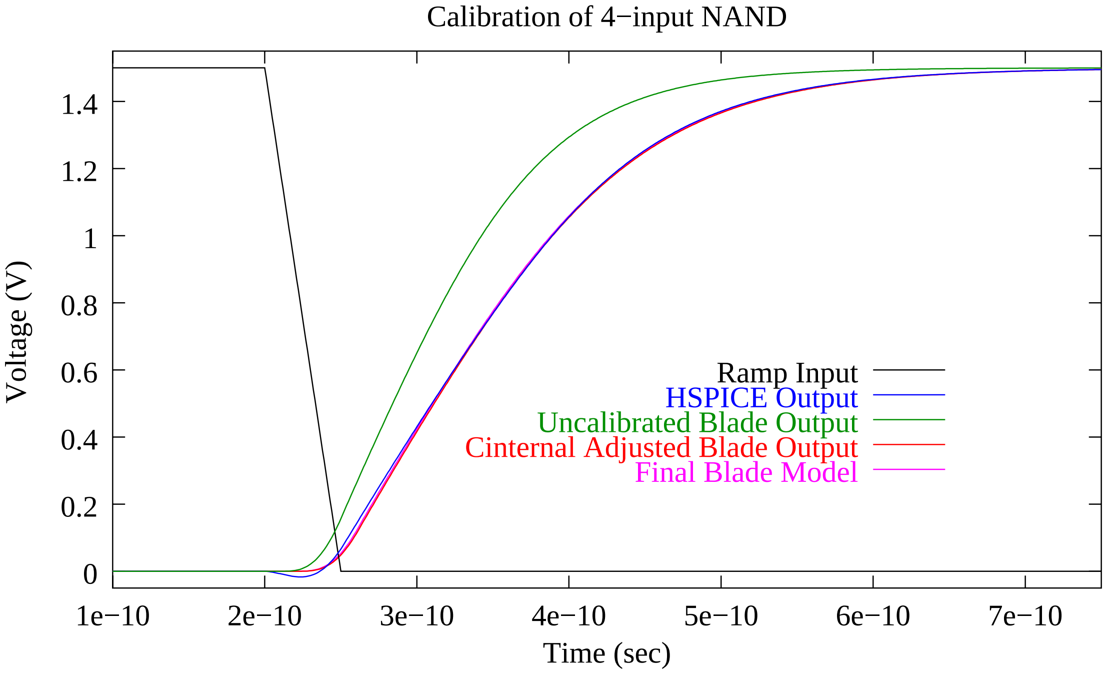


Figure 4.6: Calibration of a four-input NAND from pin A2 to output pin Z.

In the case of the four-input NAND gate, the internal capacitance had a dramatic effect on the shape of the output voltage waveform, but the parasitic time adjustment was relatively minimal. This was not always the case. Table 4.1 provides some example calibration parameters for a small selection of cell input/output pin paths. The RMS values represent the differences exhibited between the shape of the SPICE and BLADE waveforms as described in the previous chapter.

Table 4.1: Calibration Parameters for Sample Cells Paths

Cell	Input Pin	Output Pin	Internal Cap (pf)	Time Offset (nsec)	RMS Error (nsec)
AOI	A1	Z	0.0111	0.045	0.00226
INV	I	ZN	0.0028	0.003	0.00038
NAND	A2	ZN	0.02	0.010	0.00114
OR	A2 *internal*	*internal* Z	0.0065 0.002	0.004	0.00060
2-input XOR	A2	Z	0.0084	0.028	0.00115
3-input XOR	A1	Z	0.0745	0.160	0.00817

Figure 4.7 shows HSPICE and BLADE results for a two-input OR gate. The output of the internal node referenced in Table 4.1 is shown also.

4.2.2 Using the Models

Calibrated BLADE models were evaluated and compared to HSPICE runs under identical conditions. Various input waveform shapes, from ramps to noisy input signals were applied. Various output loads were tested, from simple capacitors to multi-segment π-models representing reduced-order networks. In all cases the waveforms produced by the BLADE models matched their HSPICE counterparts to within 1-2%[1].

The ability of BLADE to accurately handle noisy inputs, as shown in Fig-

---

[1] Error margins were determined by comparing the RMS of the differences between the BLADE and HSPICE waveforms to the intrinsic delay measured by HSPICE.

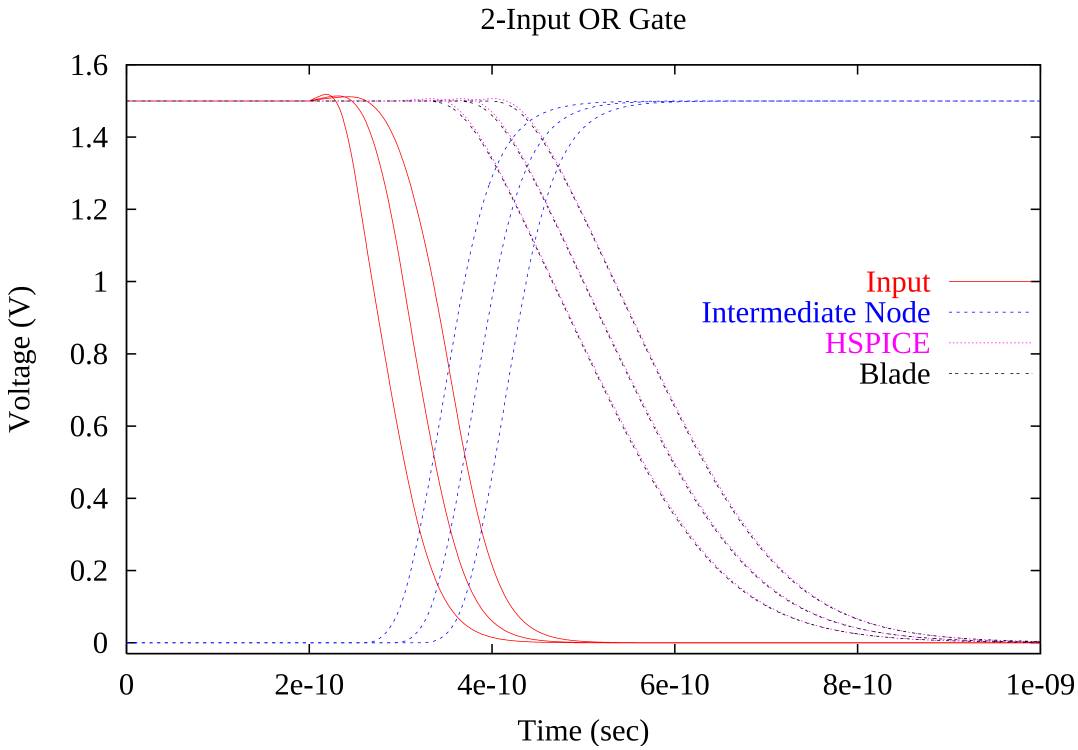


Figure 4.7: A waveform driving pin A2 of a two-input OR gate. HSPICE and Blade results are virtually identical. The internal node waveform (the output of the NOR gate feeding the INV gate) also is shown.

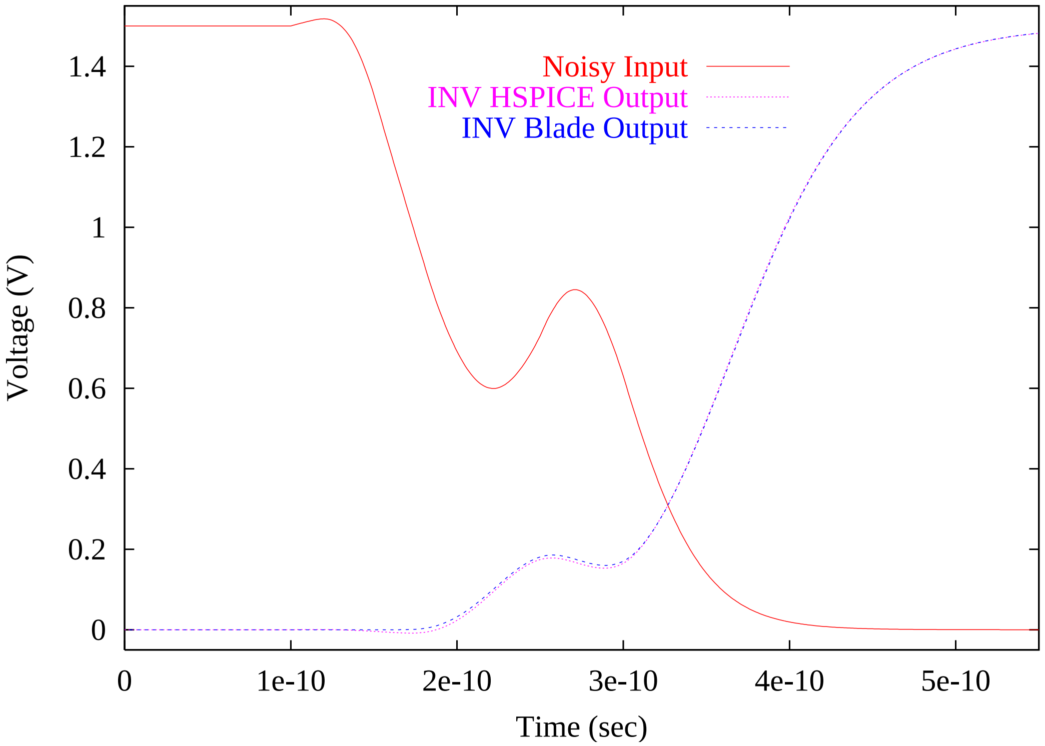


Figure 4.8: A noisy input signal applied to an inverter driving a reduced-order π-model. Blade and HSPICE outputs are virtually identical. This gate had no noise immunity characteristics, so the Blade model was not enhanced with a noise immunity filter.

ure 4.8, is unique for a simulation model that is not based on transistor-level analysis. As design feature sizes move below 0.18 μm, the capacitive coupling between interconnect will account for over 50% of the total wire capacitance. Noise caused by this interconnect coupling can increase the total stage delay by as much as 100%. Accurately modeling both cell and interconnect delay in the presence of noise will only become more critical as feature sizes continue to shrink and noise problems more frequently manifest.

Distinct tails in a cell's output voltage waveform are also accurately calculated using the Blade cell model. Figure 4.9 demonstrates that Blade models can both produce and consume such arbitrary voltages.

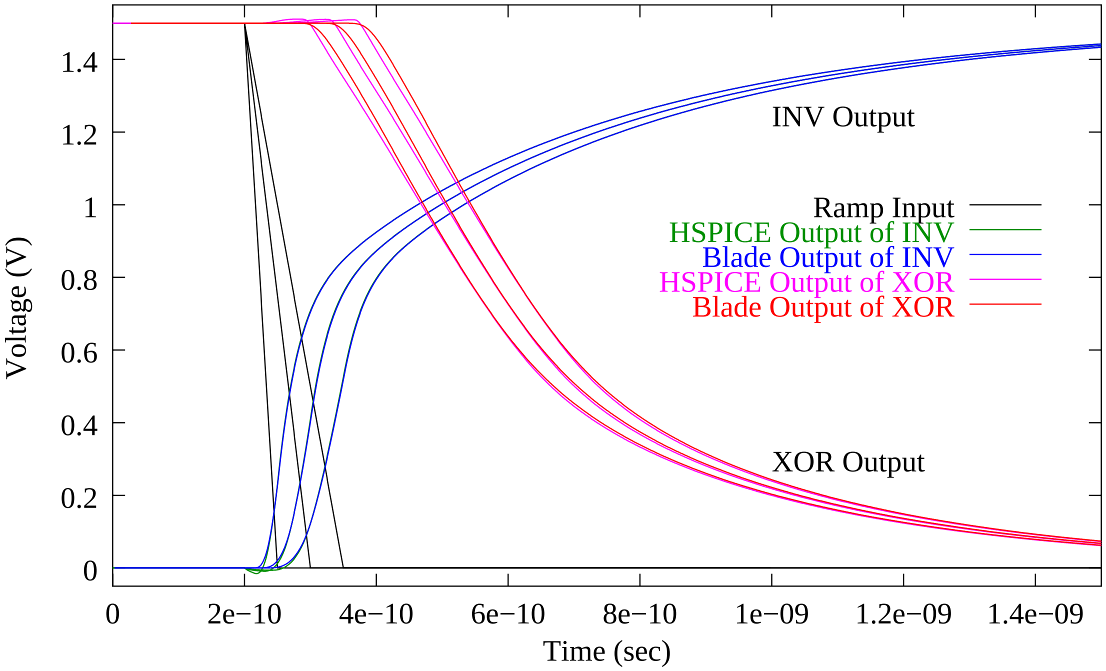


Figure 4.9: A ramp input drives an inverter to produce an output voltage waveform with a distinct tail. This waveform, in turn, drives an XOR cell. HSPICE and Blade results are nearly identical showing that Blade can both produce and consume arbitrary voltage waveforms exhibiting distinct tails.

4.2.3 Performance

Unlike SPICE, the Blade model runtime is independent of the number of transistors or parasitic elements in the cell netlist. Once the DC current information is empirically gathered and tabulated and the model is calibrated, Blade runtime is governed by the number of nonlinear iterations to convergence, the number of primitive gates in a cell, and the type of load driven.

Experimental results are presented in Table 4.2. Results are shown for a 1-

picosecond stepsize and a 1-nanosecond transient analysis period (1001 nonlinear iterations). On the average, one nonlinear iteration took 125 nanoseconds to complete. The complete simulation of a gate consumed 1.4Mb of memory. The OR gate took twice as long as it is composed of two cells: a NOR gate followed by an INV gate.

With the exception that secant iteration requires an additional BLADE model evaluation to seed the algorithm, the use of secant iteration did not negatively impact the performance of the BLADE runtime engine. After the initial two points were calculated, secant iteration looped until convergence was achieved. In the BLADE runtime engine, convergence is defined as a voltage differential of no more than 0.001V. For 1001 nonlinear iterations, the minimum number of total loops would be 1001 with 3003 BLADE model evaluations. On the average there were 1006 total loops with 3009 model evaluations over the 1-nanosecond transient analysis period.

4.2.4 The Noise Immunity Filter

Some cells, such as the INV, were highly sensitive to the presence of noise in the input waveform. Others, such as the small AOI shown in Figure 4.10, exhibited some degree of noise immunity. When the this AOI cell is used with a noise immunity filter of RC time constant 0.05 nanoseconds, the resultant model time offset was -0.008 nanoseconds.

The same noisy waveform was used to drive the largest AOI cell in the library. As shown in Figure 4.11, the presence of the noise immunity filter dra-

---

[2] The CPU time listed is only for the transient analysis portion of the HSPICE run. It does not include input, output, setup time, or other portions of the HSPICE run.

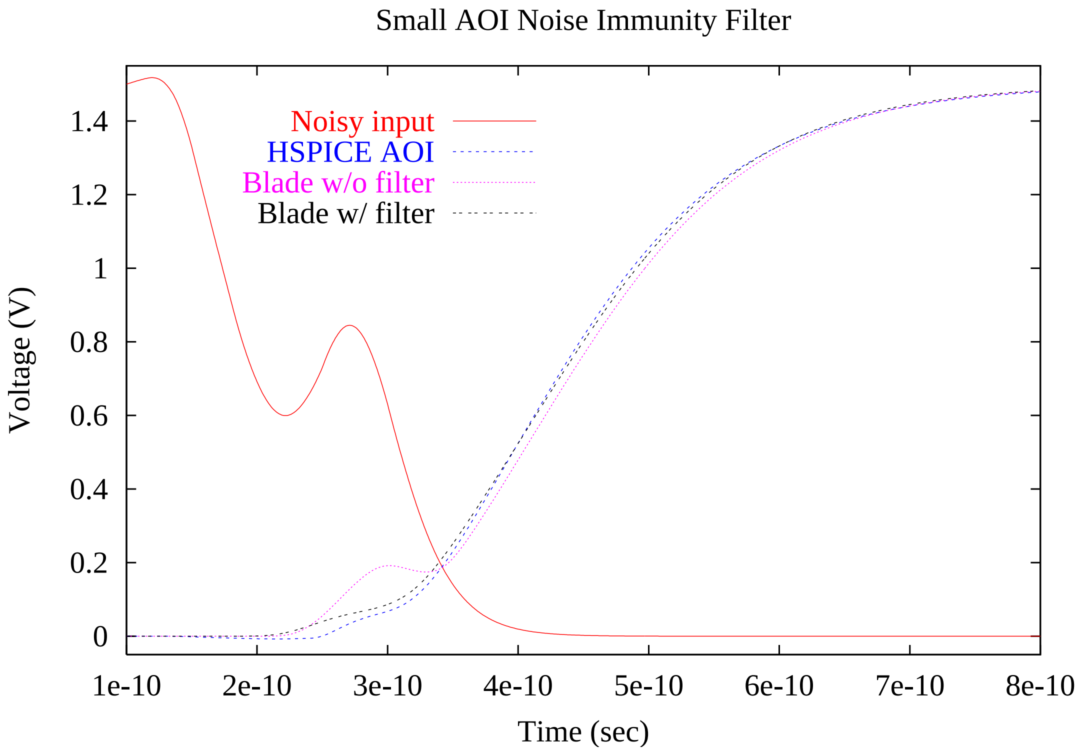


Figure 4.10: A noise immunity filter with an RC time constant of 0.05 nanoseconds, when applied to the voltage input of a small AOI cell, results in an output voltage waveform nearly identical to the SPICE waveform.

Table 4.2: Blade CPU Runtime Results

Cell	Number/Type of Elements	HSPICE Time[2]	Blade Time	Speedup
AOI	18 transistors 118 capacitors 4 diodes 141 resistors	0.67s	129μsec	5,200
INV	8 transistors 59 capacitors 1 diodes 72 resistors	0.35s	125μsec	2,800
OR	14 transistors 95 capacitors 2 diodes 127 resistors	0.53s	250μsec	2,120
XOR	35 transistors 316 capacitors 3 diodes 345 resistors	2.23s	122μsec	18,300

matically impacted the result of the Blade model. In this case, a noise filter with RC time constant of 0.026 nanoseconds was used to drive a Blade model with a time offset of 0.113 nanoseconds.

The use of a noise immunity filter against saturated linear ramp input waveforms resulted in nearly identical RMS values (when compared to prior Blade calibration runs) while causing an increase in the error margin to 4% - 5%. The Blade results were always more pessimistic than the SPICE results. Alternatively, reducing the time offset resulted in standard Blade error margins while increasing error margins for noisy signals to 4% - 5% (optimistic when compared to SPICE). Error margins for saturated linear ramps were greater than those for other input waveforms as the output of the noise immunity filter differed from the

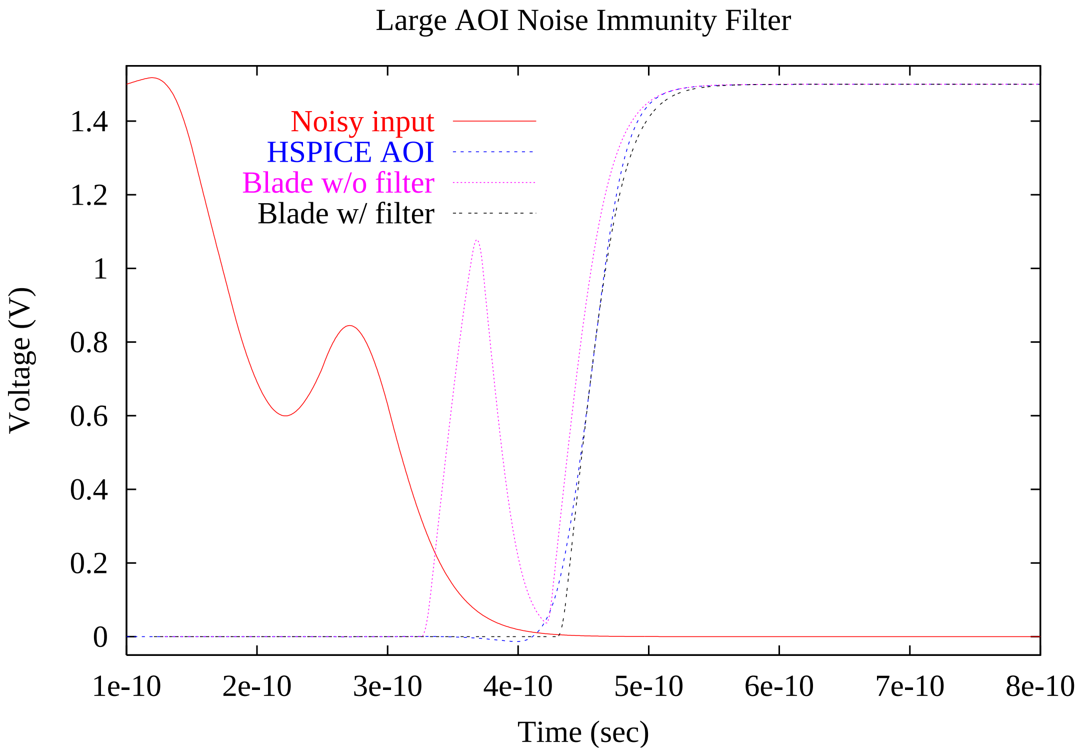


Figure 4.11: Without a noise immunity filter, the Blade model for a large AOI generates an output spike not exhibited by SPICE. However, with the filter added, the SPICE and Blade models are nearly identical.

ramps to a greater degree than for realistic waveform inputs.

4.3 RAZOR

RAZOR is written in C++ and can run as part of the BLADE runtime engine to produce stage delay values or in a standalone mode to simply compute interconnect delay. When used in conjunction with BLADE, the fixed-timestep PWL created by BLADE during model evaluation is transparently passed to RAZOR to compute the interconnect delay. When used in standalone mode, the input can be any arbitrary PWL representing a voltage waveform. If the PWL does not use a fixed timestep, a new fixed-timestep PWL is created to approximate the input.

Poles and residues representing the interconnect must be created prior to using RAZOR. During model development and performance runs, RICE [50] was used to create the interconnect representations.

The development and runtime environment for RAZOR was identical to that used for BLADE. However, unlike BLADE experiments, direct comparisons to HSPICE for interconnect-only analysis were impossible due to input line-size limitations imposed by HSPICE. A 1000-segment PWL could not be consumed by HSPICE for a 1:1 comparison of results. Thus, RAZOR analysis was performed indirectly using two different methods. In the first, a reduced (approximate) PWL was created for use by HSPICE. In the second the entire stage was simulated by HSPICE and by the BLADE/RAZOR combination and the results compared.

In order to create a reduced PWL for use by HSPICE, a separate program was written that determined the best points to pick in the PWL that most accurately represented the entire PWL. Dynamic programming was used to determine these points efficiently. Use of this program as an approach to avoid fixed timesteps in RAZOR is impractical due to the CPU time requirements. For example, reducing a 1000-point PWL to a 10-point PWL took 2 CPU seconds or approximately $25,000\times$ the CPU runtime of a RAZOR run using the full 1000 points at a fixed timestep.

4.3.1 Performance Overview

For a given number of segments to describe a voltage waveform and a given number of poles to represent the interconnect, RAZOR executes in a fixed time period. Whereas BLADE uses a nonlinear solver to determine current flow from the BLADE model to its load, RAZOR uses a purely deterministic solution for interconnect analysis.

Table 4.3 summarizes some of the performance results exhibited by RAZOR. As previously stated, interconnect analysis using RAZOR is an $O(np)$ approach where n is the number of PWL segments and p is the number of poles.

RAZOR memory consumption is governed by the number of PWL segments. For a 1000-segment PWL, RAZOR consumed 1.2Mb of memory.

In order to perform the HSPICE analysis for the runtime benchmarks, large PWLs used as input to RAZOR were reduced to 20-segment PWLs using the dynamic programming approach previously described. Thus, the HSPICE results were independent of the number of PWL segments used in RAZOR or the number of poles used in the reduced-order interconnect model.

Table 4.3: RAZOR CPU Runtime Results

PWL Segments	Poles	RAZOR Time	HSPICE Time[3]	Speedup
1000	2	80μsec	1.2s	15,000
500	2	44μsec	1.2s	27,300
1000	4	115μsec	1.2s	10,500
1000	2	82μsec	2.7s	33,000
800	2	63μsec	2.7s	43,000

4.3.2 Accuracy

Figure 4.12 shows a comparison of the BLADE and RAZOR stage model compared to HSPICE. A stage consisting of an inverter driving a large RC tree representing the interconnect was simulated using HSPICE and the BLADE/RAZOR stage model. HSPICE and RAZOR results were compared at the three terminals of the tree. The results of the two runs were virtually indistinguishable.

As shown in Figure 4.13, RAZOR is not limited to monotonic waveforms. The noisy waveform created by BLADE in the previous section can be consumed by the RAZOR interconnect model just as easily and accurately. Again, HSPICE and RAZOR results are indistinguishable.

---

[3] The CPU time listed is only for the transient analysis portion of the HSPICE run. It does not include input, output, setup time, or other portions of the HSPICE run.

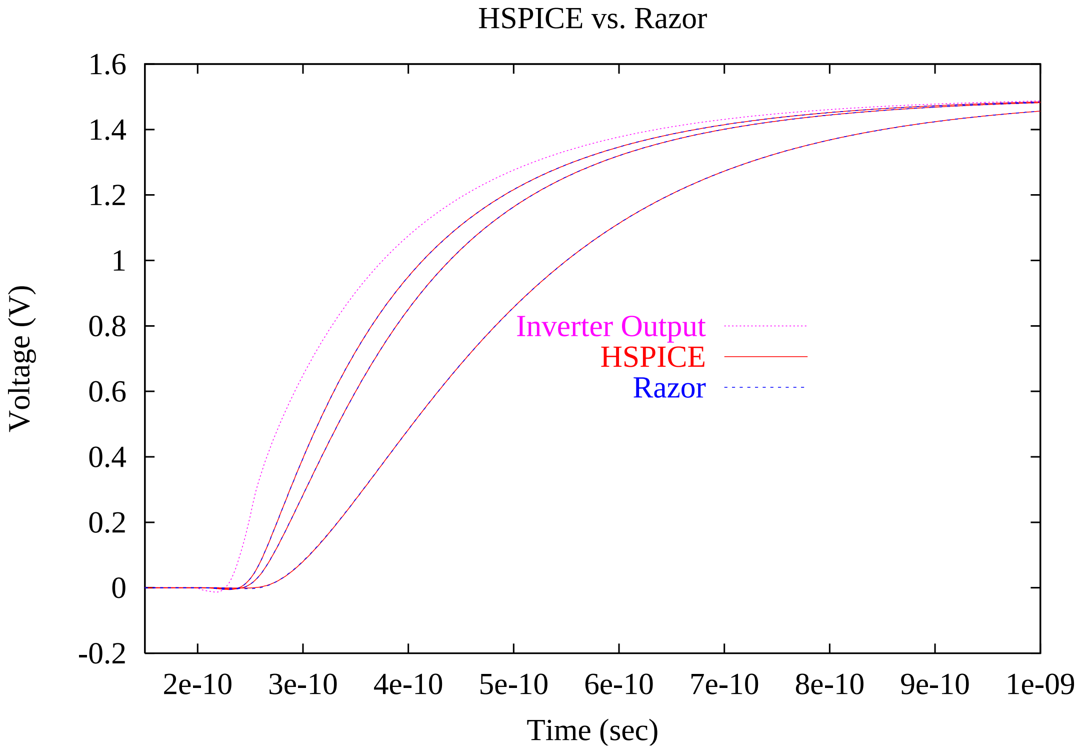


Figure 4.12: The voltage waveforms caused by an inverter driving a large RC tree representing the interconnect are shown. The BLADE inverter and RAZOR stage model are virtually identical to the same circuit in HSPICE. The plotted waveforms are those measured at the far three ends of the interconnect.

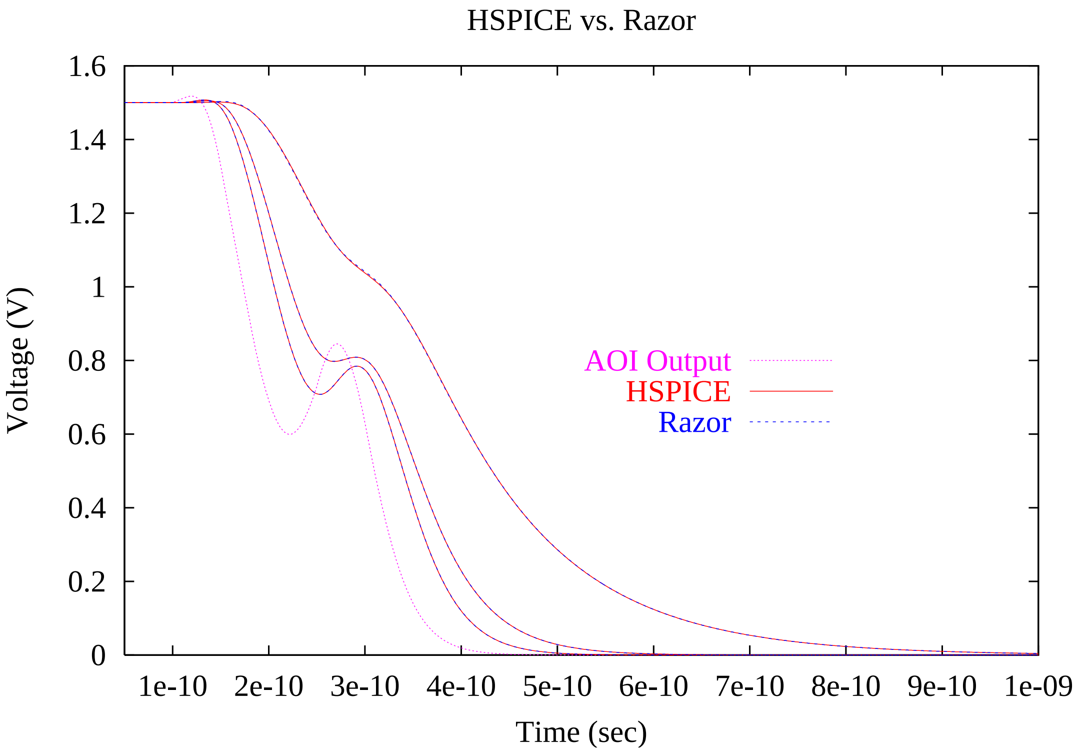


Figure 4.13: A noisy signal output by a cell driving an interconnect load. Razor results are nearly identical to those produced when HSPICE modeled the cell and interconnect simultaneously.

4.4 Final Remarks

The Blade and Razor environments uniquely provide both speed and accuracy. Two obvious questions arise: How fast? How accurate? In order to address these questions, all Blade and Razor comparisons were made using a commercial cell library simulated with the industry standard simulator, Avant! HSPICE. Cell netlists were obtained by performing parasitic extraction against the cell layout resulting in the presence of transistors, resistors, capacitors, and diodes in the simulated netlists. Additionally, these libraries were designed for leading edge, nanometer designs that exhibit the pathological effects just now being seen.

Valid comparisons between these environments only can be made if the Blade, Razor, and HSPICE control variables are identical. In other words, a transient analysis run in HSPICE using a 1-picosecond stepsize cannot be compared to a Blade run using a 2- or 3-picosecond stepsize. In creating the benchmark data, the same control variables were used by Blade, Razor, and HSPICE, to the greatest extent possible. The result is an "apples to apples" comparison between the applications.

It is also important to note that HSPICE CPU times cited with respect to Blade and Razor are only for the transient analysis portions of the HSPICE simulation. HSPICE listings contain the CPU time required for many different parts of the total run including operating point evaluation, transient analysis, input, output, error checks, and setup time. In order to obtain a true comparison between Blade/Razor and HSPICE, only the transient analysis times are compared between the systems.

Publications of models created prior to the creation of BLADE and RAZOR also contain benchmark information. For example, [21] claims a speedup of up to 1000× over SPICE. No attempt is made to compare BLADE and RAZOR against such runtime claims. None of the papers reviewed provided enough information to determine the exact conditions under which these simulations occurred. For example, there is a vast difference between Berkeley SPICE and Avant! HSPICE in terms of accuracy, speed, and transistor support. There is also little to no information available to determine what control parameters were used by the authors during either the SPICE simulations or the simulations of their models.

One obvious solution would be to implement each model reviewed on a standard platform and run comparisons locally. Unfortunately, rewriting each of these models from scratch was impractical. Additionally, the papers themselves usually did not provide enough information to accurately reproduce the authors' results. Even when existing source code written by the authors was obtained and locally compiled, as was the case for several models, the presence of hardcoded parameters and functions made comparisons against cells and interconnect available for BLADE runs impossible.

While direct runtime comparisons cannot be made to prior work easily, several conclusions can be drawn from the published material. First, the fastest models benchmarked in existing literature exhibited runtimes that were 1400× better than SPICE at best. Runtimes for BLADE showed minimum improvements of 2120× when compared to HSPICE. Second, BLADE is able to both produce and consume arbitrary waveforms, including noisy waveforms and waveforms with distinct tails. No other published macromodels were able to produce these results.

While SPICE-like transistor-level analysis programs are able to perform this type of analysis, they typically run at speeds of 25× to 50× faster than SPICE. Finally, none of the macromodel literature presented evaluations of complicated, compound cells (like the XOR) or cells with netlists that were parasitically extracted. All BLADE runs were made using a nanometer cell library used in leading-edge production designs today.

Chapter 5

Conclusion

In the last decade, the dominance of interconnect delay as a component of the total path delay has resulted in great strides in interconnect modeling while ignoring the data generator for that interconnect: the cell model. Cell models today look very much like they did 15 years ago. Unfortunately, such models are insufficient for nanometer effects that must now be accounted for.

Empirical cell models can be evaluated rapidly, but their evaluation yields only a single value used to represent intrinsic delay and output transition time. Models used by nonlinear solvers, such as SPICE, can yield highly accurate results, but only at the expense of CPU runtime and memory. The penalty associated with the use of these models is so severe that IC designers in the back end of the flow, when accuracy is critical, must use the less accurate empirical models in order to meet time-to-market constraints.

This dissertation has presented two new modeling technologies. The first, Blade, provides near-SPICE accuracy while evaluating at speeds tens of thousands of times faster than HSPICE. Razor, the second model, combines the results of Blade with an interconnect model to enable accurate stage delay modeling. The resultant stage delay model is sufficiently fast and accurate enough to be embedded within applications used during the back end of the design flow.

Unlike empirical models, Blade and Razor both produce and consume arbitrarily complex waveforms. This makes their use in environments for which characterization cannot be performed a priori, such as noisy environments, especially attractive. The combination of near-SPICE accuracy, extremely short runtimes, and the production and consumption of complex voltage waveforms uniquely distinguishes BLADE and RAZOR from other gate-level systems.

While accuracy is vital to the IC design flow, consistency between the front end and the back end of the flow is also required. Back-end models that are inconsistent with those used in the front end of the design flow can cause additional overhead and can, in fact, result in a negative performance impact on the overall design. The easiest way to achieve consistency with highly accurate back-end models is to use accurate front-end models. To meet these accuracy requirements, a new characterization and modeling methodology, over sampling and data reduction, is presented.

Over sampling and data reduction yields highly accurate characterization table models for cells. Once a cell's response surface has been determined, the best set of points with which to model that response surface can be selected. Over sampling yields this highly accurate view of the overall cell response surface. Data reduction, through a dynamic programming mechanism, reduces this large table to a manageable size for use in CAD applications. Since the introduction of over sampling and data reduction [11], hundreds of commercial cell libraries have been characterized using this technique.

Together, these technologies combine to provide IC designers with the most accurate cell and interconnect modeling without sacrificing evaluation speed.

Bibliography

[1] Emrah Acar, Florentin Dartu, and Lawrence T. Pileggi. TETA: Transistor-level waveform evaluation for timing analysis. In *IEEE Transactions on CAD*, volume 21, pages 605–616, May 2002.

[2] Charles J. Alpert, Anirudh Devgan, and Chandramouli V. Kashyap. RC delay metrics for performance optimization. In *IEEE Transactions on CAD*, volume 20, pages 571–582, May 2001.

[3] R. Arunachalam, F. Dartu, and L. T. Pileggi. CMOS gate delay models for general RLC loading. In *Proceedings 1197 IEEE Internation Conference on Computer Design: VLSI in Computers and Processors*, pages 224–229, October 1997.

[4] J. Bracken, V. Raghavan, and R. Rohrer. Interconnect simulation with asymptotic waveform evaluation. In *IEEE Transactions on Circuits and Systems*, volume 39, 1992.

[5] Alexander Chatzigeorgiou, Spiridon Nikolaidis, and Ioannis Tsoukalas. A modeling technique for CMOS gates. In *IEEE Transactions on CAD*, volume 5, pages 557–575, May 1999.

[6] B. R. Chawla, H. K. Cummel, and P. Kozak. MOTIS - a MOS timing simulator. In *IEEE Transactions on Circuits and Systems*, volume CAS-22(12), pages 901–910, December 1975.

[7] H. C. Chen and D. Du. Path sensitization in critical path problem. In *IEEE Transactions on CAD*, volume 12, pages 196–207, 1993.

[8] Liang-Chi Chen, Sandeep K. Gupta, and Melvin A. Breuer. A new gate delay model for simultaneous switching and its applications. In *38th IEEE/AMC Design Automation Conference Proceedings*, pages 289–294, 2001.

[9] Y. Cheng, M. Chan, K. Hui, M. C. Jeng, Z. H Liu, J. H. Huang, Kai Chen, P. K. Ko, and C. Hu. *BSIM3v3 Manual (Final Version).* University of California at Berkeley, December 1996.

[10] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* McGraw-Hill, Inc., 1990.

[11] John F. Croix and D. F. Wong. A fast and accurate technique to optimize characterization tables for logic synthesis. In *34th IEEE/ACM Design Automation Conference Proceedings*, pages 337–340, 1997.

[12] John Darringer, Evan Davidson, David J. Hathaway, Bernd Koenemann, Mark Lavin, Joseph K. Morrell, Khalid Rabmat, Wolfgang Roesner, Erich Schanzenbach, Gustavo Tellez, and Louise Trevillyan. EDA in IBM: Past, present, and future. In *IEEE Transactions on CAD*, volume 19, pages 1476–1497, December 2000.

[13] F. Dartu, N. Menezes, J. Qian, and L. T. Pillage. A gate-delay model for high speed CMOS circuits. In *31st IEEE/ACM Design Automation Conference Proceedings*, pages 576–580, 1994.

[14] F. Dartu and L. T. Pileggi. TETA: Transistor-level engine for timing analysis. In *35th IEEE/ACM Design Automation Conference Proceedings*, pages 595–598, 1998.

[15] Florentin Dartu, Noel Menezes, and Lawrence T. Pileggi. Performance computation for precharacterized CMOS gates with RC loads. *IEEE Transactions on CAD*, pages 544–553, May 1996.

[16] Florentin Dartu and Lawrence T. Pileggi. Calculating worst-case gate delays due to dominant capacitance coupling. In *34th IEEE/AMC Design Automation Conference Proceedings*, pages 46–51, 1997.

[17] Anirudh Devgan and Ronald A. Rohrer. Adaptively controlled explicit simulation. In *IEEE Transactions on CAD*, volume 13, pages 746–761, June 1994.

[18] W. C. Elmore. The transient response of damped linear network with particular regard to wideband amplifier. *Journal of Applied Physics*, (19):55–63, 1948.

[19] Binay J. George, Markus Wloka, and Sean Tyler. Method for deriving a piecewise linear model. *U. S. Patent*, December 13, 1994. No. 5,373,457.

[20] Mohamed Hafed, Mourad Oulmane, and Nicholas C. Rumin. Delay and current estimation in a CMOS inverter with a RC load. In *IEEE Transactions on CAD*, volume 20, pages 80–89, January 2001.

[21] Akio Hirata, Hidetoshi Onodera, and Keikichi Tamaru. Proposal of a timing model for CMOS logic gates driving a CRC π load. In *Proceedings IEEE International Conference on Computer-Aided Design*, pages 537–544, November 1998.

[22] R. B. Hitchcock, G. L. Smith, and D. D. Cheng. Timing analysis of computer hardware. *IBM Journal of Research and Development*, pages 100–105, January 1982.

[23] C. W. Ho, A. E. Ruehli, and P. A. Brennan. The modified nodal approach to network analysis. In *IEEE Transactions on Circuits and Systems*, pages 504–509, 1975.

[24] Ellis Horowitz and Sartaj Sahni. *Fundamentals of Computer Algorithms.* Computer Science Press, 1978.

[25] Steve H. Jen and Bing Sheu. A compact and unified MOS DC current model with highly continuous conductances for low-voltage IC's. In *IEEE Transactions on CAD*, volume 17, pages 169–172, February 1998.

[26] Andrew B. Kahng and Sudhakar Muddu. An analytical delay model for RLC interconnects. In *IEEE Transactions on CAD*, volume 16, pages 1507–1514, December 1997.

[27] Timothy Kam, Shishpal Rawat, Desmond Kirkpatrick, Rabindra (Rob) Roy, Gregory S. Spirakis, Naveed Sherwani, and Craig Peterson. EDA challenges facing future microprocessor design. In *IEEE Transactions on CAD*, volume 19, pages 1498–1506, December 2000.

[28] Chandramouli V. Kashyap, Charles J. Alpert, and Anirudh Devgan. An "effective" capacitance based delay metric for RC interconnect. In *Proceedings IEEE International Conference on Computer-Aided Design*, pages 229–234, November 2000.

[29] Noriya Kobayashi and Sharad Malik. Delay abstraction in combinational logic circuits. In *IEEE Transactions on CAD*, volume 16, pages 1205–1212, October 1997.

[30] Jaong-Taek Kong and David Overhauser. *Digital Timing Macromodeling for VLSI Design Verification.* Kluwer Academic Publishers, 1995.

[31] Alexander Korshak and Jyh-Chwen Lee. An effective current source cell model for VDSM delay calculation. In *IEEE International Symosium on Quality Electronic Design*, pages 296–300, 2001.

[32] Kenneth S. Kundert. Sparse matrix techniques. In A. E. Ruehli, editor, *Circuit Analysis, Simulation and Design*, chapter 6. North-Holand, 1986.

[33] W. K. C. Lam, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Exact minimum cycle times for finite state machines. In *31st IEEE/ACM Design Automation Conference Proceedings*, pages 100–105, 1994.

[34] Ying Liu, Lawrence T. Pileggi, and Andrzej J. Strojwas. *ftd*: An exact frequency to time domain conversion for reduced order RLC interconnect models. In *35th IEEE/ACM Design Automation Conference Proceedings*, pages 469–472, 1998.

[35] M. J. Maron. *Numerical Analysis: A Practical Approach.* Macmillan Publishing Co., Inc., 1982.

[36] M. D. Matson and L. A. Glasser. Macromodeling and optimization of digital MOS VLSI circuits. *IEEE Transactions on CAD*, pages 659–678, October 1986.

[37] William J. McCalla. *Fundamentals of Computer-Aided Circuit Simulation.* Kluwer Academic Publishers, 1988.

[38] Clayton B. McDonald and Randal E. Bryant. Computing logic-stage delays using circuit simulation and symbolic Elmore analysis. In *38th IEEE/AMC Design Automation Conference Proceedings*, pages 283–288, 2001.

[39] M. Monachino. Design verification system for large-scale LSI designs. *IBM Journal of Research and Development*, pages 89–99, January 1982.

[40] L. W. Nagel. *A Computer Program to Simulate Semiconductor Circuits*. PhD thesis, University of California, Berkeley, May 1975.

[41] Edouard Ngoya, Jean Rousset, and Juan J. Obregon. Newton-Raphson iteration speed-up algorithm for the solution of nonlinear circuit equations in general-purpose CAD programs. In *IEEE Transactions on CAD*, volume 16, pages 638–644, June 1997.

[42] P. R. O'Brian and L. T. Savarino. Modeling the driving-point characteristic of resistive interconnect for accurate delay estimation. In *Proceedings IEEE International Conference on CAD*, pages 512–515, 1989.

[43] Altan Odabasioglu, Mustafa Celik, and Lawrence T. Pileggi. PRIMA: Passive reduced-order interconnect macromodeling algorithm. In *Proceedings IEEE International Conference on CAD*, pages 58–65, 1997.

[44] John K. Ousterhout. A switch-level timing verifier for digital MOS VLSI. In *IEEE Transactions on CAD*, volume CAD-4, pages 336–348, July 1985.

[45] L. T. Pillage and R. A. Rohrer. Asymptotic waveform evaluation for timing analysis. In *IEEE Transactions on CAD*, volume 9, pages 352–366, 1990.

[46] L. T. Pillage, R. A. Rohrer, and C. Visweswariah. *Electronic Circuit And System Simulation Methods*. McGraw-Hill, Inc., 1995.

[47] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 1993.

[48] J. Qian, S. Pullela, and L. T. Pillage. Modeling the effective capacitance for the RC interconnect of CMOS gates. In *IEEE Transactions on CAD*, volume 13, pages 1526–1535, 1994.

[49] Vivek Raghavan and Ronald A. Rohrer. A new nonlinear driver model for interconnect analysis. In *28th IEEE/ACM Design Automation Conference Proceedings*, pages 561–566, 1991.

[50] C. L. Ratzlaff and L. T. Pillage. RICE: Rapid interconnect circuit evaluation using AWE. In *IEEE Transactions on CAD*, pages 763–776, 1994.

[51] Curtis L. Ratzlaff, Satyamurthy Pullela, and Lawrence T. Pillage. Modeling the RC-interconnect effects in a hierarchical timing analyzer. In *IEEE Custom Integrated Circuits Conference Proceedings*, pages 15.6.1–15.6.4, May 1992.

[52] Jorge Rubinstein, Paul Penfield, and Mark A. Horowitz. Signal delay in RC tree networks. In *IEEE Transactions on CAD*, volume 2, pages 202–211, July 1983.

[53] Yung-Ho Shih, Yusef Leblebici, and Sung-Mo Kang. ILLIADS: A fast timing and reliability simulator for digital MOS circuits. In *IEEE Transactions on CAD*, volume 12, pages 1387–1402, September 1993.

[54] M. Silveira, M. Kamon, and J. White. Efficient reduced-order modeling of frequency-dependent coupling inductances associated with 3-D interconnect structures. In *32nd IEEE/ACM Design Automation Conference Proceedings*, pages 376–380, 1995.

[55] Shang-Zhi Sun, David H. D. Du, and Hsi-Chuan Chen. Efficient timing

analysis for CMOS circuits considering data dependent delays. In *IEEE Transactions on CAD*, volume 17, pages 546–552, June 1998.

[56] Synopsys, Inc. *Library Compiler Reference Manual, Volume II*, 1995. Rev. 3.3b, Appendix D.

[57] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective.* Addison Wesley Publishing Company, second edition, 1993.

[58] Hakan Yalcin, Mohammad Mortazavi, Robert Palermo, Cyrus Bamji, Karem Sakallah, and John P. Hayes. Fast and accurate timing characterization using functional information. In *IEEE Transactions on CAD*, volume 20, pages 315–331, February 2001.

Vita

John Foy Crox, III was born in Louisville, Kentucky on 21 December 1963, the older of two children to Lt. Col. John F. Crox, Jr. and Joyce S. Crox. As the son of an Army officer, John spent the first 7 years of his life traveling from Germany to Ft. Knox, KY and points between.

John attended high school at St. Xavier High School in Louisville where he was awarded the Bausch & Lomb Honorary Science Award, the Saint Xavier Award for Outstanding Leadership (for his roles in student government), and the Saint Xavier Physics Award. Having earned 39 college credits while in high school, John entered the University of Louisville School of Engineering, Speed Scientific School, in 1981 to study computer engineering. While at U of L, John continued his involvement in student government as vice president of his freshman and sophomore classes. A 4.0 student, John was a member of Mortar Board and Tau Beta Pi and was a National Dean's List student. After $2\frac{1}{2}$ years John completed his B.S. and graduated from U of L *suma cum laude*.

In 1985 John changed his last name from Crox to Croix, an abbreviated form of the ancestral name *de la Croix*. He also changed his middle name to Francis.

John spent the next 8 years working for Texas Instruments, in Dallas, TX, primarily focused on computer-aided design application development. During this period he enrolled in Southern Methodist University for his M.S. degree in Computer Science at the School of Engineering under Dr. David Matula. In 1990, two years after entering SMU, John completed his degree with a 4.0 average and decided to pursue a Ph.D. degree.

In 1992, John moved to Austin, TX to work for IBM on the PowerPC design team. A few months after arriving in Austin, John enrolled at the University of Texas at Austin where he started classes in 1993 for his Ph.D. in Electrical and Computer Engineering.

John joined the AMD Athlon design team in 1995 and subsequently left in 1997 to co-found Silicon Metrics Corporation, where he is the Chief Technology Officer. He holds two patents and has outstanding applications for an additional six. John has been keynote speaker and invited guest speaker at several conferences and has written several articles and guest editorials for publications and conferences in the IC design community.

Permanent address: 1620 Sunterro Drive
Austin, Texas 78727

This dissertation was typeset with LaTeX[†] by the author.

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.