

AD-A285 794



RL-TR-94-124
Final Technical Report
September 1994



①

CURRENT ESTIMATION FOR ELECTROMIGRATION, HOT CARRIERS, AND VOLTAGE DROP

University of Illinois

Ibrahim N. Hajj, George Stamoulis, Ping-Chung Li,
Harish Kriplani, and Gautham Kamath

DTIC
ELECTE
OCT 27 1994

D

94-33335



16718

DTIC

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

94 10 26 023

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-94-124 has been reviewed and is approved for publication.

APPROVED:

Martin J. Walter
MARTIN J. WALTER
Project Engineer

FOR THE COMMANDER:

John J. Bart

JOHN J. BART, Chief Scientist
Reliability Sciences
Electromagnetics & Reliability Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (ERDD) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE September 1994		3. REPORT TYPE AND DATES COVERED Final Aug 92 - Aug 93	
4. TITLE AND SUBTITLE CURRENT ESTIMATION FOR ELECTROMIGRATION, HOT CARRIERS, AND VOLTAGE DROP				5. FUNDING NUMBERS C - F30602-92-C-0059 PE - 62702F PR - 2338 TA - 01 WT - PE	
6. AUTHOR(S) Ibrahim N. Hajj, George Stamoulis, Ping-Chung Li, Harish Eriplani, and Gautham Kamath				8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Illinois Coordinated Science Laboratory 801 S. Wright Street Champaign IL 61820				10. SPONSORING/MONITORING AGENCY REPORT NUMBER RI-TR-94-124	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory (ERDD) 525 Brooks Road Griffiss AFB NY 13441-4505				11. SUPPLEMENTARY NOTES Rome Laboratory Project Engineer: Martin J. Walter/ERDD/(315) 330-2735	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This report summarizes work in reliability analysis in VLSI CMOS circuits, particularly electromigration in metal lines, power busses and signal lines, hot carrier effects in devices, and voltage drop in power busses. In addition to process parameters, all of these effects can be shown to depend on current flow in the circuit. More specifically, electromigration and hot carrier induced degradation are both long-term effects and are related to the average current flow over time under all possible input signals that the design experiences. Maximum voltage drop, on the other hand, depends on finding the maximum current flow in the power busses, caused by a specific few inputs. Thus, our emphasis in this work is on developing fast and reliable methods for estimating average and maximum currents. In addition, we have also improved on our bus extractor to extract accurate RC models of power busses from layout information for electromigration and voltage drop estimation. This work is part of our goal of developing CAD tools for estimating physical reliability effects in VLSI designs, and to automatically modify the design, or at least suggest design changes, in order to improve the short and long term reliability of the design under realistic operating conditions.					
14. SUBJECT TERMS Electromigration, power busses, VLSI CMOS circuits				15. NUMBER OF PAGES 170	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED		18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED		19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	
				20. LIMITATION OF ABSTRACT U1	

TABLE OF CONTENTS

	Page
1. INTRODUCTION	2
2. ESTIMATION OF AVERAGE CURRENT WAVEFORMS	6
2.1 Introduction.....	6
2.2 Single Gate Probabilistic Simulation.....	7
2.3 Elimination During Switching	13
2.4 Delay Calculation	16
2.5 Output Event Merging.....	20
2.6 Signal Correlation.....	22
3. CURRENT ESTIMATION FOR HOT-CARRIER EFFECTS	29
3.1 Introduction.....	29
3.2 Hot-Carrier Effects	31
3.3 Gate Probabilistic Simulation.....	37
3.3.1 Probabilistic Waveform Description.....	37
3.3.2 Subcircuit Probabilistic Simulation Including Transition Times.....	38
3.3.3 Propagation Delay and Output Slew Rate	41
3.3.4 Substrate Current Estimation	42
3.4 Signal Overlap Handling.....	44
3.5 Implementation and Simulation Results	49
3.6 Transistor Degradation Model and Effects on Circuit Performance.....	52
3.7 HCE Resistant Design.....	59
4. MAXIMUM CURRENT AND VOLTAGE DROP ESTIMATION.....	67
4.1 Introduction.....	67
4.2 Maximum Current Estimates	70
4.3 The Proposed (iMax) Algorithm.....	73
4.3.1 Signal Representation.....	73
4.3.2 Independence Assumption	75
4.3.3 Single Gate Simulation.....	76
4.3.3.1 Calculating Uncertainty Sets.....	76
4.3.3.2 Calculating Uncertainty Intervals.....	78
4.3.4 Current Calculation	79
4.3.5 Implementation Details	80
4.3.6 Experimental Results.....	81

4.4	The Signal Correlation Problem	84
4.5	Resolving Signal Correlations.....	87
4.6	Partial Input Enumeration (PIE)	90
4.6.1	The Algorithm.....	90
4.6.2	Heuristic SC and Experimental Results	94
4.6.2.1	H_1 Heuristic	95
4.6.2.2	H_2 Heuristic	98
5.	BUS MODEL EXTRACTION	102
5.1	Introduction.....	102
5.2	Extraction of Resistance Models	102
5.3	On-the-Fly Numerical Method: The Boundary Element Method.....	111
5.4	Discretization of BEM Equations	114
5.4.1	Extension to Multiply-Connected Domains.....	118
5.4.2	Special Case of Points with Two Unknowns.....	118
5.4.3	Properties of the Resulting Linear System	120
5.4.4	Analytical Calculation of Matrix H and Matrix G Coefficients	121
5.4.5	Current Calculation	124
5.5	Comparison with Finite Element Method (FEM)	124
5.6	JET2 Capacitance Model Extraction	127
5.7	RC Models of Primitives	130
5.7.1	Rectangular Segment.....	131
5.7.2	L Primitive.....	132
5.7.3	T Primitive.....	134
5.7.4	Four-Way Junction Primitive.....	136
5.7.5	Contact Primitive.....	138
5.8	Algorithms and Data Structures of JET2	140
5.8.1	Structure of JET2	140
5.8.2	Rectangular Data Structure	143
5.8.3	Maximal Vertical and Horizontal Strip Formats	145
5.8.4	Primitive Identification.....	148
5.8.5	Contour Data Structure	149
5.8.6	FEM Analysis.....	151
5.8.7	BEM Algorithm.....	152
6.	SUMMARY AND CONCLUSIONS.....	153
7.	PUBLICATIONS AND REFERENCES	157

FINAL REPORT EVALUATION

Current Estimation For Electromigration, Hot Carriers, and Voltage Drop.
Contract F30602-92-C-0059

Reliability Analysis of VLSI circuit designs is necessary to reduce the time from concept to delivery without compromising reliability. The inclusion of reliability analysis into the design process for designs with thousands of gates and hundreds of inputs poses significant difficulty because of the computation times involved, and the very large input space. Deterministic simulations are inadequate to simulate the effects of months or years of circuit use when the number of inputs is large.

Probabilistic and Statistical techniques offer a way to obtain the averaged long term effects on circuits with many inputs and thousands of gates. This effort investigated statistical techniques applicable to digital CMOS designs. It applied probabilistic methods to hot electron degradation and incorporated input slew rate and signal correlations. It incorporated signal correlation effects into the statistical simulation for maximum current estimation.

A companion effort, "Prototype Rule Based Reliability Analysis for VLSI Circuit Designs" developed layout based reliability rules for hot electron degradation, and implemented them into a prototype reliability analysis tool. This approach has been applied to circuit designs with more than 1 million transistors in about 1 CPU hour. The rule based approach is not probabilistic, but is a worst case analysis assuming that every gate switches in every cycle.

Areas where future research should be directed include applying probabilistic and statistical methods to sequential circuits, reliability analysis at the gate level or at the hardware description language so that reliability becomes an integral part of the design process, and to extend the area of reliability issues being addressed from hot electrons and electromigration to other failure mechanisms; long and short term reliability problems.



Martin J. Walter
MARTIN J. WALTER
Project Engineer
RL/ERDD

1. INTRODUCTION

This research is concerned with the development of computer-aided methodologies and tools for assessing and improving the reliability of chip-level VLSI circuit designs. Many existing computer-aided design systems tend to ignore physical reliability constraints during the design phase, and rely on assessing circuit reliability by accelerated burn-in tests after manufacture. However, with the increase in chip complexity and the decrease in feature size, reliability issues can no longer be ignored during the design process and need to be addressed rigorously throughout the design cycle.

Reliability considerations include both long-term and short-term effects. Long-term effects are related to the physical aging of integrated circuit elements and lines. They include electromigration effects in the lines, and hot-carrier degradation and oxide breakdown in the devices. Short-term effects include voltage-drop and noise in the lines, electrostatic discharge through input protection circuits, system-noise induced latchup, and charge-particle induced soft errors.

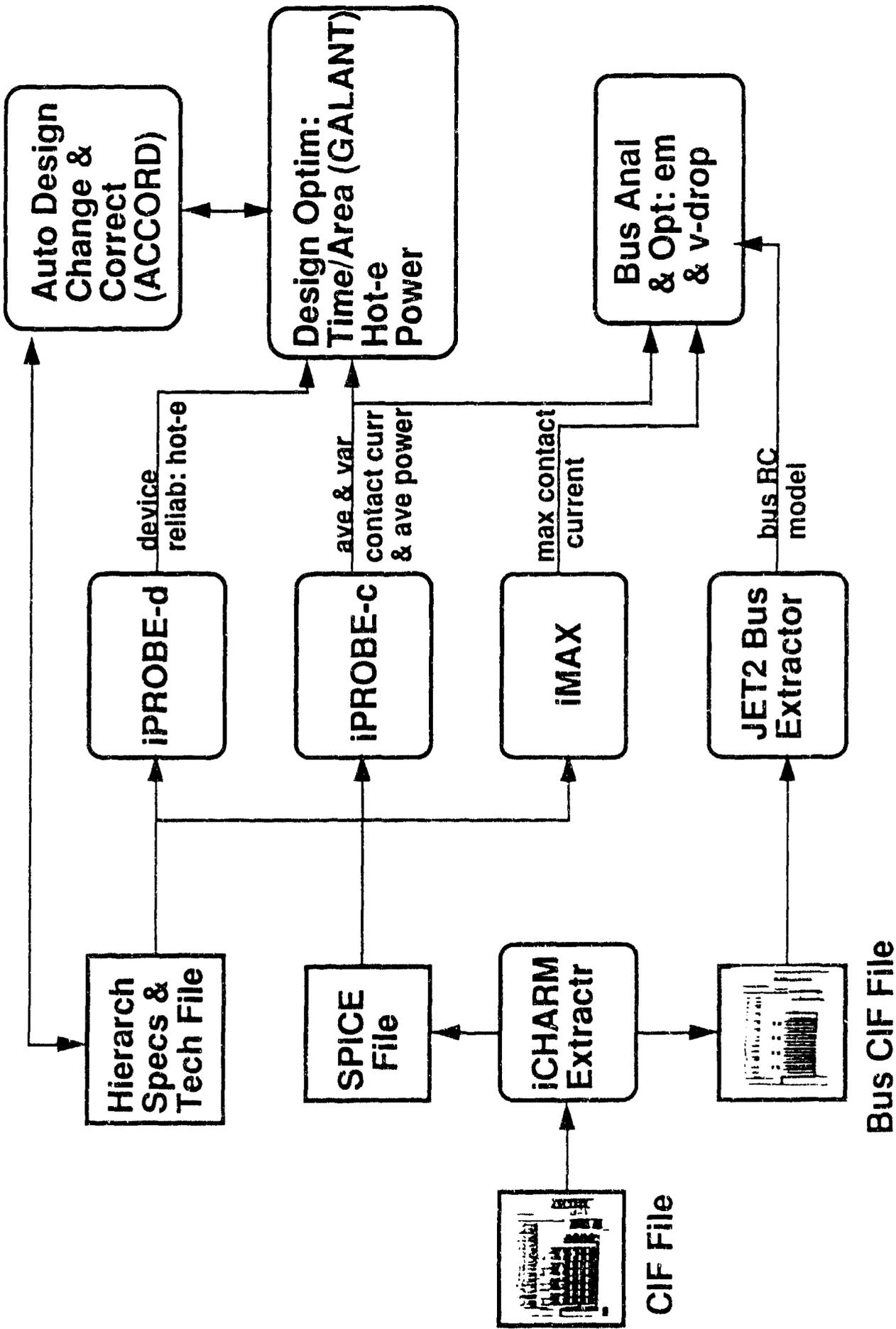
In this task, we have concentrated our efforts on three reliability issues: namely, electromigration detection and prevention, voltage-drop reduction in the power and ground busses, and hot-carrier induced degradation in the devices. These reliability issues can be shown to be related to the current flow in the circuit during switching activities. Electromigration is related to the average (as well as the variance) of the current density waveform in a metal line (averaged over all possible inputs). Hot-electron effects are related to the average current flowing through the devices, while maximum voltage-drop is related to the maximum current waveform flowing in the bus.

Before presenting our accomplishment during the past year, we give a brief overview of the CAD system that we are developing to implement and test our methodologies and algorithms. Figure 1 shows a block diagram of the CAD system. Given a design layout specified in CIF (Caltech Intermediate Format), which could be specified hierarchically, program iCHARM extracts SPICE-file from it, including interconnect parasitics, and sorts out the CIF subfiles of the power and ground busses, with bus contacts labeled to match the corresponding node connections in the extracted SPICE file. The extracted SPICE-file has the same hierarchical structure as the original CIF file.

The SPICE file forms the input to the simulators, iPROBE-c, iPROBE-d, and iMAX (The SPICE file can also be used as the input file to our other simulators to perform timing and fault simulation using input test vector sets; these other simulators are not indicated in the Figure). iPROBE-c is a probabilistic simulator which computes the average and variance current waveforms at declared contact points. iPROBE-d estimates the average relative damage due to hot-carrier effects within the circuit devices. iMAX computes an upper bound current waveform envelope at contact points. JET is a bus extractor dedicated to extracting the RC models of the bus. The outputs of JET, iMAX and iPROBE-c form the inputs to a bus analyzer and optimizer that recommends changes to the bus line widths, if necessary, in order to meet both electromigration and voltage-drop constraints. The outputs from iPROBE-c and iPROBE-d can be used as part of a multiobjective function in a design optimization program, which we are developing, that aims at optimizing the design with respect to area, timing, power, and reliability measures. We are also developing a program, ACCORD, for automatically changing the design at the logic and functional levels when such a change is recommended in the optimization process.

In Section 2, we present the accomplishments achieved in iPROBE-c, and in Section 3, the accomplishments in iPROBE-d. In Section 4, we report on the results obtained in iMAX. In Section 5, we describe the improvements done in the bus extraction, JET2. Finally in Section 6, we give a summary and conclusions.

Part of this work was also supported by the Semiconductor Research Corporation and by Texas Instruments, Inc. The authors would like to thank Professor Farid Najm for his contributions to this work; in particular his contribution to the maximum current estimation approach implemented in iMAX. They also would like to thank Carolyn Genzel for her help in preparing this report.



2. ESTIMATION OF AVERAGE CURRENT WAVEFORMS

2.1 Introduction

Average current estimation has many applications in VLSI circuit reliability analysis and design. One application is average power estimation. Another important application is electromigration estimation. Electromigration in metal lines has been shown to be experimentally related to the current density waveform flowing in the lines, as well as parameters related to the metal material and dimensions. In order to estimate electromigration effects in the power busses of a chip design under expected operating conditions, it is necessary to compute the average, as well as the variance, of the current density waveform in each section of the bus under all possible inputs that the design might experience. To do this, one has to first compute the average variance current waveforms drawn by the circuit at contact points to the bus. A brute force approach is to perform exhaustive simulation on the design, collect all the necessary data, and then compute average waveforms. Such an approach is prohibitively expensive, if not impossible, since the input space is usually very large. Another approach, which is sometimes used in industry, is to apply Monte Carlo methods to high-level functional description of the design to obtain toggle rates at all the circuit nodes, which in turn, can be used to obtain average waveform of each subcircuit or gate. However, such an approach usually does not include accurate delay and timing information since a large number of simulations are carried out. Our approach is to use high-level simulation methods to obtain signal statistics at the output of flip-flops in sequential designs. Using these signal statistics at inputs to the combinational blocks of the design, we apply probabilistic simulation techniques to obtain average and variance current waveform drawn by each gate or subcircuit including accurate timing information.

During the past year, we have carried out a number of improvements on our probabilistic simulation approach, both in accuracy and in speed. The improvements are carried out at the subcircuit or gate level, where the statistics of the current and voltage waveforms and the delays are computed more accurately, and at the global level, where signal correlations are computed and their effects on the subcircuit analysis are taken into account. In the following for completeness, we review our approach. We then explain the improvements we have done and include simulation results.

2.2 Single Gate Probabilistic Simulation

We consider MOS digital circuits that have been partitioned into channel-connected subcircuits or gates. We assume that the statistics of the signals at inputs to each gate are specified as a primary input or computed from the outputs of the driving gates in the form of probabilistic waveforms. Figure 2.1 shows an example of a typical probabilistic waveform. The statistics of the current waveform drawn by a CMOS gate after switching consist of the *expected value* and the *variance* waveforms where the expected value waveform $E[\mathbf{i}(t)]$ represents the average waveform over all possible power supply waveforms drawn by the subcircuit, and the variance waveform represents the variance of the distribution of the currents. The variance waveform is calculated as $V(\mathbf{i}(t)) = E[(\mathbf{i}(t) - E[\mathbf{i}(t)])^2]$ at every time point t .

The expected value and variance waveforms are assumed to be triangular in shape and each is specified by a peak value and a time span. To compute the peak value and the time span of the expected value and variance current pulses each gate is reduced to an equivalent inverter shown in Fig. 2.2. The p-part and the n-part of the gate are reduced to one equivalent edge each by a series-parallel reduction as follows: Each transistor in a gate has a

conductance g_{on} when it is *on* and zero when it is *off*. The probability of an nMOS transistor being *on* is the probability of its gate node being "high," while the probability of a pMOS transistor being *on* is the probability of its gate being "low." Thus the conductance of the transistor can be viewed as a random variable. The expected value $E[g]$ and the variance $V(g)$ of the conductance are defined respectively at any time t other than the time points where switching occurs as:

$$E[g] = g_{on} \times P_h(t) \tag{2.1}$$

$$V(g) = g_{on}^2 \times P_h(t) - (g_{on} \times P_h(t))^2 \tag{2.2}$$

where $P_h(t)$ the probability that the edge will conduct.

The aim of the reduction process is to find an equivalent edge between the output node and the ground node (V_{dd} node for the p-part) so that the current drawn at the ground contact (V_{dd} contact for the p-part) can be calculated. It is further assumed that the circuit can be reduced in a series-parallel fashion, which is the most prevalent case. Thus, at every step of

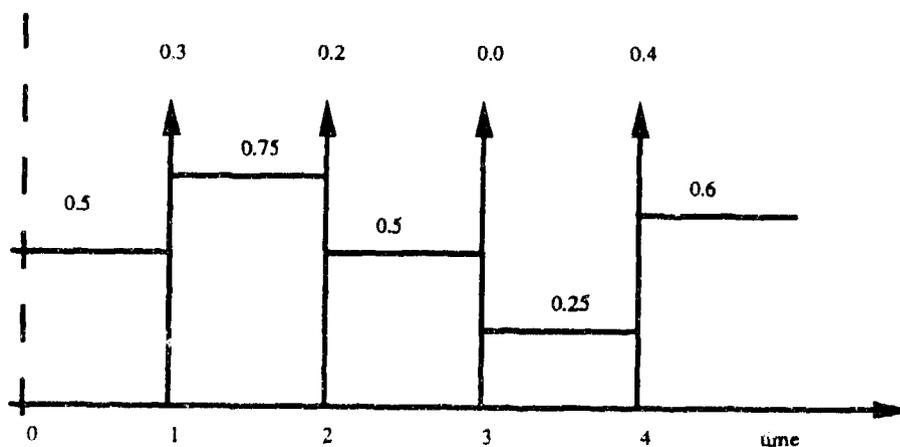


Figure 2.1 Example of a probabilistic waveform.

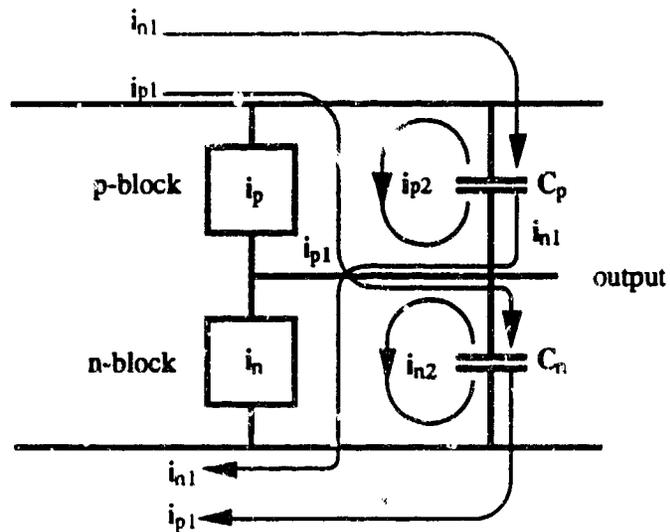


Figure 2.2 Model of a CMOS gate.

the reduction procedure two edges are combined into one. The analysis presented in the following sections will be for the n-part only, since the p-part is its complement. The following notation will be used in the sequel:

- x, y : the conductances of the two edges (transistors) to be reduced to one.
- g : the conductance of the resulting equivalent edge.
- P_{hx} : probability that x is conducting.
- P_{lhx} : probability that x is switching from *off* to *on*.

- Parallel reduction: Assuming independent inputs, the parallel case is straightforward.

The equivalent conductance is given by:

$$g = x + y \quad (2.3)$$

where x, y and g are random variables.

The stochastic quantities describing the equivalent edge, g , are given by the following

expressions:

$$P_{hp} = P_{hx} + P_{hy} - P_{hx}P_{hy} \quad (2.4)$$

$$P_{lhp} = P_{lhx}P_{ly} + P_{lhy}P_{lx} - P_{lhx}P_{lhy} \quad (2.5)$$

$$E[g] = E[x] + E[y] \quad (2.6)$$

$$V(g) = V(x) + V(y) \quad (2.7)$$

where the subscript p (for parallel) denotes those quantities associated with the equivalent edge. These results follow from simple probability theory and are *not* approximations but exact representations of the stochastic parameters of the equivalent edge.

- Series reduction: For the series case, the equivalent edge, g , is high or is switching from low to high are calculated by the following expressions.

$$P_{hs} = P_{hx}P_{hy} \quad (2.8)$$

$$P_{lhs} = P_{lhx}P_{hy} + P_{lhy}P_{hx} - P_{lhx}P_{lhy} \quad (2.9)$$

the quantities associated with the equivalent edge are here denoted by the subscript s .

In computing the expected value and the variance of the series combination, we follow a different approach from the one proposed in [1] and [2], which leads to more accurate results.

We consider the equivalent conductance to be expressed as:

$$g(x,y) = \frac{xy}{x+y} \quad (2.10)$$

Following [3], a Taylor series expansion of (2.10) gives:

$$E[g] = g(x,y) + \frac{1}{2} \left(\frac{\partial^2 g}{\partial x^2} \sigma_x^2 + 2 \frac{\partial^2 g}{\partial x \partial y} r \sigma_x \sigma_y + \frac{\partial^2 g}{\partial y^2} \sigma_y^2 \right) \quad (2.11)$$

$$\sigma_g^2 = \left(\frac{\partial g}{\partial x} \right)^2 \sigma_x^2 + 2 \left(\frac{\partial g}{\partial x} \right) \left(\frac{\partial g}{\partial y} \right) r \sigma_x \sigma_y + \left(\frac{\partial g}{\partial y} \right)^2 \sigma_y^2 \quad (2.12)$$

where σ_x^2 is the variance $V(x)$ and r is the correlation coefficient between x and y (zero if we assume independence). The right hand side is evaluated at $(E[x], E[y])$. Applying (2.11) and (2.12) to (2.10), yields:

$$E[g] = \frac{E[x]E[y]}{E[x] + E[y]} - \frac{E[y]^2 V(x)}{(E[x] + E[y])^3} - \frac{E[x]^2 V(y)}{(E[x] + E[y])^3} \quad (2.13)$$

$$V(g) = \left[\frac{E[y]}{E[x] + E[y]} \right]^4 V(x) + \left[\frac{E[x]}{E[x] + E[y]} \right]^4 V(y) \quad (2.14)$$

We have found, however, that (2.13) and (2.14) do not give accurate results especially when P_h is rather small. By using conditional probabilities we obtain very accurate results. In the conditional probability approach, the expected value and the variance of the equivalent edge are calculated *given that the equivalent edge g conducts*. With this condition (2.13) and (2.14) become:

$$E[g | x \neq 0, y \neq 0] = \frac{E[x | x \neq 0]E[y | y \neq 0]}{E[x | x \neq 0] + E[y | y \neq 0]} - \frac{E[y | y \neq 0]^2 V(x | x \neq 0)}{(E[x | x \neq 0] + E[y | y \neq 0])^3} - \frac{E[x | x \neq 0]^2 V(y | y \neq 0)}{(E[x | x \neq 0] + E[y | y \neq 0])^3} \quad (2.15)$$

$$V(g | x \neq 0, y \neq 0) = \left[\frac{E[y | y \neq 0]}{E[x | x \neq 0] + E[y | y \neq 0]} \right]^4 V(x | x \neq 0) + \left[\frac{E[x | x \neq 0]}{E[x | x \neq 0] + E[y | y \neq 0]} \right]^4 V(y | y \neq 0) \quad (2.16)$$

where

$$E[x | x \neq 0] = \frac{E[x]}{P_{hx}} \quad (2.17)$$

$$E[x^2 | x \neq 0] = \frac{E[x^2]}{P_{hx}} \quad (2.18)$$

$$V(x | x \neq 0) = E[x^2 | x \neq 0] - E^2[x | x \neq 0] \quad (2.19)$$

$E[g]$ and $V(g)$ are calculated from (2.15) and (2.16), using equations similar to (2.17)-(2.19).

Equations (2.15) and (2.16) have been tested on a large number of CMOS subcircuits with the number of transistors ranging in the subcircuit between 4 and 23, arranged in various series-parallel combinations. The results are compared with those of [1], as well as with exhaustive SPICE-like simulations. For comparison, the equations derived in [1] for series combination are:

$$\frac{1}{E[g]} = \frac{1}{E[x]P_{hy}} + \frac{1}{E[y]P_{hx}} \quad (2.20)$$

$$\frac{V(g)}{E[g]^4} = \frac{V(x)}{E[x]^4} + \frac{V(y)}{E[y]^4} \quad (2.21)$$

It is observed that in calculating the expected value, the proposed method has 3% maximum error compared to exhaustive simulation, while the previous method has 50% maximum error; while in calculating the variance, the new method has a 12% maximum error while the old one 92%.

Comparing our new results with results obtained using the original approach, we concluded that:

- the new series reduction method produces consistently accurate results,
- the error in the reduction process seems to be only slightly dependent on the input probabilities, with higher error occurring when the input probabilities are close to zero or one, a result corroborated by all the results to date, and
- the new method consistently outperforms the previous method.

2.3 Elimination During Switching

The basic circuit model of a CMOS gate is shown in Fig. 2.2. CMOS circuits draw current from the bus only when their outputs change states. By the nature of probabilistic simulation, a probabilistic event at the input always produces a probabilistic event at the output. Therefore, the calculation of a current pulse must be done whenever an input is switching. We first assume that the inputs to a subcircuit are *independent* and switch instantaneously. The effects of signal dependence on the elimination process is considered in Section 2.5 below. Since the signals are assumed to switch instantaneously, we make the following observation:

- Observation: Since the probability that two input signals to a gate switch at the same time is zero, if the edge that switches is in parallel with an edge that is conducting, then switching will have no effect on the output since the edge that is ON effectively shorts the edge that switches; as a result, the switching transistor will not cause the gate output to switch. It follows then that for the output to switch, one of the edges must switch and *all* the conducting paths between the output node and the power node *must* go through the edge that has switched.

Based on the above observation, the calculation of the equivalent conductance at switching time is simplified. Suppose that edge k is switching; then for all edges $i \neq k$ that form conducting paths with edge k , it follows from the independence assumption that:

$$P(i \text{ is ON} | k \text{ is switching}) = P(i \text{ is ON}) = P_{hi} \quad (2.22)$$

Since at switching time all the current must pass through the transistor that is switching, then all paths that bypass that transistor should not conduct, as shown in Fig. 2.3, which leads to

the following graph reduction algorithm:

- i. Assign probability 1 to the edge k that actually switches, before reduction begins.
2. Separate the edges into switching and non-switching edges. A switching edge is defined as the one that actually switches and every equivalent edge that contains it during the process of graph reduction. Otherwise an edge is non-switching.
3. Proceed to graph elimination as described in Section 2.2.
 - (a) Series reduction is performed according to (2.15) and (2.16).
 - (b) In parallel reduction we distinguish between two cases:
 - If two non-switching edges are in parallel then elimination proceeds as usual.
 - If one of the edges, which could be an equivalent edge, is switching then the edge in parallel with it is assigned $P_h = 0$ and the probability that this edge is not conducting is stored. Let α be the set of these parallel edges.
4. Calculate $E[I_k]$ and $V(I_k)$ for the peak current I_k flowing through the switching edge k , which is also the total peak current drawn by the gate, the following expressions have been derived:

$$E[I_k] = V_{dd} \times E[G_{total}] \times P_{lhk} \times \prod_{i \in \alpha} P_{li} \quad (2.23)$$

$$E[I_k^2] = V_{dd}^2 \times E[G_{total}^2] \times P_{lhk} \times \prod_{i \in \alpha} P_{li} \quad (2.24)$$

where G_{total} is the conductance of the equivalent edge between the output node and the power node.

It is evident that, because of independence, at switching time there is only one switching edge in the graph that is being reduced and that the only edge remaining after elimination is a switching edge.

The *time spans* of the current waveforms are computed from the basic gate model shown in Fig. 2.2 following the method proposed in [1], [2]. Let $E[q]$ be the expected value of the charge stored in the output capacitances C_n and C_p :

$$E[q] = V_{dd} C_n P_{lho} + V_{dd} C_p P_{hlo} \quad (2.25)$$

where P_{lho} and P_{hlo} are the probabilities that the output will go from low to high and vice versa. Also

$$E[iq] = \frac{V_{dd} C_n^2}{C_p + C_n} E[i_p] + \frac{V_{dd} C_p^2}{C_p + C_n} E[i_n] \quad (2.26)$$

where $E[iq]$ is the expected value of the charge-current product and $E[i_n]$, $E[i_p]$ are the expected values of the current drawn by the n-part and the p-part respectively. The time spans (τ_E) for the expected value and (τ_V) for the variance of the current waveforms are given by the following relations.

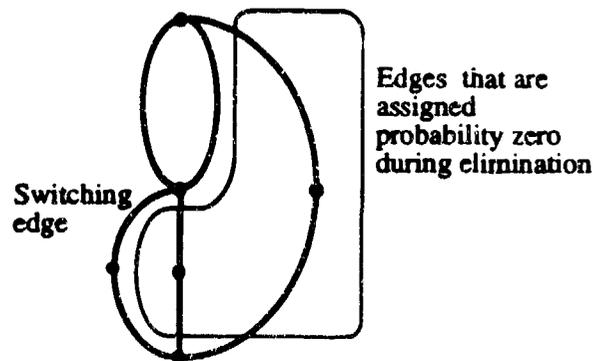


Figure 2.3 Elimination example when one edge switches.

$$\tau_E = 2 \times \frac{E[q]}{E[i]} \quad (2.27)$$

$$\tau_V = \frac{4}{3} \times \left(\frac{E[iq] - E[i]E[q]}{V(i)} \right) \quad (2.28)$$

where $i = i_n + i_p$, $E[i]$ its expected value and $V(i)$ its variance.

Example 1: This example illustrates the accuracy of calculating the expected value and variance current waveforms using the new method. Figure 2.4(a) is a two-input NAND gate, with two inputs A and B. Figures 2.4(b-d) show the expected value of the current waveforms using the new approach for $t_a \cong t_b$ (Fig. 2.4(b)), where t_a and t_b are the transition times for inputs A and B respectively, $t_b - t_a = 0.4ns$ (Fig. 2.4(c)) and $t_b - t_a = 4ns$ (Fig. 2.4(d)), compared each time with exhaustive SPICE runs. The proposed method successfully tracks SPICE results in all cases.

2.4 Delay Calculation

After having calculated the peak and the time span of the current drawn by the gate, we need to calculate the output voltage waveform that drives the fanout gates. This includes the determination of the probability that the output node is high after the transition, the probability that it will transition from low to high, and the time when the output transition will take place (i.e., the delay of the output event).

The stochastic quantities (steady-state and transition probabilities) can be derived from the following expressions:

$$P_{lh,out} = P_{h,p} \times P_{hl,k} \quad (2.29)$$

$$P_{hl,out} = P_{h,n} \times P_{lh,k} \quad (2.30)$$

$$P_{h,out}(t^+) = P_{h,out}(t^-) + P_{lh,out} - P_{hl,out} \quad (2.31)$$

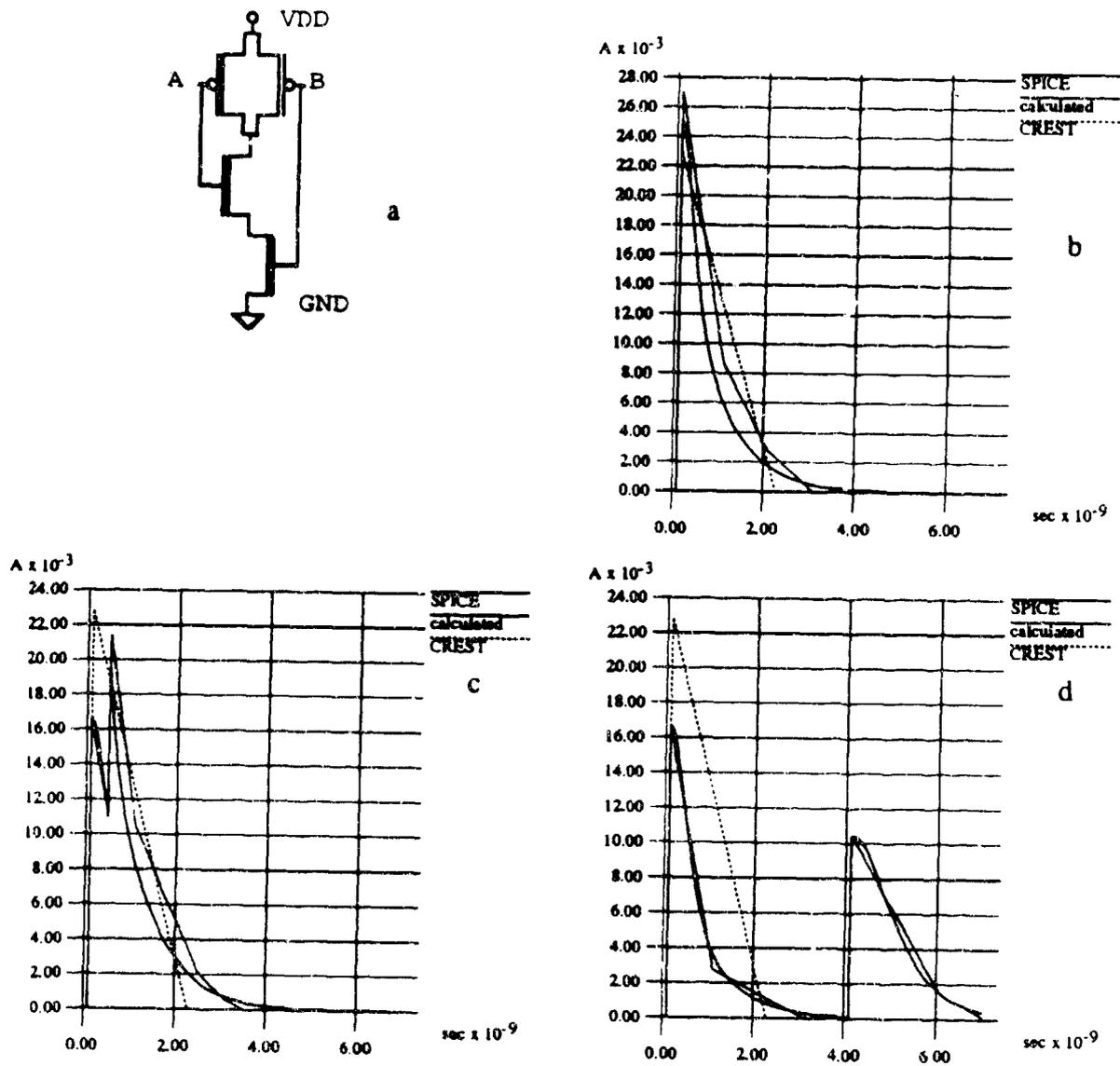


Figure 2.4 Simulation of transitions within a single interval.

where $P_{lh,out}$ is the probability that the output will go from low to high, $P_{hl,out}$ is the probability that the output will go from high to low, $P_{h,out}(t^-)$ is the probability that the output is high before the transition, $P_{h,out}(t^+)$ is the probability that the output is high after the transition, $P_{h,p}$ is the probability that the p-part of the gate conducts (as calculated by the graph

elimination process), $P_{h,n}$ is the probability that the n-part of the gate conducts, $P_{lh,k}$ is the probability that the input node that is causing this transition is going from low to high, and $P_{hl,k}$ the probability that this input node is going from high to low.

For the calculation of the delay of the output event a new method is presented based on the deterministic gate delay model, which applies to both switching from low to high and high to low, namely:

$$t_d = \ln 2 \times \frac{C_p + C_n}{G} \quad (2.32)$$

where t_d is the gate delay. Expression (2.32) is derived from a simple RC circuit driven by a step voltage input. Using simple probability theory, we arrive at the following expression for the delay of each of the transitions.

For the transition from low to high

$$E[t_{dp}] \cong \ln 2 \times (C_p + C_n) \times \left(\frac{1}{E[G_p]} + \frac{V(G_p)}{(E[G_p])^3} \right) \quad (2.33)$$

and for the transition from high to low

$$E[t_{dn}] \cong \ln 2 \times (C_p + C_n) \times \left(\frac{1}{E[G_n]} + \frac{V(G_n)}{(E[G_n])^3} \right) \quad (2.34)$$

where G_n and G_p are, respectively, the equivalent conductances for the n-part and the p-part when the output is switching. For the total average delay, the following expression is used, which is the weighted average of the delays from low to high and from high to low.

$$E[t_d] = \frac{E[t_{dn}] \times P_{out,hl} + E[t_{dp}] \times P_{out,lh}}{P_{out,hl} + P_{out,lh}} \quad (2.35)$$

Example 2: This example illustrates the accuracy of Eq. (2.35) compared to [1] and to exhaustive SPICE simulation. Figure 2.5(b) shows the graph representation of the n-part of the CMOS gate shown in Fig. 2.5(a). All inputs are in steady-state, except input A to which the probabilistic waveform shown in Fig. 2.5(c) is applied. There are two numbers associated with each edge of the graph. The top one is the probability that the gate node of the corresponding transistor is *high* and the bottom one represents its *on* conductance in $m\Omega$. The total load capacitance $C_L = C_p + C_n$ used is $10pF$.

The average delay calculated by Eq. (2.35) is found to be $3.64ns$ compared to $3.60ns$ by exhaustive SPICE simulation, while the method used in [1] yields $0.66ns$, which shows that our result is much closer to SPICE. The improvement in accuracy is the result of the improvement in the probabilistic graph elimination algorithm, the elimination during switching algorithm, and the delay calculation from Eqs. (2.33)-(2.35).

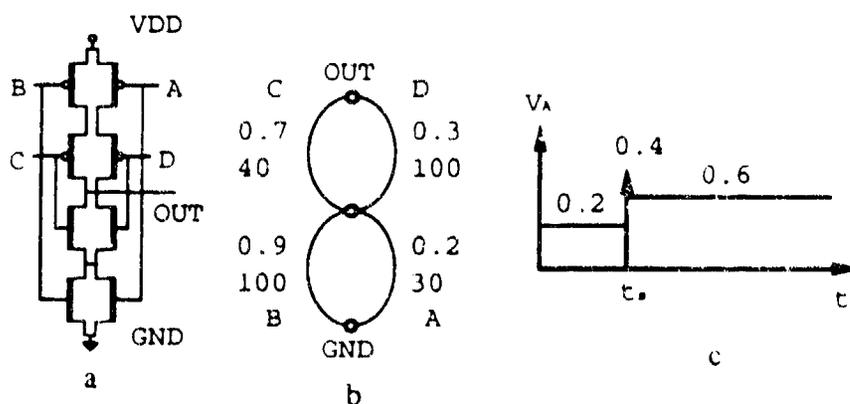


Figure 2.5 (a) A CMOS gate; (b) Graph of the n-part; (c) Input waveform at node A.

2.5 Output Event Merging

So far, in computing the current waveform drawn by a gate, it has been assumed that every event at an input to the gate produces one distinct event at its output. However, when the focus is shifted to the output voltage, depending on the delay of the gate, and the time difference between the input events, two input events may produce:

- i. One output event, as in Fig. 2.4(b);
- ii. one output event with a small glitch that does not affect the operation of any of the fanout gates, as in Fig. 2.4(c); and,
- iii. two distinct events, as in Fig. 2.4(d).

After performing a number of detailed circuit level simulations on typical gate structures, we have determined that if two output events occur within a time interval which is less than the propagation delay, t_{dr} , of the gate, then the two events should be merged into one according to Eqs. (2.36)-(2.39) below. If more than two events occur, merging is done by taking two events at a time, starting with the first two events in the queue.

If *event1* and *event2* are to be merged, then the corresponding probabilities of the equivalent event, which will be referred to as *event* ϵ , are given by:

$$P_{\epsilon h}(t^-) = P_{1h}(t^-) \quad (2.36)$$

$$P_{\epsilon h}(t^+) = P_{2h}(t^+) \quad (2.37)$$

$$P_{\epsilon lh} = P_{1h1}P_{hh2} + P_{1l1}P_{lh2} \quad (2.38)$$

where the events are assumed independent, $P_{hh} = 1 - P_{hl}$ is the probability of signal staying high before and after switching and $P_{ll} = 1 - P_{lh}$ is its probability staying low before and after switching. Since ϵ is the combination of the two events, the new event is scheduled for

time

$$t_e = \frac{(P_{lh1} + P_{hl1}) \times t_1 + (P_{lh2} + P_{hl2}) \times t_2}{P_{lh1} + P_{hl1} + P_{lh2} + P_{hl2}} \quad (2.39)$$

which is the weighted average of the times of occurrence of the two events, a result suggested by simulation results.

The NAND gate example depicted in Fig. 2.6 further illustrates the merging process. In this example we have a two-input NAND gate, which is driven by three events on its inputs, two of which are on the same input node. Assume that events 1 and 3 are closer than the t_{min} defined by the driver gate to the NAND gate. Thus, the two events are merged into the equivalent event 1a. Events 1a and 2 produce two distinct events at the output, 1a' and 2' respectively. But again these two events are closer than t_{min} for the NAND gate and are merged into the equivalent event 4.

Gate output event merging is an integral part of our probabilistic simulation approach. Its aim is to suppress small glitches at the output of the gate and to identify large glitches that

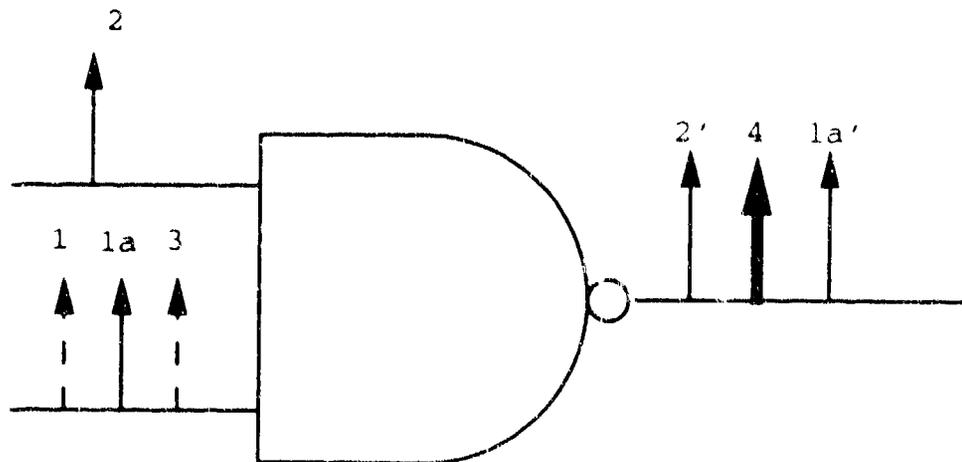


Figure 2.6 Example of merging at the input and output nodes of a gate.

may affect fanout gates. It also dramatically reduces the number of events that need to be simulated, thus reducing the overall simulation time significantly as shown in Table 2.1. In generating the results in Table 2.1, signal correlation has been ignored. The effects of signal correlation on the results are considered in the next section.

	Primary inputs	Time with no merging†	Time with merging†	Memory usage (Kbytes)
c432	37	388.1	1.5	456
c880	61	31.2	2.1	680
c1355	42	7856.2	4.1	804
c1908	34	983.4	7.2	1128
c2670	234	978.0	7.6	1512
c3540	51	59201.9	20.9	2316
c5315	179	4126.5	19.4	2676
c6288	33	>24h	95.6	3796
c7552	208	1809.9	32.2	3620
† In CPU seconds on an HP 9000/730				

2.6 Signal Correlation

So far we have assumed that all inputs to a subcircuit are independent. In this section we consider the effects of signal correlation. Correlation is a measure of the interdependency of two or more random variables. Correlation can be introduced between two or more node variables due to the presence of reconvergent fanout and/or feedback in a circuit, as in Fig. 2.7.

There are two issues that have to be considered. One is the computation of signal correlations between the circuit nodes, and the second is the calculation of the effects of signal correlations at the inputs of a gate on its probabilistic analysis. To deal with the first issue, namely to obtain signal probabilities and correlation coefficients between the nodes, we use

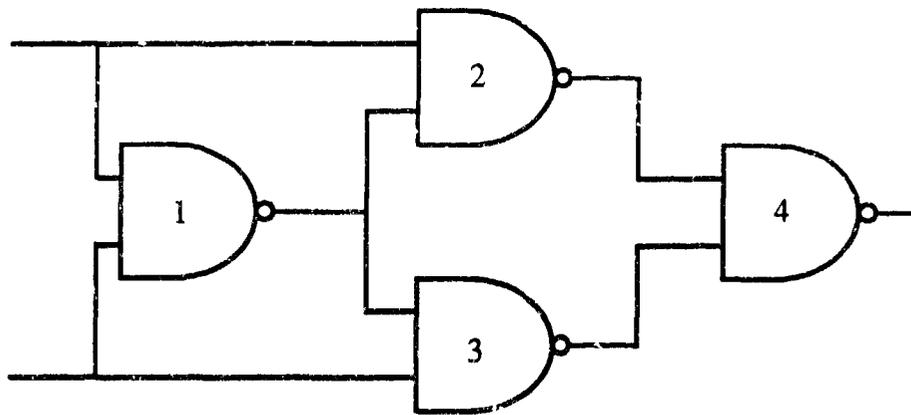


Figure 2.7 A 16-transistor implementation of an XOR gate.

the approach described in [5], with some modifications needed for probabilistic simulation. The approach in [5] estimates the node probabilities based on an estimate of the first order correlations.

The second issue, namely the analysis of a gate, given the signal probabilities and signal correlations at its inputs, is explained in detail in [7].

Example 3: Referring again to the 16-transistor XOR gate shown in Fig. 2.7, this circuit is selected because of the very strong reconvergence observed at the inputs of three of the four total gates comprising this example (namely gates 2, 3, and 4). The total average current as estimated by our approach with and without correlation considerations is shown in Fig. 2.8 along with the results of exhaustive simulation as reference. We observe that the waveform produced without correlation considerations demonstrates high error at time $1.4ns$. This is due to the fact that the inputs to gates 2 and 3 are highly correlated, and, thus, the current drawn by those two gates is overestimated. Again the same waveform shows significant error at time point $1.55ns$ where the current drawn by gate 4 (whose inputs are also correlated) is underestimated. From the same figure we can observe that the implementation with correlation

accurately follows the exhaustive current until time point $1.55ns$. The error that arises there is due to the inability of the correlation coefficient method [5] to accurately calculate the correlation coefficient between the input nodes of gate 4. Another measure of the accuracy of the methods proposed in this paper, which is directly related to the applications of probabilistic simulation (i.e., average power estimation and electromigration degradation) is to calculate the integral of the expected current. For this example the following results are obtained (5V supply voltage):

- Exhaustive: 1.335 mW
- Without correlation: 1.419 mW (6.0 error)
- With correlation: 1.387 mW (3.9 error)

As we can see the errors are small, even though the solution with correlation produces less error.

Example 4: In this example we consider the smallest of the ISCAS85 benchmark circuits, c432 [6]. To be able to perform exhaustive simulation on the circuit within reasonable time all the primary inputs are assigned to logical "1" except two. Figure 2.9 shows the results from exhaustive simulation and from probabilistic simulation with and without correlation considerations. As we can see the estimated waveforms generally follow the exhaustive one and the integrals under the curves are about the same (less than 10% error for both implementations with respect to exhaustive).

However, if we look at the current drawn by an individual gate, within the design, that is close to the output of the circuit whose input signals are correlated, it can be observed that the waveforms are different and the integrals differ by a factor of 2 when correlations are ignored,

$A \times 10^{-3}$

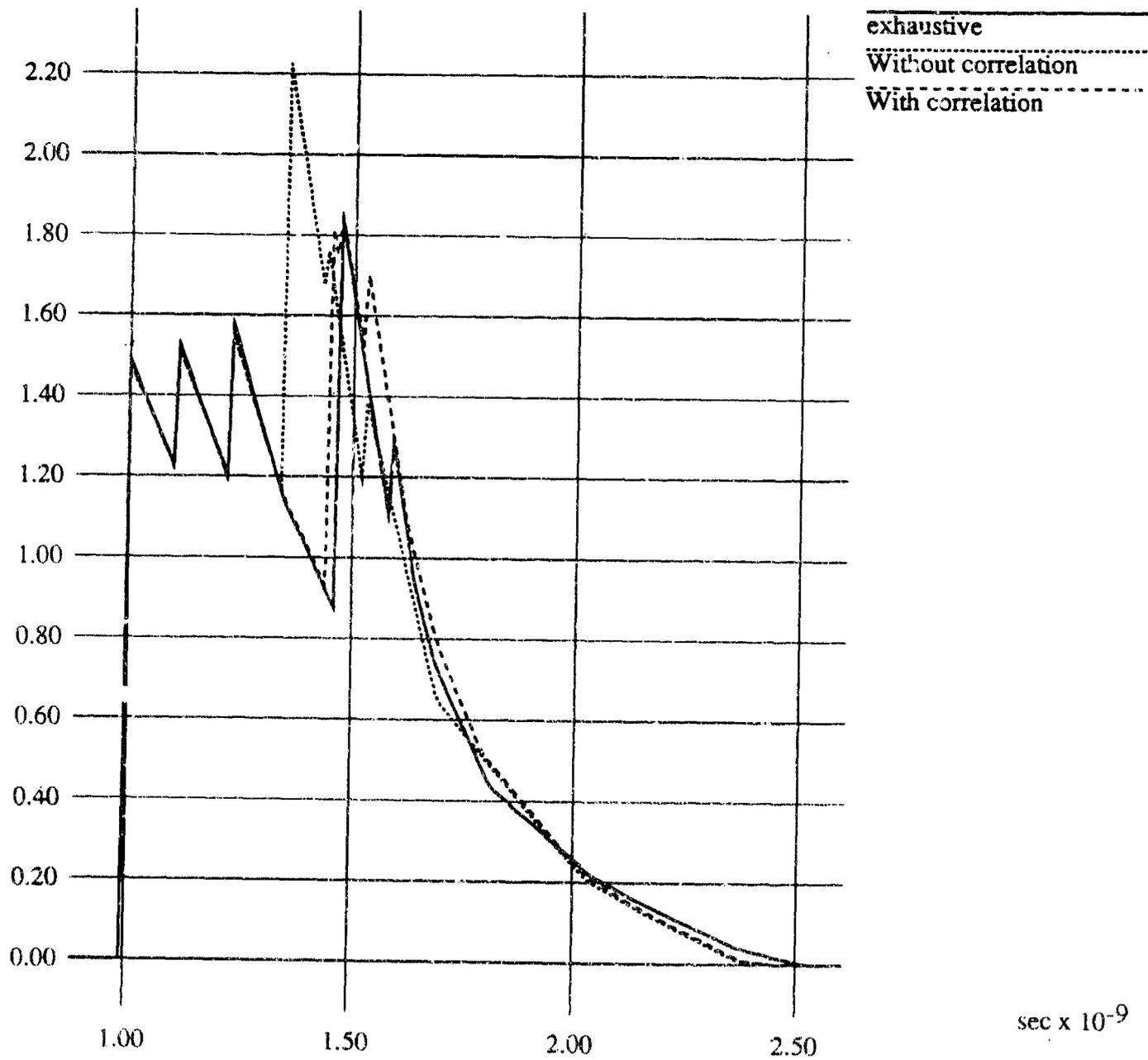


Figure 2.8 Comparison of the results on an XOR gate with and without correlation considerations.

while the error with correlation considerations is only 25%.

When the ISCAS85 benchmark circuit c432 is analyzed with all the primary inputs switching, the two waveforms produced with and without correlation considerations are marginally different, as shown in Fig. 2.9. On all the circuits we tested, the effect of correlation on the total current is small, a result consistent with previous published work [1], while the effect on individual gate currents tends to be significant.

Table 2.2 shows the impact of correlation calculations on the overall speed of the approach for the ISCAS85 benchmark circuits. The inclusion of correlation estimation increases the computation time up to threefold for the examples shown here, and could be even greater for larger examples, but the speed remains quite reasonable, and certainly better than doing an exhaustive analysis. Thus, there is a tradeoff between accuracy and speed.

	Time with no merging†	Time with merging†	Time with correlation†
c432	388.1	1.5	1.6
c880	31.2	2.1	2.8
c1355	7856.2	4.1	5.0
ci908	983.4	7.2	9.0
c2670	978.0	7.6	14.7
c3540	59201.9	20.9	40.3
c5315	4126.5	19.4	46.2
c6288	>24h	95.6	110.0
c7552	1809.9	32.2	140.2
† In CPU seconds on an HP 9000/730			

In our program two methods of calculating the expected current are implemented. The first is described in Sections 2.2 through 2.5 where correlation is ignored. It is fast, with small memory requirements and, as our examples up to now show, quite accurate. It can be used for

$A \times 10^{-3}$

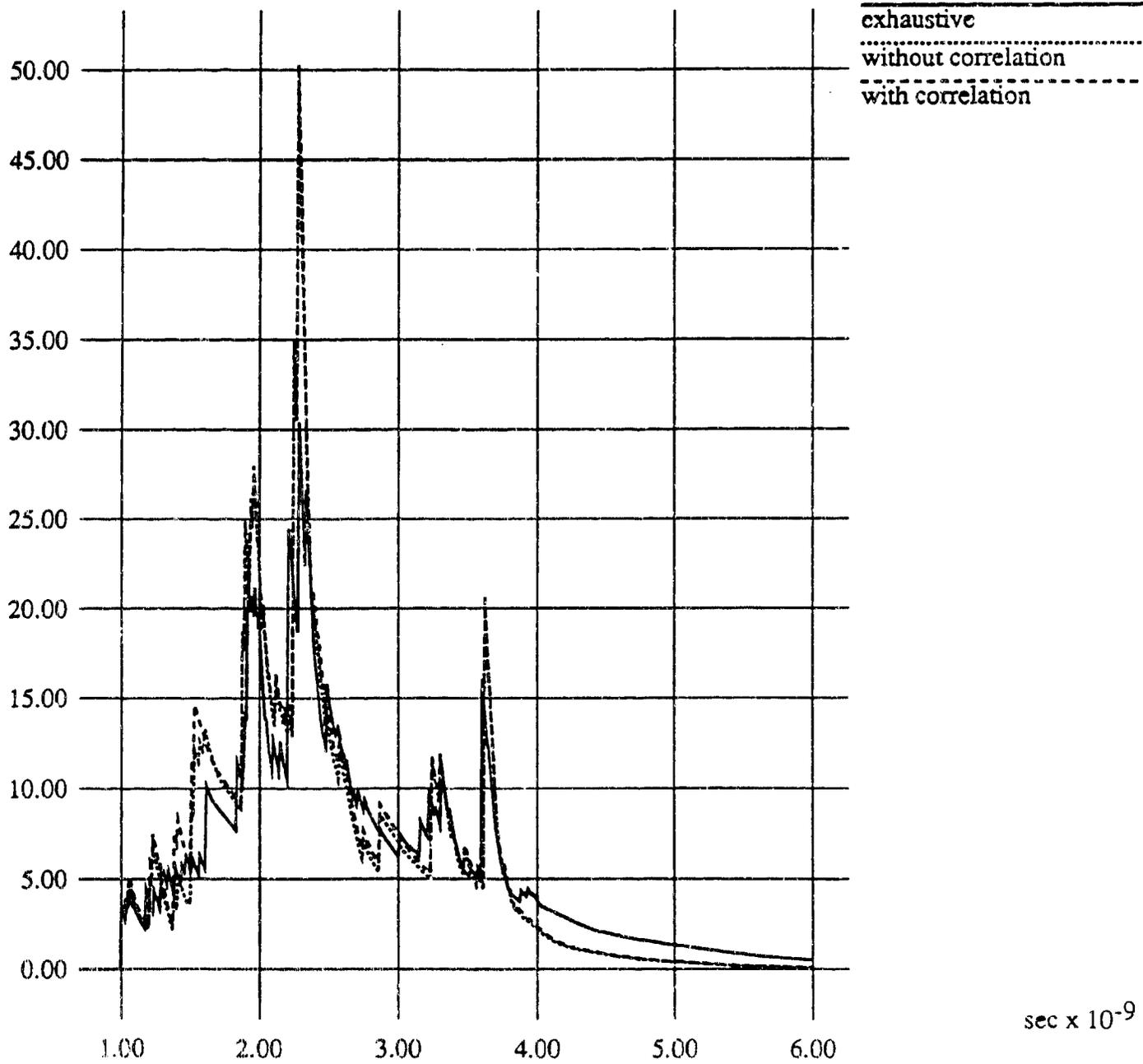


Figure 2.9 Current drawn by the entire c432 circuit.

a quick yet reliable estimate of the expected current waveform of a given circuit. The extension to this method described in Section 2.6 takes into account second order correlation, that is correlation between two nodes at a time. This is by construction more accurate than the previous one, when considering individual gates, since it takes into account correlation, but is slower (up to three times slower in the examples that were presented here) and requires more memory space. However, it does not seem to provide enhanced accuracy when the total current is being calculated. Thus, there is a tradeoff between accuracy and speed that has to be taken into account when using the new approach.

Possible explanations to the minimal impact of correlation on the total current is that the uncorrelated gate inputs in the studied circuits are more than the correlated ones and mask their effect and/or that ignoring correlation introduces random error. This error, when the estimated currents are superimposed (added), tends to zero. But, when the current drawn by each individual gate is estimated, as some applications require, large discrepancies between estimated and actual current could occur. In those applications inclusion of correlation in the computations is imperative.

3. CURRENT ESTIMATION FOR HOT-CARRIER EFFECTS

3.1 Introduction

In Chapter 2, we explained the application of probabilistic simulation to estimate the average and variance current waveforms drawn by the gates at contact points to the power bus. In this chapter, we report on the work we have done on the application of probabilistic simulation for estimating the average current waveforms flowing in the devices within the gates of a given design. These currents are then used to estimate the expected hot-carrier (HCE) induced degradation in each transistor. We then propose some design strategies to reduce such degradation and its impact on long-term circuit performance.

Existing work on HCE mainly focuses on developing degradation models of MOS transistors. These models are then linked to general purpose circuit simulators such as *SPICE* to simulate circuit performance over time [8], [9], [10], [11]. This type of approach is applicable when the circuit under simulation contains only a few transistors, and is used to verify the accuracy of the degradation models. For VLSI circuits with hundreds of thousands of transistors, however, the use of a circuit simulator such as *SPICE*, even with a simple (level one) transistor model, is not feasible, especially since HCE is a cumulative effect, and the combined effects of all possible inputs have to be evaluated.

Recently timing simulation has been considered as a tool for HCE estimation [12], [13]. The approach in [12] employs fast-timing simulation to simulate subcircuits with no damage, and more detailed circuit simulation to estimate HCE on the damaged subcircuits. In [13], a macromodel of channel-connected MOS blocks is constructed and fast timing simulation is used to predict the degradation effects over time. This approach, however, estimates the

degradation of the simplified macromodels instead of each MOS transistor inside the original circuit. In both these approaches deterministic simulation is used, and thus the most damaged subcircuit/transistor is obtained assuming that a user specified input waveform is continuously applied to the circuit. Since HCE is a long term process, the combined effects of a large number input waveforms need to be applied to find out which transistors are expected to accumulate sufficient degradation so as to affect the performance of the entire circuit. To determine the most damaged transistor with respect to all possible input waveforms using conventional deterministic timing simulators, exhaustive simulation or at least a very large number of simulations have to be done.

In this work, instead of performing exhaustive or Monte Carlo simulation, we apply probabilistic simulation techniques to estimate HCE. In this application, we are interested in computing the average current flowing in individual transistors within subcircuits during switching time. Thus more accurate and detailed subcircuit analysis is required. To attain this level of accuracy, the probabilistic voltage waveform representation is extended to include finite transition times during switching, and each subcircuit is analyzed in more detail, as will be explained in Section 3.3. With this additional information we are able to estimate the level of expected degradation individual transistors might experience, relative to other transistors in the circuit, as well as predict the effects of such degradation on the circuit performance. Our aim is not to estimate how long it takes a transistor to reach a certain level of degradation, but rather to suggest design changes so as to reduce circuit degradation that may be caused by HCE.

In Section 3.2, we describe the measures we use to estimate for hot-carrier effects. In Section 3.3, we briefly review the extension of the input probability waveform to include

transition times, and explain the analysis of a simple gate. Voltage signal overlap handling is given in Section 3.4. Implementation and simulation results are presented in Section 3.5. In Section 3.6, we propose a redesign technique derived as a result of the simulation, for reducing HCE in a circuit and thus improving the long term design reliability without increasing the design area.

3.2 Hot-Carrier Effects

Hot-carrier effects in MOS transistors are caused by the injection of high-energy carriers into the gate oxide region of the transistors. These carriers manifest themselves in the form of oxide charge trapping and interface trap generation which accumulate over time, depending on the operation of the transistor within a circuit, and cause a permanent change in the oxide and interface charge distribution [14], [15]. This change in the charge distribution causes degradation in the transistor characteristics, such as shift in the threshold voltage V_T , and decrease in the transconductance g_m and electron mobility μ_n in the channel [14]. This degradation, in turn, affects the performance of the circuit in which the transistor is embedded, such as increase in the circuit delay.

With current semiconductor process technologies, with effective channel length longer than 0.5 micron, hot-carrier-induced degradation is more severe in nMOS transistors than in pMOS [16]. Therefore, in this study we focus our attention on n-channel transistors. Our approach, however, can be applied as well to estimate HCE in pMOS transistors by using the appropriate pMOS degradation model. It has been shown that HCE in nMOS transistors in digital circuits is dominated by interface trap generation which occurs mostly when the transistor is operating in or near the saturation region [14], [17], [18], and thus the extent of

the damage depends on the operating conditions of the transistor within a circuit design. It has also been shown that the damage can be measured using the interface trap generation model [19]:

$$D_n = \frac{1}{WH} \int I_{DS} \left(\frac{I_{sub}}{I_{DS}} \right)^m dt \quad (3.1)$$

where W is the transistor width and H is a technology dependent parameter; m is approximately 3. Thus, to estimate HCE induced damage, one has to compute the drain-to-source current and substrate current waveforms, $I_{DS}(t)$ and $I_{sub}(t)$, for the lifetime duration of the transistor under all possible inputs, with the transistor model degraded in time. In our approach, however, we are interested in identifying those transistors that are most susceptible to HCE induced damage for redesign purposes, rather than estimating the lifetime of the design. We will consider the average value of D_n in (3.1), where the average is taken *over all possible inputs* to the circuit, as a measure of relative transistor damage. Alternatively, we could use the average integral of $I_{sub}(t)$ as a measure of damage. As we will show below both of these measures agree as far as detecting relative damage. We compute these averages using the undamaged transistor model.

It has been established that $I_{sub}(t)$ flows when the transistor is operating in the saturation region and is a function of the drain-to-source current, I_{DS} , drain-to-source voltage, V_{DS} , with exponential dependency on $(V_{DS} - V_{DSAT})$, and process parameters, including the channel length [14]. By performing exhaustive and detailed circuit simulation of typical complementary CMOS gates, we have identified a number of factors that affect the time interval a transistor stays in saturation during switching and consequently the $I_{sub}(t)$ current waveform in the transistor. These factors are: (1) the type of gate in which the transistor is embedded;

(2) the frequency of transistor switching that causes the gate output to switch; (3) the input slew rate and the gate output loading; and (4) the position of the transistor within the gate. We will now justify each of these factors in more detail.

1. Since I_{sub} is exponentially proportional to $(V_{DS} - V_{DSAT})$, if the transistor is prevented from entering saturation region, or at least the voltage across the channel is reduced, the substrate current can be reduced dramatically. As a consequence, the nMOS transistors in a NAND gate, for example, suffer less than those in a NOR gate configuration because the serial connection of nMOS transistors reduces V_{DS} of each nMOS, and hence hot-carrier generation is reduced.
2. Since HCE occurs when the transistor is in the saturation region, which in CMOS circuits happens only during gate output transition, it follows that the higher the switching frequency of a transistor in a gate that causes the gate output to switch, the more damage the transistor will experience over a period of time.
3. The input slew rate and the gate output capacitance also affect the duration in which a transistor stays in saturation; slower slew rate causes wider I_{sub} time-span, and hence more degradation. Similarly, larger output capacitance increases output transition time and thus the duration the transistor stays in saturation. Figure 3.1 shows the relationship between HCE induced damage (D_n and integral of I_{sub}) and the input slew rate of a CMOS inverter. Note that if the input is considered to be a step function, while in actuality it has a slew rate of 2 ns, the HCE induced damage D_n could be underestimated by twenty times! Figure 3.1 also shows the relationship between output capacitance and relative damage. We note here that the dependence of HCE on input slew rate and out-

put capacitance has also been reported in [20]. Note that both D_n and $\int I_{sub}$ show the same trend. They are both sublinear functions of the slew rate. This fact justifies using average slew rate valued in the probabilistic simulation method described in Section 3.3.

4. The position of the transistor within the gate also affects its susceptibility to HCE. Consider, for example, the NAND gate shown in Fig. 3.2. Since hot-carrier induced degradation in the nMOS transistors occurs mainly only when V_{out} switches from high to low, as will be shown below, three cases may occur: (i) M1 and M2 turn on simultaneously, (ii) M2 turns on last, and (iii) M1 turns on last. Table 3.1 shows D_n and $\int I_{sub}$ of M1 and M2, with different capacitance values for C_1 , after a simulated three months continuous operation. It is clear from the table that in cases (i) and (ii), either M1 or M2 experiences light degradation, while the other transistor experiences minimal degradation. This degradation may not be severe enough to cause circuit malfunction. On the other hand, in case (iii) M1, which is connected to the output node and switches last and thus is the one that causes the output to switch from high to low, suffers extensive degradation many order of magnitude compared to M2, and compared to both M1 and M2 in cases 1 and 2. These results can be explained as follows. In case (i) both M1 and M2 are switching at the same time, therefore, the drain-to-source voltage V_{DS} of each transistors is about half of V_{out} . As mentioned above, smaller V_{DS} due to serial transistor connection results in light damage to both M1 and M2. In cases (ii) and (iii) either transistor causes an output high-to-low transition, the initial value of V_{DS} across M1 is V_{DD} while that across M2 is $V_{DD} - V_T$. This weak high state of M2 is the main factor protecting it from severe degradation. As a result, M1 stays in the saturation region longer, has higher peak substrate current, and hence experiences more degradation. This is also true when

more than two transistors are in series; a transistor connected to the gate output stays in saturation during switching for a longer period of time compared to when it is not connected to the output node. Thus, we conclude *the top nMOS transistors that are directly connected to the output node have the potential of experiencing severe damage*. This observation has also been reported in [21].

Figure 3.3(a) illustrates the relation between V_{in} , V_{out} , I_{DS} , and I_{sub} of the nMOS transistor of a typical CMOS inverter obtained using *SPICE*. We have found that, by

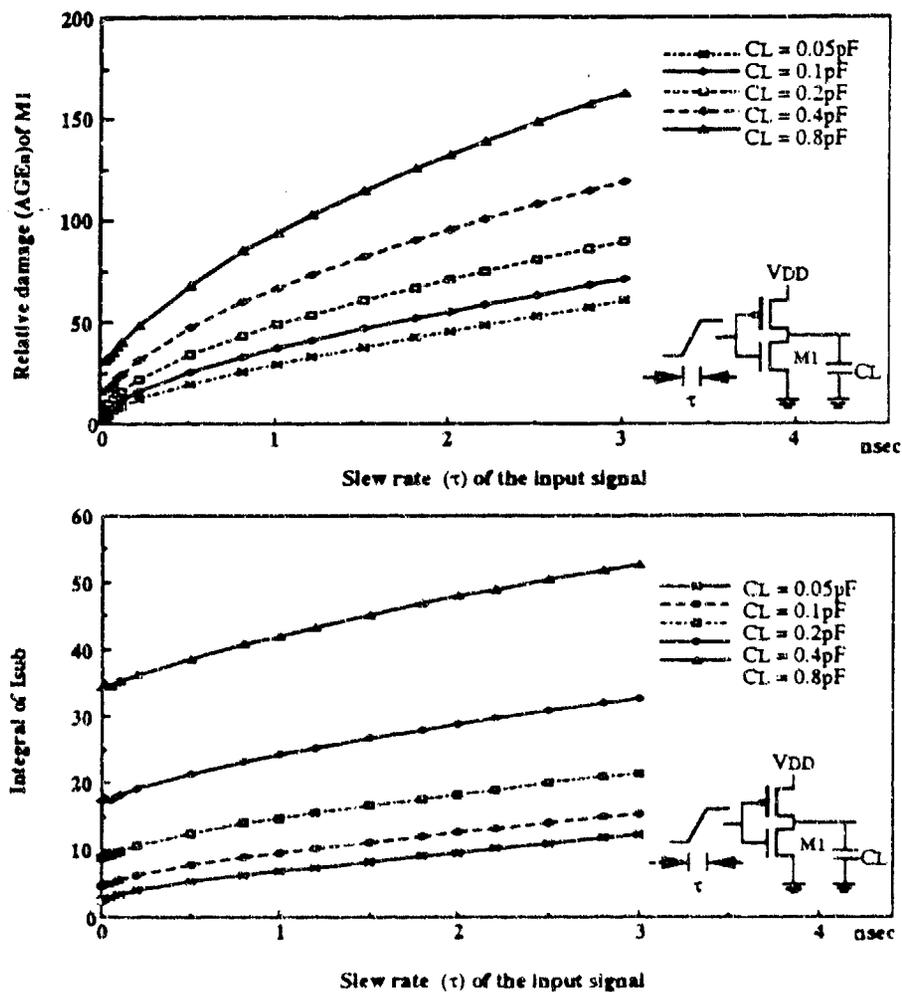


Figure 3.1 The relation between the HCE damage in the nMOS transistor of an inverter and the input slew rate.

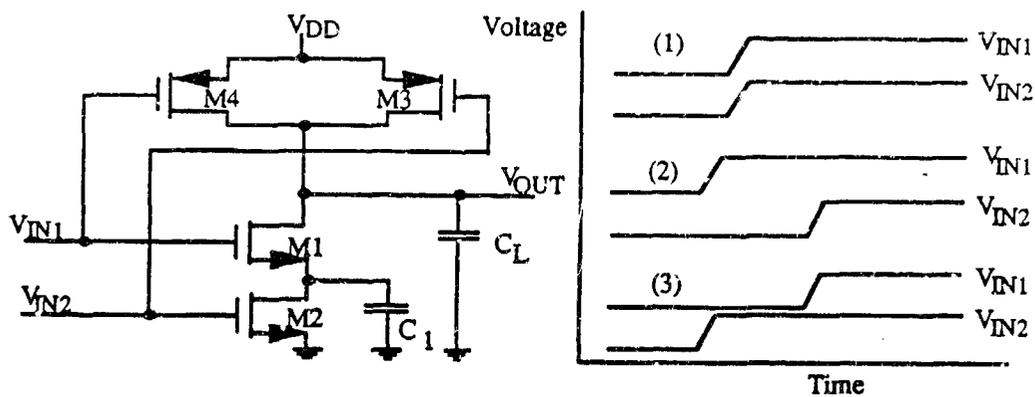


Figure 3.2 A CMOS NAND gate circuit and three input combinations leading to hot-carrier induced damage.

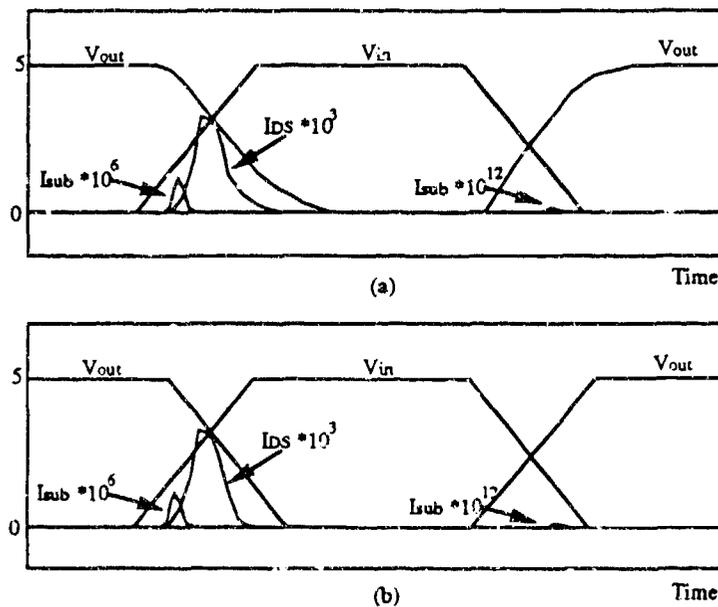


Figure 3.3 (a) The waveforms of V_{in} , V_{out} , I_{DS} , I_{sub} of the nMOS transistor of a CMOS inverter obtained using SPICE.

(b) The waveforms after replacing V_{out} with a ramp. The ramp is obtained by connecting the two points ($t_1, V_{out}(t_1) = 4$) and ($t_2, V_{out}(t_2) = 1$).

approximating V_{out} in Fig. 3.3(a) with a ramp, as drawn in Fig. 3.3(b), the error of I_{sub} peak value is within 2%. Compared with possible modeling errors, this result is accurate enough especially that our goal is to obtain the relative degradation levels of the transistors in CMOS circuits rather than the absolute degradation of each transistor. We will thus use ramp

approximation to the voltage waveforms during transition. Note also from Fig. 3.3 that HCE degradation in the nMOS transistor during output low-to-high transition is negligible, and hence we estimate HCE degradation only during output high-to-low transitions.

3.3 Gate Probabilistic Simulation

3.3.1 Probabilistic Waveform Description

As described in [1], in probabilistic simulation, a signal waveform is specified in terms of its probability of being high during a time interval and the probability of its switching from low to high at specified time points. In this section we extend this description by including the signal transition times during switching. These transition times are needed to estimate HCE more accurately, as shown in the previous section. Figure 3.4 shows a typical probabilistic waveform description. For an event occurring at time t_0 , five values are specified: $Ph(t_0^-)$, $Ph(t_0^+)$, $Plh(t_0)$, $t_{LH}(t_0)$ and $t_{HL}(t_0)$; where $Ph(t_0^-)$ is the probability of the signal

Normalized D_n	transistor	case (i)	case (ii)	case (iii)
$C_L = 0.2\text{pf}$	M1	1.28e-3	8.95e-8	0.84
$C_1 = 0.02\text{pf}$	M2	1.30e-12	3.28e-5	1.30e-12
$C_L = 0.2\text{pf}$	M1	3.38e-8	1.40e-10	1.223
$C_1 = 0.2\text{pf}$	M2	1.47e-12	6.29e-6	1.47e-12
$C_L = 0.2\text{pf}$	M1	9.99e-11	1.48e-12	1.42
$C_1 = 0.5\text{pf}$	M2	1.50e-12	4.84e-7	1.50e-12
$\int I_{sub}$	transistor	case (i)	case (ii)	case (iii)
$C_L = 0.2\text{pf}$	M1	2.104e-2	1.412e-03	1.291e-01
$C_1 = 0.02\text{pf}$	M2	1.260e-06	1.276e-03	1.260e-06
$C_L = 0.2\text{pf}$	M1	1.011e-03	1.758e-04	1.977e-01
$C_1 = 0.2\text{pf}$	M2	2.848e-06	1.985e-03	2.848e-06
$C_L = 0.2\text{pf}$	M1	1.192e-04	1.123e-05	2.420e-01
$C_1 = 0.5\text{pf}$	M2	3.784e-06	1.135e-03	3.781e-06

Table 3.1 The damage of nMOS transistors in an NAND gate with various internal capacitances.

being high right before t_0 , $Ph(t_0^+)$ the probability of the signal being high after $t_0 + t_{LH}(t_0)$, $Plh(t_0)$ the probability of the signal switching from low to high, $t_{LH}(t_0)$ and $t_{HL}(t_0)$ the average low-to-high and high-to-low transition times of the signal. If there is a complete transition, then the voltage waveform (a ramp) can be reconstructed as

$$V(t) = \begin{cases} V_{DD} * \left[\frac{t-t_0}{t_{LH}(t_0)} \right] & \text{if } V(t_0^-) = 0, t_0 \leq t \leq t_0 + t_{LH}(t_0) \\ V_{DD} * \left[1 - \frac{t-t_0}{t_{HL}(t_0)} \right] & \text{if } V(t_0^-) = V_{DD}, t_0 \leq t \leq t_0 + t_{HL}(t_0) \end{cases} \quad (3.2)$$

Note that $Phl(t_0)$, the probability of switching from high to low, can be derived from [1],

$$Phl(t_0) = Ph(t_0^-) + Plh(t_0) - Ph(t_0^+) \quad (3.3)$$

Also note that t_0 is an average reference time point and not the actual time when a transistor switches. The timing simulation algorithm described below computes the average transistor switching time relative to t_0 based on its V_T value.

3.3.2 Subcircuit Probabilistic Simulation Including Transition Times

We consider for the moment static CMOS circuits that consist of serial-parallel configurations with no pass transistors. We define a gate to be a channel-connected subcircuit. The simulation of a single gate consists of two parts: One is steady-state analysis, which

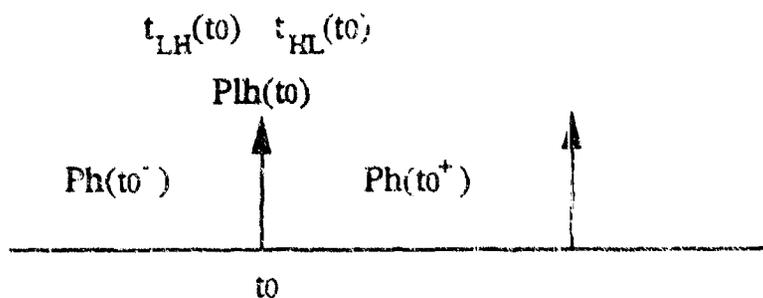


Figure 3.4 A typical input waveform for probabilistic timing simulation.

involves the computation of $Ph(t)$ at the gate output node in the time intervals before it starts and after it completes switching. The other is switching analysis which involves the computation of the average delay Δt , $Plh(t_1)$, $Phl(t_1)$, $t_{LH}(t_1)$, $t_{HL}(t_1)$, when a subcircuit output switches, where $t_1 = t_0 + \Delta t$.

Calculating $Ph(t)$ of the output node is equivalent to finding the probability of having a conducting path from the output node to V_{DD} ; i.e.

$$Ph(t)_{out} = P_{on}(t, e_1 = (out, V_{DD})) = 1 - P_{on}(t, e_2 = (out, G_{ND})) \quad (3.4)$$

Where $P_{on}(t, e)$ represents the probability of having a conducting path through equivalent edge e . If the edge represents a transistor, then $P_{on}(t, e) = Ph(t)$ (or $Pl(t)$) of the gate node of the nMOS (pMOS) transistor. For serial-parallel CMOS circuits, the graph reduction procedure consists of serial and parallel combinations of edges as described in [1].

In switching analysis we follow the approach described in Chapter 2, where signals arriving at the inputs of a gate are considered to switch at separate time instances, even if they

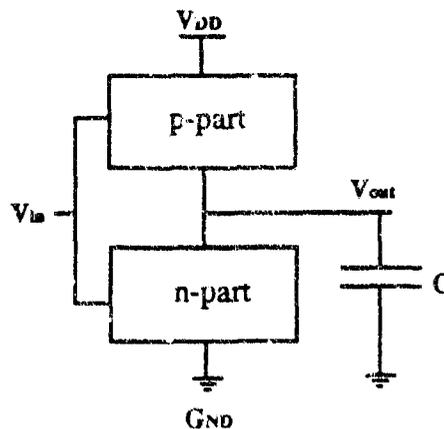


Figure 3.5 A fully complementary CMOS circuit.

appear to arrive closely together. Events arriving at a gate are propagated individually and merged at the output if they occur there within a certain time interval; and when an nMOS transistor M_{ni} in a CMOS gate switches and its switching causes a transition at the gate's output, then at least one conducting path is formed between the output and ground, and all transistors that form parallel paths with M_{ni} must be off. The same observation holds for the pMOS transistors with ground replaced by V_{DD} .

Consider the nMOS part of a CMOS logic gate shown in Fig. 3.5, where the capacitance C is lumped at the output node. Depending on the position of the switching transistor, the nMOS part is reduced to one of the three macromodels as shown in Fig. 3.6. In Fig. 3.6(a) the switching transistor is connected to the output node; in Fig. 3.6(b) it is connected to ground; and in Fig. 3.6(c) it is connected to other transistors and not to V_{out} or ground. All of the nMOS transistors, other than the switching transistor, are combined into equivalent transistors. The transconductance parameters β_1 and β_2 of the equivalent transistors are random variables, with expected values derived using series/parallel probabilistic reduction techniques given in Chapter 2. The same method is applied to pMOS part of the logic circuit. Finally,

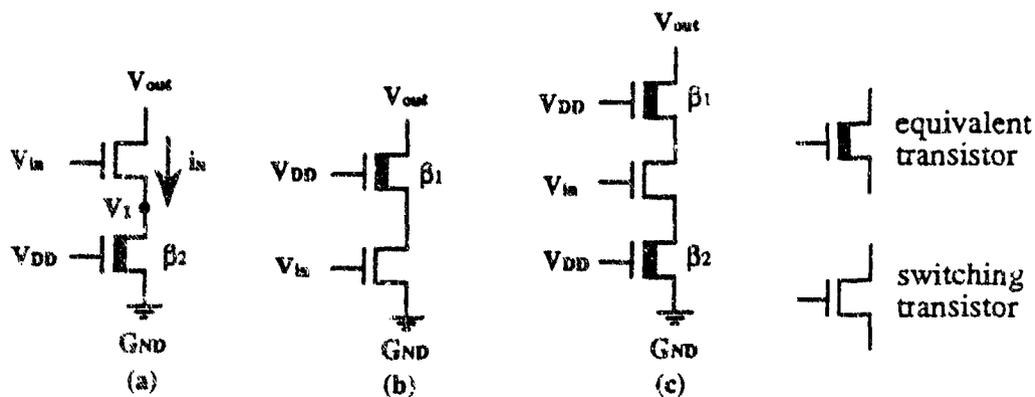


Figure 3.6 Three macromodels for the nMOS part of a CMOS logic circuit, depending on the position of the switching transistor.

the graph reduction is continued for both the n- and p-parts to obtain an equivalent inverter macromodel shown in Fig. 3.7.

3.3.3 Propagation Delay and Output Slew Rate

The approach we follow is to first calculate the output voltage waveform and the current i_n using the equivalent circuit obtained in Fig. 3.7. The output waveform is then applied to the appropriate equivalent circuit shown in Fig. 3.6(a), (b), or (c), to estimate the substrate current waveform in the switching transistors. The same method can be applied to the pMOS part as well, if necessary. The equation describing the equivalent inverter circuit in Fig. 3.7 can be written in the form:

$$\frac{dV_{out}}{dt} = \frac{i_p - i_n}{C} = f(V_{out}, t) \quad (3.5)$$

Equation (3.5) is solved with Backward Euler integration to obtain the output waveform V_{out} and the current waveforms i_p and i_n . Other faster, but less accurate methods such as [13], [22], or [23] can be applied as well. During the simulation, two voltage-time points on the output waveform, $(V_{DD} - |V_{T_p}|, t_a)$ and (V_{T_n}, t_b) , are selected in order to calculate the delay and output slew rate.

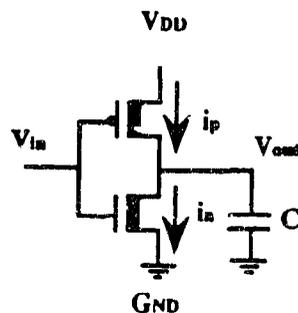


Figure 3.7 The equivalent inverter circuit of the CMOS gate in Fig. 3.5.

Based on our ramp waveform definition, for output high-to-low transition,

$$t_{HL} = (t_b - t_a) * \frac{V_{DD}}{V_{DD} - V_{T_n} - |V_{T_p}|} \quad (3.6)$$

$$\Delta t_{HL} = [t_a - t_{HL} * \frac{|V_{T_p}|}{V_{DD}}] - t_0 \quad (3.7)$$

Similar equations are applied to output low-to-high transition. In general the propagation delay of low-to-high and high-to-low transitions are different. In order to simplify the problem, we take the weighted average of the up and down transition propagation delays using their "probability to occur" as weights [1]; i.e.,

$$\Delta t_{average} = \frac{\Delta t_{lh} P lh(t)_{out} + \Delta t_{hl} P hl(t)_{out}}{P lh(t)_{out} + P hl(t)_{out}} \quad (3.8)$$

In this way, the up and down transitions of each event are always bound together after the signal is propagated through the circuit. This binding, however, is not necessary for the approach. The cost is added storage and computation. The output waveform (defined by $Ph(t_1^-)$, $Ph(t_1^+)$, $Plh(t_1)$, $t_{LH}(t_1)$, $t_{HL}(t_1)$, where $t_1 = t_0 + \Delta t_{average}$ and $t_1' = t_1 + \max(t_{HL}(t_1), t_{LH}(t_1))$) now becomes an input to the fanout subcircuits.

3.3.4 Substrate Current Estimation

After the computation of the output node voltage waveform and the current waveform through the equivalent inverter circuit, at each time point I_{sub} and D_n in the switching transistor are computed using one of the macromodels in Fig. 3.6. For example, in Fig. 3.6(a), applying KCL at the internal node 1, the following equation applies:

$$i_n(V_{out}, V_{in}, V_1) = i_n(V_1, V_{DD}, G_{ND}) \quad (3.9)$$

In this equation, the values of i_n , V_{out} , and V_{ir} are known at each time point. The only unknown variable is V_1 . Note also that the bottom equivalent transistor in Fig. 3.6(a) never enters into the saturation region. The procedure of estimating I_{sub} and D_n in Fig. 3.6(a) is listed as follows:

- (i) At each time step, Eq. (3.5) is solved with Backward Euler integration to obtain the values of V_{out} and i_n .
- (ii) The value of i_n is used to find the voltage V_1 across the equivalent transistor β_2 , with the linear region current equation of the equivalent transistor.
- (iii) The voltage $V_{DS_i} = V_{out} - V_1$ across the switching transistor is used to check if the transistor is in saturation region. If so, V_{DS_i} and $i_{DS_i} = i_n$ are applied to estimate the substrate current of the switching transistor at the particular time point. The trapezoidal method is employed to approximate D_n and the integral of I_{sub} .
- (iv) Repeat (i) to (iii) until the switching transistor no longer stays in saturation region.

Similar procedures are applied to the macromodels in Fig. 3.6(b) and (c). The output voltage waveform V_{out} and current waveform i_n are used to determine the voltage across the switching transistor, check its operating condition, and calculate its substrate current waveform as well as D_n . It follows that the average damage the switching transistor experiences with respect to one incoming event is

$$\left[\int I_{sub} \right]_{ave} = Phl \int I_{sub} dt \quad \text{or} \quad (D_n)_{ave} = Phl \frac{1}{WH} \int I_{DS} \left(\frac{I_{sub}}{I_{DS}} \right)^m dt \quad (3.10)$$

Note that, as mentioned in previous section, the HCE degradation during low-to-high transitions can be ignored. Also note that during the operation of the circuit one transistor

may be driven by several incoming events. Therefore, the average total damage the switching transistor experiences during the entire period of operation becomes

$$\left[\int I_{sub} \right]_{total} = \sum P_{hl} \int I_{sub} dt \quad \text{or} \quad (D_n)_{total} = \sum P_{hl} \frac{1}{WH} \int I_{DS} \left(\frac{I_{sub}}{I_{DS}} \right)^m dt \quad (3.11)$$

3.4 Signal Overlap Handling

When voltage waveform slew rate is included in the waveform specification, overlap during transition could occur and needs to be taken into account. Figure 3.8 shows the relation between two consecutive events at a gate output caused by two different switching transistors. In Fig. 3.8(a), the two events do not overlap, so they can be handled independently. In Fig. 3.8(b), the two events are close to each other, and thus there is certain possibility that one transition happens before the other transition is completed. As shown in this figure, the high-to-low (low-to-high) transition of the second event may occur before the low-to-high (high-to-low) transition of the first event reaches steady state. Assuming these two events at the output node are independent, the corresponding probability of having overlap is as follows:

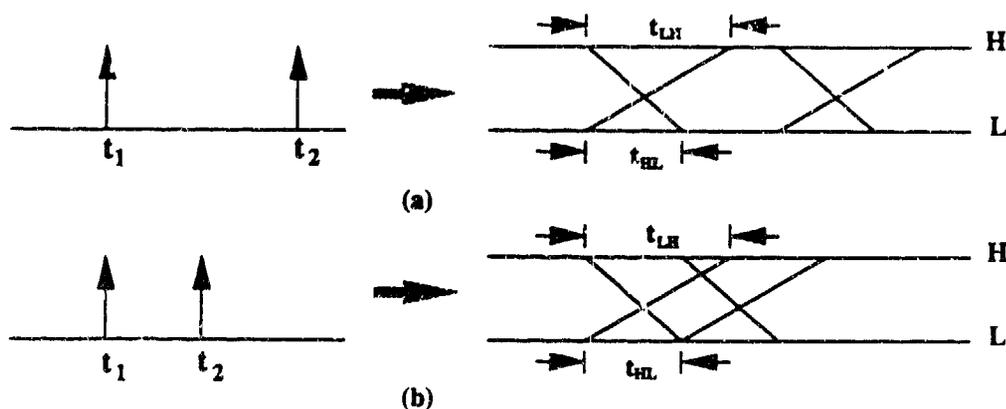


Figure 3.8 (a) No overlap between two events; (b) two events overlap each other.

$$Prob(\Lambda\text{-type overlap}) = \begin{cases} 0 & \text{If } t_2 \geq t_1 + t_{LH} \\ Plh(t_1)_{event1} * Plh(t_2)_{event2} / Ph(t_2^-)_{event2} & \text{if } t_2 < t_1 + t_{LH} \end{cases} \quad (3.12)$$

$$Prob(V\text{-type overlap}) = \begin{cases} 0 & \text{If } t_2 \geq t_1 + t_{HL} \\ Plh(t_1)_{event1} * Plh(t_2)_{event2} / Pl(t_2^-)_{event2} & \text{if } t_2 < t_1 + t_{HL} \end{cases} \quad (3.13)$$

However, the two output events are in general dependent. Depending on the relative position of the switching transistors, only one of the two overlap types would occur. If the switching transistors are in series in n-part of a CMOS gate (e.g., a NAND gate), then *V-type* overlap would occur with probability given in (3.13), but *Λ-type* overlap will not occur; if the switching transistors are in series in p-part of a CMOS gate (e.g., a NOR gate), then *Λ-type* overlap would occur with probability given in (3.12), but *V-type* overlap will not.

The reason is stated as follows: Consider the case when the switching transistors are in series in n-part of a CMOS gate. In order to have a *Λ-type* overlap, the first switching transistor must switch *on* → *off* (the output node switches from low-to-high), and the second switching transistor needs to switch *off* → *on*. However, when the first switching transistor is causing an output transition, the second switching transistor needs to be conducting since it is in series with the first one. Therefore, the second switching transistor could only switch *on* → *off* after the first event is propagated through, which is contradictory to the condition of having a *Λ-type* overlap, and hence *Λ-type* overlap could not occur when the switching transistors are in series in n-part of a CMOS gate. Similar reasoning can be applied when the switching transistors are in series in p-part of a CMOS gate.

We now use an example to explain overlap in more detail. Consider the NAND gate shown in Fig. 3.9(a). All possible waveforms incident at input nodes A and B are displayed within the shaded areas in Fig. 3.9(a). Four waveforms with equal probabilities are possible at each A and B. The expected values of the switching times at A and B are t_a and t_b , respectively. The corresponding probability waveforms are shown directly above the shaded areas. The slew rate information has been omitted from the probability waveform figures for clarity. Figure 3.9(b) shows within the shaded area all the possible output waveforms and their frequency of occurrence out of the sixteen possible scenarios obtained using exhaustive simulation. Output switching time t_1 is equal to t_a plus the expected value of the gate delay when A is switching, and t_2 is equal to t_b plus the expected value of the gate delay when B is switching. In general, these delay values could be different, depending if A or B switches.

The results show that the output signal at C will switch at t_1 twice from low to high and once from high to low. It will switch at t_2 once from low to high and twice from high to low. In one case, it will switch from high to low at t_1 and back to high at time t_2 . In eight cases out of sixteen, it will stay high and in one case it will stay low. The probabilistic waveform above the shaded area in Fig. 3.9(b), which is obtained directly by performing probabilistic simulation on the gate, matches the statistics obtained from exhaustive simulation. Figure 3.9(c) show the results when t_b is closer to t_a so that $t_2 - t_1$ is less than the gate delay. In this case, an incomplete glitch occurs at C, as shown in waveform number 5 in Fig. 3.9(b). Incomplete glitches, or overlapped output waveforms are detected in probabilistic simulation by monitoring $t_2 - t_1$, as well as the slew rate at t_1 . If $t_2 - t_1$ is less than the time it takes a waveform starting at t_1 to reach steady-state, then an incomplete glitch might occur. This case is shown in Fig. 3.9(c).

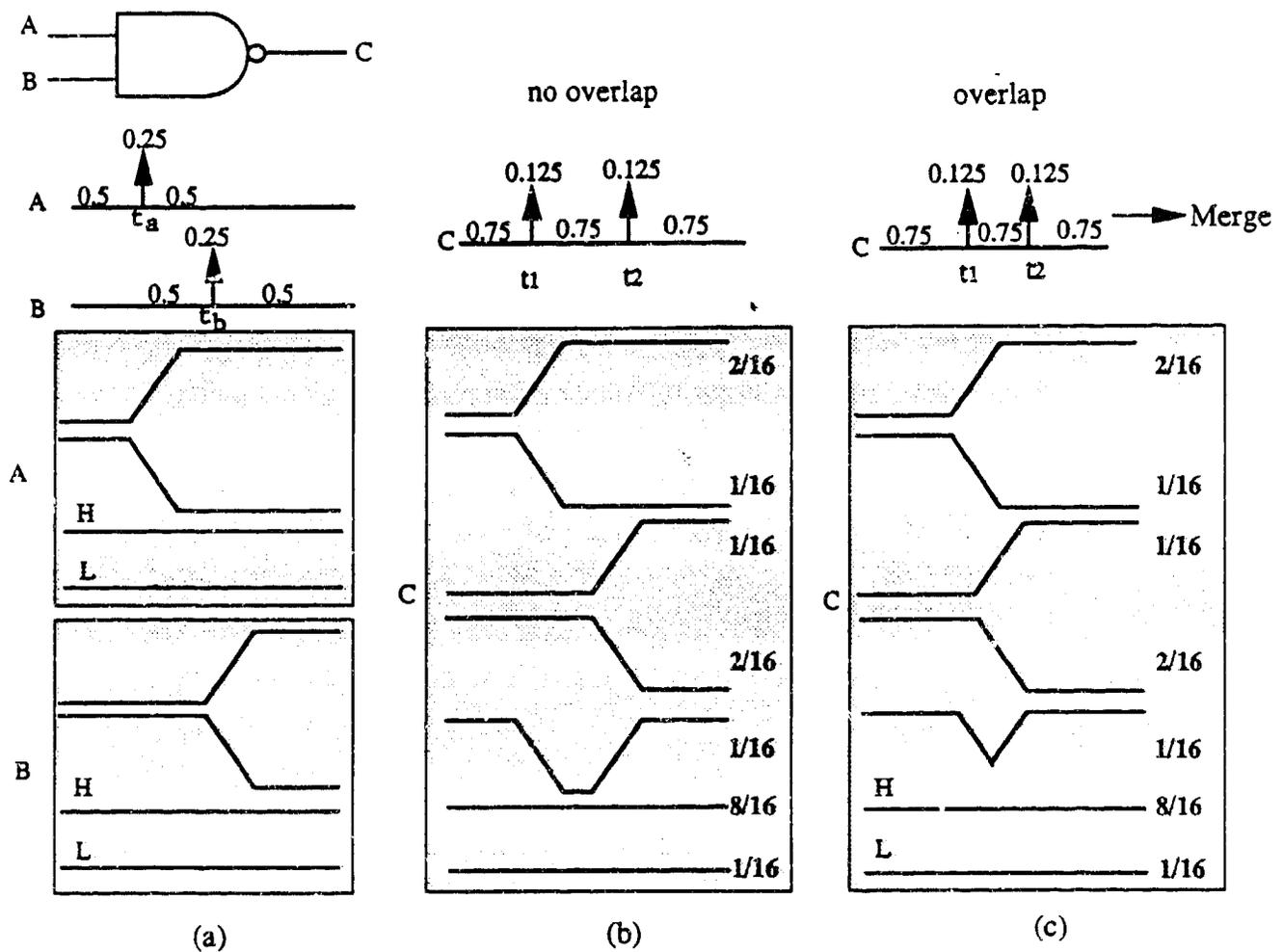


Figure 3.9 (a) A NAND gate and the input waveforms;
 (b) the output waveforms when there is no overlap;
 (c) the output waveforms when there is overlap.

Overlap can cause incomplete transitions and hence two issues need to be addressed: (1) the overlapped waveform may not propagate beyond the fanout gates; and (2) the overlapped waveform may affect the degradation within the subcircuit. Whether an overlapped waveform will propagate depends on the voltage of the intersection point of the overlapped events, as shown in Fig. 3.10. If the intersection point is close to V_{DD} for Λ -type waveform (or G_{ND} for V-type), then the two events can be considered as two complete transitions by the fanout

gates. On the other hand, if the intersection point is too low (or too high for V-type), then it is only a glitch for the fanout gates and the overlapped waveform will not be propagated.

Based on circuit simulation results on typical designs, a threshold of $V_{DD}/2$ is chosen to filter out some overlapped waveforms; that is, if the intersection point of a Λ -type (V-type) waveform is $V_{DD}/2$ or higher (lower), then the overlapped waveforms are propagated as if there is no overlap; otherwise the two events are considered overlapped and merged into one before being propagated to the fanout gates. Note that only events on the same signal line would be merged. Figure 3.11 shows the merging of two adjacent events with the following equations:

$$Plh(a) = Plh(1) + Plh(2) - Prob(\Lambda - type\ overlap) \quad (3.14)$$

if events 1 and 2 are caused by two transistors in series in the p-part of the gate. Or

$$Plh(a) = Plh(1) + Plh(2) - Prob(V - type\ overlap) \quad (3.15)$$

if events 1 and 2 are caused by two transistors in series in the n-part of the gate.



Figure 3.10 Two types of overlapped waveforms and the intersection points.

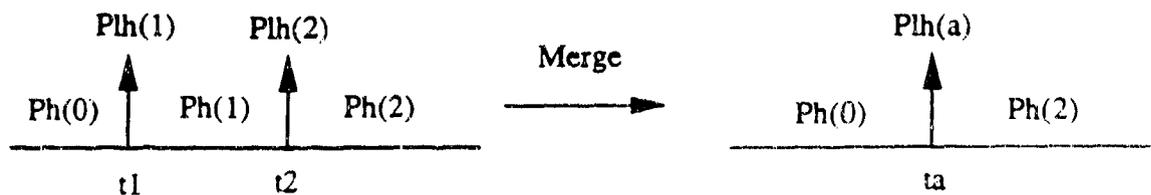


Figure 3.11 The merging of events

$$t_a = \frac{(Plh(1)+Phl(1)) \times t_1 + (Plh(2)+Phl(2)) \times t_2}{Plh(1)+Phl(1)+Plh(2)+Phl(2)} \quad (3.16)$$

In addition,

$$t_{HL}(a) = \frac{Phl(1)t_{HL}(1)+Phl(2)t_{HL}(2)}{Phl(1)+Phl(2)}; \quad (3.17)$$

$$t_{LH}(a) = \frac{Plh(1)t_{LH}(1)+Plh(2)t_{LH}(2)}{Plh(1)+Plh(2)}. \quad (3.18)$$

The effect of overlap on damage estimation is that an overlapped waveform does not draw the same amount of current as two complete transitions do, resulting in less degradation than two complete transitions. Based on circuit simulation experiments, we have found that the degradation in nMOS transistor caused during the falling edge of an incomplete Λ -type glitch is reduced, according to the following equation.

$$(D_n)_{ave} = (Phl - kProb(\Lambda - type\ overlap)) \frac{1}{WH} \int I_{DS} \left(\frac{I_{sub}}{I_{DS}} \right)^m dt \quad (3.19)$$

where Phl is the output high-to-low probability caused by the second event before merging, and where k is a function of the intersection point voltage V ,

$$k = \begin{cases} 2 - \frac{2V}{V_{DD}} & V_{DD}/2 \leq V \leq V_{DD} \\ 1 & V \leq V_{DD}/2 \end{cases} \quad (3.20)$$

The modification applies to the calculation of $(\int I_{sub})_{ave}$ as well. Note that damage occurs in the nMOS transistor during the falling edge of a waveform and none during the rising edge.

3.5 Implementation and Simulation Results

We have implemented the above approach in program *iProbe-d*. The program consists of four main parts: (1) Input circuit reading subroutines, (2) partitioning and scheduling, (3) circuit simulation based on waveform relaxation, and (4) probabilistic and timing simulation.

iProbe-d reads in *SPICE*-type circuit description and creates a circuit database, the input circuit can be defined hierarchically. Instead of flattening the input circuit, the program preserves the circuit hierarchy to save memory space.

The partitioning and scheduling parts of the program are flexible in the sense that the user may choose either to partition the circuit or keep the user specified partitions. If partitioning is activated, the circuit at the lowest level in the hierarchy is partitioned into channel-connected subcircuits. The scheduling part orders subcircuits for simulation and detects feedback loops.

The circuit simulation part is implemented for the situation when more accurate simulation is necessary and for verifying the accuracy of the results during the program development stage. The probabilistic and timing simulations are combined together. In fact, timing simulation is a special case of probabilistic timing simulation by specifying the values of Ph , Pl , Plh , Phl to be 0 or 1.

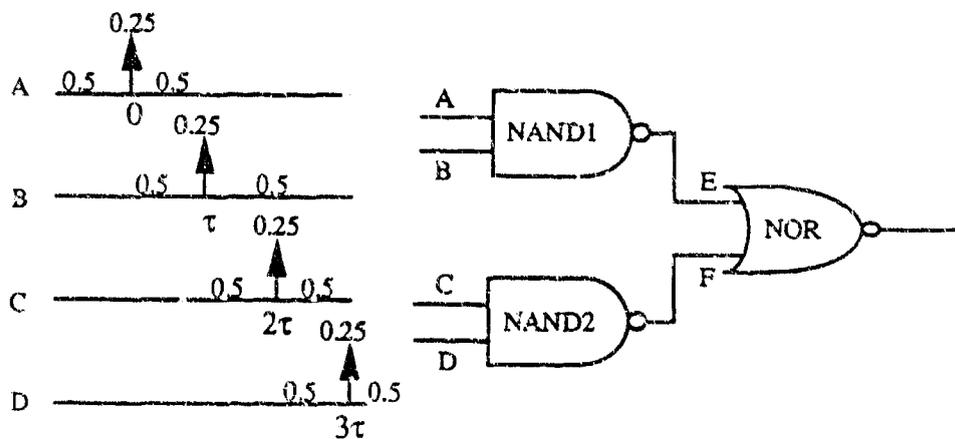


Figure 3.12 A test CMOS combinational circuit and the input statistical descriptions. t_{HL} and t_{LH} are both 1.0 ns for each input event. Simulation period $T = 20$ ns.

iProbe-d has been used to simulate several simple and benchmark circuits [6]. Figure 3.12 shows a CMOS circuit with two NAND and one NOR gates. The statistical descriptions of the four input nodes are also shown in the figure. Table 3.2 illustrates the degradation (D_n) of the six nMOS transistors using *iProbe-d*, *SPICE exhaustive simulation*, and *SPICE one run*, respectively. The inputs are assumed to be uncorrelated. The circuit is simulated for a time period $T = 20$ ns; the input events are then assumed to repeat periodically and the damage is extrapolated to three months. For each primary input node, there are four possible waveforms (staying high, staying low, switching from low to high, from high to low); hence, there are $4^4 = 256$ possible input combinations. Therefore, for the *SPICE exhaustive simulation*, 256 runs are needed. On the other hand, only one run is necessary when *iProbe-d* is employed. The input voltage waveforms of the third row in Table 3.2, *SPICE one run*, are $A = B = D = V_{DD}$ and $C = a$ periodical square wave. From the result, *iProbe-d* and *SPICE exhaustive simulation* show the same tendency, while running *SPICE* with one particular input waveform points to the wrong critical transistor. We have also compared the waveform obtained from *iProbe-d* and *SPICE exhaustive simulation* for this circuit. The slew rate and propagation delay from *iProbe-d* are within 5% error compared to the average values from *SPICE exhaustive simulation* on all the internal nodes.

Table 3.2 demonstrates the effect of overlap as well. For the two NAND gates, as pointed out in previous section, there exists only *V-type* overlap and hence the degradation in the top nMOS transistors in these two gates would not be affected. Both *iProbe-d* and *SPICE exhaustive simulation* prove the prediction. At the NOR gate, there exists *A-type* overlap and the right transistor, which switches last, has less degradation than the left one when there is overlap. The decreasing degradation in the left transistor in the NOR gate is the result of

overlap in the NOR gate as well as the event-merging at node E . A number of ISCAS85 benchmark circuits [6] have also been analyzed. These benchmark circuits have been converted to *SPICE* input file with complementary CMOS transistor circuit implementation of each gate. Table 3.3 shows the computation time on a SPARC station 10, together with the damage estimation after a simulated three months continuous operation. Here we assign one event to each primary input node, with 50ns as the simulation period, and assume the primary input signals are uncorrelated. We have also simulated C432 with the circuit simulation mode of *iProbe-d*, which in this case is faster than *SPICE*¹ since the circuit is combinational with no feedback, the simulation time for one input vector specification on a SPARC station10 is 11 seconds. This circuit has 36 inputs and exhaustive simulation would require $4^{36} \times 11$ seconds = 1.65×10^{15} years! The results show that *iProbe-d* can handle large circuits with reasonable computation time. The results shown in Table 3.3 ignore signal correlations in analyzing the gates within the circuit. To take into account signal correlation effects, the method proposed in [5] and modified in [4] to include delay information is being implemented.

3.6 Transistor Degradation Model and Effects on Circuit Performance

So far, we have estimated relative HCE degradation in a MOS transistor by estimating its substrate current I_{sub} , which is a function of drain current I_{DS} and drain to source voltage V_{DS} . The actual physical damage due to HCE in nMOS transistors is due in great part to interface charge generation, which can be represented by the interface state density N_{it} . The relation between N_{it} , I_{sub} , I_{DS} , and V_{DS} is given by [14], [24]:

¹If the circuit has feedback, the deterministic circuit simulation mode of *iProbe-d* uses waveform relaxation; in this case, *iProbe-d* may or may not be faster than *SPICE* in analyzing the circuit for one input waveform (depending on the rate of convergence).

	NAND1 top	NAND1 bottom	NAND2 top	NAND2 bottom	NOR left	NOR right
SPICE exhaustive						
$\tau = 5ns$	2.969	7.56e-5	2.975	7.80e-5	2.285	2.280
$\tau = 1.5ns$	2.966	5.63e-5	2.975	5.65e-5	1.718	0.911
$\tau = 0.5ns$	2.976	4.63e-5	2.978	4.73e-5	1.648	0.397
iProbed						
$\tau = 5ns$	2.870	6.28e-5	2.870	6.28e-5	2.045	2.045
$\tau = 1.5ns$	2.870	6.28e-5	2.870	6.28e-5	1.713	1.111
$\tau = 0.5ns$	2.870	6.28e-5	2.870	6.28e-5	1.535	0.384
SPICE one run *	0	0	6.43	0	0	9.62

Table 3.2 The relative damage of all the nMOS transistors in the circuit shown in Fig. 12. * A = B = D = high(V_{DD}); C = a periodical square wave.

ISCAS85 Circuits	# of nMOS transistors in each damage level (Normalized D_n)						CPU time (sec)
	< 1.0	1.0 - 2.0	2.0 - 3.0	3.0 - 4.0	4.0 - 5.0	> 5.0	
C432	232	37	16	65	9	53	1.00
C498	576	0	80	0	0	226	1.65
C880	443	62	40	39	52	265	3.37
C1355	776	176	56	32	0	114	4.43
C1908	653	30	29	65	86	760	7.60
C2670	971	209	151	80	125	1146	8.49
C3540	1450	340	235	171	143	1423	17.12
C5315	2147	274	374	261	192	2385	18.37
C6288	512	92	495	417	156	3384	92.08
C7552	2767	332	861	262	191	3285	26.84

Table 3.3 The computation time of *iProbe-d* on several benchmark circuits, and predicted average damage after three months continuous operation. The simulation is done on a SUN4 SPARC station10.

$$V_{DSAT} = \frac{E_{crit0}L(V_{GS} - V_T)}{E_{crit0}L + V_{GS} - V_T} \quad (3.21)$$

$$I_{sub} = \frac{A_i}{B_i} I_{DS}(V_{DS} - V_{DSAT}) \exp\left\{\frac{-B_i l_c}{V_{DS} - V_{DSAT}}\right\} \quad (3.22)$$

$$N_H + \frac{B_p X_H}{2D_H} N_H^2 = K \int_0^T \frac{I_{sub}^{2.9}}{I_{GS}^{1.9}} dt \quad (3.23)$$

Where $E_{crit0} = 10^4 V/cm$; D_H and X_H represent diffusion constants; K and B_p represent

process-dependent coefficients; $l_c = 0.2X_{ox}^{1/3}X_j^{1/2}$; where X_{ox} is the oxide thickness and X_j is the junction depth.

Since HCE can cause a shift in V_T and degradation in g_m and μ_n , it may result in individual device failure as well as in overall circuit performance degradation. In digital circuits, the latter is usually an increase of circuit delay. We thus set two criteria to evaluate HCE degradation. The first criterion is the local damage, represented by N_{it} , in each nMOS transistor; and the second criterion is the increase in total delay in the circuit, which can be determined by searching the longest true path before and after degradation. If either criterion exceeds a user specified limit, it is then necessary to redesign the circuit.

In estimating the circuit delay we employ a linearized RC model. In each subcircuit, the transistors are replaced by equivalent resistors. The subcircuit capacitors are lumped together

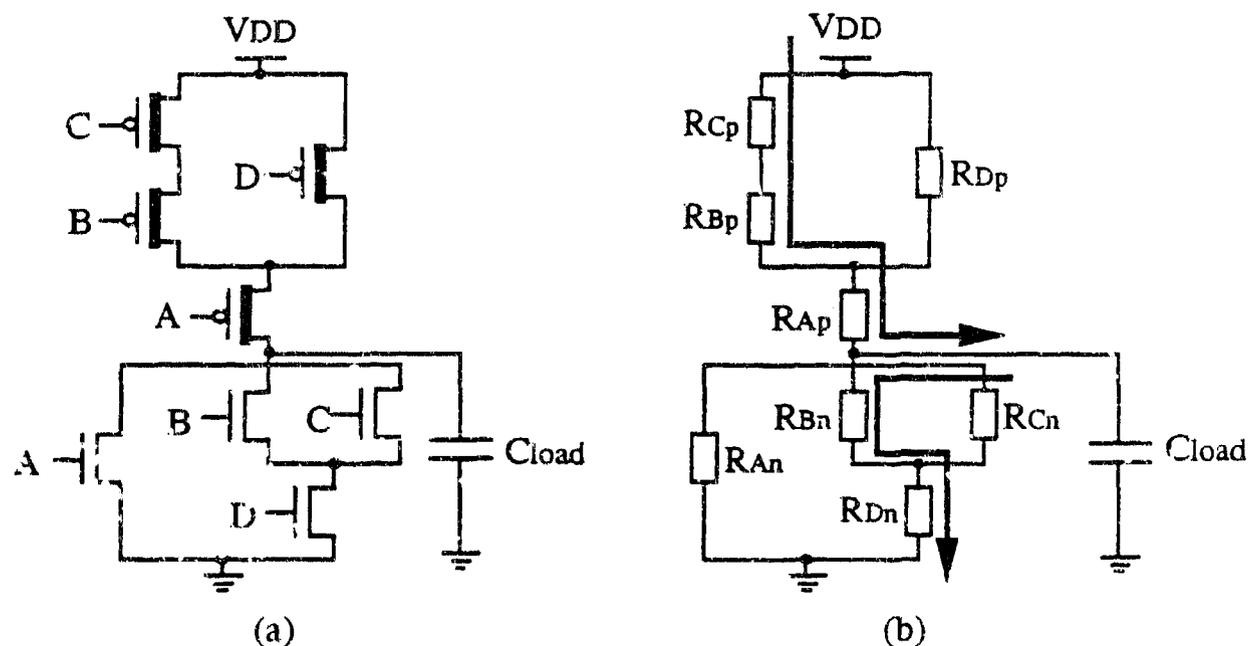


Figure 3.13 (a) A CMOS circuit and (b) its equivalent RC network and worst-case delay paths; the arrows represent longest resistive paths in the n-part and p-part of the subcircuit.

at the output node. Figure 3.13 shows a CMOS circuit and its equivalent RC network. Worst-case delay estimation is assumed. In this case, the worst-case rise and fall delay of the circuit are:

$$T_{Drise} = (R_{A_p} + R_{B_p} + R_{C_p}) * C_{load} \quad (3.24)$$

$$T_{Dfall} = (R_{B_n} + R_{D_n}) * C_{load} \quad (3.25)$$

assuming that

$$R_{A_n} = R_{B_n} = R_{C_n} = R_{D_n}, R_{A_p} = R_{B_p} = R_{C_p} = R_{D_p} \quad (3.26)$$

and

$$R_{eq}^{-1} = G_{eq} = \begin{cases} \frac{W}{L} \mu_n C_{ox} (V_{DD} - V_{T_n}) & \text{for } nMOS \\ \frac{W}{L} \mu_p C_{ox} (V_{DD} + V_{T_p}) & \text{for } pMOS \end{cases} \quad (3.27)$$

The equations above show that HCE may affect the nMOS equivalent resistance, which depends on both V_T and μ_n , but the capacitance is not affected. Earlier work on HCE assumed that the oxide-interface charge ($Q_{it} = N_{it} * q$) is uniformly distributed over the entire channel region and hence the change of threshold voltage is linearly proportional to the generated interface charge [25]. However, experiments have proved that the damage caused by HCE is localized in the channel near the drain node (for nMOS) and a uniform model cannot correctly characterize the damaged device. A one-dimensional model has been proposed by Leblebici and Kang for nMOS transistors with hot-carrier induced oxide damage [26]. In this model, the transistor channel is divided into two regions, the *damaged* and *undamaged* region. As reported in [26], based on experimental data for 1 μm technology, the damaged region is located within 0.1 μm of the drain node and the interface charge is uniformly distributed in this region. The rest of the channel remains undamaged with no generated interface charge.

For submicron technology, the length of the damaged region needs to be deduced from new experimental data.

We adopt the concept of this model and use a two-transistor degradation model, as shown in Fig. 3.14, where a damaged nMOS transistor is considered as two transistors connected in series, with different threshold voltages and mobilities. The channel length of the two transistors are $0.1 \mu\text{m}$ and $L - 0.1 \mu\text{m}$ respectively, where L is the channel length of the original transistor and the transistor with the $0.1 \mu\text{m}$ length is considered to be the damaged one. The equivalent conductances of the transistors are calculated with the following equations:

$$\mu_{ox} = 1000 * \frac{3 * 10^{11}}{N_{it}} * \frac{T}{80} \quad (3.28)$$

$$\frac{1}{\mu_{n_1}} = \frac{1}{\mu_n} + \frac{1}{\mu_{ox}} \quad (3.29)$$

$$V_{T_1} = V_T - \frac{Q_{it}}{C_{ox}} \quad (3.30)$$

$$R_1^{-1} = G_1 = \frac{W}{0.1} \mu_{n_1} C_{ox} (V_{DD} - V_{T_1}) \quad (3.31)$$

$$R_2^{-1} = G_2 = \frac{W}{L - 0.1} \mu_n C_{ox} (V_{DD} - V_T) \quad (3.32)$$

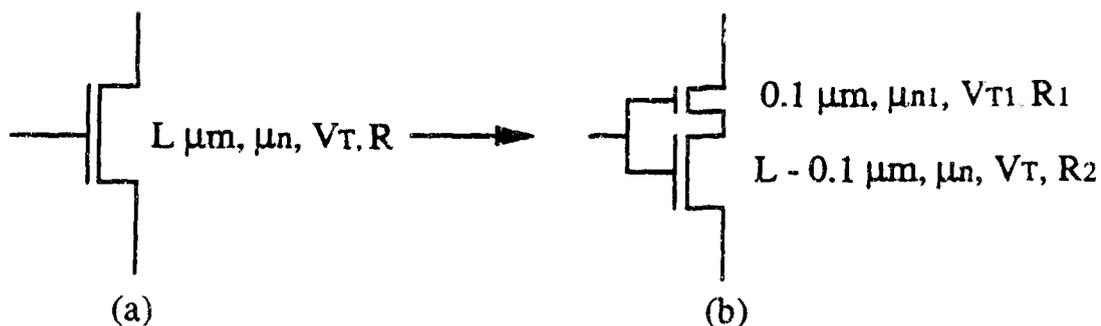


Figure 3.14 An nMOSFET (a) before and (b) after HCE degradation.

and the increase of fall delay is:

$$\Delta T_{Dfall} = \Delta R_{eq} * C_{load} \quad (3.33)$$

Note that there is no change in the rise delay since the damage in pMOSFETs is considered to be negligible.

If the circuit under consideration does not have a logic description available and its logic function cannot be retrieved from its netlist, then the delay of its longest path will be considered as its delay. In that case, a combination of topological sort and worst-case delay propagation is capable of finding the longest path and delay [27]. When the logic function of the circuit is known, then the sensitizability [28] of the longest path should be checked.

Several sensitization criteria of true/false path identification have been investigated [28]. In our work, we use static sensitization as in [29], with some modification. For a path $P = \langle f_0, f_1, \dots, f_n \rangle$, where f_0, f_1, \dots, f_n are internal nodes, to be statically sensitizable, a transition at f_i must cause a transition at f_{i+1} . This is equivalent to saying that all of the other inputs to the gates on the path must have non-controlling stable values (eg., 1 for AND gates and 0 for OR gates). The delay of the longest statically sensitizable path is defined as the delay of the circuit. With binary decision diagrams (BDD) [30], [31], we can efficiently calculate the sensitizability of a given path without backtracking. For example, in order for the path $P = \langle X, b, d, e, g \rangle$ in Fig. 3.15 to be statically sensitizable, the condition $(Z = c = 1, f = 0)$ must be satisfied. Since c and f are not primary inputs, backtracking is necessary to check whether the logic constraint can be satisfied. However, if we generate a BDD for each internal node at the outset, namely, $c = (X' + Y' + Z')$ and $f = XZ$. The sensitization problem can be transformed into a satisfiability problem [29]. In this case,

$Z \cap c \cap f' = Z \cap (X' + Y' + Z') \cap (X' + Z') \neq \emptyset$; therefore, path P is statically sensitizable.

Most previous work on critical path analysis makes the assumption that each gate has only one delay value associated with it, and hence each path has one delay. However, when the gate delay is modeled with macromodeling and RC delay definition, it is possible that the rise delay of a gate is different from its fall delay, and hence there are two delay values associated with a given path. For example, the path P in Fig. 3.15 has two delay values,

$$T_{B_{rise}} + T_{D_{fall}} + T_{E_{fall}} + T_{G_{rise}} \quad (3.34)$$

and

$$T_{B_{fall}} + T_{D_{rise}} + T_{E_{rise}} + T_{G_{fall}} \quad (3.35)$$

For XOR gates in which the output may switch either way when the input has a transition, we use the larger one of rise and fall delay to make sure the path delay is not underestimated.

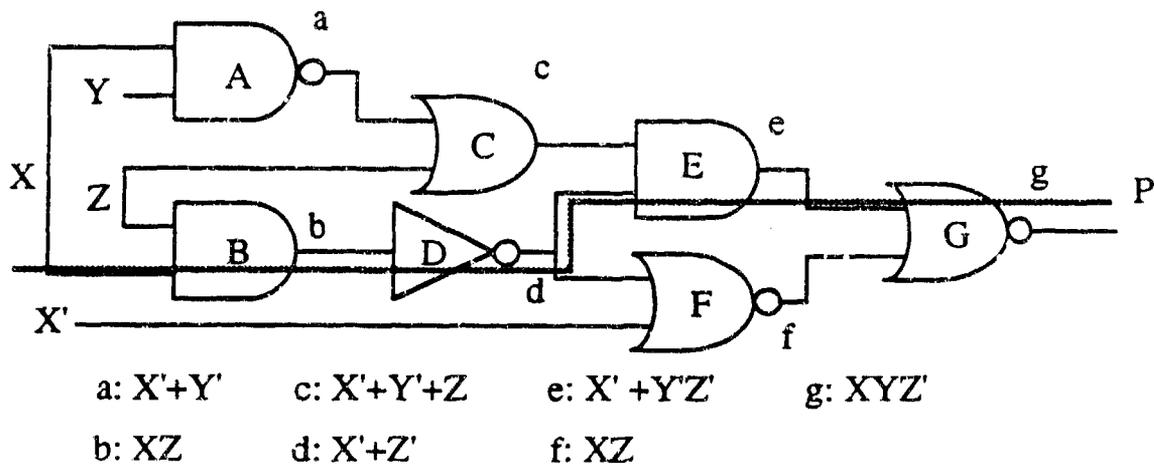


Figure 3.15 A logic circuit example.

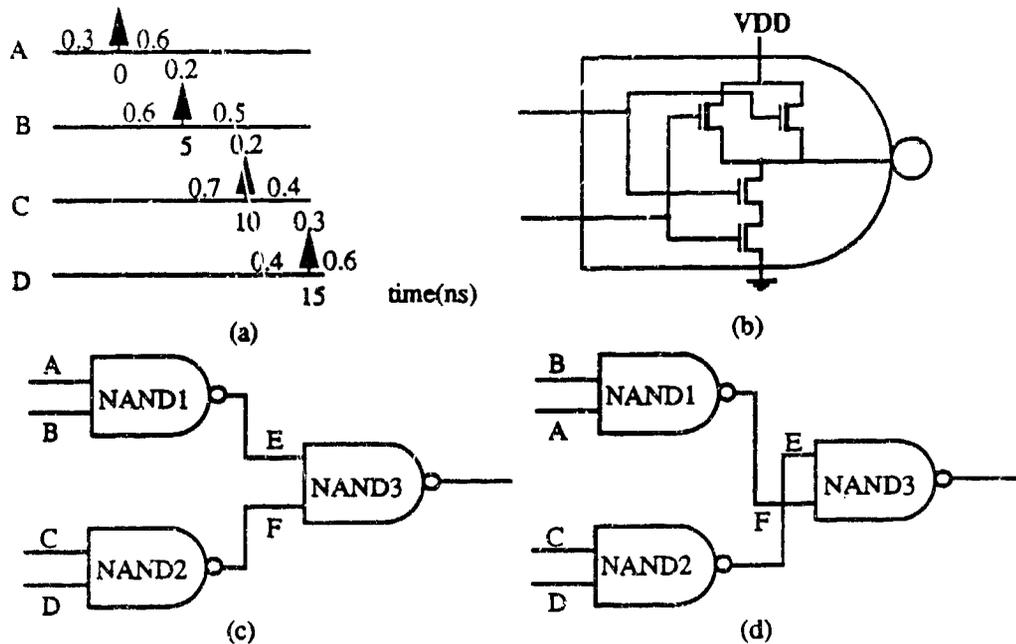


Figure 3.16 (a) Probabilistic voltage waveforms at the four input nodes, A, B, C, and D. t_{HL} and t_{LH} are both 0.2ns for each input event; (b) detailed structure of a CMOS NAND gate; (c) original connection of a three-NAND-gate circuit; (d) a better connection of the three-NAND-gate circuit for reliability.

	NAND1 top	NAND1 bottom	NAND2 top	NAND2 bottom	NAND3 top	NAND3 bottom
connection1						
D_n	2.223e-02	-	5.927e-03	-	3.323e-02	-
$\sum P_{hl_{out}}$	0.3	0.12	0.08	0.12	0.216	0.168
connection2						
D_n	8.890e-03	-	5.927e-03	-	2.584e-02	-
$\sum P_{hl_{out}}$	0.12	0.3	0.08	0.12	0.168	0.216

Table 3.4 The damage N_{it} ($\times 10^{11} \text{ cm}^{-2}$) of all the nMOS transistors in the circuit shown in Fig. 3.16(c) and (d).

3.7 HCE Resistant Design

The factors that influence HCE degradation in a transistor within a design include its feature size, its switching probability, its position in the circuit, the loading capacitance, and

input slew rate. Several strategies, based on these factors, have been previously proposed to improve circuit reliability. However, these proposed improvements have been suggested on small circuit structures in isolation, without taking into consideration the operation of the small circuit within a large design, as we do here. In the following, we first propose a design strategy to reduce HCE effects without increasing area, and then review and compare previously proposed techniques.

1. Since the damage in a transistor depends on its switching probability as well as its position in the circuit, the damage can be reduced by connecting nMOS transistors that have the least switching probability directly to the output node [24]. This is done by reconnecting the circuit input wiring so that the top nMOS transistor always has the least switching probability. For example, Fig. 3.16(a) shows the waveform description of four input nodes A, B, C, and D. The circuit under test is a three-NAND-gate circuit with detailed gate structure in Fig. 3.16(b). When the circuit is connected as in Fig. 3.16(c), the HCE degradation is listed in row 1 of Table 3.4, as well as the switching probability of each nMOS transistor. It is noticed that the bottom transistors of gate NAND1 and NAND3 have lower switching probability value. After reconnecting the input of these two gates as in Fig. 3.16(d), the HCE degradation is reduced, as shown in the second row of Table 3.4. The advantage of this method is that the circuit area is not increased. However, this method of reordering the transistors cannot be applied to NOR gates in which all the nMOS transistors are directly connected to the output. In this case, some other method, such as increasing the area of the affected transistor or applying method 3 below could be used, with some sacrifice in performance. A better approach would be to eliminate NOR gates with high switching activity through functional transformations.

2. By adding a self-bootstrapping nMOS transistor, as shown in Fig. 3.17(a), the peak substrate current in transistor NM1 can be reduced by as much as 3.9 times according to [32] and the interface-state generation can be suppressed by one to two order of magnitude. In addition, this structure also has shorter rise and fall time compared to conventional NAND logic circuit. A disadvantage of this structure is an increase in circuit area; therefore, it is preferred to add a small self-bootstrapping nMOS transistor. In addition, this structure cannot be used to improve NOR gates.

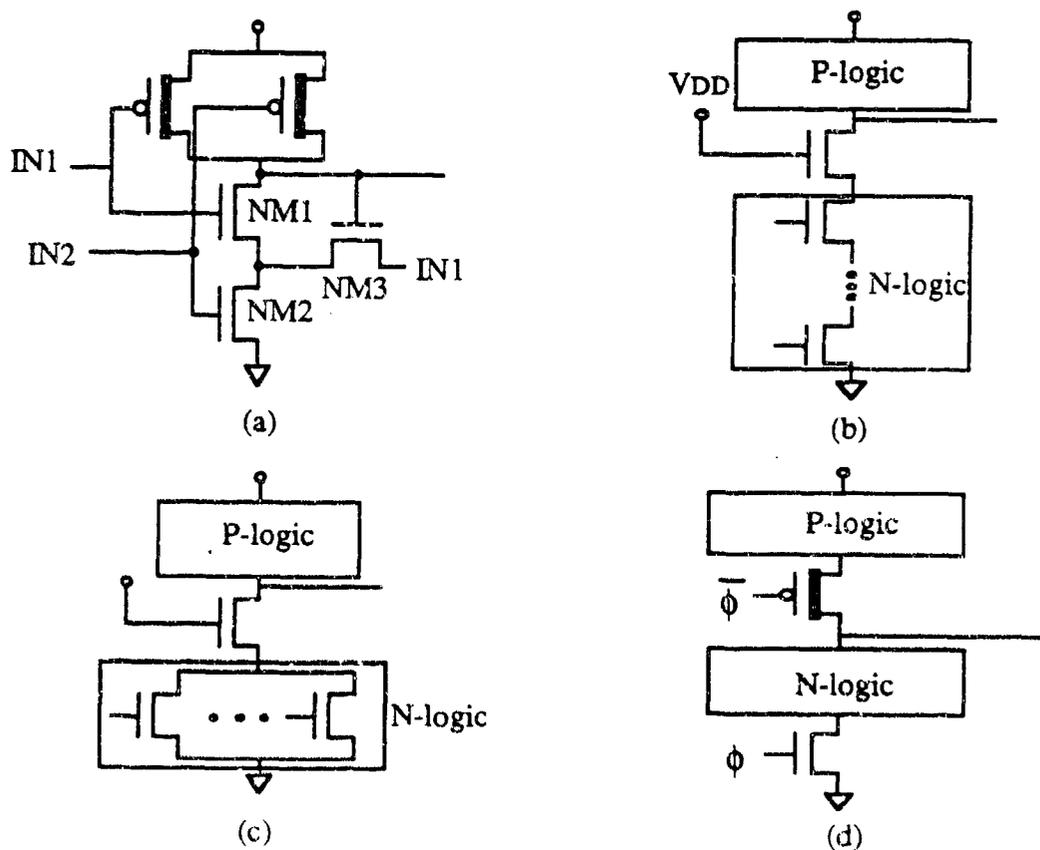


Figure 3.17 Several HCE resistant circuit design strategies.

3. Another HCE suppressing circuit redesign technique, shown in Fig. 3.17(b), has been proposed in [33], where a normally-on nMOS transistor is added between the n-part of a CMOS circuit and the output node. Since the added transistor is always on, the logic function of the circuit is not affected. In addition, the added transistor is always operating in the linear region and hence has negligible hot-carrier generation. The rest of the nMOS transistors are not directly connected to the output node, and thus are relatively safe from HCE degradation. This technique can also be applied to NOR gates, as shown in Fig. 3.17(c). Its disadvantage is the increase in layout area; therefore, it should be selectively used.
4. In Clocked CMOS (C^2 MOS) circuits where the clock signal arrives after all the inputs settle, we recommend that the control nMOS transistors be put anywhere but the top. With this restriction, the top nMOS transistor(s) never switches last, i.e., their switching does not cause output transitions, and hence HCE degradation can be prevented. Fig. 3.17(d) illustrates a HCE resistant C^2 MOS design. The same strategy can be applied to dynamic CMOS design as well. However, the effects of such a design strategy on circuit delay and performance needs to be investigated.

A CMOS circuit design, shown in Fig. 3.18, has been proposed for testability in [34], where it is proposed that, with the four combinations (00, 01, 10, 11) of the control signals ϕ_1 and ϕ_2 , all stuck-open and stuck-on faults of the circuit can be detected. The circuit operates as follows: (1) When $\phi_1 = 0, \phi_2 = 1$, the circuit operates normally and implements the same function as the one without the transistors N_ϕ and P_ϕ (shorted). (2) When $\phi_1 = 1, \phi_2 = 0$, both N_ϕ and P_ϕ are off, and the output state does not change even if the input signals have transitions. (3) When $\phi_1 = 0, \phi_2 = 0$, only P_ϕ is on and all the

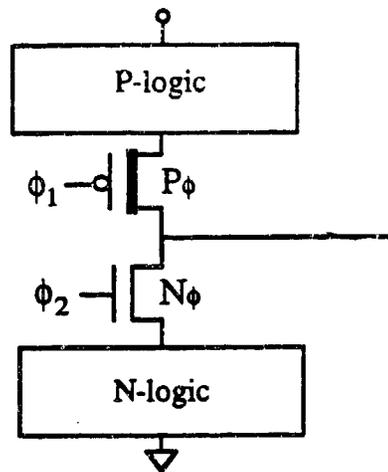


Figure 3.18 A CMOS circuit structure for testability.

stuck-open and stuck-on faults of the pMOSFET network can be detected. (4) Finally, when $\phi_1 = 1, \phi_2 = 1$, only N_ϕ is on and all the stuck-open and stuck-on faults of the nMOSFET network can be detected. We found that this structure can be employed for improving reliability as well. There are two situations: (1) If the circuit is in normal operation most of the time, i.e., the control signals rarely switch, then this design is similar to the structure in Fig. 3.17(b). (2) If the system clock is used as the control signal, then we need to modify the design by moving N_ϕ to the bottom of the nMOSFET network, similar to the case of C^2 MOS design.

The algorithms and strategies above have been included in *iProbe-d* and used to simulate and redesign some benchmark circuits. The results of simulating the ISCAS85 benchmark circuits are shown in Table 3.3. After obtaining the damage and switching probability of each transistor, the critical path analysis algorithm is employed to check the global degradation criterion. Table 3.5 shows the circuit delay before and after HCE degradation. The delay increase mostly comes from INVERTERS and NOR-type gates; all of the NAND-type gates

with more than two inputs have less than 0.05ns delay increase. This observation matches our suggestion that NAND-type gates are more HCE resistant than NOR-type gates. Table 3.6 shows delay degradation for the same circuits when $V_{DD} = 5.5v$. Note that by increasing V_{DD} , the circuit becomes faster, but the damage is increased.

Table 3.5 The circuit delay (longest path) of several benchmark circuits before and after HCE degradation. $V_{DD} = 5v$. Delay in nsec.

Circuit	No. Gates	No. Nodes	Delay before HCE	Delay after HCE	Delay increase
432	160	198	20.88	20.95	0.34%
C499	202	245	16.46	16.50	0.24%
C880	383	445	19.02	19.05	0.16%
C1355	546	589	20.11	20.15	0.20%
C1908	880	915	27.71	27.98	0.97%
C2670	1193	1428	31.38	31.57	0.61%
C3540	1669	1721	37.87	38.13	0.69%
C5315	2307	2487	34.69	34.98	0.84%
C6288	2416	2450	80.04	80.93	1.09%
C7552	3512	3721	28.78	29.01	0.80%

Table 3.6 The circuit delay (longest path) of several benchmark circuits before and after HCE degradation when $V_{DD} = 5.5v$. Delay in nsec.

Circuit	No. Gates	No. Nodes	Delay before HCE	Delay after HCE	Delay increase
432	160	198	17.98	18.21	1.28%
C499	202	245	14.18	14.30	0.85%
C880	383	445	16.40	16.61	1.28%
C1355	546	589	17.32	17.49	0.98%
C1908	880	915	23.90	25.54	6.86%
C2670	1193	1428	27.07	27.84	2.84%
C3540	1669	1721	32.67	33.89	3.73%
C5315	2307	2487	29.93	31.13	4.01%
C6288	2416	2450	69.19	77.81	12.46%
C7552	3512	3721	24.83	25.69	3.46%

Preliminary results of redesign improvements are shown in Table 3.7. This table shows the impact of three strategies, namely, (1) increasing transistor width, (2) adding normally-on transistors, and (3) interchanging gate input lines on benchmark circuit C3540 with $V_{DD} = 5.5v$. We set the criteria of redesign to be normalized N_{it} equal to 5.0 for individual transistor damage and 0.05ns for subcircuit delay increase. We also assume the equivalent resistance of the added normally-on transistor to be one fourth that of the other nMOSFETs. As can be seen from this table, increasing transistor widths can improve circuit delay but have less improvement on transistor damage. Adding normally-on transistors, on the other hand, does eliminate HCE degradation in all the critical transistors, while the circuit delay is slightly improved compared to the damaged circuit. Both strategies increase the total circuit area by 20%, assuming the normally-on transistor occupies same area as one regular nMOSFET. This increase may not be acceptable, and other redesign strategies that do not impact the area need to be considered. The impact of interchanging input lines on HCE is shown in the third row in the table. Note that for this example the improvements in damage and delay resulting from interchanging input lines at critical NAND gates along longest paths are minimal. The reason is that in this example the switching probabilities at the inputs of critical NAND gates are not too different, and interchanging the connections does not improve reliability in a noticeable way. This study indicates that design for reliability should be done as part of the design synthesis process rather than afterwards.

Table 3.7 (1) Increasing the width of critical transistors ($N_i > 5.0$) by 75%;
 (2) Adding a normally-on transistor on top of n-logic of critical subcircuits;
 (3) Interchanging gate input lines. $V_{DD} = 5.5V$.
 Circuit delay before HCE was 32.67 nsec.

C3540	No of nMOSS transistors in each damage level						Circuit delay after HCE
	<1.0	1.0-2.0	2.0-3.0	3.0-4.0	4.0-5.0	>5.0	
initial	950	177	142	118	189	2167	33.89
(1)	959	177	142	177	451	1846	32.84
(2)	3211	176	138	72	155	0	33.26
(3)	967	199	150	112	215	2109	33.87

4. MAXIMUM CURRENT AND VOLTAGE DROP ESTIMATION

4.1 Introduction

Voltage drop is another important reliability issue which is related to the current flow in the power bus. Excessive supply currents can severely affect both circuit reliability and performance by causing excessive voltage drops in the Power and Ground (P&G) lines. Excessive voltage drops manifest themselves as *glitches* on the P&G lines, and cause erroneous logic signals (soft errors) and degradation in switching speeds. Severity of the voltage drop problems intensify with the continuing push for denser chips and finer technologies. With higher currents flowing in narrower lines, the voltage drops in the PG lines go up and quickly become a limiting factor in the design of VLSI chips. Furthermore, a lower supply voltage means that the noise margins for the correct operation of the transistors on the chip decrease. In order to avoid logic errors, the circuit needs to be appropriately designed to take care of increased voltage drops and reduced noise margins. Worst case voltage drops are determined by maximum current flows rather than averages. Thus this research subtask is focused on the problem of estimating *maximum currents* in the P&G lines.

Current drawn by a CMOS circuit depends upon the specific input pattern applied at its inputs. An *input pattern* for a circuit with n inputs is defined as a vector of n excitations, where each excitation could be any one of four possibilities i.e., *low*, *high*, *high to low* or *low to high*. For different input patterns, different transient current waveforms are drawn at the contact points. Therefore, in the presence of such input dependent and transient current waveforms, we need to define what we mean by the maximum current waveform at a contact point.

Accurate estimation of the maximum current waveform at every contact point is extremely difficult since for that we need to determine current waveforms corresponding to all possible input patterns. If the circuit has n primary inputs then we need to explore the set of 2^{2n} input patterns to calculate the maximum current waveforms. This makes the problem practically impossible to handle by any of the known search procedures for large circuits.

Several papers (such as [36]-[39]) have appeared in literature on the estimation of P&G currents from deterministic input patterns. These methods offer significant improvement in execution times compared to SPICE, while providing acceptable accuracy in the current waveforms. These methods can be used for finding maximum currents for small circuits having a few inputs, by calculating the current waveforms corresponding to all possible input patterns. However they are not much helpful for large circuits, as they do not guide us in selecting an input pattern that leads to the maximum currents.

In [35], Chowdhury et al., have addressed the problem of maximum current estimation for large circuits. The circuit is first divided into a set of macros, where each macro consists of a combinational interconnection of logic gates. Considering each macro separately, either an exact search technique (namely, branch and bound) or a heuristic technique is used to find the maximum of its transient currents, assuming that the macro has only one contact point and its inputs switch simultaneously. In the analysis of the bus, to calculate maximum voltage drops, the authors assume that all the macros draw their maximum currents simultaneously. Because of these assumptions, their methodology overestimates the worst case currents and voltage drops. Secondly, due to the huge size of the input space, their branch and bound search technique is slow on large circuits. Furthermore, their heuristic approach does not guarantee an upper bound on the maximum currents.

In [40], Devadas et al., have addressed a similar problem. They consider the estimation of worst case power dissipation in CMOS combinational circuits. They reduce this problem to a weighted max-satisfiability problem, on a set of multi-output Boolean functions obtained from the circuit logic description. These functions are appropriately weighted to account for different load capacitances. They then use either a disjoint cover enumeration algorithm or the branch and bound algorithm to solve the (*NP*-complete) max-satisfiability problem. They are able to account for the glitches at various internal nodes. However, for a multilevel logic circuit, even under a unit gate delay assumption, the functions generated by their algorithm are fairly complex. Consequently, even for small circuits, their analysis is slow. Analysis of multi-level circuits under a general delay model was not attempted.

In this chapter, we propose a better measure of the maximum current waveform at a contact point called *maximum envelope current* (**MEC**) waveform. We then propose a pattern independent, linear time (in the number of gates) algorithm (**iMax**) that provides tight upper bounds on the **MEC** waveforms. The proposed approach represents a trade-off between execution speed and tightness of these bounds. In order to maintain reasonable execution times, the **iMax** algorithm neglects various signal correlations that exist inside a circuit. As will be shown later, while in most cases **iMax** produces good upper bound waveforms, in some cases the loss due to signal correlations can be significant. We then propose a new *partial input enumeration* (**PIE**) algorithm that efficiently resolves these correlations and leads to significant improvement in the upper bound waveforms. The **PIE** algorithm is based on (1) intelligently selecting a few *critical* inputs and (2) enumerating a limited number of cases at these inputs to produce an overall improvement in the upper bound waveform at every contact point. It turns out that the choice of these critical inputs is the key to improving the upper bound. We

present two heuristics for automatically selecting the critical inputs, that have shown good results in practice. While the PIE algorithm is slower than the simple iMax algorithm, we demonstrate good speed and accuracy performance results on circuits with over twenty thousand gates. Furthermore, the algorithm has the attractive property that it does an *iterative improvement*, so that one can stop the algorithm at any time and obtain a better upper bound than the simple *iMax* result.

In the next section, we discuss various assumptions on which our algorithms are based and describe the proposed maximum current estimate. After that, we present the iMax algorithm in detail in Section 4.3. Experimental results on several benchmark circuits using *iMax* are also provided in this section. The signal correlation problem is described in Section 4.4. This is followed by a discussion of possible methods that can be used to resolve the signal correlations in Section 4.5. In Section 4.6, we present the partial input enumeration algorithm along with extensive experimental results on several benchmark circuits.

4.2 Maximum Current Estimates

We consider synchronous digital circuits consisting of combinational blocks separated by latches (Fig. 4.1) such that all the inputs to each combinational block switch simultaneously,

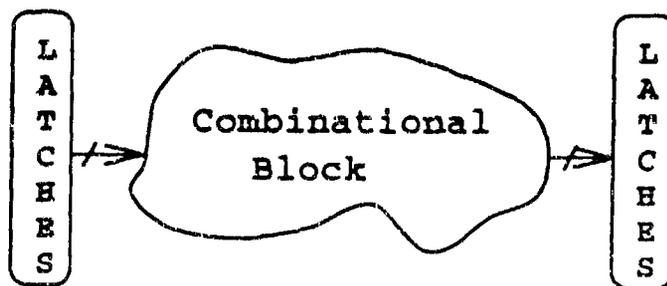


Figure 4.1 An example of a latch controlled synchronous digital circuit.

when they do. This effectively eliminates the time domain uncertainty about the input transitions. This assumption has also been used by all the previous approaches.

We assume that the delay of each gate in the circuit is specified. Different gates can have different delays. Further, we assume that each time the output of a gate switches, a pulse of current approximated by a triangular waveform is drawn from the supply lines, as shown in Fig. 4.2. The duration of this pulse is computed from the delay of the gate and its height (peak value) is pre-characterized. Two separate values for the peak current may be specified, corresponding to *low to high* and *high to low* transitions at the gate output.

Given the specific clocking scheme of the synchronous circuit, the maximum current waveforms from different combinational blocks can be appropriately shifted in time depending upon the individual clock trigger. We will thus focus on the analysis of a single combinational block whose inputs switch at time zero.

We define *excitation* at a node (or net) at any time t as the stimulus (or signal value) present at the node at that time. At any time, a node in the circuit could be either stable at *low* or *high*, or could transition from *high* to *low* or from *low* to *high*. Thus, the excitation could be any single value from the set $\mathbf{X} = l, h, hl, lh$, where $l = \text{low}$, $h = \text{high}$, $hl =$

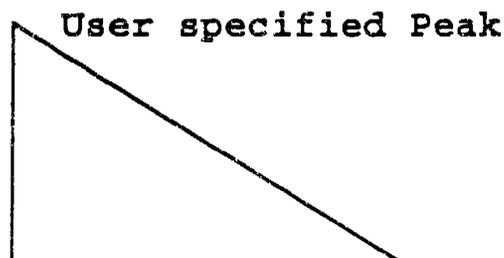


Figure 4.2 Model of a gate current pulse.

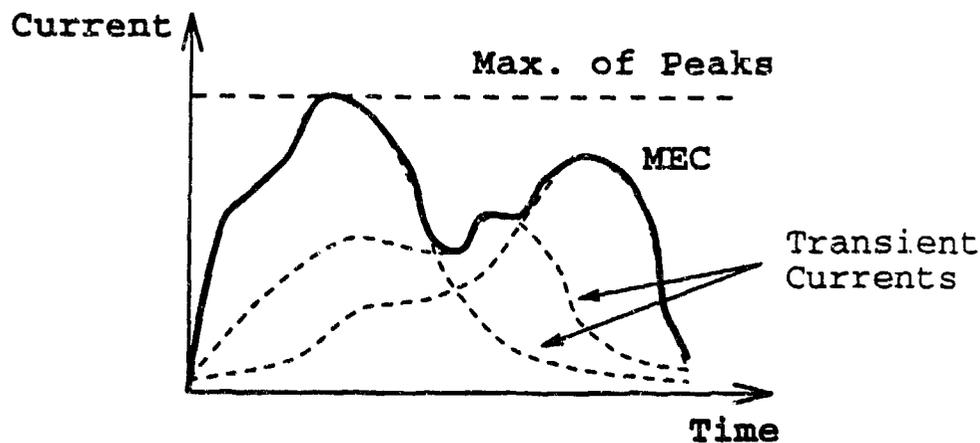


Figure 4.3 The Maximum Envelope Current (MEC) waveform.

high to low transition and $lh = \text{low to high transition}$.

For each input pattern that is applied to the circuit, a transient current waveform is drawn from the P&G lines at a contact point. As shown in Figure 4.3, instead of representing the maximum current at a contact point by a single dc value, in our approach, we represent it by a waveform whose value at any time is the maximum current value that the circuit can draw at that time. We call this the *Maximum Envelope Current (MEC)* waveform.

Suppose that a circuit under consideration has n inputs and when an input pattern $p = (e_1, e_2, \dots, e_n)$, where $e_i \in \mathbf{X}, 1 \leq i \leq n$, is applied to the circuit, a transient current waveform $I_p(t)$ is drawn from the circuit at the contact point. Denote the set of all possible input patterns that can be applied to the circuit by $U = \{(e_1, e_2, \dots, e_n) \mid e_i \in \mathbf{X}, 1 \leq i \leq n\}$. If the value of the MEC waveform at any time t at the contact point is denoted by $I_{\text{MEC}}(t)$, then the following equation applies

$$I_{\text{MEC}}(t) = \max_{p \in U} I_p(t) \quad (4.1)$$

Clearly, the MEC waveform is the maximum envelope of all transient current waveforms corresponding to all possible input patterns. There is a unique MEC waveform at every

contact point

If the power or the ground bus of a circuit is represented by an equivalent RC network, then we have the following result:

Theorem 1: The voltage drop $\left[V_k^{\text{MEC}}(t) \right]$, calculated at any node k in the power or ground bus when the MEC waveforms are applied at the contact points, is an upper bound on the voltage drop $\left[V_k^p(t) \right]$ occurring at that node when any input pattern p is applied to the circuit, i.e., $V_k^{\text{MEC}}(t) \geq V_k^p(t)$, for all t .

4.3 The Proposed (iMax) Algorithm

The proposed pattern independent, linear time algorithm operates at the gate level description of the circuit. Unless specified by the user, we assume that nothing is known about the specific excitations at the primary inputs, except that they may transition (only) at time zero, i.e., each primary input may carry any excitation from the set \mathbf{X} at time zero. We call this an *uncertainty* about these input signals. The basic idea of the proposed algorithm is to propagate this uncertainty present at the inputs inside the circuit, so that, at the output of every logic gate, we know the set of all possible excitations and their associated timing. From this, the worst case gate currents are computed, as explained below.

4.3.1 Signal Representation

Perhaps the first question that comes to mind is what kind of information one maintains in order to represent the signal uncertainty about internal circuit nodes. Ideally, one would like to compute the set of *all* possible transitions (along with their timing information) that occur at the output of every gate in the circuit. That would certainly be enough to estimate

the MEC waveforms. However, due to the uncertainty at the primary inputs and the general gate delay model used, the number of possible transitions at internal gates grows exponentially, and quickly becomes a bottleneck. To avoid this problem, we maintain information, not about individual transitions, but about *intervals* during which the output of the gate *might* switch. Thus, for each of the excitations *low*, *high*, *hl* and *lh*, we store a list of intervals during which a node might carry that excitation. These intervals, which might overlap, serve to describe the signal uncertainty. We call these intervals *uncertainty intervals*.

Definition 1 (Uncertainty Set $X_n(t)$): The uncertainty set at time t for a node n defines the set of all excitations that the node can possibly assume at that time. $X_n(t) \subseteq X$.

Definition 2 (Uncertainty Waveform): The uncertainty waveform describes the signal uncertainty present at a node as a function of time. At time t , the set of values taken by the waveform is the uncertainty set for the node at that time.

An example of the uncertainty waveform is given in Fig. 4.4. In this figure, we show an uncertain signal $U(t)$ represented as four sets of intervals¹ along the time axis. Thus, if $u(t)$ is a logic signal that belongs to the family $U(t)$, i.e., $u(t) \in U(t)$, then $u(t)$ will be *low* up to t_1 , will switch from *low* to *high* sometime between t_1 and t_2 , will then be *high* up to t_3 , etc. Thus, at any time between t_1 and t_2 , the signal can be either *high* or *low* and may switch any number of times between these values, but will be *high* at t_2 . Notice that between t_6 and t_7 the signal may make *any* number of *low* to *high* and/or *high* to *low* transitions. At the primary inputs, signals are represented by such waveforms with a single point of *possible* transition at time 0. As internal signals are generated, the number of possible transition points

¹One set of intervals for each *low*, *high*, *hl* and *lh* excitations.

increases. In order to contain the complexity, we then start to merge neighboring transition points into intervals. In general, this strategy can be stated as follows: when the number of intervals associated with a gate corresponding to any excitation exceeds a certain user-specified threshold (**Max_No_Hops**), we repeatedly merge closest-neighbor intervals, so as to keep their count below the threshold.

4.3.2 Independence Assumption

While propagating information at a logic gate, we know the uncertainty waveforms at each of its inputs and we would like to derive the corresponding waveform at its output. However, one cannot do this accurately without knowing how some of these inputs, if any, are *correlated*. For instance, certain *combinations* of the gate input excitations may not be possible. Unfortunately, maintaining information about correlation between various circuit nodes is very expensive. We, therefore, use a *conservative* approximation, one that does *not*

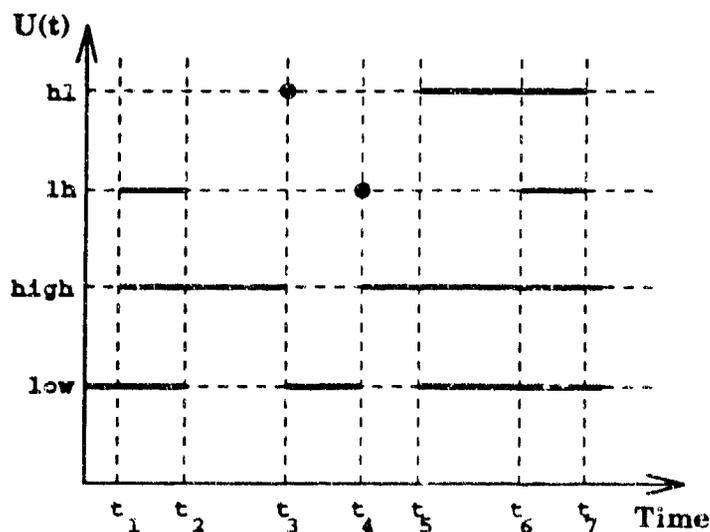


Figure 4.4 An illustration of uncertainty waveform at a gate output.

underestimate the MEC waveforms, as follows. If we assume that *all* combinations of the gate input excitations are possible, i.e., the gate inputs are *independent*, then the worst case current in that case *will be* an upper-bound on the gate current for the case when the inputs are dependent. In other words, the worst case current over *all* combinations of inputs is certainly an upper bound on the worst case current over *some*.

4.3.3 Single Gate Simulation

Given the type of a Boolean gate and the independence assumption for the uncertainty waveforms at its inputs, we now describe how the uncertainty waveform at the output of the gate is calculated. This process is divided into the following two parts:

1. Calculation of the uncertainty set at the output of the gate at a time t .
2. Calculation of uncertainty intervals at the output of the gate.

4.3.3.1 Calculating Uncertainty Sets

One can calculate all possible excitations at the output of the gate at time t from the excitations present at the inputs at time $t - D$, where D is the delay of the gate. Let us denote the uncertainty set at the i^{th} input of the gate at time $t - D$ by X_i . Let us further suppose that the gate has m inputs. Then the set of all possible input patterns that lead to an excitation at the output of the gate at time t can be represented by $\{(x_1, x_2, \dots, x_m) \mid x_i \in X_i, 1 \leq i \leq m\}$. For each input pattern, the output of the gate can be easily determined from the Boolean equation of the gate. Thus, by calculating the output of the gate corresponding at each input pattern, the uncertainty set at the output of the gate at time t can be determined. This process would require one to generate and evaluate $|X_1| |X_2| \dots |X_m|$ input patterns. This worst case complexity can be greatly reduced by the

following observations.

1. The above input pattern generation and evaluation process can be stopped when the uncertainty set at the output of the gate becomes equal to \mathbf{X} . Obviously, trying out any more input patterns would not lead to any improvement in the uncertainty set at the output of the gate.
2. An input to a gate is *completely ambiguous* at time t if its uncertainty set at time t is \mathbf{X} . If all the inputs to the gate are completely ambiguous at time $t - D$, then the output of the gate is also completely ambiguous at time t .
3. All the gates in a circuit can be divided into the following two categories.
 - (a) Gates whose outputs depend not only on the excitations present on its inputs but also on the total count of input lines, e.g., XOR, XNOR etc., gates.
 - (b) Gates whose outputs do not depend on the count of the inputs. The outputs of such gates depend only on the specific excitations present on the inputs, e.g., NAND, NOR etc., gates.

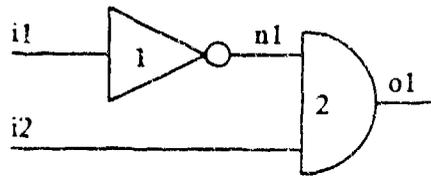
For all the gates which belong to the second category above, all the input lines which have the same uncertainty sets can be merged into a single line. This observation leads to a reduction in the number of input lines and thus the number and size of the input patterns.

All of these observations lead to tremendous savings in the calculation of uncertainty sets at the output of the gate and thus contribute to the speed of the algorithm.

4.3.3.2 Calculating Uncertainty Intervals

In iMax, since signals are represented in the form of uncertainty intervals at the inputs of the gate, therefore, the output of the gate would also be in the form of uncertainty intervals. An interval at the output of the gate could begin or end at time t only if an interval begins or ends at any of the inputs at time $t - D$. Between the time at inputs when an interval begins or ends, and the next interval begins or ends, the sets of excitations that the inputs can assume do not change and therefore no corresponding uncertainty interval can begin or end at the output during that time shifted by D . Based upon these observations, the uncertainty intervals at the output of the gate can be easily calculated.

An example illustrating how uncertainty waveforms at various circuit nodes are calculated is shown in Fig. 4.5.



Input Description : $i1, i2 \in \{l, h, hl, lh\}$ at time 0.

Uncertainty Intervals :

$i1, i2$: $lh[0, 0], hl[0, 0], l[0, \infty), h[0, \infty)$

$n1$: $lh[1, 1], hl[1, 1], l[0, \infty), h[0, \infty)$

$o1$: $lh[2, 2][3, 3], hl[2, 2][3, 3], l[0, \infty), h[0, \infty)$

if **MAX_NO_HOPS** = 1 then

$o1$: $lh[2, 3], hl[2, 3], l[0, \infty), h[0, \infty)$

Key : Excitation[Interval Begin, Interval End]

Figure 4.5 An example of uncertainty waveforms calculation.



Figure 4.6 Current waveform due to an uncertainty interval.

4.3.4 Current Calculation

After the uncertainty waveform at the output of a gate is known, its current contribution is calculated next. Since the output of the gate could switch at any time during an uncertainty interval, therefore, a triangular pulse of current could be drawn at any time during that interval (shifted backwards by the delay of the gate) from the P&G lines due to these transitions, as shown in Fig. 4.6. Hence, by taking an envelope of all possible triangular current pulses, we get the worst case current contribution of a gate for an uncertainty interval. At every gate, there are two types of uncertainty intervals that result in some switching activity at the output and therefore, there are two possible current waveforms, one due to the *hl* uncertainty intervals, called **hlCurrent** and the other due to *lh* uncertainty intervals, called **lhCurrent**. Since at any time, the output of the gate could switch either from *high* to *low* or from *low* to *high*, therefore, by taking an envelope of the **hlCurrent** and **lhCurrent** waveforms, we get the maximum current contribution of the gate. Once all the gate currents are calculated, the current waveforms at the contact points are calculated by combining the individual currents of those gates that are tied to it.

4.3.5 Implementation Details

The above approach has been implemented in a program in C. In the program, the circuit is first *levelized* so that the output of a gate at level j does not feed any other gate at a level less than or equal to j . Any user-specified restrictions on certain inputs are then imposed, while all other inputs are assumed to take all possible excitations from the set X . After this, the circuit is analyzed in a level by level fashion, starting from the lowest level, by propagating the uncertainty waveforms at the inputs of every gate to its output. As a result, we get a current waveform that is a point-wise upper bound on the **MEC** waveform at every contact point.

In order to assess the quality of the solution at every contact point, we need to determine, how close the upper bound waveforms obtained from **iMax** are to the exact **MEC** waveforms. One way of doing this would be to perform an exhaustive enumeration over all possible input patterns and actually calculate the **MEC** waveforms at every contact point. However, this would be very expensive and practically impossible for circuits with more than about 10 inputs (Note: $4^{10} = 1,048,576$). We, therefore, resort to the following random optimization approach. We repeatedly apply different input patterns, randomly selected from the set of all possible patterns, to the circuit. Then we use a logic simulator to calculate the outputs of various gates. From these gate outputs, the P&G current waveforms at every contact point are easily calculated. At every contact point, by maintaining an upper-bound envelope of the current waveforms obtained for different input patterns, we basically get a lower-bound on the **MEC** waveform. Naturally, the more patterns are simulated the closer this waveform will get to the **MEC** waveform. Ideally, one would like to see the upper-bound obtained from **iMax** come as close to this lower-bound as possible. The program that implements this random

optimization technique is called **iLogSim** (Current Logic Simulator).

In our experiments, **iLogSim** calculates the lower bound by trying out several thousand randomly generated input patterns. However, such a technique does not make any use of the current waveform of a currently applied input pattern in generating the next (hopefully better) input pattern. By making use of this information, we can generate better input patterns and thus avoid wasting time in randomly enumerating the input space. We have experimented with one such iterative optimization scheme, namely the *simulated annealing* (SA) algorithm [41], [42]. However, simulated annealing algorithm needs an objective function to indicate how well the search is progressing. We have supplied the peak of the current waveform as the objective function to the algorithm. Therefore, in our experimental results, we compare the peak of the current waveform obtained from **iMax** to that obtained from SA. If one is interested in a comparison of the waveforms at different or several places (and not just the peaks), the lower bound waveform should be optimized at several places by the SA algorithm, or the **iLogSim** program should be run for a large number of input patterns.

4.3.6 Experimental Results

In all the circuit examples considered below, two assumptions are made. First, a fixed number is assigned to each gate as its delay value. This delay value can be different for different gates. Second, the peak of the transition current for every gate for both *lh* and *hl* transitions is taken to be 2 units of current. The results of simulated annealing were collected after trying about 100,000 input patterns for each circuit. Without any detailed layout information, we have assumed that all the gates in a circuit are connected to a single contact point, and we report the peak values of the current waveforms obtained from **iMax** and SA.

Table 4.1 lists the results of running **iMax** and SA algorithms on eight small CMOS circuits. The number (10) next to **iMax** in the table indicates the value of **Max_No_Hops** parameter. In most of the cases, the results of **iMax** are in perfect agreement with the SA results.

Circuit	No. Gates	No. Inputs	iMax10	SA	Ratio
BCD Decoder	18	4	32.41	32.41	1.00
Comparator A	31	11	54.47	54.47	1.00
Decoder	16	6	26.75	26.75	1.00
P. Decoder A	29	9	44.11	44.11	1.00
P. Decoder B	31	9	49.35	49.35	1.00
Full Adder	36	9	58.00	55.44	1.05
Parity	46	9	47.57	47.57	1.00
Alu (SN74181)	63	14	78.89	70.87	1.11

In Table 4.2, we report similar results on the ten ISCAS-85 benchmark circuits [6]. We observe that for all the circuits, the linear time **iMax** algorithm took only a few seconds of cpu time on a sun SPARCstation ELC compared to several hours of time needed by the SA algorithm (typical times needed for trying 10,000 patterns by SA are shown in the table). Furthermore, for most of these circuits, the ratio of the upper bound obtained from **iMax** to the lower bound obtained from SA is less than 1.57. There are two possible reasons for this mismatch. Firstly, it is quite possible that the lower bound obtained from SA is not very close to the **MEC** waveform. Since all the circuits have at least 32 inputs, the space of possible input patterns is huge, and the lower bound waveform obtained after trying about 100,000 input patterns may not be very close to the **MEC**. For smaller circuits (Table 4.1) where the input space is not so huge, 100,000 input patterns were enough to get a lower bound estimate of **MEC** waveform that is quite close to the actual waveform, as is reflected by the results.

The second possible source of mismatch is our conservative independence assumption for the signals at the gate inputs. One can improve on this assumption by attempting to resolve the signal correlations, as discussed in the following sections.

Circuit	No. Gates	No. Inputs	Peak Currents			CPU Times	
			iMax10	SA	Ratio	iMax10	SA(10k)
c432	160	36	181.9	162.0	1.12	1.2s	9m 40s
c499	202	41	247.9	186.6	1.33	1.2s	10m 46s
c880	383	60	418.5	321.4	1.30	3.0s	26m 32s
c1355	546	41	633.8	415.8	1.52	4.6s	36m 18s
c1908	880	33	732.1	445.6	1.64	7.6s	1h 34m
c2670	1193	233	1169.2	866.1	1.35	7.9s	2h 14m
c3540	1669	50	1740.4	866.1	2.01	12.7s	3h 39m
c5315	2307	178	2312.6	1558.9	1.48	16.0s	4h 40m
c6288	2406	32	4096.2	3193.2	1.28	37.8s	61h 58m
c7552	3512	207	4339.7	2768.6	1.57	28.4s	7h 28m

We next discuss the effect of varying the **Max_No_Hops** parameter on the performance of **iMax**. Table 4.3 lists the peak values of the upper bound waveforms for ISCAS-85 circuits for different values of **Max_No_Hops**. In parentheses, we also tabulate the cpu times (in sec.) needed by the algorithm. As the value of **Max_No_Hops** increases, the number of intervals being merged at every gate decreases and therefore, the cpu time needed by the algorithm increases. This also improves the peak value of the current waveform, as shown in the table. However, with an increase in **Max_No_Hops** parameter, while the cpu time continues to increase, the improvement in peak value is not significant beyond **Max_No_Hops** = 10. Similar behavior is observed for the entire current waveform and not just for the peak value. In Fig. 4.7, we plot the upper bound waveforms for c1908 for three different values of **Max_No_Hops**. The difference between the upper bound waveforms for **iMax10** and **iMax ∞**

is almost negligible. Similar plots are obtained for all other circuits. Therefore, we conclude that a value between 5 and 10 seems to be a good choice for **Max_No_Hops** parameter.

Table 4.3 iMax results vs. Max_No_Hops				
Circuit	iMax: Max_No_Hops			
	1	5	10	∞
c432	236.3 (0.4)	185.1 (0.9)	181.9 (1.2)	181.7 (2.1)
c499	292.1 (0.4)	247.9 (0.8)	247.9 (1.2)	247.9 (1.3)
c880	550.6 (0.8)	424.1 (2.0)	418.5 (3.0)	415.1 (11.8)
c1355	749.9 (1.3)	646.8 (2.7)	633.8 (4.6)	633.8 (19.5)
c1908	908.3 (2.0)	740.5 (5.0)	732.1 (7.6)	724.9 (86.3)
c2670	1476.0 (2.3)	1188.6 (5.4)	1169.2 (7.9)	1166.4 (21.7)
c3540	2088.2 (3.7)	1752.4 (8.2)	1740.4 (12.7)	1730.3 (170.0)
c5315	2632.0 (5.1)	236.5 (11.2)	2312.6 (16.0)	2309.0 (109.5)
c6288	4134.8 (10.4)	4098.4 (20.7)	4096.2 (37.8)	4096.1 (7086.0)
c7552	5163.1 (8.0)	4401.6 (20.0)	4339.7 (28.4)	4325.8 (177.8)

4.4 The Signal Correlation Problem

In general, signals at internal nodes of a circuit are *correlated*. This limits the number of transitions that can possibly occur at the outputs of the gates, an effect that is ignored by the

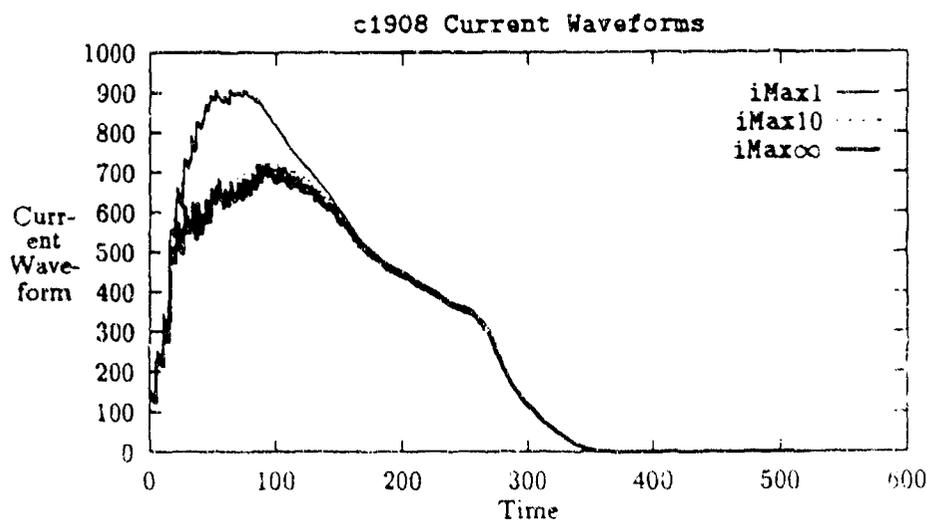


Figure 4.7 iMax current waveforms for different values of **Max_No_Hops** parameter.

iMax algorithm. Two examples of how signal correlation limits the number of transitions are illustrated in Fig. 4.8.

In Fig. 4.8(a), signal lines x_1 and x_2 are correlated (in this case, they carry the same signal). Depending upon the specific excitation present at x , only one of the two gates can switch at a time. However, since **iMax** ignores the signal correlation present between x_1 and x_2 , it calculates the uncertainty sets at the outputs of the two gates as shown in the figure and thus erroneously concludes that both gates may switch at the same time, and therefore adds two triangular current pulses due to both gates switching simultaneously to the overall current waveform. It is this kind of approximation that contributes to a loose **iMax** upper bound. Similarly, in Fig. 4.8(b), the output of the inverter is correlated with its input and therefore, the NAND gate may never switch. However, ignoring this correlation, **iMax** concludes that the NAND gate *can* switch.

As is clear from these examples, the source of the signal correlation problem, in general, is a gate (or input) whose output *fans out* to several other gates. Such gates are called *multiple*

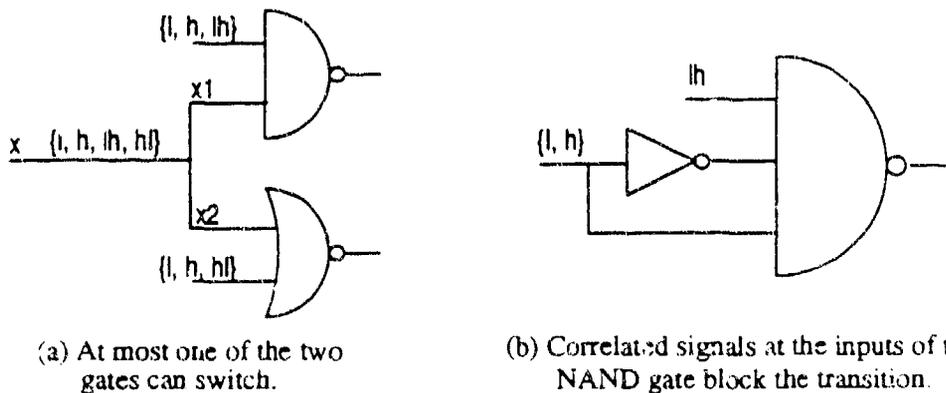


Figure 4.8 Two examples to illustrate the signal correlation problem.

fan-out (MFO) gates. The general situation is shown in Fig. 4.9, where a MFO gate G with output node n fans out to nodes n_1, n_2, \dots, n_k that in turn feed gates G_1, G_2, \dots, G_k . In this figure, inputs to the gates G_1, G_2, \dots, G_k (which are n_1, n_2, \dots, n_k respectively) are correlated. Due to this correlation, even though the output of each gate (G_1, G_2, \dots, G_k) can assume all possible excitations as calculated by **iMax**, they may not *simultaneously* carry their worst case excitations. As one goes deeper into the circuit, where these correlated outputs reconverge and feed the same gate, the inputs of that gate become correlated (e.g., NAND gate in Fig. 4.8(b)). Such gates are called *reconvergent fan-out (RFO) gates*. With correlated signals at the inputs of a gate, the number of transitions that can possibly occur at its output is reduced. The signal correlations considered above, which exist among various nodes throughout the circuit, are called *spatial correlations*.

Besides the spatial correlations, there is another set of correlations that the **iMax** algorithm completely ignores. The excitation assumed by a node at time t , restricts the set of pos-

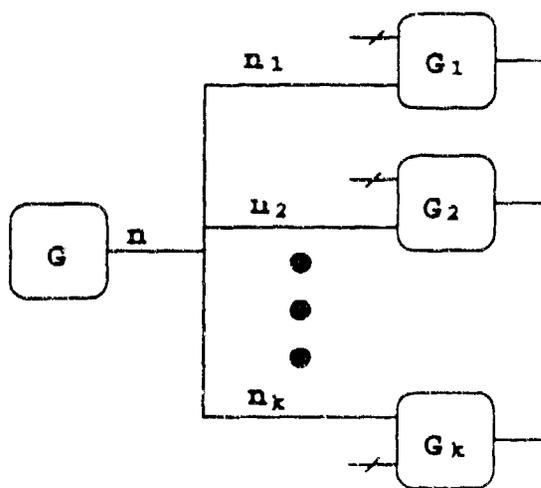


Figure 4.9 An example of a Multiple Fan-out Gate.

sible excitations that the node can assume at an earlier or a later time. For example, if a node is low at time t , then it can either stay at low or switch from high to low at time t^- and it can either stay at low or switch from low to high at time t^+ . These correlations which exist in the time domain are called *temporal correlations*.

The **iMax** algorithm completely ignores all spatial and temporal signal correlations and, therefore, overestimates the supply currents. The advantage of ignoring correlations in the algorithm is its, very desirable, linear time performance.

4.5 Resolving Signal Correlations

The upper bound produced by the **iMax** algorithm can be made exact by doing a brute-force enumeration at the inputs of the circuit and calculating an envelope of the current waveforms produced. In enumeration, since unambiguous input patterns are applied to the circuit, there is no "uncertainty" present at the inputs and therefore, signal correlations do not become an issue. In a similar fashion one can improve the results of the **iMax** algorithm by doing a *partial enumeration* at a few selected nodes in the circuit.

An example of how partial enumeration helps improve the upper bound can be seen from Fig. 4.8(a). In this circuit with no enumeration, **iMax** would assume that the signal lines x_1 and x_2 are mutually independent and therefore infer that both NAND and NOR gates can switch at the same time. However, if we do partial enumeration at signal line x , then we would generate four cases corresponding to when $x = l$, $x = h$, $x = hl$ and $x = lh$. When $x = l$ or hl , only the NOR gate switches. Similarly, when $x = h$ or lh , only the NAND gate switches. Thus, by splitting the problem into four sub-problems, we have improved our result, i.e., found that only one of the two gates may switch at any given time.

While enumerating a node, we only need to process a small subset of the gates. We define the *Cone of Influence*, $COIN(n)$, of a node n as the set of all the gates that can possibly be affected by a change in excitation at the node. Thus, a gate is in the $COIN(n)$ of a node n if it is either directly fed by n or is connected to the output of a gate that is in $COIN(n)$. While enumerating a node, we only need to consider those gates that are in its $COIN$.

One technique to partially enumerate the internal nodes of a circuit, called *Multi-Cone Analysis (MCA)*, was reported in [43]. The motivation behind such an approach was to be able to enumerate at the outputs of the MFO gates, which are the sources of the signal correlation problem. However, from the results in [43] and Tables 4.6, 4.7 below, it can be seen that the **MCA** approach offers only a modest improvement in the upper bound. There are several reasons for this.

As shown in Table 4.4, there are usually several MFO gates/inputs in a circuit and all of these nodes should be enumerated to properly resolve the signal correlation problem. From our experience with ISCAS-85 benchmark circuits, we have found that the **COINs** of several of these nodes overlap and therefore, to properly handle signal correlations, these nodes should be enumerated *simultaneously*. Furthermore, because of the presence of glitches in a circuit, signals at internal nodes span several time points (i.e., signal transitions occur at several time points). To take care of the temporal correlation problem, the node should be enumerated at each of these time points. Simultaneous enumeration is an extremely expensive process specially when there are several nodes and each node needs to be enumerated at several time points. For example, to enumerate two nodes simultaneously, the cpu time needed to enumerate each node gets multiplied. To avoid this multiplicative growth of cpu

time, we made several simplifying assumptions in the implementation of the **MCA** algorithm [43]. Because of these simplifications, the algorithm led to only mild improvement in **iMax** results.

Table 4.4 Number of MFO gates/inputs in ISCAS-85 circuits.

Circuit	No. Inputs	No. MFO	Circuit	No. Inputs	No. MFO
c432	36	153	c2670	233	1129
c499	41	170	c3540	50	1647
c880	60	357	c5315	178	2184
c1355	41	514	c6288	32	2384
c1908	33	855	c7552	207	3405

There are usually several RFO gates in a circuit. If at the output of a RFO gate, a false transition is predicted by **iMax** (such as in Fig. 4.8(b)), then as this transition propagates through the circuit to the output nodes, it causes several other false transitions in the circuit along its way. Therefore, locating and suppressing such transitions in **iMax** is important. However, these false transitions can be isolated only by doing a simultaneous enumeration at the source MFO gate(s) that created them. In effect, one needs to construct the *supergate* [16] for each RFO node in the circuit and for each supergate, do a simultaneous enumeration at its MFO inputs. However, these supergates can be as big as the entire circuit and therefore enumerating their inputs becomes intractable. We have implemented an approximate algorithm based on enumerating *primary stem regions* [45], [46]. Our results show only modest improvement in the upper bound waveforms.

From our experience with enumerating internal nodes of a circuit, as explained above, we infer that improving **iMax** upper bound waveforms by enumerating internal nodes is very expensive and does not offer a practical solution for VLSI circuits. In the next section, we present an alternative partial input enumeration approach that significantly improves the **iMax**

results and represents a good speed-accuracy trade-off.

4.6 Partial Input Enumeration (PIE)

As shown in Table 4.4, there are usually many more MFO nodes than primary inputs in a circuit. Secondly, as stated in Section 4.2, all the inputs to a circuit switch at most once at time zero. Therefore, there is only one time point at which a primary input needs to be enumerated. This is in contrast to an internal circuit node which usually needs to be enumerated at several time points. These observations, combined with the fact that **iMax** is an extremely fast algorithm led us to explore the following *partial input enumeration (PIE)* algorithm to improve the **iMax** upper bound at every contact point.

4.6.1 The Algorithm

Let x_1, x_2, \dots, x_N be the N primary inputs of a circuit under consideration. Let X_i represent the uncertainty set for input x_i at time zero. The *input search space* for the circuit consists of the set of all valid input patterns that can be applied to the circuit. Mathematically, the input search space is $\{(e_1, e_2, \dots, e_N) \mid e_1 \in X_1, e_2 \in X_2, \dots, e_N \in X_N\}$. For brevity, we denote this by (X_1, X_2, \dots, X_N) . Suppose, for the purposes of this illustration, for a particular input x_i , $X_i = \mathbf{X}$. Then the input search space (X_1, X_2, \dots, X_N) for the circuit can be divided into four disjoint parts, namely $(X_1, X_2, \dots, \{l\}, \dots, X_N)$, $(X_1, X_2, \dots, \{h\}, \dots, X_N)$, $(X_1, X_2, \dots, \{hl\}, \dots, X_N)$ and $(X_1, X_2, \dots, \{lh\}, \dots, X_N)$. We can compute the maximum current waveforms (at every contact point) for each of these four parts by running the **iMax** algorithm and in each case, restricting the excitation on input x_i to the value in its respective uncertainty subset. Since the four parts combined together constitute the complete search space, by taking an upper bound envelope of the four current waveforms at every contact point, we can still

guarantee an upper bound on the respective MEC waveforms. Since, in each of the four runs of iMax, specific excitations are present at input x_i , signal correlations due to x_i disappear and the resulting current waveform should be an improvement on the original upper bound. In a similar fashion, the upper bounds for the individual subcases can be improved.

The set of inputs selected for enumeration has a direct influence on the quality as well as the cost of the solution obtained. If all the inputs are selected and enumerated then the upper bound obtained at every contact point would be exact. However, doing this is practically impossible for most circuits. The extent to which an input contributes to signal correlations inside a circuit is different for different inputs. For example, in Fig. 4.8(a), enumerating input x is more beneficial than enumerating any of the other two inputs. Hence, by selecting and enumerating inputs in an intelligent fashion, we can significantly improve the iMax upper bounds, without spending too much cpu time.

We have developed an intelligent *best first search* (BFS) algorithm [47] that is very effective in selecting and enumerating inputs and thereby improving the upper bound at every contact point. The algorithm starts with the iMax upper bound and some known lower bound and repeatedly expands "*search nodes*" (**s_nodes**) in a best first fashion. Various **s_nodes**, generated during the search, correspond to partially specified input states in which some inputs have specified excitations (e.g., say *low*), while others have "uncertain" (or unspecified) excitations. An *objective function* is associated with the search and during the search, **s_nodes** which correspond to the best objective value are repeatedly expanded. This function will be explained later in this section. Because of this best first strategy, there is a gradual reduction in the value (at any time) of the upper bound waveform at every contact point in the circuit. This *iterative improvement* is a very important feature of the algorithm for large

circuits where an exhaustive exploration of the input space is practically impossible. The BFS algorithm can be stopped at any intermediate stage and the current best upper bounds at various contact points can still be reported.

The BFS search starts with the initial uncertain state i.e., $s_node = (X_1, X_2, \dots, X_N)$ and a known lower bound, which is the objective value for some input pattern. During the search, a s_node with the highest objective value is repeatedly selected and its descendent s_nodes are generated by enumerating an input, as explained in the following outline:

Remark: *List* is an ordered list of s_nodes , arranged in decreasing **objective** values.

1. *List* \leftarrow starting s_node (Initial uncertain state).

Upper Bound \leftarrow objective value of the starting s_node .

Lower Bound \leftarrow objective value for a specific input pattern. (otherwise 0.0).

2. While **Stopping Criterion** is not satisfied, do

2.1 Remove the top s_node from the *List*.

2.2 Calculate next input number to enumerate from the **Splitting Criterion**.

2.3 Generate all (≤ 4) children s_nodes by enumerating the input and calculate their **objective** values.

2.4 If these children are leaf s_nodes , then update the **Lower Bound**, else, insert them in *List*, after pruning if any.

2.5 **Upper Bound** \leftarrow objective value of the top s_node in *List*.

3. Report the best upper bound current waveform at every contact point. **STOP**.

where a leaf s_node is one which has exactly one excitation associated with each of its uncertainty sets (i.e., it corresponds to an input pattern). The following functions are used in the

outline above.

Objective Function: This function specifies the quantity that is being minimized during the search. There are usually several contact points within a combinational block and we would like to minimize the (**iMax**) upper bound waveform at each of these points, so that they are close to their respective **MEC** waveforms. One possibility is to minimize the peak of a weighted sum of the upper bound waveforms, where these weights are determined depending upon how much "influence" the contact point has on the overall voltage drops. We are currently working on this problem. In the experiments reported below, we have assumed these weights to be unity and we minimize the peak value of the sum of all the upper bound waveforms. This corresponds to minimizing the worst case total current of the combinational block.

Stopping Criterion: We stop the search if any of the following two conditions is satisfied.

- a. **Best Upper Bound** \leq **Lower Bound** \times **ETF**.
- b. **Number of s_nodes generated** \geq **User specified parameter (Max_No_Nodes)**.

The *Error Tolerance Factor (ETF)* is a user-specified parameter that provides control over the final desired accuracy of the algorithm. The value of this parameter is always bigger than 1. The first condition above specifies that when the highest upper bound value is within **ETF** factor of some known lower bound, then the search can be terminated. In large circuits where calculating an exact upper bound value by running the search to completion is extremely expensive, and an overestimation by 20% to 30% may be acceptable, such a parameter can be extremely useful. The second condition puts a hard limit (**Max_No_Nodes**) on

the number of **s_node**s that are to be generated in the search.

Pruning Criterion: During the search, if we come across a **s_node** for which the upper bound satisfies the following condition:

$$\text{Upper Bound} \leq \text{Lower Bound} \times \text{ETF}$$

then, such a **s_node** can be deleted from the search as its upper bound value is already acceptable. This pruning criterion deletes several unnecessary **s_nodes** during the search and thus keeps the memory usage down.

Splitting Criterion (SC): This criterion specifies the input which should be enumerated next from any **s_node** during the search. In the next subsection, we describe two heuristic functions for doing this.

The BFS algorithm always processes **s_nodes** which are on its current *wavefront*, see Fig. 4.10. At the start, this wavefront consists of only one **s_node**, namely the initial uncertain state. As the search progresses, this wavefront moves forward through the input search space. An input pattern leading to the maximum objective value could belong to any of the **s_nodes** on the wavefront. Therefore, when the algorithm terminates, at every contact point, we compute an envelope of the current waveforms of all the **s_nodes** on the wavefront and report that as an upper bound waveform.

4.6.2 Heuristic SC and Experimental Results

We now describe two heuristics for the splitting criterion that have shown good results in practice. The first heuristic selects an input which has the highest sensitivity while the second one selects an input based upon the *influence* it has inside the circuit.

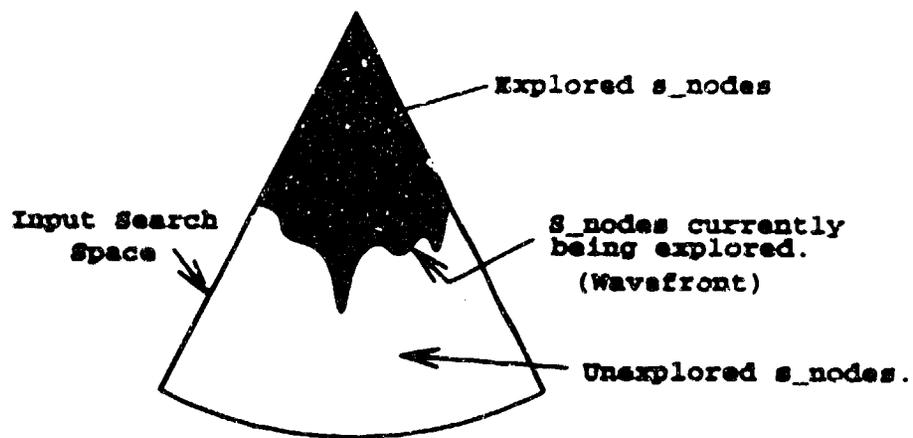


Figure 4.10 Exploring s_nodes through the input search space.

4.6.2.1 H_1 Heuristic

Let us suppose that during the search, we are at a particular s_node n and we select an input x_i for enumeration. If we assume that the uncertainty set for x_i at time zero is X , then by enumerating x_i , we would generate four children s_nodes , as shown in Fig. 4.11. We assume that the objective value of s_node n is denoted by obj_n and the objective values of the children s_nodes are denoted by obj_l, obj_h, obj_{hl} and obj_{lh} . If

$$\Delta obj_i = obj_n - \max\{obj_l, obj_h, obj_{hl}, obj_{lh}\},$$

then by enumerating x_i , we can improve the objective value of s_node n by an amount Δobj_i . We can repeat this calculation for every input and then select an input which gives rise to the maximum improvement in the objective value.

However, if Δobj_i is zero for all the inputs, which happens very often in practice, then the above selection process would not work well. For a specific input x_i , $\Delta obj_i = 0$ means that the objective value of one of its children s_nodes is equal to obj_n . However, for the

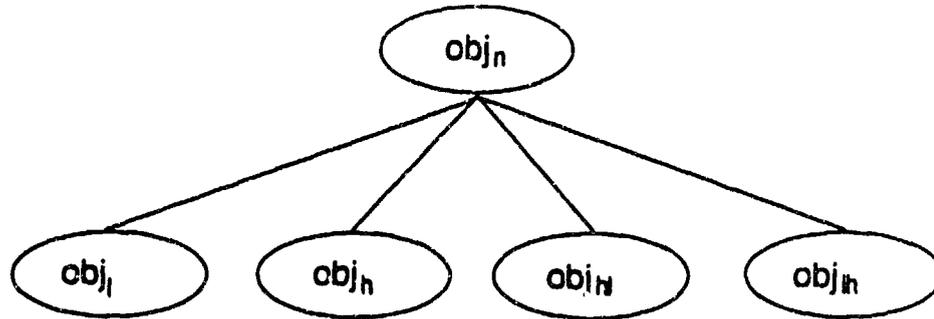


Figure 4.11 An example to illustrate the H_1 .

remaining children s_nodes , the objective value may not be equal to obj_n and this information can be used in assigning credit (or relative importance) to x_i . Based on these observations, we have come up with the following H_1 heuristic function corresponding to an input x_i :

$$H_1 = A \times (obj_n - obj_1) + B \times (obj_n - obj_2) + C \times (obj_n - obj_3) + (obj_n - obj_4)$$

where obj_1 , obj_2 , obj_3 and obj_4 are the objective values of the children s_nodes , generated by enumerating x_i , arranged in decreasing order and A , B and C are three constants such that $A \gg B \gg C \gg 1$. At any s_node during the search, we compute the heuristic values for every input and select an input with the maximum associated heuristic value. This splitting criterion is called *dynamic (H_1) splitting criterion because at each s_node , it calculates the heuristic value for every input and then selects the best input for enumeration.*

The results of partial input enumeration using the BFS algorithm and using the dynamic (H_1) splitting criterion for the nine small circuits are documented in Table 4.5. For all the circuits, the algorithm was run to completion, i.e., till the upper bound became equal to the

lower bound ($ETF = 1$). The results clearly show that the **PIE** algorithm is very efficient in scanning the input space. As an example, the last circuit in the table (Alu) has 14 inputs and therefore, the number of possible input patterns for this circuit is $4^{14} = 268,435,456$. The **PIE** algorithm was able to scan the entire search space after generating just 233 **s_nodes**. This also shows that the upper bound produced by the **iMax** algorithm is very tight for these circuits.

Circuit	Dynamic (H_1) Splitting Criterion			Static (H_1) Splitting Criterion		
	No. S_nodes Generated	iMax runs in SC	Time	No. S_nodes Generated	iMax runs in SC	Time
BCD Decoder	17	40	1.9s	17	17	1.2s
Comparator A	109	664	87.7s	277	45	35.1s
Comparator B	45	264	30.6s	45	45	9.7s
Decoder	25	84	3.6s	25	25	1.8s
P. Decoder A	37	180	11.4s	37	37	3.9s
P. Decoder B	37	180	11.7s	37	37	4.2s
Full Adder	53	296	29.2s	53	37	7.7s
Parity	37	180	18.7s	37	37	6.4s
Alu (SN74181)	233	1952	378.2s	2525	57	352.7s

As can be seen from Table 4.5, the number of **iMax** runs needed in the dynamic splitting criterion far exceeds the number of **s_nodes** generated. At a **s_node**, to calculate the H_1 value for a particular input x_i , we need to run the **iMax** algorithm $|X_i|$ times. If a **s_node** has k inputs which are possible candidates for enumeration (i.e., their $|X_i| > 1$), then we need to run the **iMax** algorithm $\sum_{i=1}^k |X_i|$ number of times to find the best input to enumerate next. For bigger circuits, with large number of inputs, this time will be even more dominant rendering the **PIE** algorithm prohibitively expensive. Therefore, we have experimented with other less expensive alternatives.

Instead of calculating the heuristic function value (H_1) for every input at every `s_node` during the search, we calculate the heuristic value for every input at the beginning of the search. All the inputs are arranged in the decreasing order of their heuristic values. During the BFS search, inputs are selected in this fixed order. This criterion is called *static (H_1) splitting criterion*. The amount of time spent in the static splitting criterion is fixed and is equal to $\sum_{i=1}^N |X_i|$ runs of the `iMax` algorithm for a circuit with N inputs. The results of the `PIE` algorithm using the static (H_1) splitting criterion are also summarized in Table 4.5. With static splitting criterion, the number of runs of the `iMax` algorithm needed in the splitting criterion goes down, but the number of `s_nodes` generated during the search goes up for some circuits. However, for all the circuits, we observe an overall reduction in the cpu times needed for the algorithm to complete.

4.6.2.2 H_2 Heuristic

The number of gates that are affected by a change in excitation at an input is a good heuristic measure of how much influence the input has on the upper bound waveforms. Therefore, inputs which affect more number of gates (i.e., which have larger `COINs`) should be enumerated before others. This leads us to another (static) splitting criterion H_2 , in which we calculate the size of the `COIN` i associated with each input x_i . As with H_1 , all the inputs are arranged in the decreasing order of H_2 values and during the search, inputs are selected in this fixed order. We will show in the next section that, while both static H_1 and H_2 give good results in practice, H_2 is much better in terms of speed and has accuracy comparable to H_1 .

The results of partial input enumeration using both H_1 and H_2 static splitting criteria for the `ISCAS-85` benchmark circuits are shown in Table 4.6. In the table, under various `iMax`,

MCA and BFS columns, we show the ratio of the respective upper bound to the lower bound obtained from simulated annealing. The numbers in parentheses under the BFS columns indicate the number of *s_nodes* that were generated before stopping the search (i.e., the *Max_No_Nodes* parameter; 1k stands for 1000). Total cpu times needed by the algorithm on a sun SPARC station ELC (with *Max_No_Nodes*=100) are also shown in the table. From Table 4.6, we note that for all the circuits, the ratio of the upper bound to the lower bound is at most 1.52 (as opposed to a worst case of 2.02 for the simple *iMax* algorithm). This ratio can be further improved by running the *PIE* algorithm for longer durations. We emphasize that, since we can only compare the upper bound to a *lower bound*, the numbers in the table are only upper bounds on the error. It is prohibitively expensive to measure the true error.

Circuit	iMax	MCA	Static H_1 SC			Static H_2 SC		
			BFS (100)	BFS (1k)	Time (100)	BFS (100)	BFS (1k)	Time (100)
c432	1.12	1.12	1.08	1.05	5m 14s	1.12	1.12	1m 34s
c499	1.33	1.20	1.33	1.33	4m 40s	1.33	1.33	1m 23s
c880	1.31	1.26	1.25	1.22	17m 16s	1.28	1.26	4m 5s
c1355	1.52	1.52	1.52	1.52	21m 28s	1.52	1.52	6m 13s
c1908	1.64	1.55	1.49	1.46	33m 17s	1.58	1.54	11m 51s
c2670	1.35	1.34	1.29	1.28	1h 57m	1.35	1.35	11m 56s
c3540	2.01	1.95	1.45	1.36	51m 12s	1.59	1.37	17m 3s
c5315	1.48	1.44	1.42	1.40	3h 2m	1.48	1.47	26m 2s
c6288	1.28	1.28	1.28	1.27	2h 5m	1.28	1.28	57m 28s
c7552	1.57	1.55	1.52	1.50	6h 21m	1.53	1.53	45m 4s

While the improvement over the original *iMax* algorithm is not large in all the cases, in those cases where the *iMax* bound was very loose, such as c3540, the new *PIE* algorithm with H_1 or H_2 heuristic gives *significant* improvement: the ratio of 2.02 (maximum over-

estimation by 1.02) is now 1.37 (maximum over-estimation by 0.37) with H_2 , a reduction in the maximum over-estimation by about 64%.

We also emphasize the following attractive property of the algorithm: a significant amount of improvement in the upper bound occurs in the first few s_nodes (about 50-200) of the algorithm. This is shown in Fig. 4.12 for c3540, where the ratio of the upper bound to the lower bound is plotted as a function of cpu time for the first 1000 s_nodes .

The cpu time needed for generating the input list by the H_2 splitting criterion is negligible compared to the time needed by the H_1 criterion. For VLSI circuits with several hundred inputs, where the time needed by the H_1 criterion may be large, H_2 criterion may be used instead. As can be seen from Table 4.6 (also see Table 4.7), the results produced by using either splitting criteria are quite comparable, specially for those circuits where iMax did not produce a good upper bound.

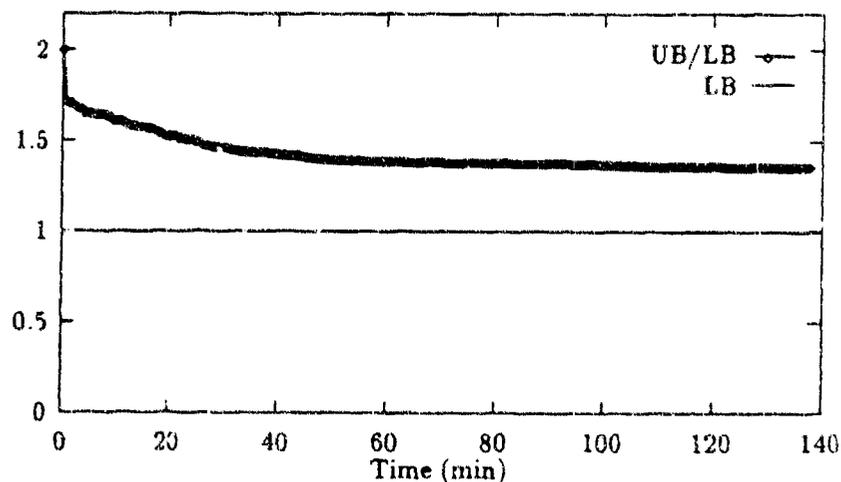


Figure 4.12 'Upper Bound / Lower Bound vs Time' plot for c3540.

Circuit	No. Gates	iMax	MCA	Static H_1 SC			Static H_2 SC		
				BFS (100)	BFS (1k)	Time (100)	BFS (100)	BFS (1k)	Time (100)
s1423	657	1.35	1.32	1.32	1.29	37m 22s	1.35	1.34	7m 43s
s1488	653	2.21	2.10	1.40	1.08	5m 32s	1.41	1.06	2m 49s
s1494	647	2.18	2.08	1.37	1.06	5m 35s	1.39	1.05	2m 51s
s5378	2779	1.38	1.37	1.29	1.25	2h 23m	1.30	1.23	13m 21s
s9234	5597	1.76	1.74	1.51	1.47	7h 24m	1.56	1.56	37m 18s
s13207	7951	1.37	1.35	-	-	-	1.30	1.26	36m 53s
s15850	9772	1.81	1.80	-	-	-	1.64	1.57	1h 11m
s35932	16065	1.66	1.66	-	-	-	1.56	1.56	2h 6m
s38417	22179	1.73	1.70	-	-	-	1.72	1.68	2h 46m
s38584	19253	1.45	1.38	-	-	-	1.39	1.37	2h 15m

In order to demonstrate the applicability of the partial input enumeration algorithm for VLSI circuits with several thousand gates, we have also experimented with the ISCAS-89 benchmark circuits [48]. For these synchronous sequential circuits, we have extracted the combinational blocks by deleting the flip-flops. These combinational blocks have gate counts ranging up to 22,000 and number of inputs ranging up to 1750. The results of the **PIE** algorithm on some of the ISCAS-89 circuits (combinational blocks) using both H_1 and H_2 splitting criteria are summarized in Table 4.7. It is clear from the table that even for circuits of this size, our algorithms show good speed and accuracy performance.

5. BUS MODEL EXTRACTION

5.1 Introduction

In this chapter, we describe the work accomplished on the extraction-bus models from layout information. The aim is to develop fast, accurate, and general techniques for extracting the RC network models of power busses in VLSI chip design. The models are used to estimate electromigration and voltage drop in the busses. The main results accomplished on the extraction subtask this past year, compared to our previous work are: (1) the ability to extract the resistive models of bus partitions that do not fit precharacterized 'standard' partitions; this is done using the Boundary Element Method (BEM); (2) improved modeling of the capacitance by the use of finite element partitioning and model reduction techniques to reduce the complexity of the extracted model without sacrificing accuracy.

5.2 Extraction of Resistance Models

For simplicity of presentation and without loss of generality, we will assume a sheet resistance of $1 \Omega/\text{square}$. The layout is first partitioned into primitive geometrical shapes. The idea behind partitioning is to simplify the computation since, if we ignore capacitance, the sum of resistances of the partitioned pieces will equal the resistance of the unpartitioned piece. The partitioning strategy relies on partitioning a section of the layout along equipotential lines that will occur when a voltage source is applied to one of the terminals and the others are grounded. Usually equipotential lines that are approximately straight are chosen. The distributions of equipotential lines for some common shapes are shown in Fig. 5.1. For two-terminal segments, like those in Fig. 5.1, the equipotential lines will be the same no matter which terminal is the grounded one. This is not true for the four-terminal segment in Fig.

5.2, thus making three or more terminal segments, such as T and + segments, harder to partition along equipotential lines.

After partitioning, a resultant N terminal shape is modeled as an N port resistor network having no internal nodes, where each resistor connects one terminal to another, yielding $N(N-1)/2$ resistors. Conceptually, to compute R_{ij} (the resistance connecting terminal i to each other terminal j), a unit voltage is applied to terminal i and all other terminals are grounded. Let I_j be the current flowing through terminal j. Then $R_{ij} = 1/I_j$.

If partitioning does not follow the equipotential partition constraint, then error is introduced. By applying the general multiterminal model described above to each of the incorrectly

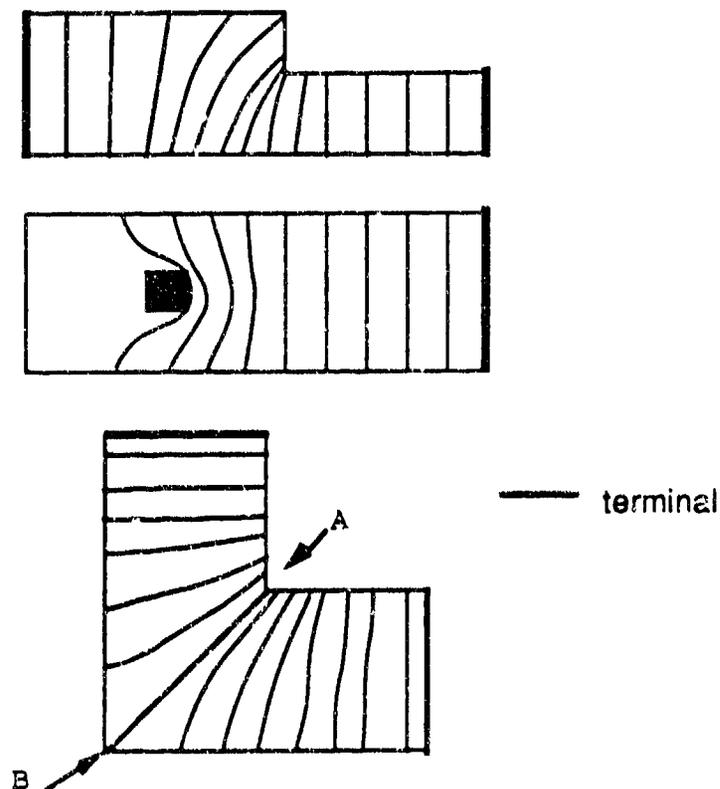


Figure 5.1 Potential distribution in some common two-terminal shapes.

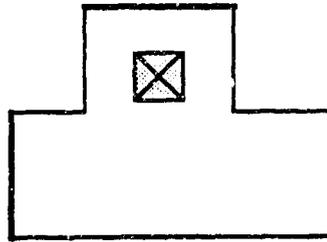


Figure 5.2 Four-terminal resistor.

partitioned pieces, an equipotential line is forced to exist, by virtue of the unity voltage excitation, where it normally does not exist when actual current is flowing. This causes the error. Looking at Fig. 5.3, the arbitrary partition yields a 1% error in the total resistance even if the exact resistance of the individual pieces is known.

However, often a resistance method applied to a large piece that cannot be partitioned anymore yields a larger error in resistance than the same method applied to an incorrect partitioning. Thus relaxing the partitioning constraint (as described in the next section) so as to increase the partitioning of large pieces will increase accuracy. For example, returning to Fig. 5.3, if the BEM used in JET2 is applied to the whole piece, a 4% error occurs. If, however,

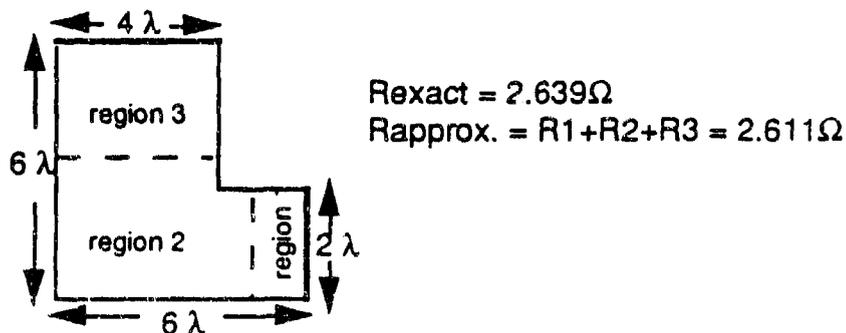


Figure 5.3 Nonequipotential partition.

this method is applied to the arbitrary partitions and their resistances are summed, an error of 2.6% occurs, which is actually an improvement.

Partitioning is even more important for three (or more) terminal shapes where the path of current flow between two terminals is shared by a third terminal. The first two terminals become weakly coupled, corresponding to a large resistance between them. Also if the distance between two terminals is relatively greater than between others (as is commonly the case with multiterminal shapes), weak coupling results. Many numerical methods, including the BEM used in JET2, are inaccurate in estimating the resistance between two weakly coupled points. Each partitioning line in effect introduces an internal node which splits the segment's network model into two networks. Let us call these two new networks A and B, which are effectively decoupled from each other. This eliminates trying to compute the value of the coupling between the two weakly coupled terminals as long as one of the terminals is in A and the other is in B. Also the distance between the new internal node and any terminal in network A is smaller than the distance between any terminal in A and any terminal in B, and vice versa. Thus the coupling between the internal node and any terminal in network A is stronger than the coupling between any terminal in A and any terminal in B, and vice versa. Thus in JET2 we will relax the constraint of partitioning along known equipotential lines, as described in detail at the end of this section.

Previous approaches to extracting the resistance of the bus segments could be classified into table look-up and on-the-fly numerical methods. Table look-up methods are faster but on-the-fly numerical methods are more general. Look-up methods generally pattern match the partitioned segment with the closest case for which it has an analytical formula or table look-up values. In our approach, we use a combination of the two.

Many extractors use table look-up and various heuristics to model bus segments. One early heuristic was to model the corner rectangle of material in an L-bend as $0.55 R_s$ [49], where R_s is the sheet resistance. Thus the L-bend shown in Fig. 5.1 would be modeled as a 2.55Ω resistor. For $w_1 = w_2$, this approximation is fine. But for $w_1 \gg w_2$, the resistance of the corner becomes large. Thus the heuristic breaks down for this case. It also gives no indication on how to model the four-terminal shape in Fig. 5.2. The method in [50] partitions bus segments along equipotential lines and analyzes the resulting expected simple polygons using approximate empirical formulas. The method would still have trouble with the segment in Fig. 5.2, because in the case of multiterminal shapes, it concentrates on the current flow between two terminals at a time, ignoring the effects that the other terminals have on the current flow, and hence the effect on the resistance value between the two terminals under consideration. This is caused by the lack of unique equipotential lines for segments with more than two terminals, as pointed out earlier. Also as current crowding effects become more pronounced, the heuristics employed in [50] would break down. These current crowding effects are the same effects that cause the L-bend heuristic error previously mentioned. This is equivalent to having the approximately straight equipotential lines that were originally chosen as a partition start to deviate significantly in location and shape as w_1/w_2 changes. While shapes such as in Fig. 5.2 might not be very common, for the sake of completeness, a general resistance extractor should be able to handle them accurately without human intervention, such as modifying the layout to suit the heuristics used in [51].

The original JET program [51] also is a table look-up program. Straight equipotential lines occur at places of symmetry, such as between points A and B in Fig. 5.1, and approximately straight lines occur one square away from bends or contacts as noted in [51] and as

seen in Fig. 5.1. Since JET only handles Manhattan geometries, it partitions one square away from all bends or contacts. Therefore, a segment must have at least two squares of metal between any bends or contacts for further partitioning. JET then pattern matches the piece exactly to the shapes and respective network models shown in Figs. 5.4 (a)-(f), if possible. It then uses table look-up values for the resistors. These primitive shapes are the most commonly occurring ones in a VLSI bus. If a partition does not match a stored pattern, then JET skips it, flags an error, and then terminates. Like the method in [50], JET cannot parse segments like the one in Fig. 5.2. It cannot even parse segments the method in [50] could handle, such as in Fig. 5.5. In Fig. 5.5, the requirement that corner bends be at least two squares away from each other yields a nonprimitive shape.

Table 5.1 shows, for the VDD and GND busses of a real test chip, the relative frequencies of occurrence of the primitives and nonprimitives according to the partitioning criteria used in JET. Note the significant number of nonprimitives. Table 5.2 categorizes the multiterminal nonprimitives by the number of edges. This gives a measure of the extent of partitioning. The computation time for large nonprimitives is equal to that of many small nonprimitives because of the $O(N^3)$ running time of the resistance calculation algorithm used in JET2. These data were compiled using a modification of the new extractor, JET2.

type of shape	GND		VDD	
	number	Pct. of total	number	Pct. of total
simple	1180	49.0%	987	47.1%
width change	0	0.0%	0	0.0%
L	8	0.3%	8	0.4%
T	206	8.6%	263	12.6%
four-way	3	0.1%	4	0.2%
contact	206	8.6%	173	8.3%
nonprimitive	806	33.5%	666	31.6%
all	2409	100%	2095	100%

JET2 relaxes the partitioning criteria used in JET. If two geometric features that cause bends in the current flow are two or more squares away from each other, then JET2 does what JET does. Otherwise, if the distance between the two geometric features is at least 2/3 of a square, then the segment is partitioned approximately half way between the geometric features. Table 5.3 shows the same data as in Table 5.2 for the same chip, except using this

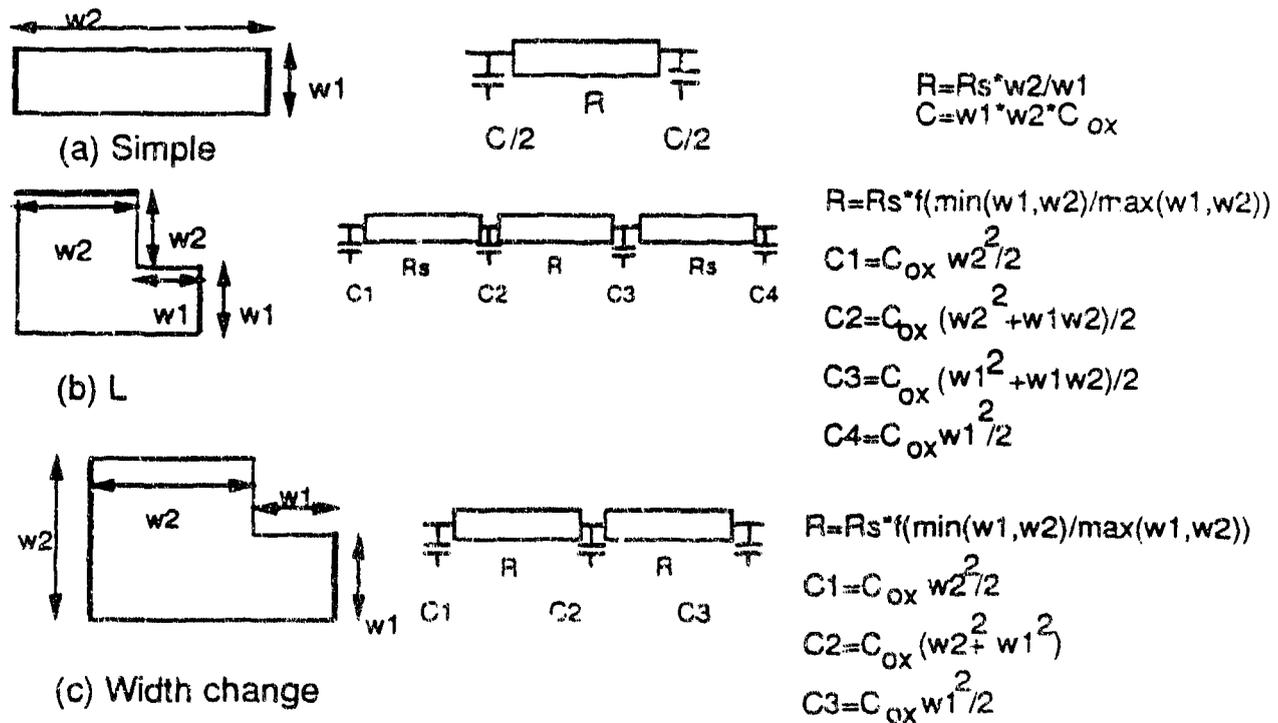
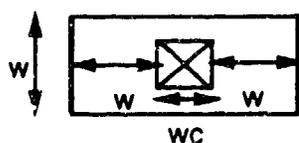
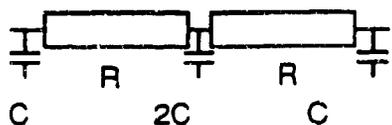


Figure 5.4 JET primitives.

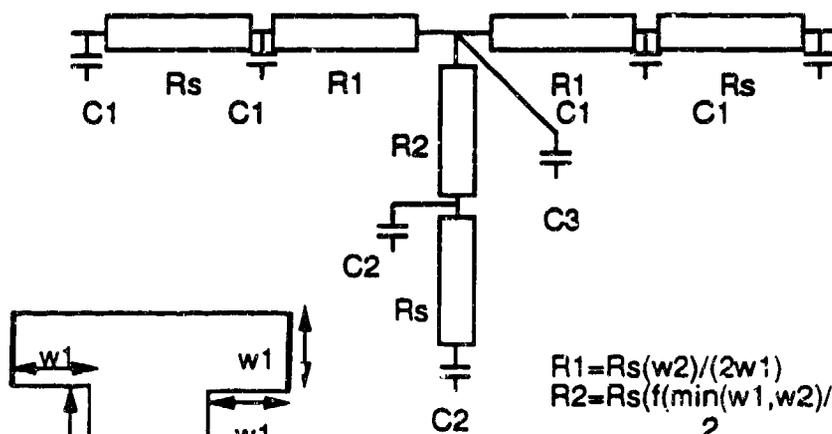


(d) Contact



$$R = R_s * f(w_c/w_c)$$

$$C = C_{ox} [(2w + w_c)w - w_c^2] / 4$$



(e) T

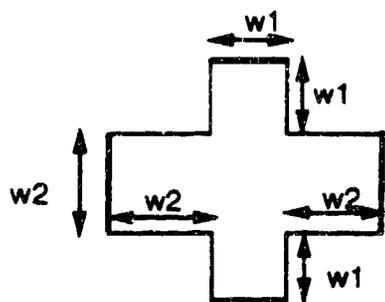
$$R1 = R_s(w2) / (2w1)$$

$$R2 = R_s(f(\min(w1, w2) / \max(w1, w2)) - 2 - (w2) / (2w1))$$

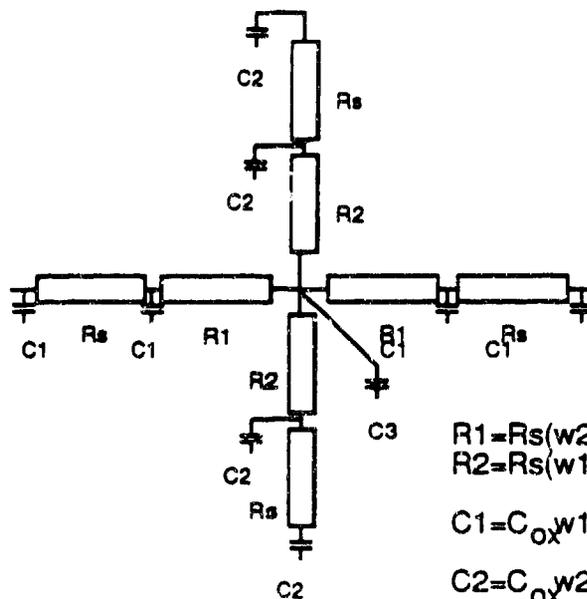
$$C1 = C_{ox} w1^2 / 2$$

$$C2 = C_{ox} w2^2 / 2$$

$$C3 = C_{ox} w1w2$$



(f) Four-way



$$R1 = R_s(w2) / (2w1)$$

$$R2 = R_s(w1) / (2w2)$$

$$C1 = C_{ox} w1^2 / 2$$

$$C2 = C_{ox} w2^2 / 2$$

$$C3 = C_{ox} w1w2$$

Figure 5.4 (continued)

new criterion. Note that the number of nonprimitives has increased, yet there are no computationally expensive ones with a large number of edges.

Table 5.2 Edge frequency for layout using JET partitioning criteria.

GND		VDD	
No. of edges	Occurrence	No. of edges	Occurrence
8	249	8	319
10	220	10	57
12	162	12	148
14	75	14	37
16	42	16	5
18	40	18	2
20	4	20	12
24	4	22	1
36	1	24	4
44	1	26	1
56	1	30	1
		32	1
		52	2
		64	2
		66	1
		72	1

Table 5.3 Edge frequency for layout using JET2 partitioning criteria.

GND		VDD	
No. of edges	Occurrence	No. of edges	Occurrence
6	331	6	116
8	537	8	562
10	255	10	70
12	65	12	32
14	34	14	34
16	37	16	9
18	2	18	1
20	1	22	2
		24	1
		26	1
		48	2

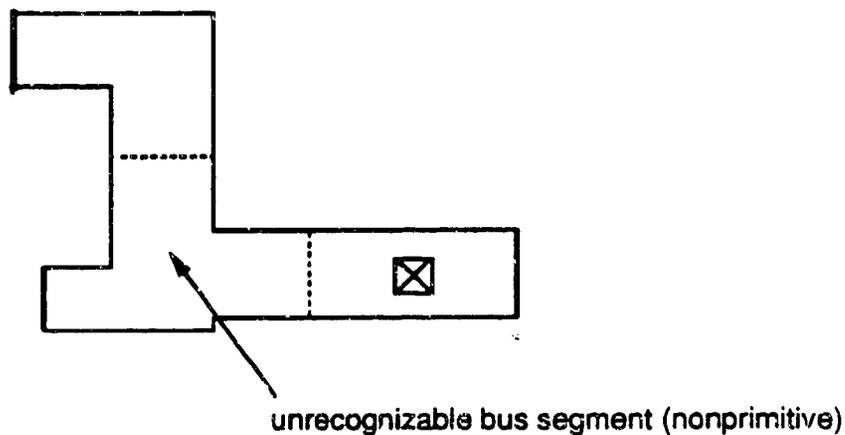


Figure 5.5 JET partitioning itself into an unrecognizable situation.

5.3 On-the-Fly Numerical Method: The Boundary Element Method

JET2 uses the same concept of splitting the bus into primitives as JET, but in addition it uses BEM to analyze the nonprimitives.

The potential u in any general linear resistive region (Fig. 5.6) obeys Laplace's equation in two dimensions (assuming constant thickness) in its interior Ω :

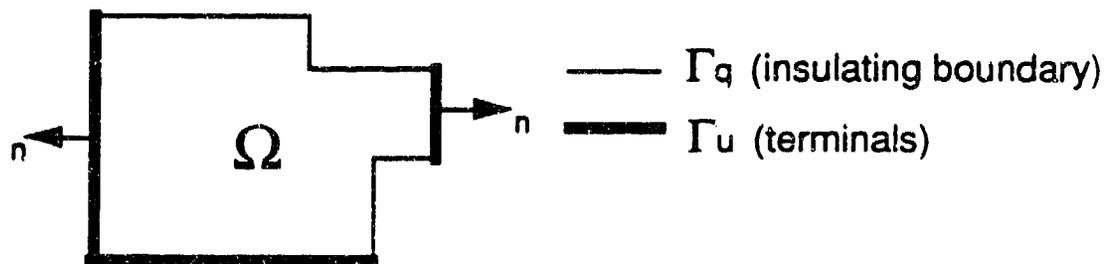


Figure 5.6 Domain for Laplace's equation.

$$\nabla^2 u = 0 \quad (5.1a)$$

the Dirichlet boundary condition on its conducting boundary Γ_u (the terminals) is:

$$u = \bar{u} \quad (5.1b)$$

and the Neumann boundary condition on its insulating boundary Γ_q is:

$$q = 0 \quad (5.1c)$$

where $q = \partial u / \partial n$ and n is the unit outward normal to the boundary. This follows from

$$0 = J \cdot n = \sigma E \cdot n = -\sigma \frac{\partial u}{\partial n} \quad (5.1d)$$

Γ_u and Γ_q need not be continuous, but $\Gamma = \Gamma_u + \Gamma_q$. The current through any conducting subboundary Γ_i is given by

$$I = \frac{1}{R_s} \int_{\Gamma_i} q d\Gamma \quad (5.2)$$

where R_s is the sheet resistance.

There are a number of standard methods for solving Laplace's equation [52], [53]. We have chosen the BEM [54], [55] because of its flexibility, generality, and computational efficiency.

For convenience, from now on the Laplacian operator is represented by Δ . The arc length element $d\Gamma$ and the area element $d\Omega$ will be omitted from integrals since they are understood. Consider the same domain Ω and boundary Γ . Green's second identity [52] (which is derived from the Divergence Theorem) states that for any arbitrary functions u and v which have continuous second partial derivatives in Ω and continuous first partial derivatives in the combined domain $\Omega + \Gamma$:

$$\int_{\Gamma} u \frac{\partial v}{\partial n} - \int_{\Gamma} v \frac{\partial u}{\partial n} = \int_{\Omega} u \Delta v - \int_{\Omega} v \Delta u \quad (5.3)$$

Let u be the solution to our PDE as before, while we still have a degree of freedom in choosing v . Using $\Delta u = 0$ and rearranging (5.3), we obtain

$$-\int_{\Omega} u \Delta v + \int_{\Gamma} u \frac{\partial v}{\partial n} - \int_{\Gamma} v \frac{\partial u}{\partial n} = 0 \quad (5.4)$$

Now choose a function v such that

$$\Delta v = -\delta(P) \quad (5.5)$$

where $-\delta(P)$ is the 2-D Kronecker delta function located at point P anywhere in Ω . This is called Green's function (or fundamental solution), otherwise, known as a delta source. In 2-D, Green's function is

$$u^* = \frac{1}{2\pi} \ln r \quad (5.6a)$$

$$r = \sqrt{(x-x_p)^2 + (y-y_p)^2} \quad (5.6b)$$

where r is the absolute value of the distance between the observation point at (x, y) and the delta source at (x_p, y_p) . Substituting (5.5) into (5.4) yields

$$u(P) + \int_{\Gamma} u \frac{\partial v}{\partial n} - \int_{\Gamma} v \frac{\partial u}{\partial n} = 0 \quad (5.7)$$

$$P \in \Omega$$

This equation is expressed in terms of only the unknown potential and outward normal flux at the boundary, except for the term $u(P)$. To eliminate $u(P)$ we choose to take the point P of the delta source on Γ instead of in Ω . If P is on a smooth part of the boundary (i.e., not on a corner), then it can be shown [61] that the area integral reduces to $0.5u(P)$. If P sits on a corner, then it reduces to $cu(P)$ where c depends on the local geometry. It will be shown later that this constant does not have to be calculated.

If, for convenience, we replace $\partial u / \partial n = q$, as we have done before, let $v = u^*$ (to remind us that v is a potential as well, but that of a hypothetical delta source), and analogously let $\partial u^* / \partial n = q^*$, we obtain as a final contour integral equation

$$cu(P) + \int_{\Gamma} uq^* - \int_{\Gamma} qu^* = 0 \quad (5.8)$$

$$P \in \Gamma$$

$c = 0.5$ if the delta source is on a smooth boundary.

The boundary element equation (5.8) can be applied as follows: once we choose the form of the approximating function for u and q on the boundary, the equation gives a constraint on the choice of coefficients of the approximating function so as to minimize the error between the approximations and the exact answer. If the coefficients are put in terms of values of u and q at discrete points on the boundary, then the constraint becomes one on these u and q values. Given enough constraints, we can form a system of equations to solve for these u and q values.

5.4 Discretization of BEM Equations

To discretize (5.8) into a system of linear equations, we construct an approximation to u and q by first dividing the boundary into N segments as shown in Fig. 5.7. Then we designate the u and q values of the endpoints of each segment as variables. Note that dividing the boundary into N segments yields N endpoints and, hence, $2N$ variables since each endpoint j has a u_j and q_j variable associated with it (there is an exception that will be dealt with later). However, for every point on Γ_u (not just the endpoints), u is known and q is unknown. The exact opposite occurs for Γ_q . Thus there will be only N unknowns, with the other N known values combining to form a source term. Next, on each segment, we approximate the local u solution

on the segment by a function of the u value of the endpoints. Similarly, the local q solution is approximated by the q value of the endpoints. These functions can be expressed by interpolation functions:

$$u = T_{j0}(s)u_j + T_{j1}(s)u_{j+1} \quad \text{for segment } j_0 \quad (5.9)$$

$$q = T_{j0}(s)q_j + T_{j1}(s)q_{j+1} \quad \text{for segment } j_0 \quad (5.10)$$

$$T_{j0} = \begin{cases} 1 & \text{if } s = 0 \\ 0 & \text{if } s = 1 \end{cases} \quad (5.11)$$

$$T_{j1}(s) = \begin{cases} 0 & \text{if } s = 0 \\ 1 & \text{if } s = 1 \end{cases} \quad (5.12)$$

where s is the normalized arc length along the segment as shown in Fig. 5.8. For example, if u and q are to linearly vary between the endpoint values, we obtain

$$T_{j0} = 1 - s \quad (5.13)$$

$$T_{j1} = s \quad (5.14)$$

$$u = (1-s)u_j + su_{j+1} \quad \text{for segment } j_0 \quad (5.15)$$

$$q = (1-s)q_j + sq_{j+1} \quad \text{for segment } j_0 \quad (5.16)$$

This is called a *linearelement* and is a common approximating function used on boundaries with Manhattan geometries. Higher-order polynomial approximations are possible and require k points per segment for an order k approximation.

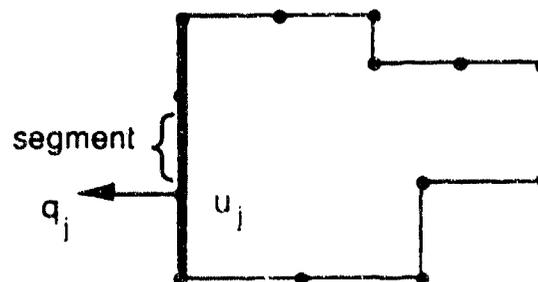


Figure 5.7 Division of boundary into segments.

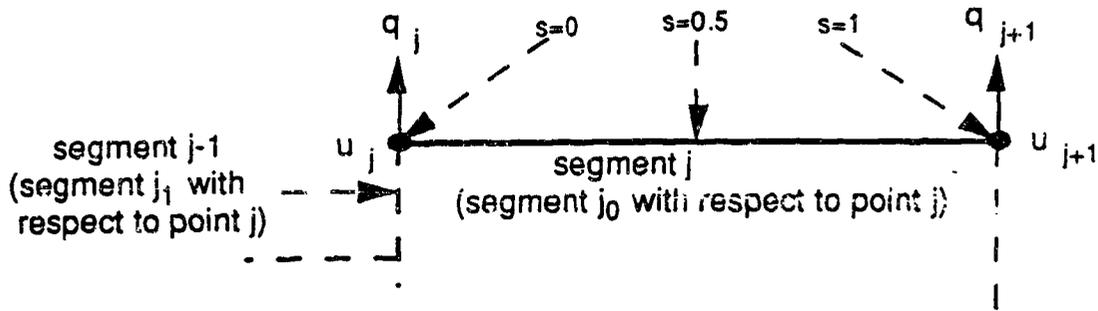


Fig. 5.8 Typical Segment

Note that both q and u use the same order of interpolation functions in (5.9) and (5.10). This is not necessary, but is done because it is convenient. Note also in looking at Fig. 5.7 that each point j is shared between segment j and segment $j-1$. Thus each point has two interpolation functions associated with it.

Next we choose to place the delta source at point i . Substituting (5.9) and (5.10) into (5.8), we obtain the following equation, which is a matrix equation since i can be any of the N endpoints:

$$\sum_{j=1}^N H_{ij} u_j = \sum_{j=1}^N G_{ij} q_j \quad i = 1..N \quad (5.17)$$

$$H_{ij} = \begin{cases} \sum_{k=0}^1 \int_{\Gamma_k} T_{jk} q_j d\Gamma = \sum_{k=0}^1 H_{ijkk} & j \neq i \\ c + \sum_{k=0}^1 \int_{\Gamma_k} T_{ik} q_i d\Gamma = c + \sum_{k=0}^1 H_{iik} & j = i \end{cases} \quad (5.18)$$

$$G_{ij} = \sum_{k=0}^1 \int_{\Gamma_k} T_{jk} u_j d\Gamma = \sum_{k=0}^1 G_{ijk} \quad (5.19)$$

where c is defined in (5.8). As mentioned before, calculation of c and hence H_{ii} is more complex if point i is at a corner. We can avoid the calculation of c and calculate H_{ii} directly by noticing that in a physical resistor of any shape, forcing the boundary condition $u = 1$ on

Γ_u makes $u = 1$ and $q = 0$ everywhere on Γ . Thus substituting $u_j = 1$ and $q_j = 0$ into (5.17) yields

$$H_{ii} = - \sum_{j=1, j \neq i}^N H_{ij} \quad (5.20)$$

As mentioned before, because half of the $2N$ variables are known, we can formally write the final linear algebraic system:

$$\begin{aligned} Ax &= b \\ A &= \{a_{ij} \mid a_{ij} = H_{ij} \text{ if point } j \in \Gamma_u \text{ else } a_{ij} = -G_{ij}\} \\ x &= \{x_i \mid x_i = u_i \text{ if point } i \in \Gamma_q \text{ else } x_i = q_i\}^T \\ b &= -H \cdot z \text{ where } z = \{z_i \mid z_i = u_i \text{ if point } i \in \Gamma_q \text{ else } z_i = 0\}^T \end{aligned} \quad (5.21)$$

As mentioned before, the known u variables multiplied by their corresponding column of H coefficients are transferred to the right-hand side to form the source term. The known q variables are not similarly treated because all of the known q variables are equal to zero by the Neumann boundary condition.

As described before, to find the $P(P-1)/2$ resistors for our representation of the P port resistor, P different Dirichlet boundary conditions are needed. Hence, P source terms are required. Using LU factorization, approximately $\frac{N^3}{3} + PN^2$ operations are necessary to solve this system.

Again the advantage of the BEM approach is that only the boundary of the shape is broken up, not the interior. Thus the generated matrix sizes are much smaller, especially for complex shapes, than those for finite element method (FEM) [52]. Also generation of grid points on a boundary is much simpler than the generation of triangular meshes in the interior, as is done in FEM. Also, the outward normal flux q is solved for directly in BEM, whereas in

traditional FEM one has to subtract two nearby voltage values.

5.4.1 Extension to Multiply-Connected Domains

So far we have considered simple domains, as in Fig. 5.6, where the boundary is continuous and, hence, the contour used for the contour integral is straightforward. For more complex regions with internal contacts or holes, as shown in Fig. 5.9(a), we can visualize the contour integral path (dotted line) starting at the outside boundary, covering part of it, then crossing the interior to cover the two holes in the manner shown, then doubling back on itself to cover the rest of the outer boundary. Those segments where the path crosses over itself have outward normals that point in opposite directions as indicated. Looking back at the contour integrals in (5.9), we see that these segments' contributions cancel out, resulting in the contours shown in Fig. 5.9(b). Note that the two internal contours are counterclockwise while the outer one is clockwise. This will always be the case. Thus, the net contour integral of a disjoint boundary equals the sum of the contour integrals of the disjoint boundaries *with each internal boundary evaluated in a direction opposite to the outer boundary*.

5.4.2 Special Case of Points with Two Unknowns

Earlier it was mentioned at the beginning of Section 5.4 that each point has one unknown and one known variable associated with it. The exception, shown in Fig. 5.10, occurs at corners in the Dirichlet Γ_u boundary. u is known, but there are *two* unknown q variables since there are two outward normals and the outward normal flux is not zero for both of them. The authors of [55] describe two ways of dealing with this problem. The first is the partially discontinuous element. In this method, two separate but close nodes are used to model the corner. This preserves the rule that each point has only one unknown associated with it. A

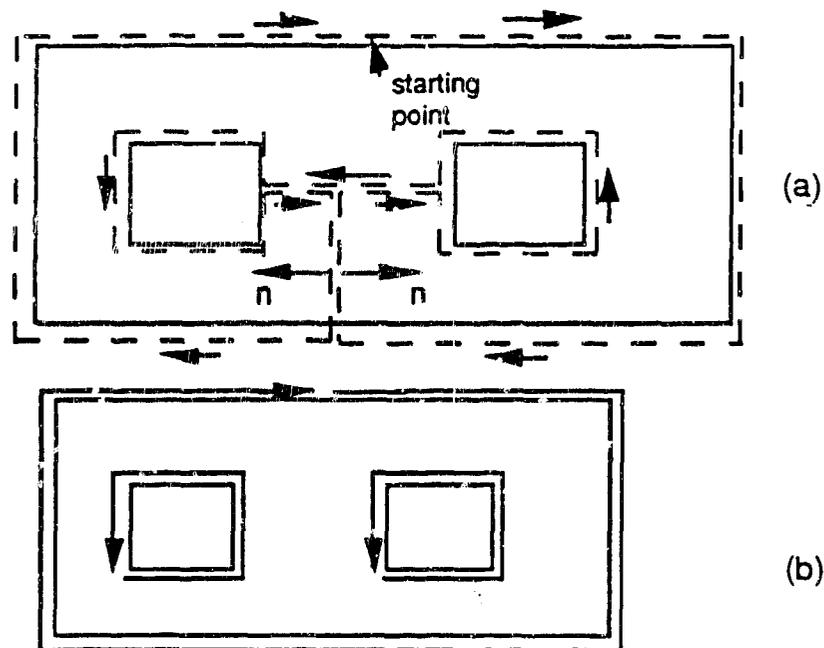


Figure 5.9 Multiply connected regions and their contours.
 (a) total contours (b) net contours

more accurate method, adopted in [55] and adopted by us, is the double node where the corner node is treated as having two unknowns q_{j1} and q_{j0} . The G_{ij} coefficient of a normal node is made of the sum of the two contributions from the two segments to its immediate left and right, as in (5.19); here, the coefficient for q_{j1} is only the contribution from the segment $j1$ and similarly for q_{j0} . In other words,

$$G_{ij1} = \int_{G_{j1}} T_{j1} u^* d\Gamma \quad (5.22)$$

$$G_{ij0} = \int_{\Gamma_{j0}} T_{j0} u^* d\Gamma \quad (5.23)$$

Because of the extra unknown, an extra equation will be needed. This can be formed by placing the delta source anywhere else on the boundary, except at the segment endpoints (where all the previous delta sources have been placed). As in [55], we choose to place it at the midpoint of segment $j1$. Unlike the other N equations, the potential and flux at the delta source

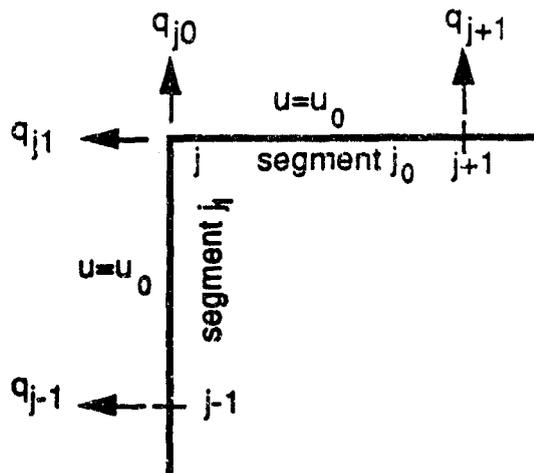


Figure 5.10 Double-node condition.

point are not considered unknowns to be solved for since by the very definition of the linear approximation we are using, these quantities are only the arithmetic mean of those at the endpoints of the segment. Thus the H_{ii} coefficient does not have the same meaning for the extra equation as for the other normal N equations. Because the delta source is on a smooth surface, the c term in (5.8) is 0.5. Since the delta point has no unknown of its own, and hence no coefficient of its own to add this c correction term, we can add it to the H coefficient associated with one of the segment endpoints, i.e., H_{ij} or H_{ij-1} .

5.4.3 Properties of the Resulting Linear System

The flux and potential produced by a delta source placed at one part of the boundary are nonvanishing at every other part of the boundary. Thus, any point couples to every other point and the resulting matrix A defined in (5.21) is full. The matrix is, in general, asymmetric since the H and G matrices of which it is comprised are asymmetric as well, because the H and G coupling coefficients between two points do not depend only on the distance between the points (which is the same no matter which point has the delta source placed on

it). The coefficients also depend on the length and orientation of the segments the points are on. If these are not symmetric, then neither is the matrix. Also the system is not guaranteed to converge using relaxation methods, thus requiring the full LU factorization to solve the system.

Thus, even though the BEM system of equations is significantly smaller than the corresponding FEM or the finite difference method (FDM) [52] system which yields the same accuracy, it does not have the advantages of sparsity and symmetry that the FEM system does. Also the computation of the coefficients in BEM are more expensive than in FEM or FDM. Thus the question remains which system is faster to solve. Comparison in [55] of the computation time between both BEM and FDM using the same shapes has shown that the BEM system is still faster, by factors as large as 3 to 7. The more complicated the shape, in fact, the larger the savings. A simple FEM program written as part of this work has shown that FEM with uniform grids to be slower than BEM as well for the same accuracy, even with sparsity and symmetry of the system taken advantage of.

In [55], for simple shapes most of the computation time was due to computing the matrix coefficients. The authors evaluate the integrals in (5.18) and (5.19) numerically, probably by some special techniques because the kernels of the integrands become singular at the delta source point. In our approach, we try to decrease computation time and improve accuracy by calculating the coefficients analytically.

5.4.4 Analytical Calculation of Matrix H and Matrix G Coefficients

To calculate the contribution of a segment to H_{ij} and G_{ij} , without loss of generalization, we shift the coordinate system so that the delta source point i is at the origin and rotate the

axes so that the segment with point j as its endpoint is vertical and assume that the contour integration direction is the positive y direction ($d\Gamma = dy$). We also assume that the outward normal n is i_x . Point j is located at y_1 and point $j + 1$ at y_2 with x as the directed distance between the delta source and the segment (Fig. 5.11). If the actual contour direction is in the negative y -direction, then H_{ij} and G_{ij} are negated. The H_{ij} is also negated if the actual outward normal in the layout is $-i_x$.

The interpolation function T_{j0} in (5.35) and (5.36) and u^* and q^* in terms of this new coordinate system are

$$T_{j0} = \frac{y_2 - y}{y_2 - y_1} \quad (5.24)$$

$$u^* = \frac{1}{2\pi} \ln r = \frac{1}{4\pi} \ln(x^2 + y^2) \quad (5.25)$$

$$q^* = \frac{\partial u^*}{\partial n} = \frac{\partial u^*}{\partial x} = \frac{-x}{2\pi \ln(x^2 + y^2)} \quad (5.26)$$

Substituting the above into the contributions H_{ij0} and G_{ij0} of (5.18) and (5.19), respectively, yields:

$$\begin{aligned} H_{ij0}(x, y_1, y_2) &= \frac{-x}{2\pi(y_2 - y_1)} \int_{y_1}^{y_2} \frac{y_2 - y}{x^2 + y^2} dy \\ &= \frac{-1}{4\pi(y_2 - y_1)} \left[2y_2 \left[\tan^{-1} \frac{y_2}{x} - \tan^{-1} \frac{y_1}{x} \right] + \ln \frac{x^2 + y_2^2}{x^2 + y_1^2} \right] \end{aligned} \quad (5.27)$$

$$\begin{aligned} G_{ij0}(x, y_1, y_2) &= \frac{-1}{4\pi(y_2 - y_1)} \int_{y_1}^{y_2} (y_2 - y) \ln(x^2 + y^2) dy \\ &= \frac{1}{8\pi(y_2 - y_1)} \left[2y_2 \left[y \ln(x^2 + y^2) - 2y + 2x \tan^{-1} \frac{-1y}{x} \right] \right. \\ &\quad \left. - (x^2 + y^2) \left[\ln(x^2 + y^2) - 1 \right] \right]_{y_1}^{y_2} \end{aligned} \quad (5.28)$$

where terms have been grouped as much as possible to reduce the number of evaluations of logarithmic functions first and then the number of floating point divides and multiplies. The singularities of the kernel are integrated out.

Note that for certain zero values of x , y_2 or y_1 the above functions can be simplified. To save computation, we check for these conditions and use intermediate variables to eliminate redundant operations.

The contribution of the other segment (with points j and $j-1$ as its endpoints) yields H_{ij1} and G_{ij1} and is calculated much the same way, except that T_{j1} is used as the interpolation function since point j now fulfills the same role for this segment as point $j+1$ did for the first segment. In fact, if one designates the coordinate of point $j-1$ as y_0 and redefines x with respect to the new segment, one finds that

$$H_{ij1}(x, y_0, y_1) = -H_{ij0}(x, y_1, y_0) \quad (5.29)$$

$$G_{ij1}(x, y_0, y_1) = -G_{ij0}(x, y_1, y_0) \quad (5.30)$$

Thus one function can be used to evaluate both parts of H_{ij} and similarly for G_{ij} .

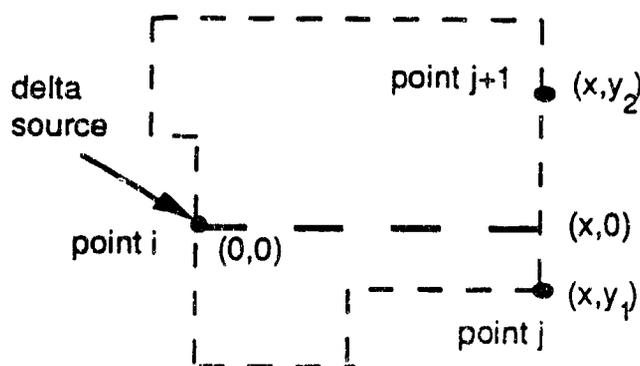


Figure 5.11 Setup for analytical calculation of H_{ij} and G_{ij} coefficients.

5.4.5 Current Calculation

After the system is solved, all of the q 's are known and can be used to solve for the current. The current through any terminal equals the sum of the currents through the segments that compose that terminal. To find the current through the segment in Fig. 5.11, let q_1 and q_2 be the solved values at y_1 and y_2 , respectively. Then on the segment

$$q = \frac{(y-y_2)q_1 + (y_1-y)q_2}{y_1-y_2} \quad (5.31)$$

Substituting this in the definition of I in (5.2) yields

$$\begin{aligned} I &= \frac{1}{R_s(y_2-y_1)} \int_{y_1}^{y_2} (y-y_2)q_1 + (y_1-y)q_2 dy \\ &= \frac{(q_1+q_2)(y_2-y_1)}{2R_s} \end{aligned} \quad (5.32)$$

In summary, in this section we have derived the boundary element method which solves Laplace's equation by first transforming it from a statement about the potential over an area domain into an equation on the boundary of that domain. Then this resulting contour equation is approximately solved by discretizing it into a linear algebraic system using linear approximations to the potential and flux on the boundary. The method was further generalized for multiply connected regions. Solving the linear system yielded the outward normal flux at discrete points on the boundary, which were then used to calculate the current flowing through the terminals in the original problem.

5.5 Comparison with Finite Element Method (FEM)

In this section we compare BEM with a FEM method that directly computes resistance. The FEM will be used later in conjunction with BEM to extract the capacitance model.

Again, in this section, we assume normalized $R_s = 1$ and normalized $C_{ox} = 1$.

In [56] a method of replacing a triangle of an FEM mesh with an equivalent three-terminal delta RC network and a rectangle with its four-terminal RC equivalent is derived. The rectangle is of interest to us since the layout shapes considered in this thesis are all rectangular due to the Manhattan geometry. The rectangular mesh model and its equations are shown in Fig. 5.12.

$$R1 = 2x/y \quad (5.33a)$$

$$R2 = 2y/x \quad (5.33b)$$

$$C = xy \quad (5.33c)$$

Replacing all of the mesh rectangles with their equivalent delta networks yields a detailed RC network model of the segment being modeled. Considering the resistance network alone, by collapsing all nodes on one terminal into a single terminal node and using delta-Y transformation of to eliminate internal nodes yield the equivalent star model. In Section 5.6 below, we show how this method can be used in conjunction with BEM to extract the capacitance of bus segments. However, for now, we consider it to be representative of FEM methods to do a comparison with BEM for resistance extraction.

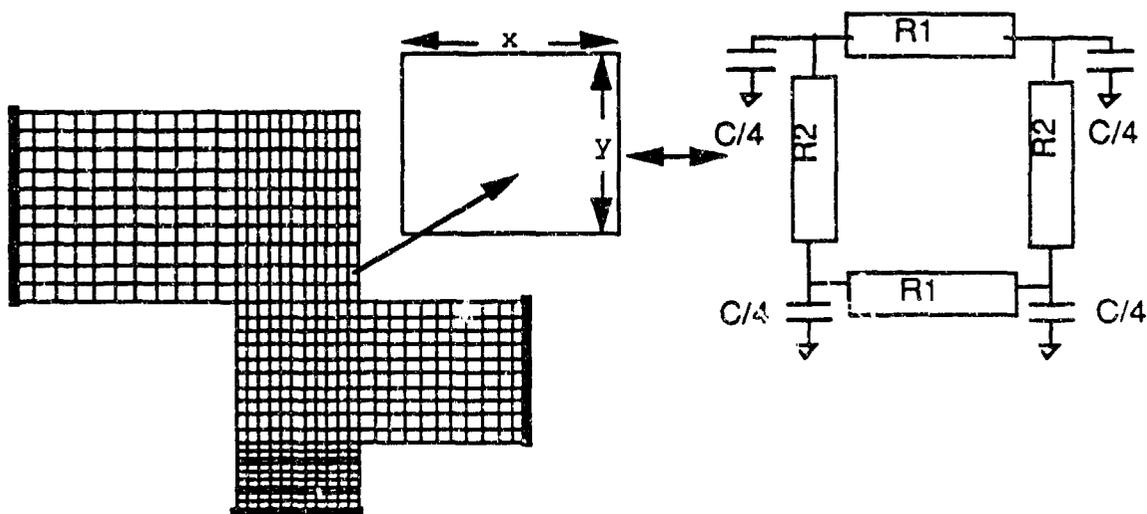


Figure 5.12 RC equivalent of rectangular mesh element.

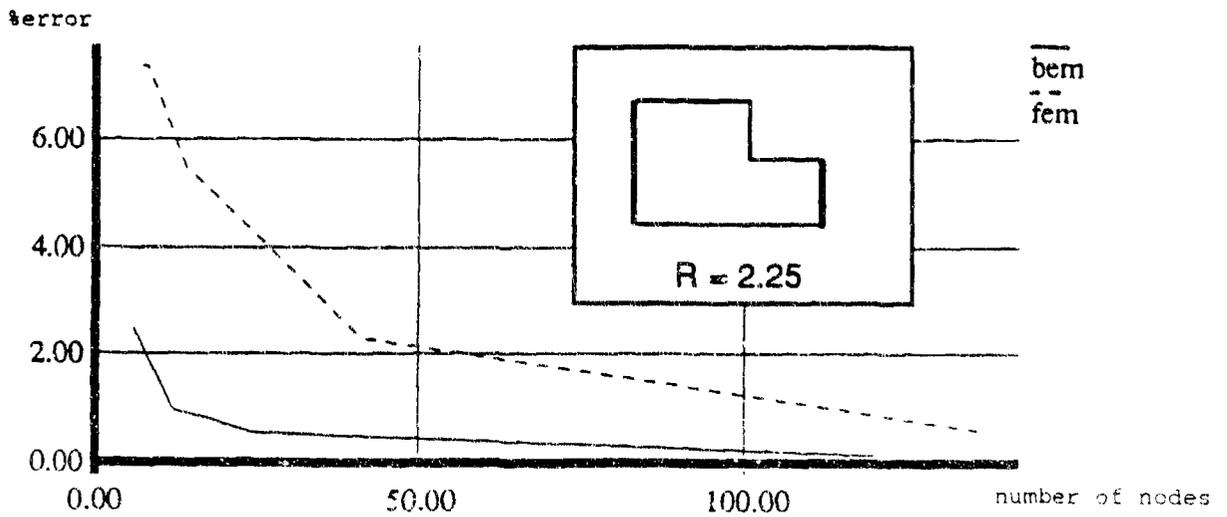
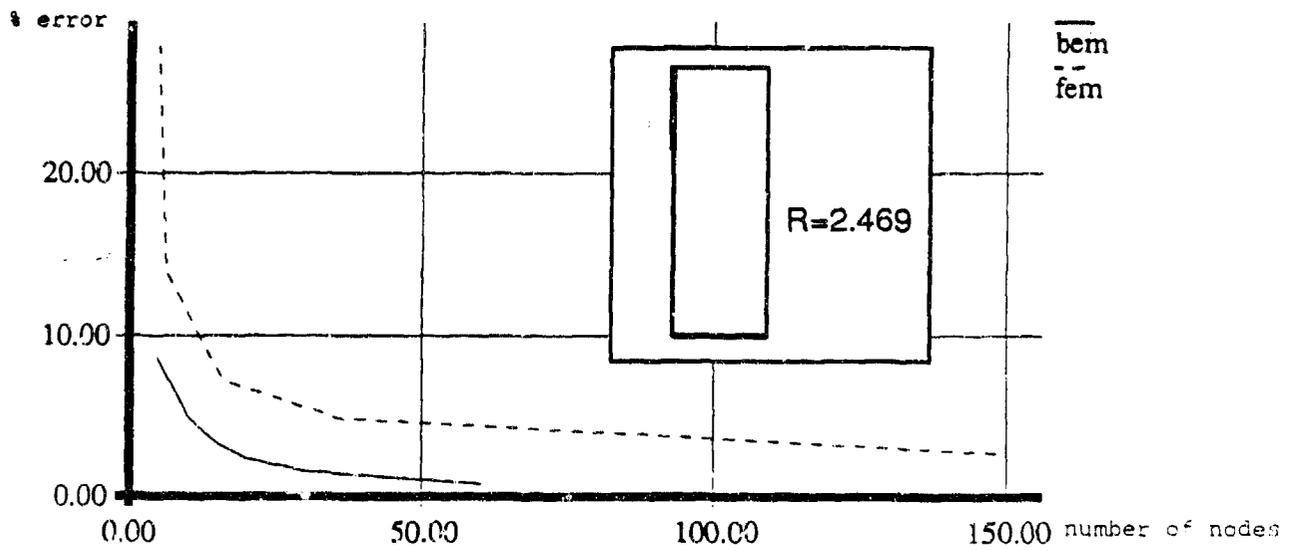
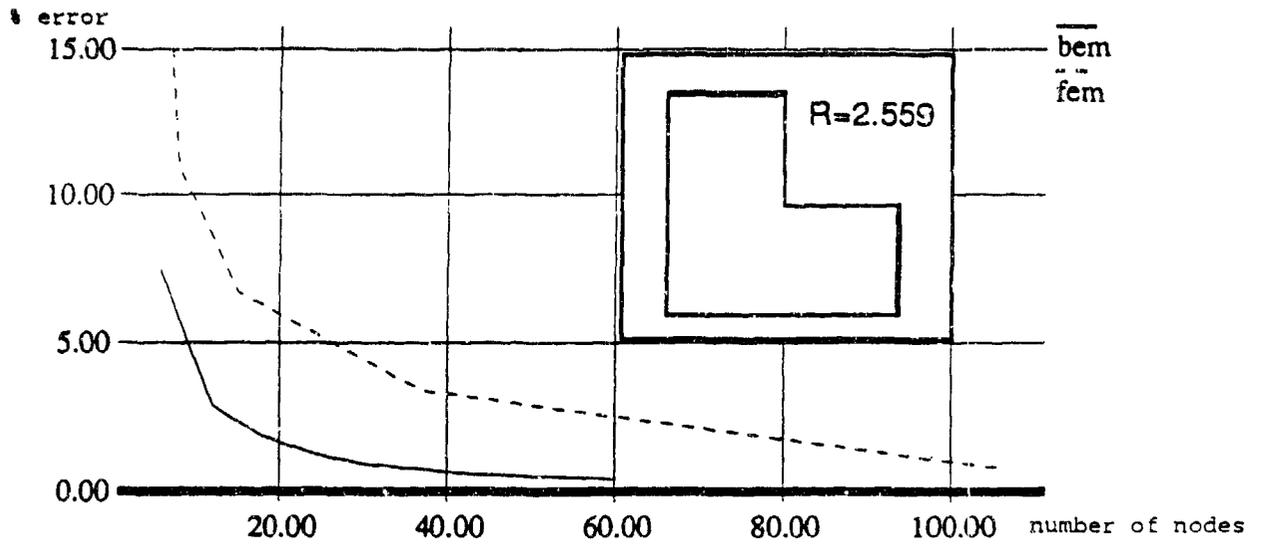


Figure 5.13 BEM vs. FEM in resistance calculation.

now, we consider it to be representative of FEM methods to do a comparison with BEM for resistance extraction.

Shown in Fig. 5.13 are shapes for which the author of [56] uses the above FEM method with an *adaptive mesh* to calculate the resistance. The resultant absolute percentage error in resistance vs. number of nodes is plotted next to each shape. The performance of BEM using a simple *uniform grid* is also plotted. Clearly BEM outperforms FEM since for every desired level of accuracy, BEM requires a significantly fewer number of nodes, even without the benefit of an adaptive mesh.

5.6 JET2 Capacitance Model Extraction

In JET2, a bus segment is simply modeled as a parallel plate capacitor to determine the total capacitance to ground. JET2 improves over JET in distributing this capacitance among the terminal nodes of the segment's RC model. The next section describes how a combination of the finite element method and a special node reduction technique determines the distribution of capacitance among the terminals so as to accurately model the transient response of the bus segment. For primitives shown in Fig. 5.4, the resulting capacitance to ground at each node is stored in a look-up table. For nonprimitives, the capacitance model extraction is done on-the-fly.

We first divide the bus segment into many rectangles. Then the FEM method described in Section 5.5 is used to replace each rectangle with a simple RC model (Fig. 5.12). This yields a detailed RC network for the whole bus segment which accurately models its distributed RC effects. Note that since each rectangular element has a total capacitance to ground equal to its parallel plate capacitance, the sum of all the capacitances in the overall segment's

RC model is equal to its parallel plate capacitance as well.

The FEM method usually produces a very large RC model for the segment, depending on the number of grids chosen to model the segment. We would like to simplify the network into a delta model while still retaining the essence of its transient response. Reference [56] presents an RC network node reduction method based on Elmore time constants [57] that does just that. The final simplified network has the following properties:

- 1) The Elmore delay between any two terminal nodes is the same as that of the original network.
- 2) The terminal resistance parameters of the network are preserved.
- 3) No additional coupling capacitances are introduced.
- 4) The total capacitance to ground of the final network is equal to that of the original network.

Thus the transmission properties of the detailed RC network are accurately preserved. The node elimination formulas [56] for networks that contain no coupling capacitance are

$$g'_{ij} = g_{ij} + \frac{g_{ik}g_{jk}}{\sum_{m=1, m \neq k}^N g_{mk}} \quad i \neq j \wedge i \neq k \wedge j \neq k \quad (5.34a)$$

$$C'_i = C_i + \frac{C_k g_{ik}}{\sum_{m=1, m \neq k}^N g_{mk}} \quad i \neq k \quad (5.34b)$$

where the above is for the elimination of node k . All of the above four properties are preserved after each node elimination and hence are preserved in the final network. After elimination of all internal nodes, a delta model is obtained with all the capacitances still connected to ground.

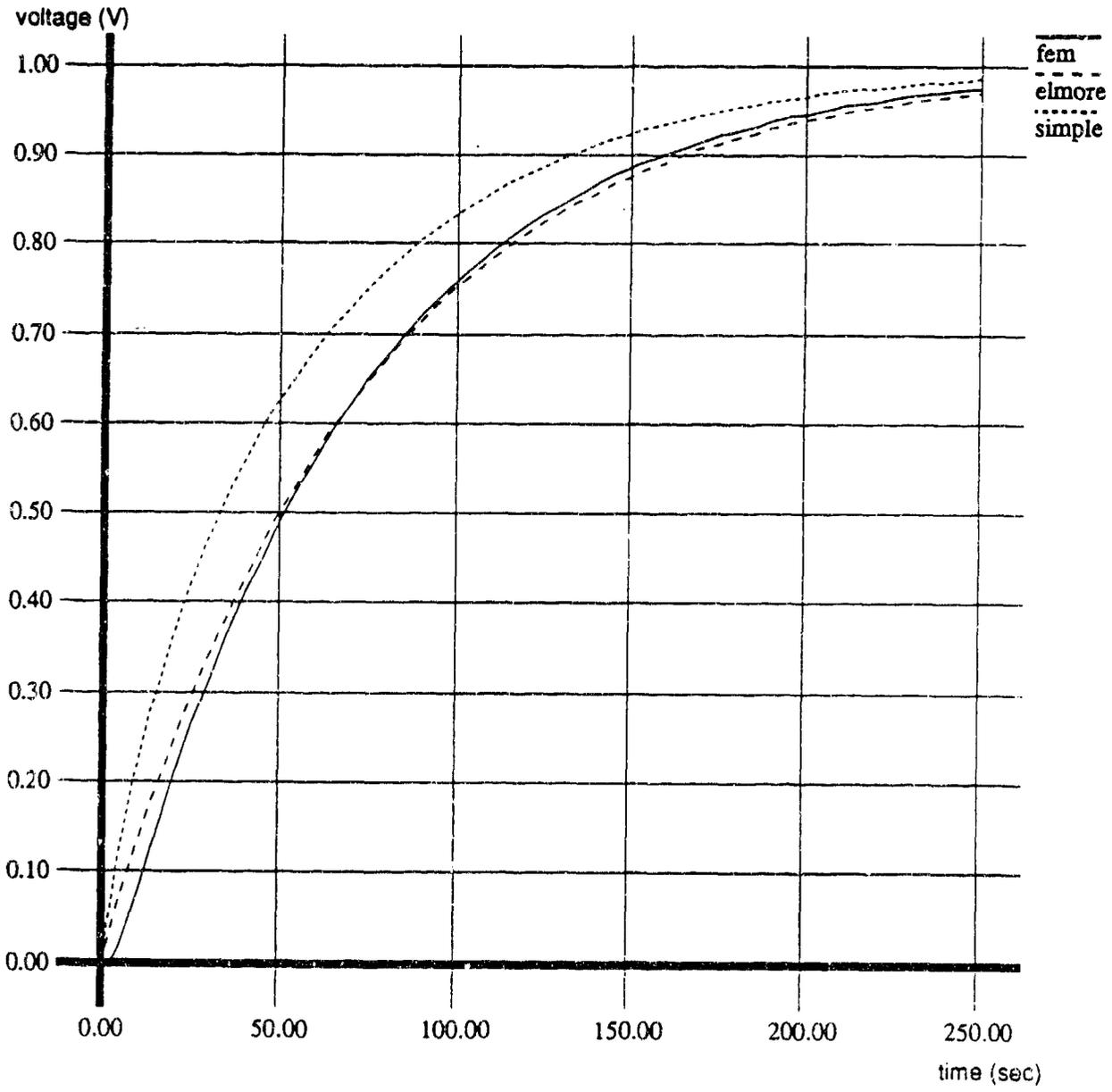
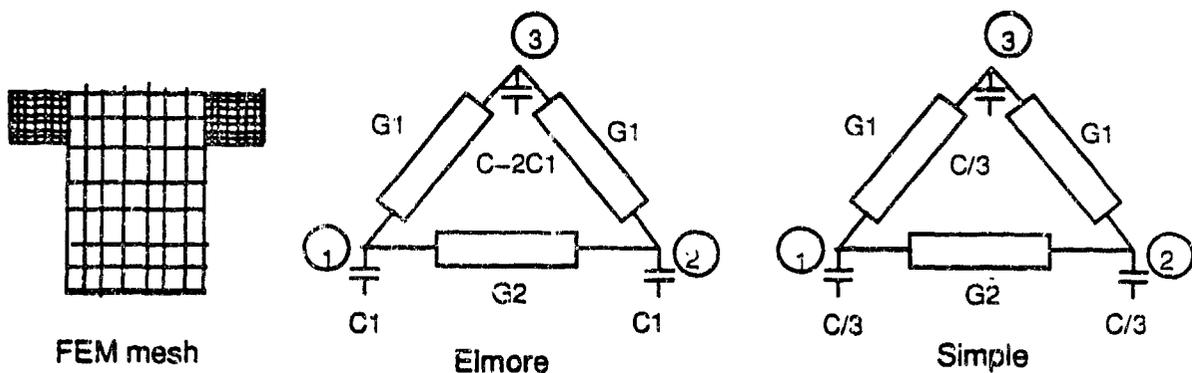


Figure 5.14 Step response of T primitive using three RC models.

Shown in Fig. 5.14 is a comparison of different RC models for the unit step response of the T-junction primitive of $w_2/w_1 = 5$ (see Fig. 5.24 below for meaning of w_1 and w_2) for zero initial conditions. The input terminal is one of the symmetric terminals with the other terminals floating. The output terminal is the asymmetric one. The most accurate model is "fem" which uses a rectangular mesh of 400 nodes. Thus we want the other two simpler models to approximate the response of this first model. The other two models are "elmore," which uses the above node reduction technique to reduce the model to three terminals, and "simple," also reduces the model to three terminals with the same resistance values as "elmore," but with the total capacitance C equally distributed to each node as $C/3$. The figure shows that the Elmore model is very accurate.

Because the Elmore node reduction technique does not alter the multiport resistance parameters of the network, one can determine them from some other method that converges faster to the correct resistance values, such as BEM. Hence the resistances from the simplified network are ignored, and the resistance values obtained by BEM, which are more accurate, are retained. The lumped capacitances at each terminal of the simplified network are kept and used for the capacitances of our model. This hybrid approach is used in JET2, as explained next.

5.7 RC Models of Primitives

The rectangular segment and the L primitive have analytical resistance models. The resistances for the other primitives are stored in tables. For all of the primitives, except for the rectangular segment, the fraction of ground capacitance distributed to a terminal node is stored in tables.

5.7.1 Rectangular Segment

The model of Fig. 5.15, where $R = L/W$ and C is the total ground capacitance, has the same Elmore delay as a detailed distributed RC model [56].

The added improvement in this model is that C now takes into account fringing capacitance as well as parallel plate capacitance. Using the model in Fig. 5.16, from [58] we have

$$C = \epsilon l \left[\frac{w - \frac{t}{2}}{h} + \frac{2\pi}{\ln\left(1 + \frac{2h}{t} + \sqrt{\frac{2h}{t}\left(\frac{2h}{t} + 2\right)}\right)} \right] \quad w \geq \frac{t}{2}$$

$$C = \epsilon l \left[\frac{w}{h} + \frac{\pi(1 - 0.5043 \frac{t}{2h})}{\ln\left(1 + \frac{2h}{t} + \sqrt{\frac{2h}{t}\left(\frac{2h}{t} + 2\right)}\right)} + 1.47 \right] \quad w < \frac{t}{2}$$
(5.35)

where t = thickness of the metal, h = height above the ground plane, and l = length of the resistor, and the metal is surrounded by a material of dielectric constant ϵ . The first term in the bracket is the normal parallel plate contribution and the second term is due to the fringing field. The above formulas are accurate except when $t \ll h$. If t and h are not specified, JET2 uses only the parallel plate model and C_{ox} . For the case of metal2, which might be embedded in a different dielectric than metal1 (Fig. 5.17), the fringing term in (5.35) is multiplied by a correction factor of $(1 + \epsilon_2/\epsilon_1)/2$ [13].

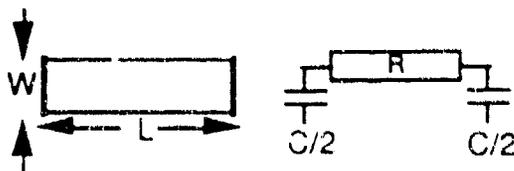


Figure 5.15 Rectangular segment model.

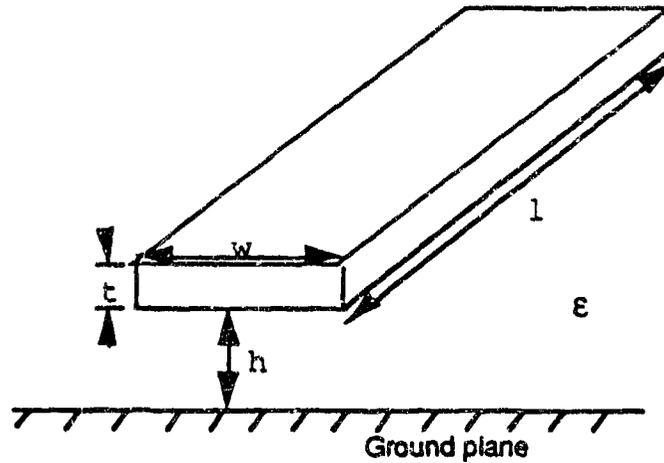


Figure 5.16 Ground capacitance model for rectangular segment.

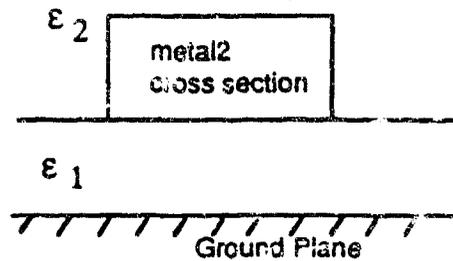


Figure 5.17 Differing dielectrics.

5.7.2 L Primitive

In the model shown in Fig. 5.18, R is given analytically in [59] as

$$R = R_s \left[\frac{1}{a} - \frac{2}{\pi} \ln \left(\frac{4a}{a^2+1} \right) + \frac{a^2-1}{\pi a} \cos^{-1} \left(\frac{a^2-1}{a^2+1} \right) \right], \quad a = \frac{\max(w_1, w_2)}{\min(w_1, w_2)} \geq 1 \quad (5.36)$$

where width ratio a reflects $R(w_1, w_2) = R(w_2, w_1)$. The resistance R versus a is plotted in Fig. 5.19. The total capacitance to ground C for this and the rest of the primitives is derived from the parallel plate model. In this case, the total capacitance is:

$$C = C_{ox}(w_1^2 + w_2^2 + w_1 w_2) \quad (5.37)$$

The plot of the normalized Elmore capacitance C_1/C of the smaller width terminal versus

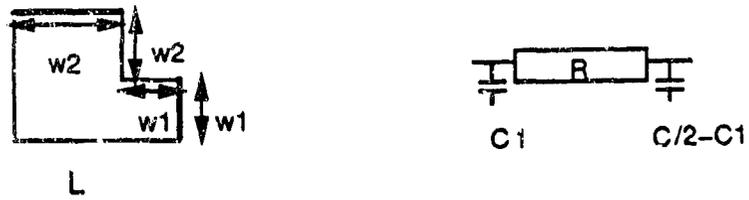


Figure 5.18 L-bend model from JET2.

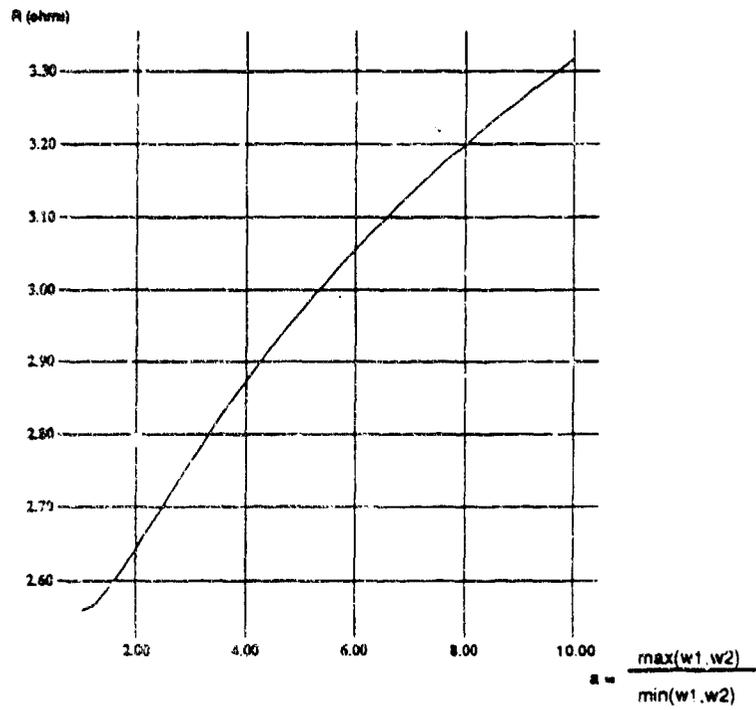


Figure 5.19 L-bend resistance vs. width ratio.

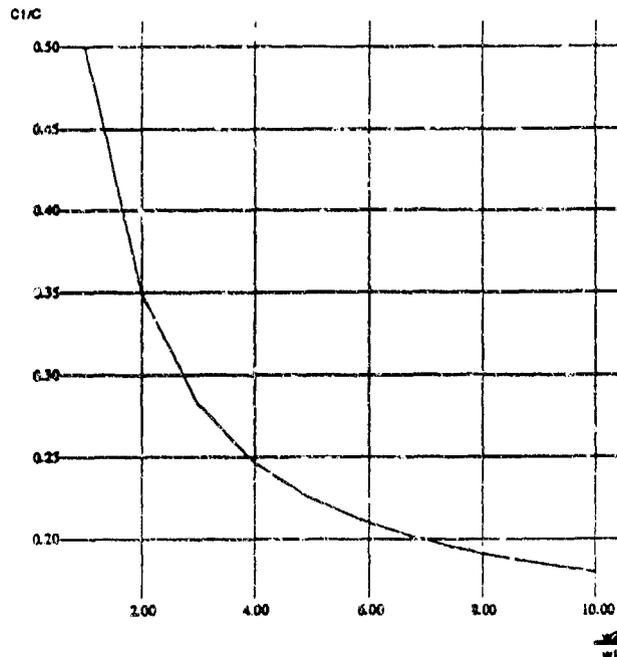


Figure 5.20 C_1/C vs. width ratio (L bend).

integer values of a is shown in Fig. 5.20. They are computed by using the RC network node reduction method mentioned above applied to a fine rectangular mesh. These normalized capacitances are stored in a look-up table indexed by the width ratio. Normalized capacitances for other width ratios are determined by linear interpolation and extrapolation. This table look-up method is also used for the rest of the primitives in determining the Elmore capacitance. The capacitances of the other terminals in the primitives are determined by symmetry and the fact that all of the Elmore capacitances add up to C .

5.7.3 T Primitive

The three-terminal model is shown in Fig. 5.21. Note that because of symmetry, only two conductances have to be computed. The conductances are computed for integer values of w_2/w_1 and w_1/w_2 (plotted in Fig. 5.22) using BEM with a very high number of nodes. These

conductances are stored in a look-up table indexed by the width ratio. Conductances for other ratios are linearly interpolated or extrapolated from table values. This look-up table method is also used for the rest of the primitives. The total capacitance is

$$C = C_{ox}(2 w_1^2 + w_2^2 + w_1 w_2) \quad (5.38)$$

and the normalized Elmore capacitance for the symmetric terminal is plotted in Fig. 5.23 for integers w_2/w_1 and w_1/w_2 . This primitive is an example of the hybrid BEM-FEM approach since the conductances are precomputed by BEM and the capacitances are precomputed by FEM.

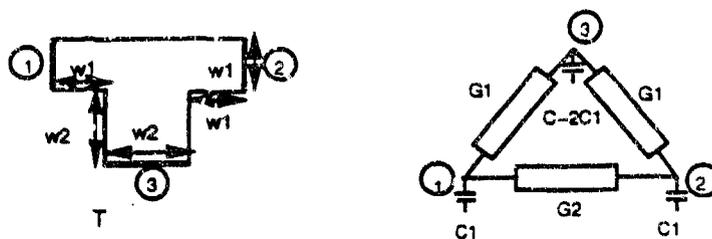


Figure 5.21 T-section model.

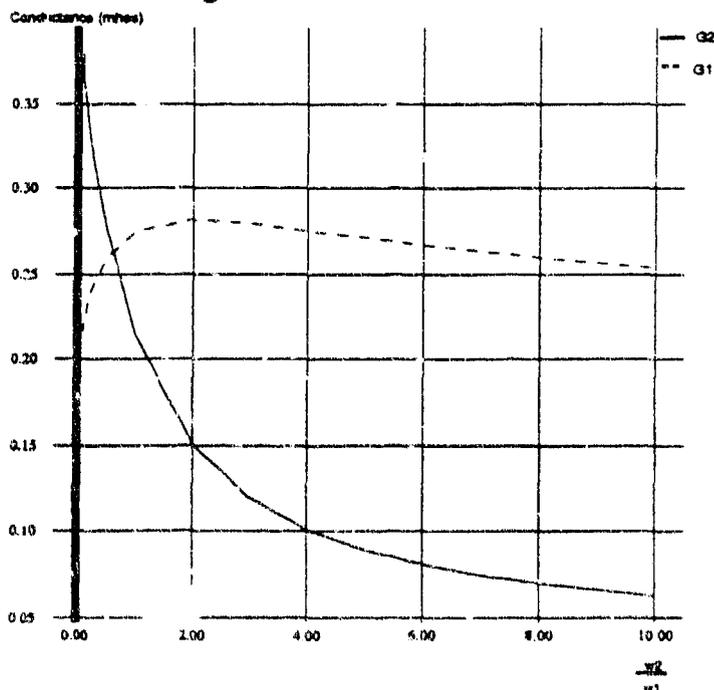


Figure 5.22 T-section conductance vs. width ratio.

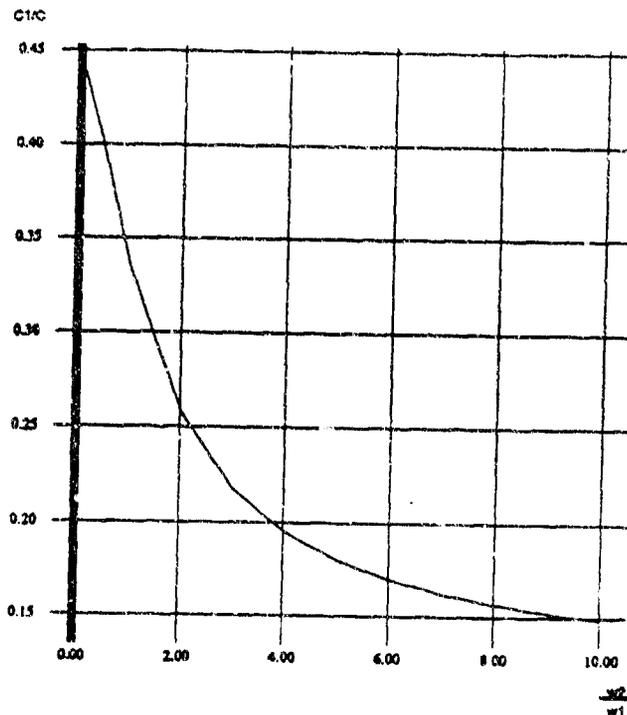


Figure 5.23 C_1/C vs. width ratio (T junction).

5.7.4 Four-Way Junction Primitive

The four-terminal model is shown in Fig. 5.24. Because of symmetry, only three conductances have to be computed. Actually in the look-up table only two conductances have to be stored (G_1 and G_2) because $G_3(w_2, w_1) = G_2(w_1, w_2)$ due to further symmetry. The conductance versus width ratio is plotted in Fig. 5.25. The capacitance is

$$C = C_{ox}(2w_1^2 + w_2^2 + w_1w_2) \quad (5.39)$$

and the normalized Elmore capacitance of the smaller terminal is plotted in Fig. 5.26.

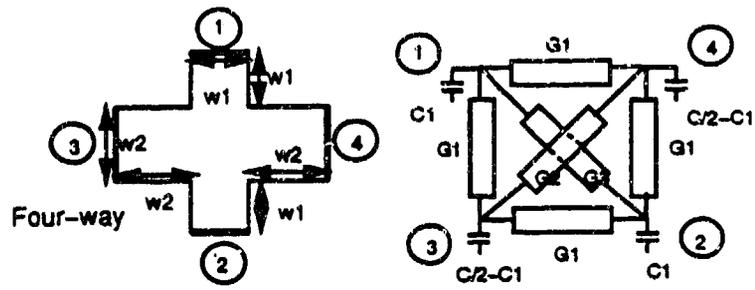


Figure 5.24 Four-way junction model.

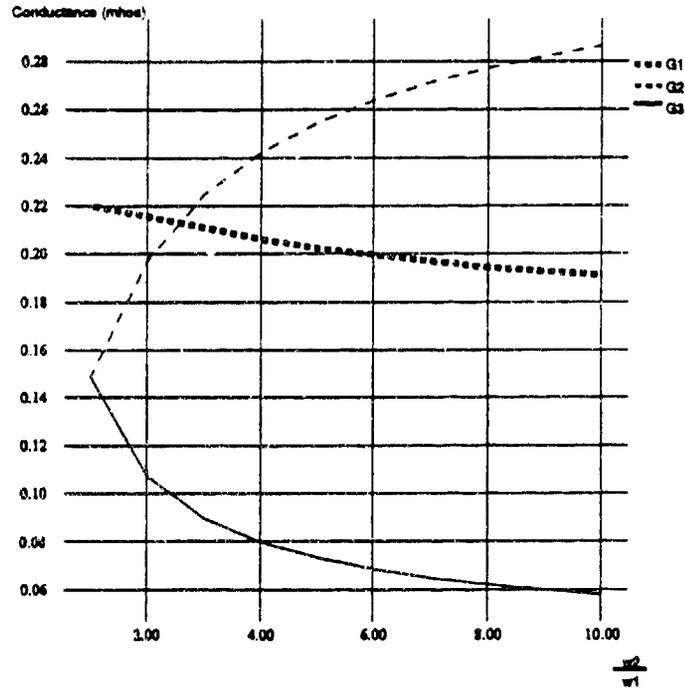


Figure 5.25 Four-way conductance vs. width ratios.

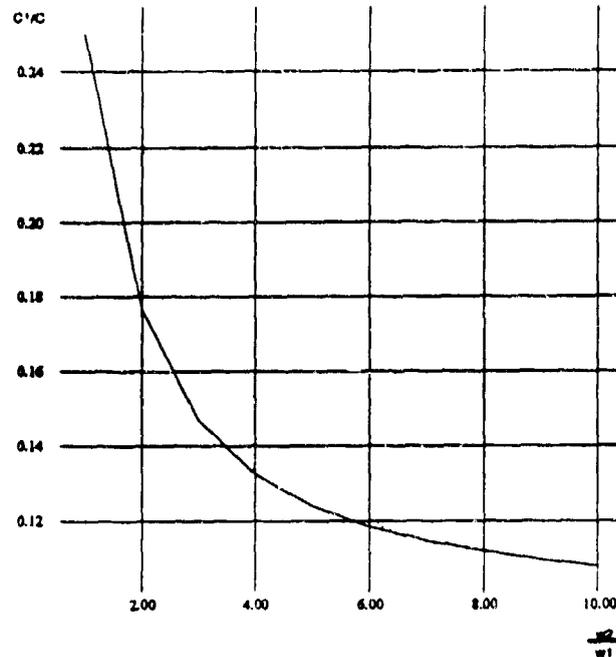


Figure 5.26 C_1/C vs. width ratio (four-way).

5.7.5 Contact Primitive

The three-terminal model is shown in Fig. 5.27. Because of symmetry, only two conductances have to be computed. Besides being dependent on the width ratio, the conductance is also a function of how far the contact is off center, measured by $shift = \min(a/(w-w_c), 1-a/(w-w_c))$, where the meaning of a is shown in Fig. 5.27. $Shift$ thus varies from 0 to 0.5. The conductance versus width ratio and $shift$ is plotted in Fig. 5.28. The total capacitance is

$$C = C_{ox}(w(2w + w_c) - w_c^2) \quad (5.40)$$

and the normalized Elmore capacitance of the center terminal is plotted in Fig. 5.29. Like the conductance, the normalized Elmore capacitance is a function of the $shift$ parameter as well.

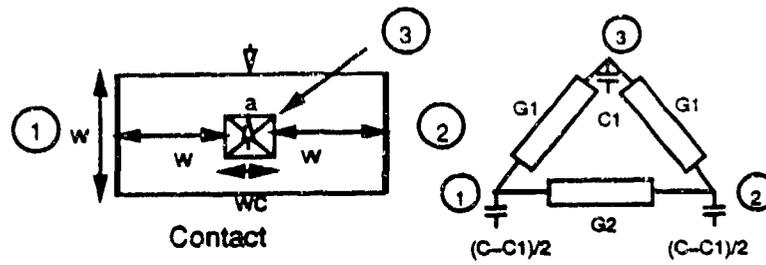


Figure 5.27 Contact primitive.

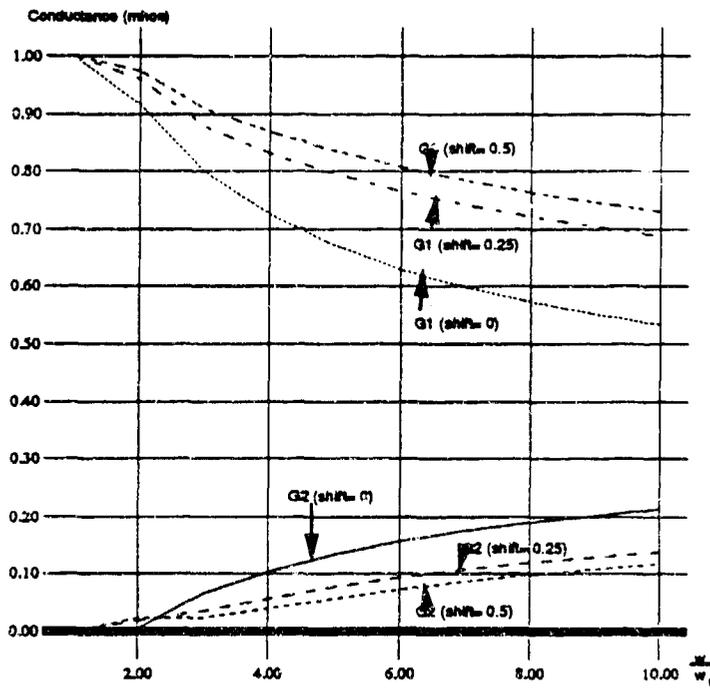


Figure 5.28 Contact primitive conductances vs. width ratio.

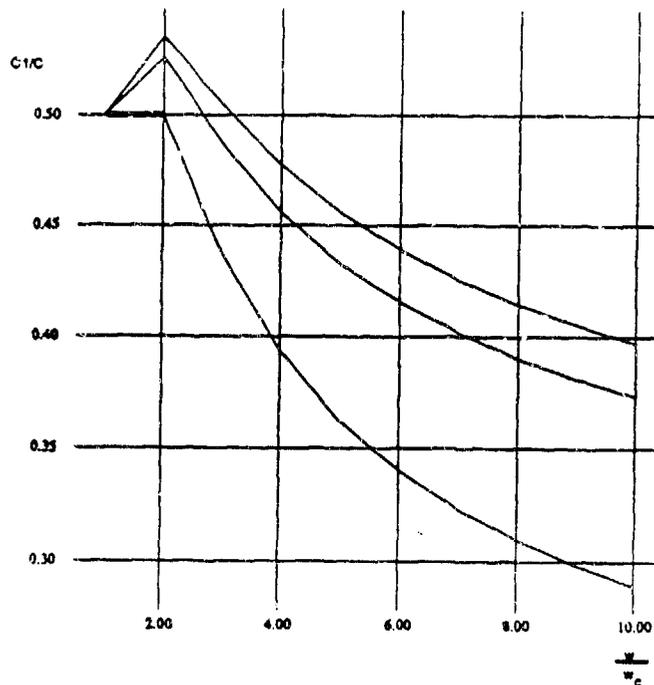


Figure 5.29 $C1/C$ vs. width ratio (contact primitive).

5.8 Algorithms and Data Structures of JET2

5.8.1 Structure of JET2

The basic structure of JET2 is shown in Fig. 5.30 and a more detailed algorithm is given below:

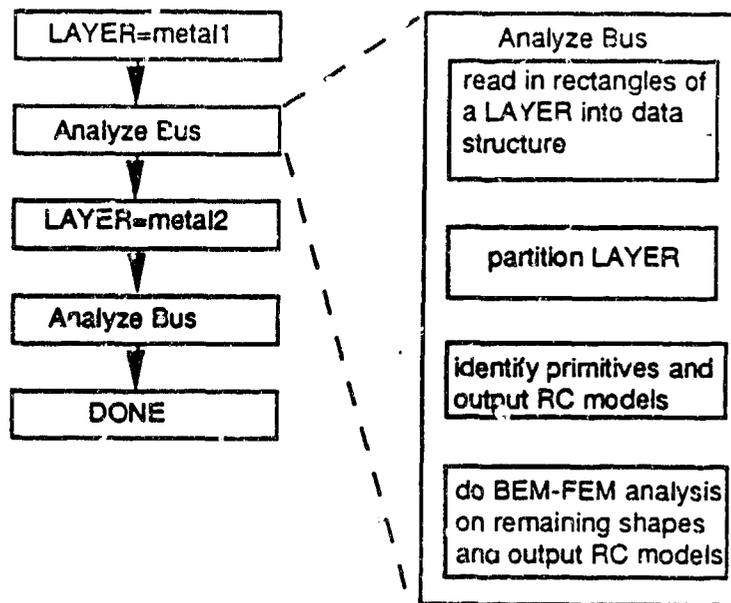


Figure 5.30 Basic structure of JET2.

Main JET2 algorithm

```

{Read technology parameters;
read in metal1, metal2, diffusion contact, vias
duplicate via list;
for each metal layer n
  {break metal where overlaps via and diffusion
  merge contacts and metal lists
  put layout in maximal vertical strip format
  partition layout horizontally
  put layout in maximal horizontal strip format
  partition layout vertically
  assign node number to equipotential lines
  identify primitives and output RC model
  for each nonprimitive
    {determine boundary
    do rough FEM analysis
    do BEM on boundary
    output RC model
    }
  }
}
  
```

A file containing a list of rectangles specifying metal1, metal2 and vias, which comprise the bus as well as the diffusion contacts that connect the bus to transistors, is read into a data structure of rectangles. Vertical and horizontal equipotential lines are selected by examining each rectangle to see if it can be split according to the partitioning criterion given in Section 5.2. During this partitioning step, horizontal and vertical simple resistors are identified, since unlike other primitives, they are composed of only one rectangle and we already examine each rectangle individually in the partitioning step. A horizontal resistor has horizontal current flow and is bounded by vertical equipotential lines. A vertical resistor has vertical current flow and is bounded by horizontal equipotential lines. Node numbers are then assigned to the edges of rectangles that coincide with these partitions. Next, starting with the first rectangle i in the data structure, we start accumulating rectangles that touch this rectangle and each other. Accumulation is stopped when the group of abutting rectangles topologically match a primitive or when it is impossible for the group to match any primitive. Both possibilities are determined by a decision tree. If a group does match a primitive, it is removed from the data structure and has its RC model outputted. Regardless of matching, we go to rectangle $i + 1$ and repeat. Once all primitives have been identified, the data structure is scanned again for groups of rectangles that touch each other and are bounded by equipotential or insulating edges. The boundary of this group is extracted and put into a data structure of points and the rectangles are removed from the data structure. The structure of points is analyzed by a boundary element routine which outputs the structure's RC model. This is done for each group of rectangles until the original data structure has no more noncontact rectangles left.

Once this whole process is done for metal1, it is repeated for metal2. The two circuit netlists for metal1 and metal2 are connected by the node numbers of the vias, which are given

in the input file to JET2. Thus the vias are implicitly treated as perfect nodes; hence, current crowding effects around the vias are ignored. Accurate modeling of vias still has to be done.

The main data structures and algorithms used in the program will now be presented.

5.8.2 Rectangular Data Structure

The layout data for a given layer are read into a doubly linked list of rectangles whose elements are of the form:

```
struct rectlist
{
  int rx, uy, lx, dy;
  char right, left, up, down;
  short node_up, node_down, node_right, node_left;
  char type;
  boolean visited;
  struct rectlist *next, *prev;
}
```

rx and *uy* are the *x* and *y* coordinates, respectively, of the upper-right corner of the rectangle. *lx* and *dy* are coordinates of the lower-left corner. *right*, *left*, *up* and *down* mark each of the four edges of the rectangle as being in 1 of 3 states: INSULATOR, EQUIPOT, or TOUCH. INSULATOR means the edge has no current flowing through it, i.e., it does not touch any other rectangle. EQUIPOT means the edge is an equipotential line. This occurs when either the edge is completely engulfed by a contact or was created by partitioning.

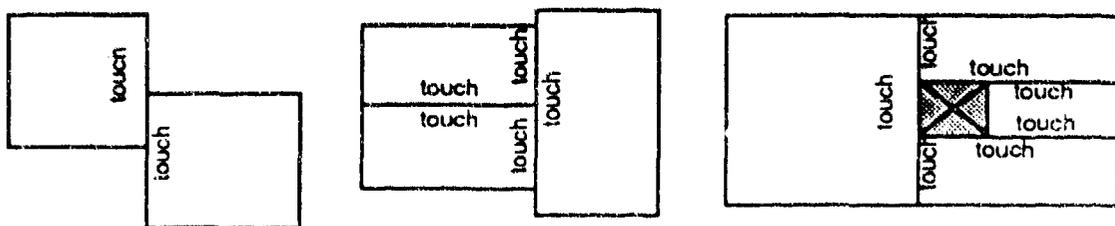


Figure 5.31 Edges with status of TOUCH.

TOUCH refers to any edge not covered by the previous states (Fig. 5.31). A "TOUCH" edge indicates that current bends as it passes through that edge.

`node_up`, `node_down`, `node_right`, `node_left` apply to their respective edges if that edge's status is EQUIPOT. They specify the node in the final RC model that their edge corresponds to. If the edge's status is not EQUIPOT, then the value of `node_**` is NO_NODE_NUM. These node numbers are either already assigned to the edge during data structure initialization if their rectangle is listed as a diffusion contact in the input file or they are assigned after partitioning as mentioned in Section 5.8.1.

`type` takes on the values CONTACT, SIMPLE, and REGULAR which indicate if the whole rectangle is either a contact, simple resistor, or neither. This is necessary since contact rectangle lists are merged with a metal list. `visited` (TRUE or FALSE) indicates, during a recursive traversal of all the rectangles in the list, whether the rectangle has been processed already. `next` and `prev` are pointers to the next and previous rectangles in the list. The first and last elements in the list do not correspond to any rectangle. They are just a header and tail. The data structure field values for a shape are shown in Fig. 5.32.

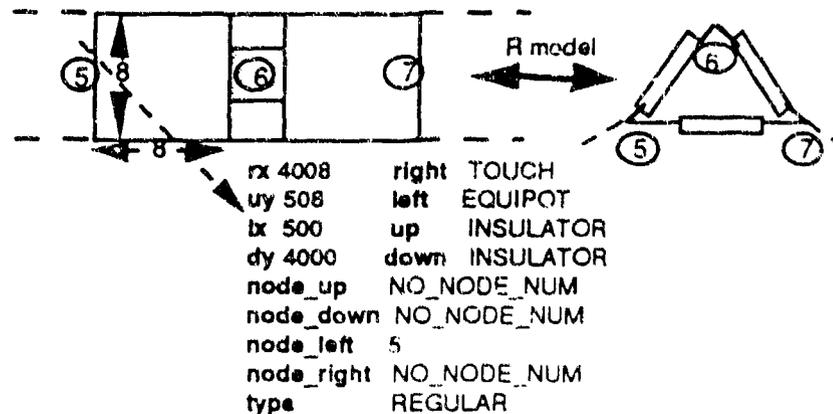
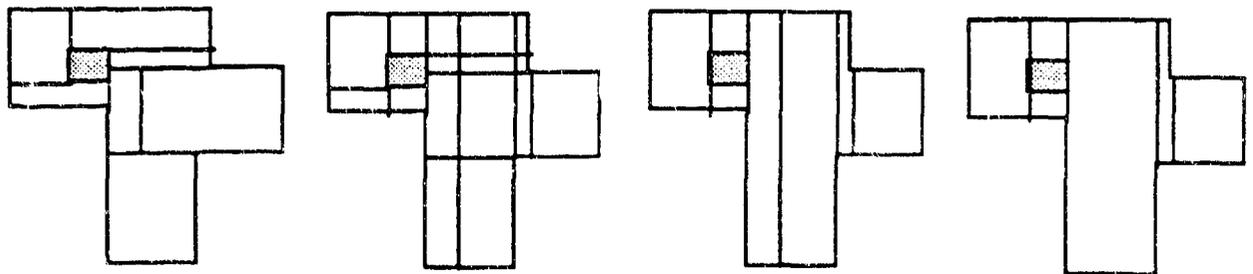


Figure 5.32 Values of fields in *rectlist* data structure for typical rectangle.

5.8.3 Maximal Vertical and Horizontal Strip Formats

To horizontally partition the layout with vertical equipotential lines, it is necessary to first put the metal layout into maximal vertical strip (MVS) format. The two properties of MVS format are: 1) all the rectangles of one layer are as vertically tall as possible (i.e., no rectangle touches any other rectangle of the same layer on the top or bottom edge) and 2) these vertical rectangles are as wide as possible. Figure 5.33(a) shows a metal layout not in MVS format. Figure 5.33(d) shows the same layout in MVS format. The algorithm for putting an arbitrary layer of rectangles (given by the data structure in the previous section) into MVS format [51] is given below with the intermediate steps shown in Fig. 5.33(a)-(d):



(a) Initially not MVS (b) split vertically and horizontally (c) combine vertically (d) combine horizontally

Figure 5.33 Steps in MVS algorithm.

Procedure MVS

1. For each rectangle, split current rectangle into left and right pieces if either left or right edge of another rectangle is within the left and right limits of current rectangle.

2. For each rectangle, split current rectangle into upper and lower pieces if either upper or lower edge of another rectangle is within upper and lower limits of current rectangle.
3. Combine rectangles in y -direction as much as possible.
4. Combine rectangles in x -direction as much as possible.

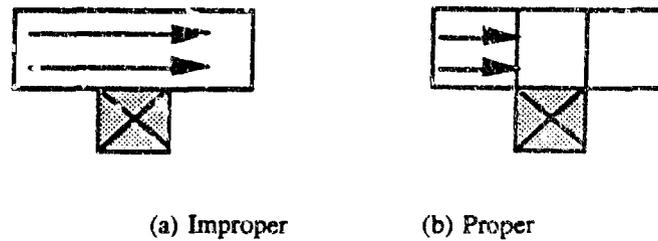


Figure 5.34 Exception to Step 4 of MVS algorithm.

Property 1 maximizes the vertical cross-sectional area that the horizontal current flows into and ensures that the top and bottom edges are not touched by any other rectangle of the same layer. Property 2 ensures that the left and right edges of the rectangle are bordered either by an insulating or equipotential region or a rectangle of the same layer that causes a bend in the current flow. The only problem is that a bend in the current flow can also be caused by a contact, which is a rectangle in another layer, as shown in Fig. 5.34(a). Thus step 4 of the MVS algorithm where the rectangles are combined horizontally if possible has to be modified to take care of this case. The correct partition is shown in 5.34(b). Now the top and bottom edges of each rectangle are guaranteed not to have a status of TOUCH.

After MVS formatting, any rectangle with a top and bottom edge status of INSULATOR is a candidate for partitioning by vertical lines as the various cases of Fig. 5.35 show.

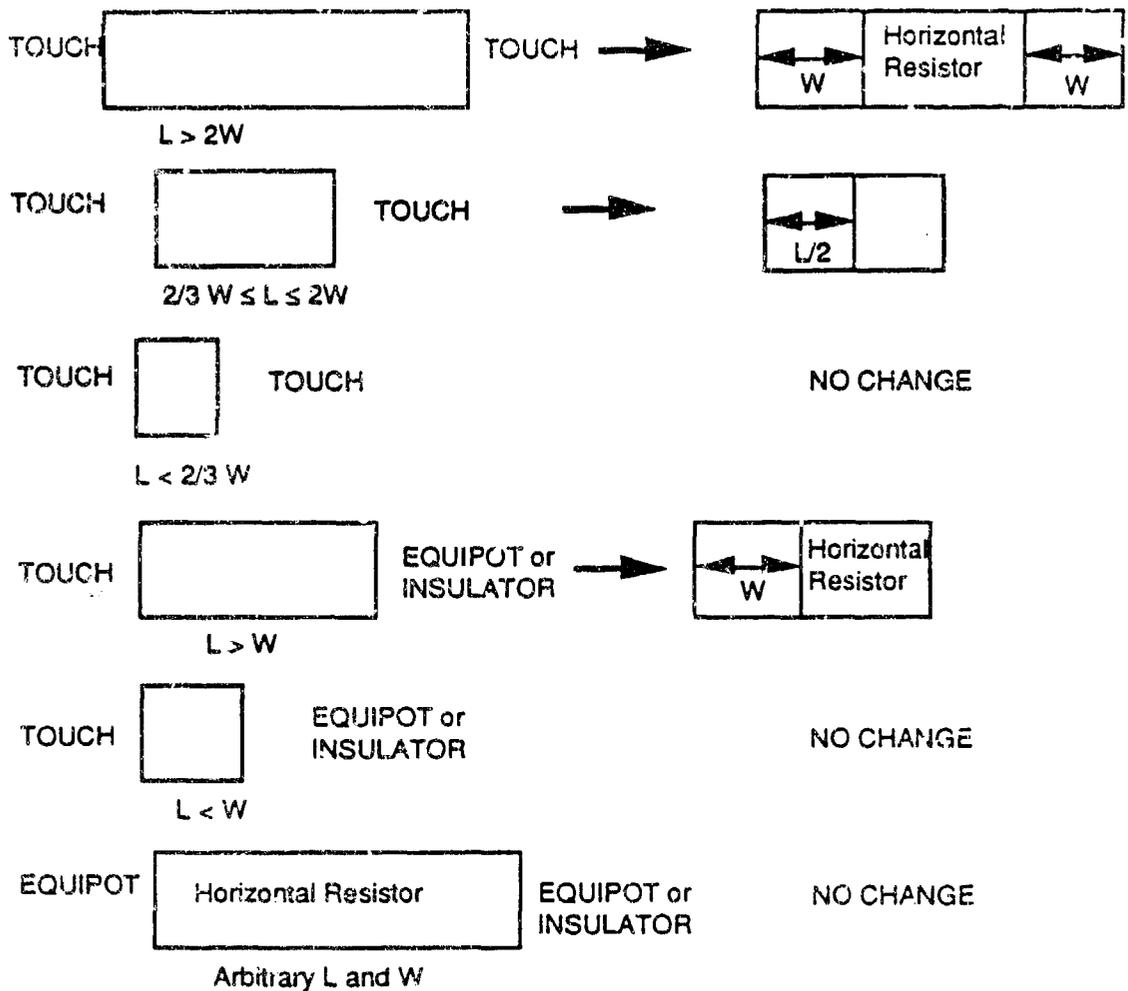


Figure 5.35 Partitioning of various cases.

Vertical partitioning uses maximal horizontal strip (MHS) format, which is completely analogous to MVS. The only difference is that since MHS formatting is done after MVS formatting and horizontal partitioning, it is not allowed to split or combine any horizontal resistors, thus undoing previous work.

5.8.4 Primitive Identification

As mentioned before, primitive identification is done on the fly as adjacent rectangles are accumulated. This saves time by avoiding the further search of adjacent rectangles when the accumulated group is clearly not a primitive any longer. Searching for a rectangle that touches the current one is an $O(N^{0.5})$ process in the average case for a layout of N rectangles and is an $O(N)$ process in the worst case and should be avoided when possible. A binary decision tree/algorithm does this identification for the noncontact primitive topologies in MHS format given in Fig. 5.36. Each added rectangle causes a branch in the tree to be taken and possible matches to certain primitives to be eliminated. If a match is found, the group of rectangles is removed from the list and matching continues with the next rectangle on the list. The contact primitive topologies shown in Fig. 5.37 are handled in a totally separate search of the data structure and have their own decision trees for the sake of simplicity and debugging.

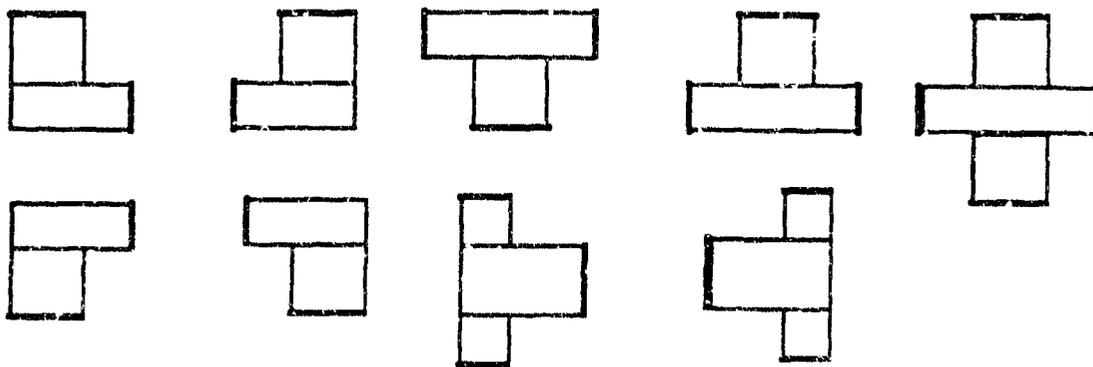


Figure 5.36 MHS form of all possible topologies of noncontact primitives.

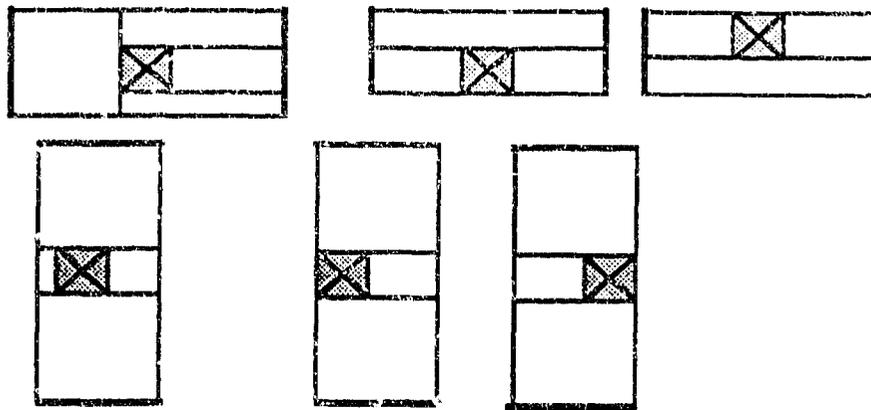


Figure 5.37 MHS form of all possible topologies of contact primitive.

5.3.5 Contour Data Structure

The contour of the remaining nonprimitives is stored in the following data structure, which is visually represented in Fig. 5.38:

```

struct contour {Boolean is_contact;
                int portnum;
                struct point firstpoint, lastpoint;
                }
struct point {int x, y;
              short nodenum,numprevious,numnext;
              boolean corner, nextq;
              signed char fluxdir;
              }

```

As seen in Fig. 5.38, the total contour is a linked list of individual contours, with the first contour being the external boundary and the remaining contours, if any, being holes and internal contacts. Each contour is a circularly linked list of points, with the external boundary points stored clockwise and other points stored counterclockwise. The difference in orientation is necessary for proper evaluation of the contour integrals. **is_contact** is TRUE if the contour is a contact. **firstpoint** and **lastpoint** point to the first and last points of the contour. **portnum** indicates how many terminals the contour has.

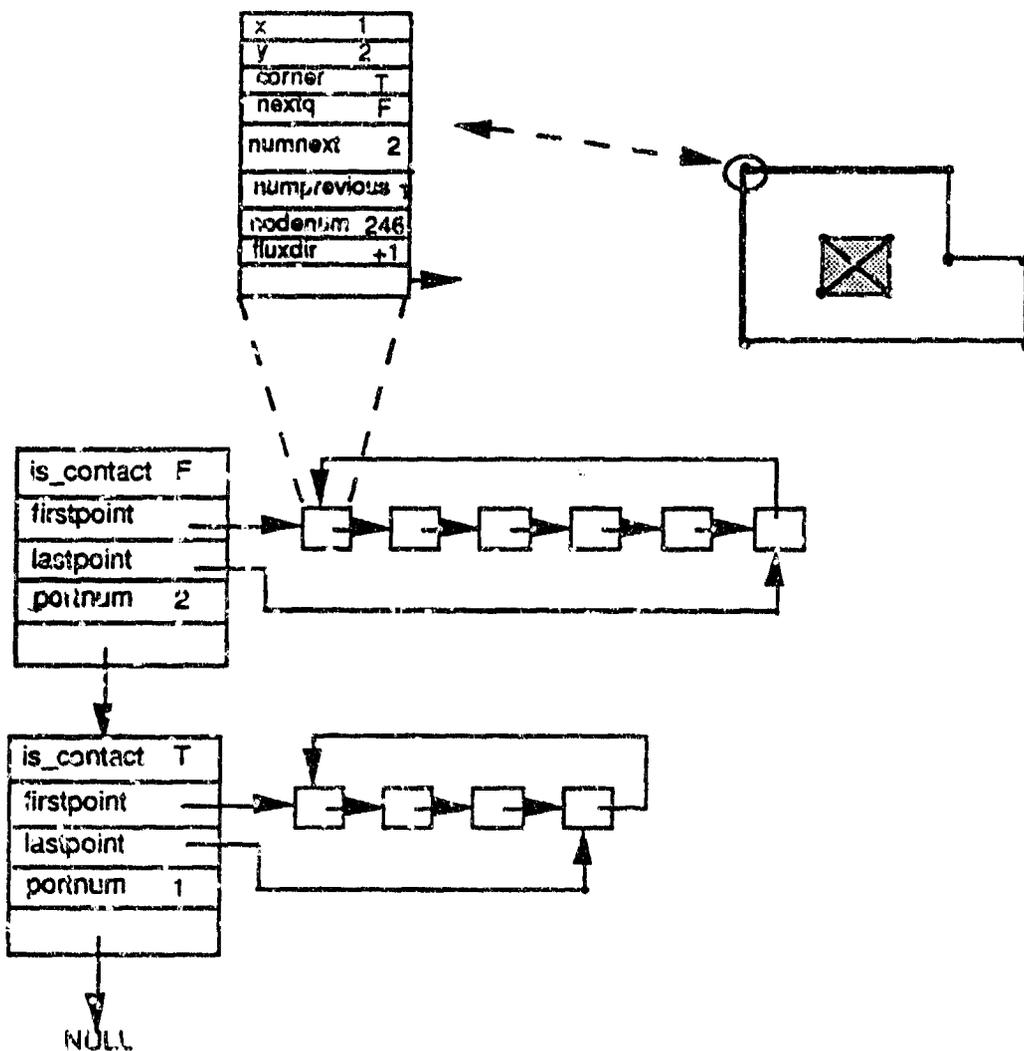


Figure 5.38 Contour data structure for a non-primitive.

`x` and `y` are the coordinates of the point. `node_num` is the node number in the final RC model of the node that the point belongs to. It equals `NO_NODE_NUM` if the point belongs to an insulating edge. When BEM is applied, a system of linear equations is formed with each point corresponding to an unknown. `numnext` is the index of the unknown belonging to that point. For double nodes (Section 5.4.2) with two unknowns, `numnext` is the index for q_{j0} and `numprevious` of q_{j1} (Fig. 5.9). `corner` is TRUE if the point is at a corner. `nextq` is

TRUE if the contour segment between the point and the next point is a conducting edge, i.e., q is unknown on that segment. **fluxdir** = 1 if the outward normal n on the segment between the point and the next point is i_x or i_y ; otherwise, **fluxdir** = -1. **next** points to the next point on the contour. The contour of a group of abutting rectangles is produced by the following algorithm from [60]:

1. Place vertical edges of all rectangles in a list (an edge is two points with the first point pointing to the second point and the second point being clockwise on the rectangle to the first).
2. Sort points in the list according to increasing y-coordinate.
3. Sort groups of points with same y-coordinate according to increasing x-coordinate.
4. Link point $2k - 1$ with point $2k$ for $1 \leq l \leq N/2$ (N is the number of points) to form the horizontal edges.

This algorithm is $O(N \log N)$ since sorting is asymptotically the most expensive step.

5.8.6 FEM Analysis

To do a rough finite element analysis on the nonprimitives, the abutting group of rectangles in the contour extraction method of is removed from the main list and placed in to their own list. Then they are broken into smaller rectangles according to steps 1 and 2 of the MVS procedure. After each corner is numbered, the FEM method of Section 5.5 is used to form an RC model, which is then reduced using the Elmore time constant-preserving node reduction technique of Section 5.6.

5.8.7 BEM Algorithm

Given below is the BEM algorithm, executed after the contour formation step:

Procedure BEM

if number of terminals $P \geq 2$

```
{ multiply all coordinates in the contour by 3;
  divide each contour segment into 3 segments;
  clear matrix A (N x N) and source matrix B (N x P);
  for each delta source p at a node on the contour
  for each observation point q at a node on the contour
    { form matrix A coefficient;
      form contribution to source vectors on left;
    }
  solve matrix equation AX=B;
  determine and output conductances from matrix X;
}
```

The contour coordinates are multiplied by 3; since they are stored as integers, dividing a boundary segment into three pieces must yield integer coordinates also.

6. SUMMARY AND CONCLUSIONS

In this report, we have described our progress on the development of current estimation techniques for reliability analysis of VLSI circuits.

In Chapter 2, we have described our progress in developing and implementing the probabilistic simulation approach for estimating average and variance current waveform drawn by digital CMOS gates at contact point to the power bus for electromigration analysis and for average power estimation. Improvements in both computational speed and accuracy have been made. In particular, a new method of calculating the stochastic properties of the equivalent edge of a channel-connected subcircuit during elimination has been derived, which improves the accuracy of the estimation. The calculated expected value for all the examples we simulated was within 3% of the expected value calculated through exhaustive SPICE-like simulation, while our previous method was at times 50% off. For the variance our method was within 12% of the exhaustively calculated value, while the previous method was an order of magnitude away in the worst case. The calculation of the stochastic quantities of the equivalent edge during switching is performed in a different fashion. The new method is straightforward and, as the examples show, more accurate than the previous one. In addition, the calculation of the delay through a gate is treated from a different perspective, producing more accurate results compared to the previous method. Finally, with the introduction of the correlation coefficient method into probabilistic simulation the important issue of signal dependencies is addressed and, although some error remains, execution time and memory usage are kept within reasonable bounds, which permits the simulation of large circuits. The approach has been implemented in program iProbe-c. We are currently working on extending the approach to include current estimation in sequential circuits, gates with pass transistors,

and mixed-signal designs.

In Chapter 3, we have described the application of probabilistic simulation techniques for estimating hot-carrier induced degradation within the transistors of VLSI CMOS digital circuits. Since HCE in a transistor depends on the current flowing through it when the transistor is operating in the saturation region, which in turn, is a function of the input signal slew rate, we have extended the probabilistic approach to include signal slew rate. We have implemented the method in another version of iProbe, called iProbe-d. We have also proposed a redesign strategy, based on the simulation results, that calls for interchanging signal connections, depending on the average switching frequencies of the signals, at inputs to transistors connected in series to reduce overall HCE degradation. Since HCE degradation affects the timing of a circuit, we have also combined HCE degradation estimation with critical path analysis. The aim is to analyze HCE caused degradation effects on timing along critical paths in the design and recommend redesign strategies to optimize long term circuit performance. More work needs to be done in this area.

In Chapter 4, we have described a linear time algorithm (**iMax**) that computes maximum currents in the supply lines. Most of the previous algorithms on maximum current estimation suffer from exponential complexity and are not adequate for large circuits. Our approach avoids exponential complexity by adopting a pattern independent approach. The results produced by the algorithm are within acceptable bounds for most circuits. We have also presented a new *partial input enumeration* algorithm that partially resolves the signal correlations and further improves the upper bound obtained from **iMax**. The algorithm is based on the *best first search* (BFS) technique and represents a good time-accuracy trade-off. The **PIE** algorithm involves a search procedure, but this search need not be carried too deep to obtain

good results. The algorithm is quite applicable to VLSI circuits, as is demonstrated by the experimental results on circuits with up to 22,000 gates. In our future research, we plan to extend the study to include better gate delay and current models and to identify troublesome voltage drop sites in supply lines, using RC models, from the maximum current estimates.

In Chapter 5, we explain the methods used in the RC bus extractor JET2. Unlike our previous extractor on which it is based, JET2 does not require human intervention to edit sections of the layout to complete the extraction. It does this by using the BEM with a uniform grid of three linear boundary elements per edge to model the multiport conductances of a bus segment. It uses finite element analysis and an Elmore time constant-preserving node reduction technique to model the distributed capacitance of bus segments. For primitives, the resulting lumped capacitance at each node as a percentage of the total capacitance to ground is stored in a look-up table. The lumped capacitance for nonprimitives is done on-the-fly using FEM with a coarse grid. These methods lead to more accurate models for both the primitives and nonprimitives.

JET2 can still be improved. Most of the runtime on large layouts is spent on the BEM and specifically on the computation of the boundary element coefficients rather than geometric processing of the layout and primitive identification. This can be reduced by either having a larger static table of primitives or storing shapes in a dynamic table as they are encountered to build a set of primitives particular to that layout. Another solution would be to improve the calculation time of the coefficients. Numerical integration via Gaussian quadrature might be effective in trading some speed for accuracy. Another improvement is the handling of non-Manhattan geometries, most probably those with a fixed number of slopes. This would require major changes in the data structure and geometric preprocessing algorithms. The BEM is

however well-suited to nonrectilinear geometries and would require minor changes in the functions that compute the linear system coefficients. The rough FEM analysis however would need a full-blown triangularization algorithm. Finally as feature sizes scale down, accurate 3-D capacitance models will be needed, especially as coupling capacitance become more important. JET2 has only a crude 3-D model for an interconnect that can be modeled as a simple resistor. This model does not take coupling capacitance into account.

In summary, this work has focused mainly on analysis and modeling issues, which are the first steps needed for including physical reliability issues in large circuit design. Even though more work still needs to be done on improving the analysis and modeling techniques, the next step is to integrate the results into a design system so that reliability measures can be included at all levels in the design process, together with other design objectives, such as timing, area, and system performance.

7. PUBLICATIONS AND REFERENCES

- [1] F. Najm, R. Burch, P. Yang, and I. Hajj, "Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 9, no. 4, pp. 439-450, April 1990.
- [2] F. N. Najm, I. N. Hajj and P. Yang, "An Extension of Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 10, no. 11, pp. 1372-1381, November, 1991.
- [3] Athanasios Papoulis, *Probability, Random Variables and Stochastic Processes*, McGraw-Hill Book Co., 2d edition, 1984.
- [4] G. I. Stamoulis and I. N. Hajj, "Improved Techniques for Probabilistic Simulation Including Signal Correlation Effects," *Proc. 30th ACM/IEEE Design Automation Conference*, Dallas, TX, pp. 379-383, June 1993.
- [5] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and P. Ricco, "Estimate of Signal Probabilities in Combinational Logic Networks," *IEEE European Test Conference*, pp. 132-138, 1989.
- [6] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," *Proc. 1985 International Symposium on Circuits and Systems*, 1985.
- [7] G. I. Stamoulis, I. N. Hajj, and F. N. Najm, "Techniques for Probabilistic Simulation of VLSI Circuits," submitted to *Transactions on Computer-aided Design*.
- [8] S. Aur, D. Hocevar, and P. Yang, "Hotron - A Circuit Hot Electron Effect Simulator," *Proc. IEEE ICCAD*, pp. 256-259, November 1987.
- [9] W. J. Hsu, S. M. Gowda, and B. J. Sheu, "VLSI Circuit Design with Built-in Reliability using Simulation Techniques," *Proc. IEEE CICC*, pp. 19.3.1-19.3.4, May 1990.
- [10] Y. Leblebici and S. M. Kang, "An Integrated Hot-Carrier Degradation Simulator for VLSI Reliability Analysis," *Proc. IEEE ICCAD*, pp. 400-403, November 1990.
- [11] M. M. Kuo, K. Seki, P. M. Lee, J. Y. Choi, P. K. Ko, and C. Hu, "Simulation of MOSFET Lifetime under AC Hot-Electron Stress," *IEEE Transactions Electron Devices*, Vol. 35, pp. 1004-1011, July 1988.
- [12] Y. Leblebici, P. C. Li, S. M. Kang, and I. N. Hajj, "Hierarchical Simulation of Hot-Carrier Induced Damages in VLSI Circuits," *Proc. IEEE CICC*, pp. 29.3.1-29.3.4, May 1991.

- [13] Y.-H. Shih, Y. Leblebici, and S. M. Kang, "New Simulation Methods for MOS VLSI Timing and Reliability," *Proc. IEEE ICCAD*, pp. 162-165, November 1991.
- [14] C. Hu, S. Tam, F. C. Hsu, P. K. Ko, F. Y. Chan, and K. W. Terrill, "Hot-Electron-Induced MOSFET Degradation - Model, Monitor, and Improvement," *IEEE Transactions Electron Devices*, Vol. ED-32, pp. 375-384, February 1985.
- [15] B. Doyle, M. Bourcerie, J. Marchetaux, and A. Bouldou, "Interface State Creation and Charge Trapping in the Medium-to-High Gate Voltage Range ($v_d/2 \geq v_g \geq v_d$) During Hot-Carrier Stressing of n-MOS Transistors," *IEEE Transactions Electron Devices*, pp. 744-754, March 1990.
- [16] D. B. Jackson, D. A. Bell, B. S. Doyle, B. J. Fishbein, and D. B. Krakauer, "Transistor Hot Carrier Reliability Assurance in CMOS Technologies," *Digital Technical Journal*, pp. 100-113, Spring 1992.
- [17] W. Weber, L. Risch, W. Krautschneider, and Q. Wang, "Hot-Carrier Degradation of CMOS-Inverters," *IEDM Technical Digest*, pp. 208-211, 1988.
- [18] W. Weber, "Dynamic Stress Experiments for Understanding Hot-Carrier Degradation Phenomena," *IEEE Transactions Electron Devices*, Vol. 35, pp. 1476-1486, September 1988.
- [19] P. M. Lee, M. M. Kuo, K. Seki, P. K. Ko, and C. Hu, "Circuit Aging Simulator (CAS)," *IEEES Technical Digest*, pp. 134-137, December 1988.
- [20] Y. Leblebici, W. Sun, and S. M. Kang, "Parametric Macromodeling of Hot-Carrier Induced Dynamic Degradation in MOS VLSI Circuits," *IEEE Transactions Electron Devices*, Vol. 40, no. 3, pp. 673-676, March 1993.
- [21] E. R. Minami, K. N. Quader, P. K. Ko, and C. Hu, "Prediction of Hot-Carrier Degradation in Digital CMOS VLSI by Timing Simulation," *IEDM Technical Digest*, pp. 20.5.1-20.5.4, December 1992.
- [22] D. Overhauser and I. N. Hajj, "A Tabular Macromodeling Approach to Fast Timing Simulation Including Parasitics," *Proc. IEEE ICCAD*, November 1988.
- [23] A. I. Kayssi, K. A. Sakallah, and T. M. Burks, "Analytical Transient Response of CMOS Inverters," *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, Vol. 39, pp. 42-45, January 1992.
- [24] P. C. Li, G. I. Stamoulis, and I. N. Hajj, "A Probabilistic Timing Approach to Hot-Carrier Effect Estimation," *Proc. IEEE ICCAD*, pp. 210-213, November 1992.

- [25] F. Hsu and S. Tam, "Relationship Between MOSFET Degradation and Hot-Electron-Induced Interface-State Generation," *IEEE Electron Device Letters*, pp. 50-52, February 1984.
- [26] Y. Leblebici and S. M. Kang, "A One-Dimensional MOSFET Model for Simulation of Hot-Carrier Induced Device and Circuit Degradation," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 109-112, May 1990.
- [27] T. I. Kirkpatrick and N. R. Clark, "Pert as an Aid to Logic Design," *IBM Journal of Research and Development*, pp. 135-141, March 1966.
- [28] P. McGeer and R. K. Brayton, "Efficient Algorithms for Computing the Longest Viable Path in a Combinational Network," *Proc. ACM/IEEE Design Automation Conference*, pp. 561-567, July 1989.
- [29] Y. C. Ju and R. A. Saleh, "Incremental Techniques for the Identification of Statically Sensitizable Critical Paths," *Proc. ACM/IEEE Design Automation Conference*, pp. 541-546, June 1991.
- [30] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computer*, Vol. C-35, pp. 677-691, August 1986.
- [31] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient Implementation of a BDD Package," *Proc. ACM/IEEE Design Automation Conference*, pp. 40-45, July 1990.
- [32] H. J. Park, K. Lee, and C. K. Kim, "A New CMOS NAND Logic Circuit for Reducing Hot-Carrier Problems," *IEEE Journal of Solid-State Circuits*, Vol. 24, pp. 1041-1047, August 1989.
- [33] T. Sakurai, et al., "Hot-Carrier Generation in Submicrometer VLSI Environment," *IEEE Journal of Solid-State Circuits*, Vol. SC-21, pp. 187-192, February 1986.
- [34] L. Bruni, G. Buonanno, and D. Sciuto, "Transistor Stuck-at and Delay Faults Detection in Static and Dynamic CMOS Combinational Gates," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 431-434, May 1992.
- [35] S. Chowdhury and J. S. Barkatullah, "Estimation of Maximum Currents in MOS IC Logic Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 9, no. 6, pp. 642-654, June 1990.
- [36] F. Rouatbi, B. Haroun, and A. J. Al-Khalili, "Power Estimation Tool for Sub-Micron CMOS VLSI Circuits," *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, pp. 204-209, Santa Clara, CA, November 8-12, 1992.

- [37] A. Nabavi-Lishi and N. Rumin, "Delay and Bus Current Evaluation in CMOS Logic Circuits," *Proc. of IEEE/ACM International Conference on Computer-Aided Design*, pp. 198-203, Santa Clara, CA, November 8-12, 1992.
- [38] U. Jagau, "SIMCURRENT - An Efficient Program for the Estimation of the Current Flow of Complex CMOS Circuits," *Proc. of IEEE International Conference on Computer-Aided Design*, pp. 396-399, Santa Clara, CA, November 11-15, 1990.
- [39] A.-C. Deng, Y.-C. Shiau, and K.-H. Loh, "Time Domain Current Waveform Simulation of CMOS Circuits," *Proc. of IEEE International Conference on Computer-Aided Design*, pp. 208-211, Santa Clara, CA, November 7-10, 1988.
- [40] S. Devadas, K. Kentzer, and J. White, "Estimation of Power Dissipation in CMOS Combinational Circuits using Boolean Function Manipulation," *IEEE Transactions on Computer-Aided Design*, no. 3, pp. 373-383, March 1992.
- [41] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, no. 4598, pp. 671-680, May 13, 1983.
- [42] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. Norwell, MA: Kluwer Academic Publishers, 1987.
- [43] H. Kriplani, F. Najm, and I. Hajj, "Maximum Current Estimation in CMOS Circuits," *Proc. of 29th ACM/IEEE Design Automation Conference*, pp. 2-7, Anaheim, CA, June 8-12, 1992.
- [44] S. Seth, L. Pan, and V. D. Agrawal, "Predict - Probabilistic Estimation of Digital Circuit Testability," *The 15th International Symposium on Fault Tolerant Computing*, pp. 220-225, Ann Arbor, MI, June 19-21, 1985.
- [45] F. Maamari and J. Rajski, "A Reconvergent Fanout Analysis for Efficient Exact Fault Simulation of Combinational Circuits," *IEEE 18th International Symposium on Fault Tolerant Computing*, pp. 122-127, Tokyo, Japan, June 27-30, 1988.
- [46] S. Dey, F. Brglez, and G. Kedem, "Corolla Based Circuit Partitioning and Resynthesis," *Proc. of 27th ACM/IEEE Design Automation Conference*, pp. 607-612, Orlando, FL, June 24-28, 1990.
- [47] J. Pearl, *Heuristics - Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley, 1984.
- [48] F. Brglez, D. Bryan, and K. Koźmiński, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. of International Symposium on Circuits and Systems*, pp. 1929-1934, May 1989.

- [49] R. Geiger, P. Allen, and N. Strader, *VLSI Design Techniques for Analog and Digital Circuits*. New York: McGraw-Hill Publishing Co., p. 45, 1990.
- [50] M. Horowitz and R. W. Dutton, "Resistance Extraction from Mask Layout Data," *IEEE Transactions on Computer-Aided Design*, Vol. CAD-2, no. 3, pp. 145-150, July 1983.
- [51] H. Cha, "Current Density Calculation using Rectilinear Region Splitting Algorithm for Very Large Scale Integration Metal Migration Analysis. M.S. thesis, University of Illinois at Urbana-Champaign, UIIU-ENG-90-2223, August 1990.
- [52] K. E. Gustafson, *Introduction to Partial Differential Equations and Hilbert Space Methods*. New York: John Wiley and Sons, pp. 38-39, 1980.
- [53] C. Pearson, *Numerical Methods in Engineering and Science*. New York: Van Nostrand Reinhold Company, pp. 178-180, 1986.
- [54] J. J. Connor and C. A. Erebbia, "Boundary Element Methods," in *Boundary Element Techniques in Computer-Aided Engineering*, C.A. Brebbia, Ed. New York: Martinus Nijhoff Publishers, 1984.
- [55] Z. Wang and Q. Wu, "Two-Dimensional Resistance Simulator using the Boundary Element Method," *IEEE Transactions on Computer-Aided Design*, Vol. II, no. 4, pp. 497-504, April 1992.
- [56] A. J. van Genderen, *Reduced Models for the Behavior of VLSI Circuits*. Netherlands: Delft University Press, pp. 9-50, 1991.
- [57] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics*, Vol. 19, pp. 55-63, January 1948.
- [58] C. P. Yuan and T. N. Trick, "A Simple Formula for the Estimation of the Capacitance of Two-Dimensional Interconnects in VLSI Circuits," *IEEE Electron Device Letters*, Vol. EDL-3, no. 12, pp. 391-393, December 1982.
- [59] P. M. Hall, "Resistance Extraction for Thin Film Patterns," *Thin Solid Films*, Vol. 1, pp. 277-295, March 1968.
- [60] D. L. Souvaine and I. Bjorling-Sachs, "The Connor Problem for Restricted-Oriented Polygons," *Proc. of the IEEE*, Vol. 80, p. 1450, September 1992.
- [61] G. D. Kamath, "RC Extraction of VLSI Power Bus using Primitive Splitting, Finite Element Analysis, and the Boundary Element Method. M.S. thesis, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, June 1993.