

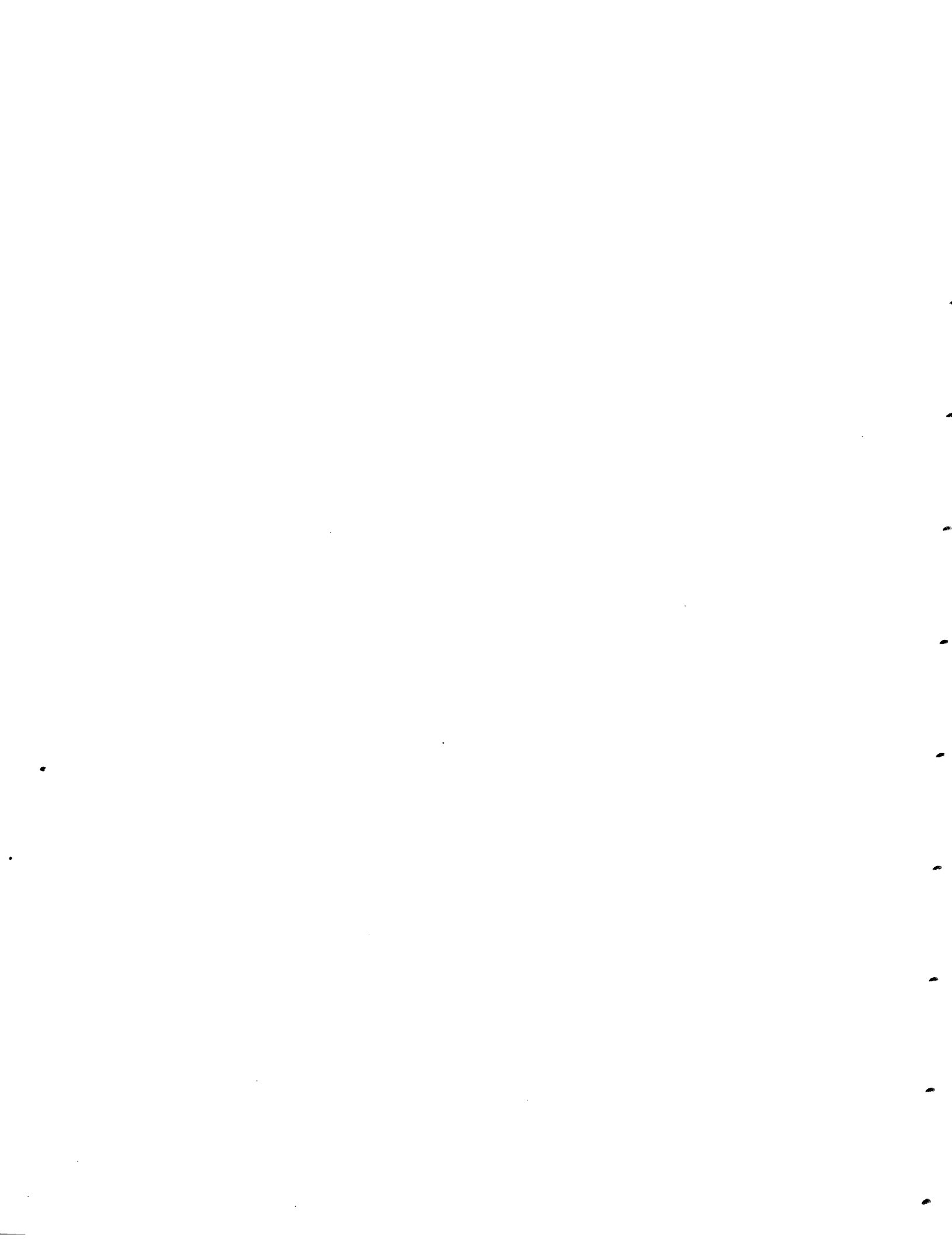
# ***Modeling and Simulation of VLSI Interconnections with Moments***

***RLE Technical Report No. 543***

***May 1989***

Steven Paul McCormick

**Research Laboratory of Electronics  
Massachusetts Institute of Technology  
Cambridge, MA 02139 USA**



# Modeling and Simulation of VLSI Interconnections with Moments

by

Steven Paul McCormick

Submitted to the Department of Electrical Engineering and Computer Science  
on March 17, 1989, in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

## Abstract

This thesis presents a new CAD simulation method for determining waveform estimates of MOS circuits. The methods are particularly useful in determining the delay times and coupling noise voltages of interconnection networks. Additionally, switching devices are also accurately emulated with a macromodeled equivalent circuit.

The moment representation, based on the Laplace Transform, is used as the model for both signals and transfer functions of interconnection networks. A large interconnection network can be modeled by a small number of polynomial terms, making network analysis much simpler, computationally. A new matrix algorithm is developed for solving for the moment representation of almost any linear network. Transistor switching circuits are modeled by macromodeled linear network equivalents.

On CMOS test circuits, after several logic stages the moment representation waveforms deviated no more than 10% from SPICE's waveforms, but were computed substantially faster than SPICE. The simulation speed is further improved by "compiling" results, in which an entire switching and interconnection circuit is represented by a macromodeled moment representation approximation.

Thesis Supervisor: Jonathan Allen

Professor





---

---

# Acknowledgments

First, I would like to thank Jonathan Allen, my advisor, for his inspiration and encouragement, and for providing the flexibility to define my own work as it progressed. Thanks also to my thesis readers, John Wyatt and Jacob White, for actually reading all 182 pages and for providing good feedback.

I thank my friends—particularly Keith Nabors, Andy Ayers and Don Baltus, who expressed occasional interest in this work, and Bob Armstrong, Cyrus Bamji, Barry Thompson and Mark Reichelt, who have been around on the eighth floor through the whole thing. I also thank Dorothy Fleischer.

Mostly, I thank my wife, Lynne, for three things: her technical assistance in the form of the work frequently cited in Chapter 6, her emotional support, and her patience as I strived to make this thesis as good as possible.

Lastly, I gratefully acknowledge my sponsor, the Joint Services Electronics Program, for supporting this research.

This research was supported in part by the Joint Services Electronics Program under contract number DAAL03-86-K-0002.



---

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>15</b>
1.1	Existing Performance Extraction Techniques . . . . .	16
1.1.1	Direct Method Simulation . . . . .	16
1.1.2	Specialized Simulators . . . . .	17
1.1.3	Waveform Bounding of <i>RC</i> Trees . . . . .	17
1.1.4	Second Order <i>RC</i> Tree Model . . . . .	18
1.1.5	Transmission Line Modal Analysis . . . . .	18
1.2	Simulation with the Moment Representation . . . . .	19
1.3	Overview of Remaining Chapters . . . . .	22
<b>2</b>	<b>VLSI Interconnection Properties</b>	<b>25</b>
2.1	Propagation Characteristics of VLSI Interconnections . . . . .	25
2.2	Coupling Characteristics of VLSI Interconnections . . . . .	27
2.3	Interconnection Propagation on Sample Technologies . . . . .	28
2.4	Analysis of Silicon MOS Interconnections . . . . .	32
2.4.1	Effects of Classical Scaling on Interconnections . . . . .	34
2.4.2	MOS Scaling in Practice . . . . .	38
2.5	Analysis of VLSI Transmission-Line Interconnections . . . . .	39
2.5.1	Transmission Line Propagation . . . . .	39
2.5.2	Transmission Line Coupling . . . . .	41
2.6	Discussion . . . . .	43
<b>3</b>	<b>The Moment Representation</b>	<b>45</b>
3.1	Node Voltage Transitions and Waveforms . . . . .	45
3.2	Representation Definition . . . . .	48
3.2.1	Expressed in terms of $\hat{v}(t)$ . . . . .	49
3.2.2	Expressed in terms of the $V(s)$ rational function . . . . .	50
3.3	Properties of the Moment Representation . . . . .	51
3.3.1	Shifting in Time . . . . .	52
3.3.2	The Linearity Property . . . . .	52
3.3.3	Derivatives of Waveforms . . . . .	53
3.3.4	Linear Network Properties . . . . .	53
3.3.5	Non-Linear Network Properties . . . . .	53
3.4	Representations of Simple Waveform Functions . . . . .	53
3.5	Time Domain to Moment Representation Conversion . . . . .	55
3.6	Moment Representation to Time Domain Conversion . . . . .	55

3.6.1	Basic principles . . . . .	55
3.6.2	Polynomial Exponential . . . . .	56
3.6.3	Double Exponential . . . . .	60
3.6.4	Choice of $g(t)$ . . . . .	64
3.7	Discussion . . . . .	66
<b>4</b>	<b>General Linear Network Solutions</b>	<b>69</b>
4.1	Formulating the Moment Polynomial Matrix Equations . . . . .	71
4.1.1	Distributed Circuit Elements . . . . .	74
4.1.2	Moment Polynomial Matrix Properties . . . . .	74
4.2	Solution of the Moment Polynomial Matrix Equation . . . . .	76
4.2.1	Gaussian Elimination . . . . .	76
4.2.2	LU Decomposition . . . . .	78
4.2.3	Finite Truncation in Moment Polynomial Gaussian Elimination . . . . .	79
4.2.4	Computing Node Orders . . . . .	82
4.2.5	Pivoting for Accuracy . . . . .	83
4.2.6	Pivoting for Sparsity . . . . .	84
4.3	Examples . . . . .	84
4.4	Computational Requirements . . . . .	85
4.5	Discussion . . . . .	92
<b>5</b>	<b>Transmission Line Solutions</b>	<b>95</b>
5.1	Modal Analysis of Coupled Lossless Transmission Lines . . . . .	96
5.2	Modal Analysis with the Moment Representation . . . . .	101
5.3	Discussion . . . . .	105
<b>6</b>	<b>Macromodels for Non-Linear Networks</b>	<b>105</b>
6.1	MOS Network Decoupling . . . . .	106
6.1.1	Cells . . . . .	106
6.1.2	Sub-networks . . . . .	106
6.1.3	Macromodel Cells . . . . .	107
6.2	Fundamentals of Macromodeling . . . . .	108
6.3	Transistor Driver Cell Model . . . . .	109
6.3.1	Input Capacitance . . . . .	109
6.4	Macromodel Input Parameters . . . . .	110
6.4.1	Input Waveform Parameter . . . . .	110
6.4.2	Output Load Parameter . . . . .	113
6.5	Macromodel Output Functions . . . . .	117
6.6	Extracting Macromodel Function Values . . . . .	120
6.6.1	Macromodel Extraction CPU Time . . . . .	128
6.6.2	Macromodel Memory Requirements . . . . .	128
6.7	Transistor Transmission Cells . . . . .	129
6.7.1	Conducting Transmission Cell . . . . .	131
6.7.2	Switching Transmission Cell . . . . .	133
6.8	Examples . . . . .	139
6.9	Computational Requirements . . . . .	144
6.10	Discussion . . . . .	144

<b>7</b>	<b>Circuit Model Compilation</b>	<b>147</b>
7.1	Circuit Model Levels . . . . .	147
7.1.1	Compiled Circuit Model . . . . .	149
7.1.2	Multi-Stage Compiled Circuit Model . . . . .	152
7.2	Computation Requirements . . . . .	152
<b>8</b>	<b>Conclusions and Future Work</b>	<b>155</b>
8.1	Future Work . . . . .	156
<b>A</b>	<b>Moment Polynomial Operations</b>	<b>159</b>
A.1	Definitions . . . . .	159
A.2	Minimum Polynomial Orders . . . . .	160
A.3	Truncation Order Rules . . . . .	161
A.4	Floating Point Operation Counts . . . . .	161
<b>B</b>	<b>Truncation Orders During Gaussian Elimination</b>	<b>163</b>
B.1	Moment Representation Admittance Properties . . . . .	163
B.2	MPNA matrix properties . . . . .	164
B.2.1	Gaussian Elimination of Nodal Analysis Matrices . . . . .	165
B.3	Truncation Orders for Gaussian Elimination . . . . .	168
B.3.1	Backward elimination . . . . .	168
B.3.2	Forward Elimination . . . . .	169
<b>C</b>	<b>Using Moments as Macromodel Parameters</b>	<b>173</b>



---

---

## List of Figures

1-1	Elmore delay of a waveform. . . . .	17
1-2	Transitions of node voltage, $v(t)$ . . . . .	20
1-3	Waveform modeling in the simulator. . . . .	21
2-1	Lumped element circuit approximations for propagation classes. . . . .	26
2-2	Lumped element circuit approximation for coupled lines. . . . .	29
2-3	Two-port circuit symbols for coupled $RC$ and $LC$ lines. . . . .	30
2-4	Eight layer Wafer Scale Integration interconnections. . . . .	31
2-5	Multi-layer ceramic substrate technology. . . . .	31
2-6	Integrated circuit groundplanes. . . . .	32
2-7	Scaling of IC interconnections. . . . .	35
2-8	Sample circuit and model for studying delay. . . . .	36
2-9	Interconnection dimensions of 1.5 $\mu\text{m}$ process. . . . .	37
2-10	Delay vs. line length for 1.5 $\mu\text{m}$ process. . . . .	37
2-11	Delay vs. line length for 0.5 $\mu\text{m}$ process. . . . .	38
2-12	Ground and coupling capacitance vs. conductor spacing. . . . .	40
2-13	GaAs dimensions contrasted with silicon's. . . . .	41
2-14	Transmission line coupling waveforms. . . . .	42
3-1	Interfaces between representation domains. . . . .	46
3-2	Individual transitions of node voltage, $\hat{v}(\hat{t})$ . . . . .	47
3-3	Graphical meaning of some moment representation terms. . . . .	51
3-4	Basis functions for the polynomial exponential waveform approximation. . . . .	59
3-5	Polynomial exponential waveform approximations. . . . .	61
3-6	Effects of iterating to find double exponential poles. . . . .	62
3-7	Procedure for selecting assumed waveform shape. . . . .	65
4-1	Thevenin and Norton equivalent circuits. . . . .	71
4-2	Circuit Element template patterns. . . . .	72
4-3	Distributed $RC$ element template . . . . .	75
4-4	Circuit requiring node pivoting. . . . .	84
4-5	Two connecting $RC$ trees . . . . .	86
4-6	Charge sharing . . . . .	87
4-7	10-stage $RC$ line . . . . .	88
4-8	Distributed $RC$ tree . . . . .	89
4-9	10-stage coupled $RC$ lines . . . . .	90
4-10	Two-stage coupled $RLC$ lines . . . . .	91

4-11	Computations vs. node count. . . . .	93
5-1	Response of single transmission line. . . . .	96
5-2	Waveforms from MPNA and modal analysis of transmission lines. . . . .	97
5-3	Coupled transmission line geometries. . . . .	97
5-4	Modal analysis equivalent representation. . . . .	99
5-5	Method of characteristics equivalent circuit. . . . .	99
5-6	Modal analysis circuit. . . . .	100
6-1	Sub-network boundaries. . . . .	108
6-2	Legal and illegal macromodel cells for a cross-coupled inverter circuit. . . . .	109
6-3	Linear network equivalents for macromodel driver cell . . . . .	111
6-4	MOSFET input capacitance. . . . .	112
6-5	Representative input voltage waveforms with identical Elmore times, $M_1$ . . . . .	112
6-6	Critical range of output voltage. . . . .	114
6-7	Approximating circuit for computing $V_{step}(t)$ . . . . .	114
6-8	Transistor driver cell linear circuit equivalents. . . . .	118
6-9	SPICE simulation of true inverter and linear circuit equivalents. . . . .	119
6-10	Full Adder Karnough map with output transition possibilities. . . . .	120
6-11	Macromodel extraction test circuits. . . . .	121
6-12	Test input waveform. . . . .	122
6-13	Approximate linearization of a macromodel function. . . . .	130
6-14	Conducting transmission cell . . . . .	131
6-15	Switching transmission cell . . . . .	131
6-16	Linear circuit equivalents for a conducting transmission cell . . . . .	132
6-17	Conducting transmission cell macromodel extraction circuit. . . . .	134
6-18	Linear circuit equivalents for a one-transistor switching transmission cell. . . . .	135
6-19	Second order linear circuit equivalents for a two-transistor switching transmission cell. . . . .	136
6-20	Test circuit for extracting switching transmission cell macromodel values. . . . .	137
7-1	Circuit model levels. . . . .	148
B-1	Circuit representation during gaussian elimination . . . . .	166
C-1	Macromodel plot of critical region waveform slope vs. $X_2$ and $X_3$ . . . . .	175



---

---

## List of Tables

2-1	Interconnection characteristics of sample technologies. . . . .	33
2-2	Comparison of maximum coupling capacitance ratios of 1.5 $\mu\text{m}$ and 0.5 $\mu\text{m}$ process. . . . .	39
2-3	Transmission line characteristics of sample technologies. . . . .	41
3-1	Terms of moment representation for one transition. . . . .	50
3-2	Laplace transforms and waveform representation terms for simple functions. . . . .	54
3-3	Common Assumed Waveform Shapes . . . . .	57
4-1	Equivalent computations of examples for each method . . . . .	92
6-1	Macromodel functions and scalar values of a third-order macromodel set. . . . .	119
6-2	Number of single precision floating point numbers for a macromodel set. . . . .	128
6-3	Simulation time in CPU seconds. . . . .	144
7-1	Parameters for complete system CPU time estimates. . . . .	153
7-2	Simulation times for different circuit model levels in hours/MIP. . . . .	153
7-3	Third-order circuit model translation times in hours/MIP. . . . .	154
A-1	Maximum number of floating-point operations for polynomial operations. . . . .	162



---

---

# Introduction

VLSI circuitry has created enormous possibilities for digital system design, but has also created a new set of design problems. A system must be extensively evaluated before fabrication of even a prototype, since parts are not off-the-shelf and may take valuable months to make. Determining the performance of digital circuits before fabrication is an essential task. VLSI circuit designers must rely on Computer Aided Design (CAD) tools. Most often, the performance of a VLSI system is determined through simulation. General, numerical simulators (like SPICE [1]) are accurate, but are too slow for simulating large circuits. Popular alternatives to SPICE are simulators with simple circuit models that are computed rapidly.

Previously, estimating speed performance through simple models was relatively easy. Circuit speeds of older MOS technologies was controlled by transistor drive and total load capacitance. Circuit speeds of bipolar technologies were mainly controlled by the same parameters. Printed circuit board delays were largely ignored since chip output switching times have exceeded printed circuit board interconnection times. Increasingly, however, the performance of all VLSI technologies is being dominated by interconnection performance. As transistor dimensions shrink, device speed performance improves. However, interconnection delays across an entire chip are becoming much longer. This is true for MOS, bipolar and GaAs technologies, even though interconnection circuit models are quite different. With faster switching chips, propagation delays on printed circuit boards and chip carriers are also increasingly significant.

As chips and printed circuit boards become more complex another difficulty is increasingly observed on all VLSI technologies—cross-talk noise between coupled interconnections. Coupling on silicon MOS chips can intensify by the increased relative sidewall heights of finer linewidth interconnections, and as always, coupling is observed between overlapping conductors. Coupling is far greater on GaAs and Silicon-on-Sapphire chips than silicon chips because of semi-insulating substrates. On printed circuit boards, increased coupling is due to finer linewidths and spacings, an increased number of layers, and faster transitions. Chapter 2 discusses interconnection propagation delay and coupling in detail, and in general, shows why it is becoming

more important to examine interconnection performance.

This thesis focuses on circuit modeling methods that can be used for fast computer simulation of interconnection behavior—that is, the circuit modeling is optimized for circuits with interconnection dominance. The modeling in this thesis determines voltage waveforms on any node, which can subsequently be used for determining delay times and noise levels.

The problem of extracting an equivalent circuit from layout geometries is not considered in this thesis. It is assumed that interconnection behavior can be modeled by linear circuit elements (distributed or discrete) and that these can be extracted from physical layouts. Circuit extraction from interconnections is discussed in [2,3,4,5] and circuit connectivity extraction is discussed in [6,7].

## 1.1 Existing Performance Extraction Techniques

First, we will examine existing simulation and performance modeling methods for interconnection circuits.

### 1.1.1 Direct Method Simulation

Currently, direct method circuit simulators like SPICE [1] and ASTAP [8] are the mainstay of IC circuit designers for simulating circuit performance. These flexible simulators accept almost any circuit with linear or non-linear, discrete elements and determine the voltage waveform on any node.

Direct method simulation is much too costly for simulating complete chips and certainly for simulating complete systems or system backplanes. For instance, the circuit size of SPICE simulations is effectively limited to hundreds of nodes, thus, limiting these simulators to analysis of individual cells. As circuit performance moves from transistor dominance to interconnection dominance, direct method simulators need to devote less time to costly non-linear transistor model evaluation. But, with direct method simulators, distributed elements of interconnections must be broken into a series of discrete elements, adding excessively to the already limiting number of circuit nodes.

Relaxation and iteration simulation methods [9,10,11] offer a simulation speed and circuit size improvement over direct methods for MOS digital circuits. This is achieved by breaking circuits, usually on capacitive boundaries, and solving the sub-circuits separately, where the waveform on one sub-circuit controls the simulation of the next sub-circuit, which control the next, and so forth. When the interaction between separate cells is strong, the sub-circuits must be solved iteratively. Relaxation methods enable circuit size to be increased to thousands of nodes—still a size prohibiting entire chip simulation.

An even more comprehensive simulation method solves Maxwell's Equations for interconnections. These simulators perform the most accurate simulation of interconnections, including non-TEM transmission line effects, skin effect, etc. But, they are limited to very small circuits,

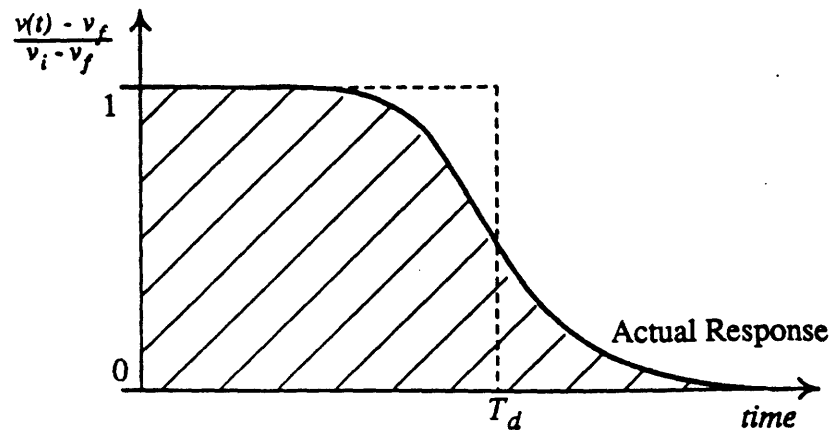


FIGURE 1-1: Elmore delay of a waveform.

in some cases they are able to simulate just a single crossing of two interconnections.

### 1.1.2 Specialized Simulators

Because of circuit size limitations with direct method simulators many specialized, faster simulators have been developed—usually at the expense of accuracy. A prime example of this is RSIM [12] for MOS circuits. This program started as a switch-level logic simulator, and simple delay models were added to it. Each logic state transition on a node causes a delay model computation for a time estimate of the transition. The delay models in RSIM are based on transistor resistance and load capacitance—models which are inadequate for interconnection dominated circuits.

Improved delay models have been developed which can be built into MOS switch-level simulators. The next sections describe improved delay models which are specific to interconnection dominated circuits.

### 1.1.3 Waveform Bounding of RC Trees

Waveform bounding methods are used to find an Elmore time approximation, a low bound and a high bound of the voltage waveform on a circuit node. The Elmore time [13] actually equals the area under the step response waveform if the initial and final values are shifted to 1 v. and 0 v., respectively. That is, the Elmore time equals the shaded area in Figure 1-1. The Elmore time can be used to make a waveform estimate, most often in the form of an exponential, i.e., in the form of

$$v(t) = V_0 e^{-t/t_d}$$

These methods were first developed by Penfield *et al.* [14] for RC tree circuits consisting of resistive tree networks, with a capacitance to ground at each node. Distributed RC elements

are also allowed. Later, the class of permissible circuit was extended to include  $RC$  meshes [15] and general  $RC$  networks [16]. The tightness of the bounds was improved in [17].

These methods have several limitations. First, on some nodes the Elmore time waveform estimates are poor and the bounds are loose. This is quite noticeable for nodes near the driven end of an  $RC$  line. Second, the  $RC$  circuit forms are limited and do not model all desirable interconnection situations for  $RC$  lines. These methods, for instance, cannot model capacitive coupling or resistive substrate effects (like those shown in Figure 2-1(e) and (f)). Third, non-linear elements can only be modeled by linear element approximations.

#### 1.1.4 Second Order $RC$ Tree Model

Horowitz [18] has developed second order  $RC$  tree methods as an extension of the previous  $RC$  tree methods. Two things were added in this work. The first addition is improved, second-order waveform estimates that match the Elmore time, the second-order moment (*moment* is extensively defined in Chapter 3) and the sum of the open-circuit time constants (or term  $b_1$  of the transfer function,  $\frac{a_0+a_1s+a_2s^2+\dots}{1+b_1s+b_2s^2+\dots}$ ). The second addition is the ability to model switching pass gates in the middle of an  $RC$  tree.

This work addresses the first limitation presented in the last section. However, it still contains the other limitations mentioned for  $RC$  tree methods.

#### 1.1.5 Transmission Line Modal Analysis

Much effort has been devoted to modeling both lossless and lossy transmission lines with computationally faster algorithms. A good discussion of this is in [19]. Direct method simulation is particularly inefficient for transmission line circuits, since generally the distributed transmission lines must be modeled by a very large number of discrete nodes, more than is needed for  $RC$  lines. A much faster analysis technique is achieved with modal analysis. Modal analysis is possible in either the time domain or the frequency domain.

Time domain modal analysis works only for lossless transmission lines and assumes a constant dielectric medium and constant coupling along the length of the transmission lines.<sup>1</sup> Time domain modal analysis is advantageous in that it is possible to simulate coupled networks and non-linear drivers and loads (if solved with direct methods). The major disadvantage is that lossy lines cannot be simulated unless the line is divided into many sections connected by discrete resistors [22].

Frequency domain modal analysis can simulate lossy lines, but only with linear driver and load networks. Again, it assumes a constant dielectric medium and constant coupling along the length of the transmission lines. To operate in a time domain simulator, an FFT interface

---

<sup>1</sup>Transmission line analysis with varying coupling coefficients has been addressed in [20] and [21].

is needed to translate between the time domain and frequency domain. Mild non-linearities in frequency domain modal analysis has been investigated by [23].

Both time domain and frequency domain modal analysis has been added to direct method simulators. Accordingly, performance is often constrained by other features of the simulator. These methods are incorporated into the simulator methods presented in this thesis (see Chapter 5) and operate with much improved speed.

## 1.2 Simulation with the Moment Representation

In general, the specialized interconnection simulation methods described in the previous sections suffer several drawbacks. The most noticeable one is that no one method can be used universally for all configurations of interconnection simulation. A different method must be applied to each propagation class (i.e., *LC* vs. *RC* propagation) and to each type of circuit stimulation (i.e., a single-ended driver vs. a switched transmission gate vs. a noise spike ...). Additionally, many of these methods have limited capacity to simulate non-linear circuit elements. Most use a linearized approximation for non-linear drivers and loads. This often leads to approximate solutions with large errors, particularly when input waveforms fluctuate over a wide range of input slopes.

Direct method circuit simulators have the flexibility to overcome these drawbacks, but these simulators are too slow to accommodate the large scale simulations needed for VLSI design.

The moment representation simulation methods, introduced in this thesis, address these drawbacks. The moment representation simulator uses nodal analysis, so, like direct method simulators, the circuit configurations can be very flexible. It can simulate both *LC* and *RC* interconnections with or without coupling, charge sharing circuits, single-ended drivers, pass gates and so forth. A major difference, however, is that direct methods operate in the time domain while the moment representation simulator operates in the moment representation domain, which is a subspace of the frequency domain. The moment representation domain better accommodates linear interconnection circuits, especially with distributed circuit elements.

While the frequency domain usually precludes non-linear circuit elements, this thesis presents a moment representation macromodeling method which very accurately constructs a linear circuit equivalent for a non-linear transistor circuit. The equivalent is valid through the entire duration of any transition. The linear equivalents have *variable* circuit elements with values that are macromodeled functions of input signal slope and output load. This macromodeling method permits very rapid and accurate simulation of non-linear elements and is favorable for combined linear and non-linear circuits.

Several features of the moment representation allow very rapid simulation of circuits:

1. An entire logic transition is solved in the moment representation simulator with one matrix equation solution. On the other hand, direct methods require at least one matrix

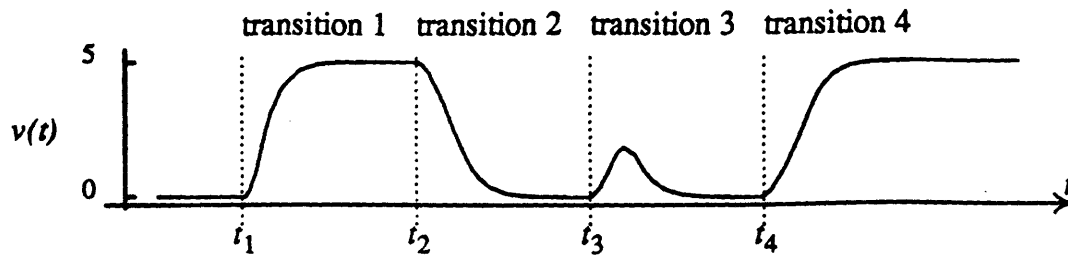


FIGURE 1-2: Transitions of node voltage,  $v(t)$ .

equation solution at each incremental time-step—several dozen time-steps may be needed for each logic transition.

2. The macromodeling algorithms for non-linear circuits are much faster than Newton-Raphson iteration used in direct methods.
3. The response of circuits can be *compiled* by a preprocessor into a very compact moment representation form. This is analogous to finding a linear circuit transfer function, except that non-linear elements are also permitted.

An experimental simulator was developed to test the moment representation simulation methods. The moment representation algorithms were embedded in an event-driven logic simulator in much the same way that RSIM's simple timing models were added to a switch-level logic simulator [12]. The logic simulation algorithms are well-known, so they are discussed in the thesis only in the briefest fashion. Details of event-driven logic simulation are found in [12,24]. Waveform related issues of event-driven simulation, such as what to do when events overlap or when events cancel, are detailed in [25,26]. The simulator algorithms were tested on digital circuits containing MOS transistor and interconnection models. Simulator results are interspersed throughout the thesis at appropriate locations. The following paragraphs briefly outline the operation of the experimental simulator.

The voltage on any node is broken into a series of *transitions* separated by periods of d.c. voltage, as shown in Figure 1-2. Each transition has a starting time, designated  $t_1$  through  $t_4$  in Figure 1-2, a d.c. transition value, and a waveform. The simulator models a node voltage by transitions—at the starting time of a transition, the node voltage is defined by the waveform portion of the transition; when the waveform decays to a final, constant value,<sup>2</sup> the node voltage is given by a constant d.c. value. The simulator must keep track of the d.c. value and update it at the end of each transition. As shown in Figure 1-2, the transition can represent any type of perturbation in the waveform, and is not always a logic transition.

<sup>2</sup>For an asymptotically decaying waveform this point is taken to be where the voltage decays to within 2% of the final voltage.



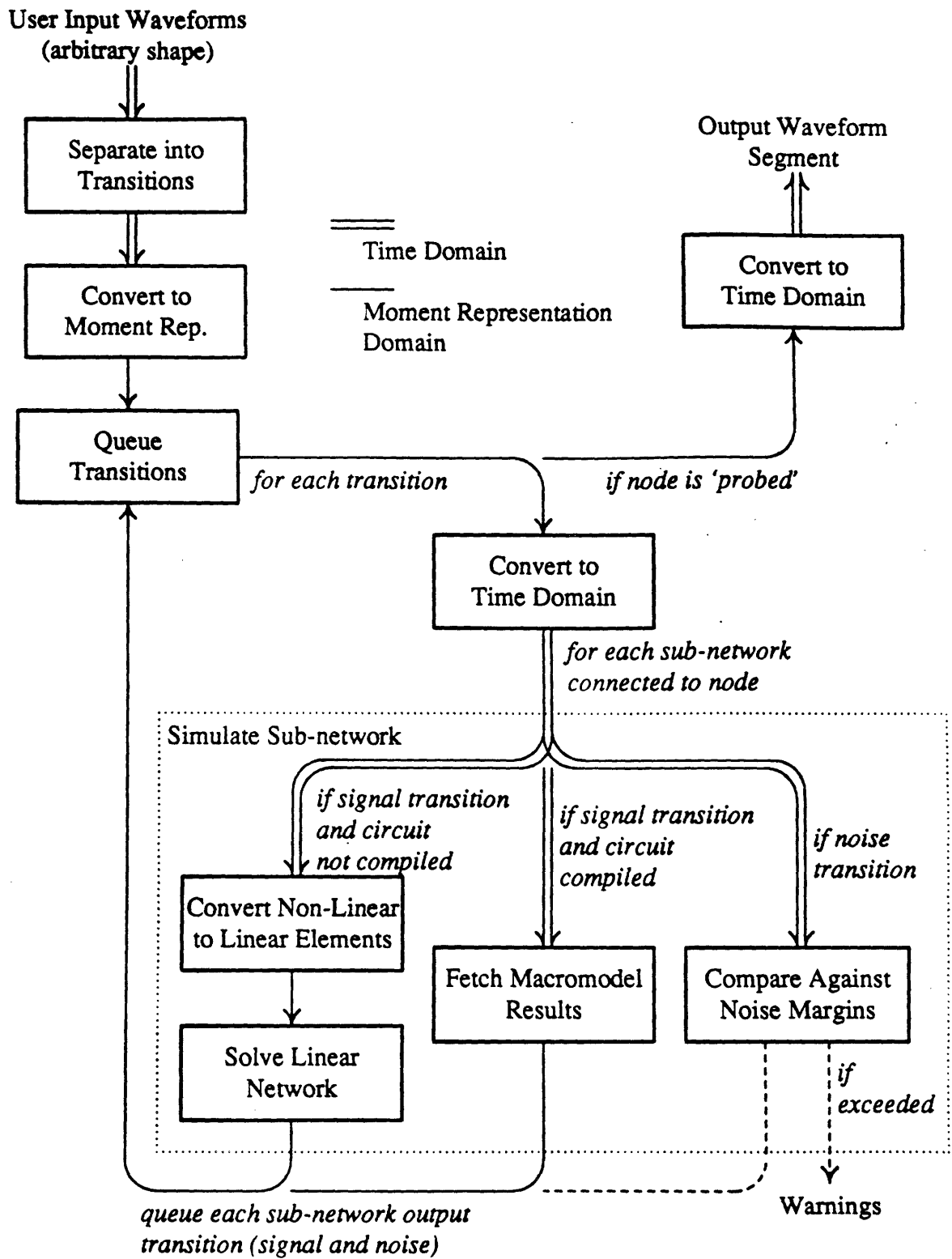


FIGURE 1-3: Waveform modeling in the simulator.

Within the event-driven simulator, transition waveforms are represented in two forms: in the time domain and in the moment representation domain. The different forms are used in different places in the simulation algorithms to gain maximum efficiency in both simulation time and simulation memory requirements. Figure 1-3 shows the general flow of operations in the simulator and also shows locations in the simulation algorithms where the two waveform representations are used. The double lines indicate where the time domain is used, all other places use the moment representation.

Initially, the user supplies a set of input transitions to the simulator. These are converted to the moment representation domain and are inserted into the transition event queue.

During simulation, transitions are pulled from the queue sequentially by starting time. The node on which the transition occurs is said to be *activated*. The moment domain waveform of the transition is then converted to the time domain. As we will see in Chapter 6, this is required for macromodel function evaluation. Chapter 6 also describes how an MOS circuit is split into disjoint sub-networks. Each sub-network with an input connection to the active node is simulated separately. The choice of simulation algorithm depends on whether the transition represents a signal change or noise spike and whether the sub-network has a compiled circuit model. For a noise transition, the peak noise voltage is compared against noise margins for the active node. A warning is issued to the user if exceeded. For a signal transition, if a compiled circuit model exists, then precomputed simulation results are fetched from macromodel tables, otherwise, the sub-network must be solved, first by converting non-linear elements into equivalent linear networks, and then by solving the linear network. In either case, if the signal transition on any sub-network causes an output change, then the new transition(s) are added to the event queue.

The user may select to *probe* the output waveform of any node. All output transitions for a probed node are converted to the time domain and are assembled into a single, continuous waveform for user viewing.

### 1.3 Overview of Remaining Chapters

Chapter 2 discusses interconnection properties, and demonstrates the importance of examining interconnections of advanced technologies. The information in Chapter 2 is not crucial to understanding later chapters and may be skipped.

Remaining chapters describe the moment representation simulation algorithms. Chapter 3 defines and characterizes the moment representation. It presents the methods for converting between the time domain and moment representation domain and discusses the transition model for waveforms.

There are several advantages to circuit modeling with the moment representation methods; the major advantage is presented in Chapter 4. It shows a method for solving any linear circuit

composed of resistors, capacitors, inductors and independent sources in any configuration. Furthermore, distributed circuit elements, such as distributed  $RC$ 's or  $LC$ 's which are so prevalent as interconnection models, can be included as simple two-terminal elements without loss of accuracy.

Chapter 5 outlines some special algorithms for solving transmission line circuits with the moment representation. It shows that existing transmission line *modal* analysis methods can be used with the moment representation to give very good results.

The macromodeling feature of the moment representation simulator is introduced in Chapter 6. First it describes how MOS transistor circuits are divided into subnetworks, and then it describes an accurate method for converting a non-linear circuit into an equivalent linear circuit. The linear circuit is then suitable for solution with the methods of Chapter 4.

A final feature of the moment representation is presented in Chapter 7. It shows that a circuit that is to be simulated through many input transitions can be *compiled* into a small macromodeled moment representation description. Subsequent simulations of compiled circuit models are much faster.



---



---

## VLSI Interconnection Properties

When studying VLSI circuits, many types of interconnections must be considered which have dissimilar signal propagation behaviors. In this chapter, propagation characteristics will be studied, first in a general manner, then for specific VLSI technologies.

### 2.1 Propagation Characteristics of VLSI Interconnections

Interconnections on VLSI chips, printed circuit boards and high-density ceramic chip carriers have different properties. For our purposes, all interconnection properties can be illustrated by equivalent *lumped element circuit approximations*. We will assume that circuit elements are constant over all time and frequency. The lumped element circuit approximation of a general *RLGC* interconnection is shown in Figure 2-1(b). If a very large number of these two-port circuits are chained together, the propagation behavior resembles the behavior of its equivalent interconnection.

Partial differential equations describing propagation along a general *RLGC* interconnection are:

$$\begin{aligned}\frac{\partial i(x,t)}{\partial x} + C \frac{\partial v(x,t)}{\partial t} + G v(x,t) &= 0, \\ \frac{\partial v(x,t)}{\partial x} + L \frac{\partial i(x,t)}{\partial t} + R i(x,t) &= 0,\end{aligned}\tag{2.1}$$

or in the frequency domain,

$$\begin{aligned}\frac{d i(x, j\omega)}{dx} + j\omega C v(x, j\omega) + G v(x, j\omega) &= 0, \\ \frac{d v(x, j\omega)}{dx} + j\omega L i(x, j\omega) + R i(x, j\omega) &= 0.\end{aligned}\tag{2.2}$$

Typically, one or more of the terms can be neglected, depending on the signal frequency,  $\omega$  (which is approximated for a transition with rise or fall time,  $\tau$ , by  $\omega \approx 2\pi/\tau$ ) and on values of  $R$ ,  $L$ ,  $G$ , and  $C$ ; the line resistance, inductance, conductance and capacitance per unit length, respectively. Accordingly, different *propagation classes* result—the more prevalent classes are listed below. We will always ignore  $G$  since only good insulators are assumed.

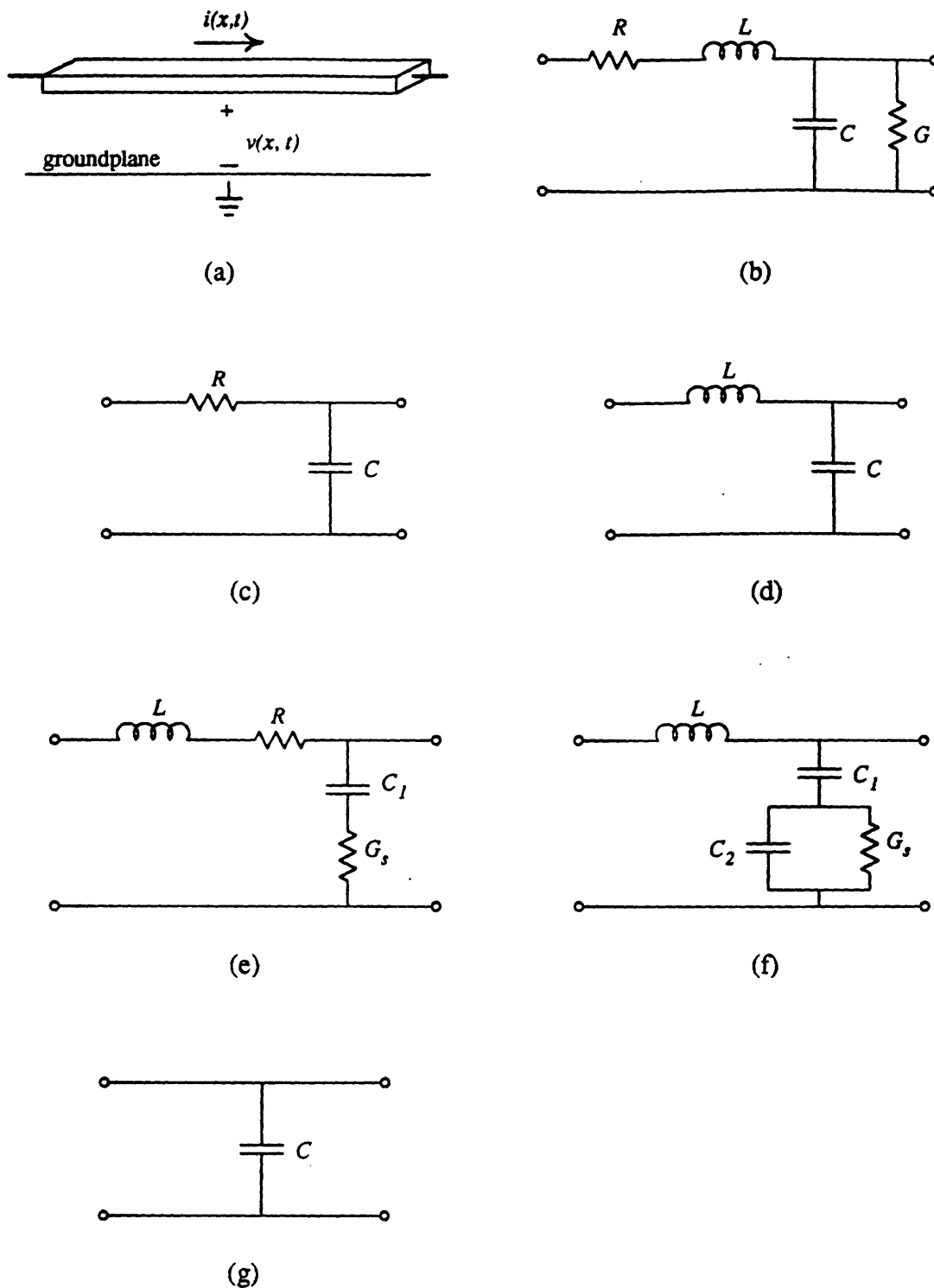


FIGURE 2-1: Lumped element circuit approximations for propagation classes.

(a) transmission line, (b) general  $RLGC$  propagation, (c)  $RC$  propagation, (d) TEM propagation, (e) and (f) resistive substrate propagation, and (g) tracking propagation.

- *RC propagation* prevails as the most common propagation class for silicon MOS integrated circuits. These interconnections have (1) very fine line widths, leading to large  $R$ 's, and (2) a closely spaced groundplane, leading to small  $L$ 's, thus,  $R \gg \omega L$ . With the  $G$  and  $L$  terms absent in this class (Figure 2-1(c)), Equations (2.1) reduce to the *diffusion* partial differential equation—also applicable for describing heat flow, impurity concentration movement, etc.
- *LC propagation* or *Transverse Electromagnetic (TEM)* propagation contains only the  $L$  and  $C$  terms of Equation (2.1) (also illustrated in Figure 2-1(d)) since  $\omega L \gg R$ . These interconnections are lossless transmission lines. Because of its faster, wave-like propagation, *LC propagation* is more desirable than *RC propagation*. Efforts have been made to ensure this propagation class in Wafer Scale Integration circuitry [27] and in VLSI packaging interconnections [28], mostly by raising the interconnection higher above the groundplane. This propagation class also exists in GaAs technologies, but not for MOS technologies, as it is difficult to reduce the  $R$  term below the  $L$  term at typical MOS dimensions and switching speeds.
- As one would expect, there is a transition region between the previous two propagation classes where the resistive and inductive components have comparable magnitudes. The resulting differential equations are more difficult to solve with heuristic methods and are often neglected in interconnection analysis. Interconnections that exhibit wave-like propagation but have a non-negligible resistive component are *lossy transmission lines*. Some consideration is given to this propagation class in later sections.
- Hasegawa [29] has studied signal propagation on metal lines above a highly resistive silicon substrate. Due to the non-perfect ground plane, the transmission line models more resemble those of Figure 2-1(e) and (f). Despite the harder analysis, Seki [30] has demonstrated some speed advantages to a lossy substrate.
- If the interconnection time constant is much shorter than the input signal time constant, it suffices to consider an interconnection as a single node in which propagation is instantaneous. In this *tracking* propagation class, an entire interconnection is modeled by a single lumped capacitor as shown in Figure 2-1(g). This propagation class is included here for completeness. But, as we will see in future sections, this type of propagation is becoming less pertinent to VLSI interconnections.

## 2.2 Coupling Characteristics of VLSI Interconnections

As we will soon see, coupling characteristics of VLSI interconnections are also becoming important with technology improvements. In general, coupling between two interconnections may

have inductive, capacitive and resistive components as shown in Figure 2-2(b). The behavior on line 1 is described by

$$\begin{aligned} \frac{d i_1(x, j\omega)}{dx} + j\omega C_{11} v_1(x, j\omega) - j\omega C_{12} v_2(x, j\omega) - G_{11} v_1(x, j\omega) + G_{12} v_2(x, j\omega) &= 0 \\ \frac{d v_1(x, j\omega)}{dx} + j\omega L_{11} i_1(x, j\omega) + j\omega L_{12} i_2(x, j\omega) + R_1 i_1(x, j\omega) &= 0 \end{aligned} \quad (2.3)$$

where

$$C_{11} = C_{10} + C_{12},$$

$$L_{11} = L_{10} - L_{12},$$

and

$$G_{11} = G_{10} + G_{12}.$$

In matrix form the equations for all lines are

$$\begin{aligned} \frac{d \mathbf{i}(x, j\omega)}{dx} + j\omega \mathbf{C} \mathbf{v}(x, j\omega) - \mathbf{G} \mathbf{v}(x, j\omega) &= \mathbf{0}, \\ \frac{d \mathbf{v}(x, j\omega)}{dx} + j\omega \mathbf{L} \mathbf{i}(x, j\omega) + \mathbf{R} \mathbf{i}(x, j\omega) &= \mathbf{0}. \end{aligned} \quad (2.4)$$

These equations are the telegrapher's equations. Again, for all examples in this thesis we assume perfect insulators around the interconnections, and the  $G$  terms vanish.

The arguments are the same here as in the previous section for neglecting either the resistive or the inductive components or both. We will consider the components depending on the relative magnitudes of  $R$  and  $\omega L$ .

- With  $RC$  propagation, inductive terms disappear, leaving the lumped element circuit approximation shown in Figure 2-2(c).
- $LC$  propagation leaves only the  $L$  and  $C$  terms as shown in Figure 2-2(d).
- Tracking propagation has only the lumped capacitors shown in Figure 2-2(e).

Coupled  $RC$  and  $LC$  interconnections have the circuit symbols pictured in Figure 2-3.

## 2.3 Interconnection Propagation on Sample Technologies

In this section, interconnection characteristics of various VLSI technologies are examined. A propagation class is linked to each technology.

Technologies we will consider are:

1. silicon MOS,
2. silicon bipolar,



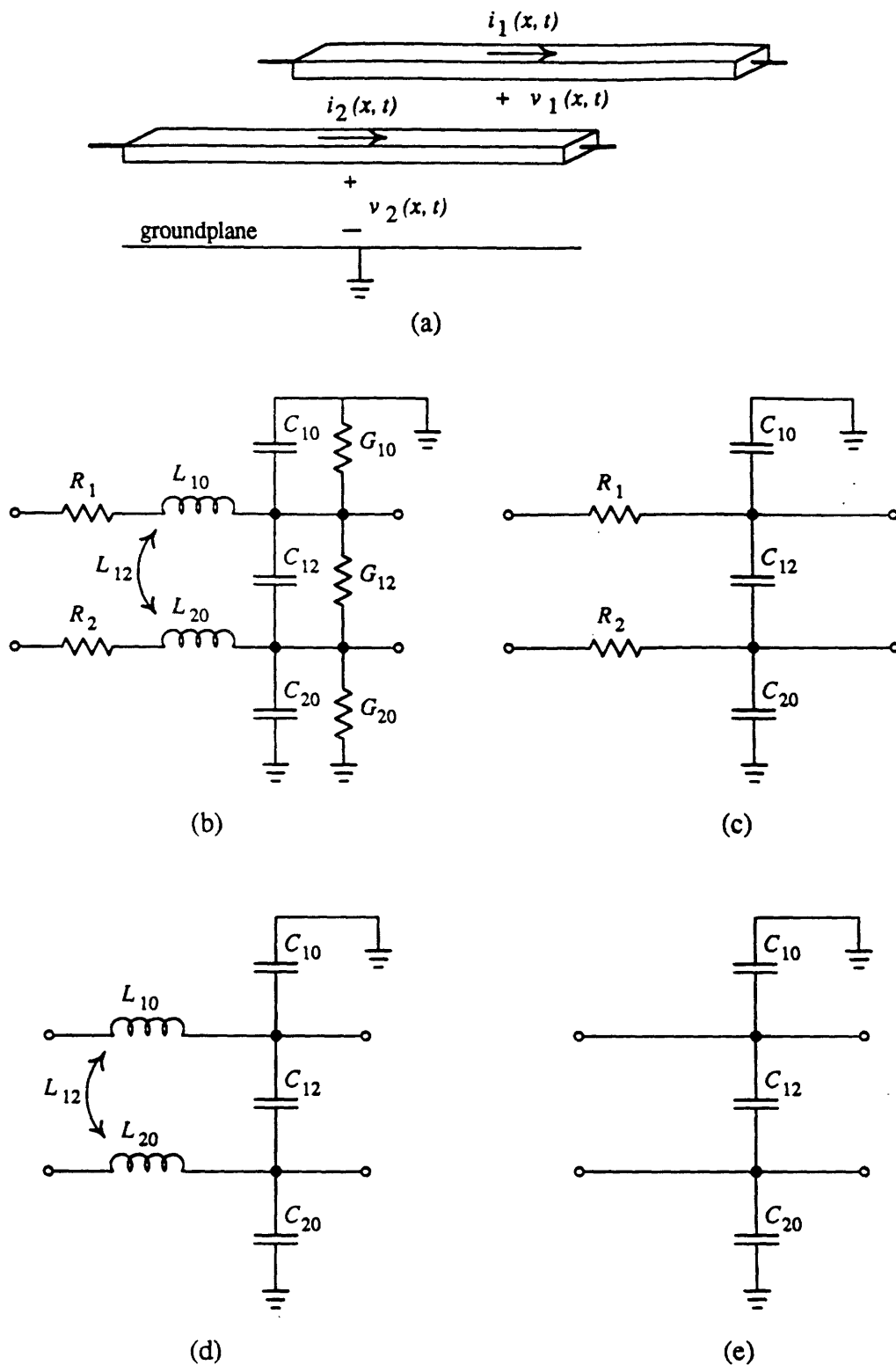


FIGURE 2-2: Lumped element circuit approximation for coupled lines.

(a) actual, coupled transmission lines, (b) general *RLGC* propagation, (c) *RC* propagation. (d) TEM propagation, and (e) tracking propagation.

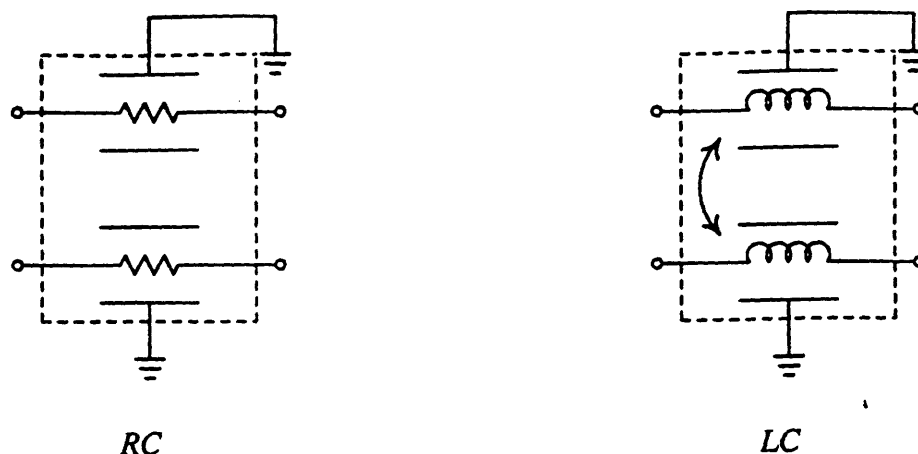


FIGURE 2-3: Two-port circuit symbols for coupled *RC* and *LC* lines.

3. silicon-on-sapphire (SOS),
4. GaAs,
5. an improved interconnection technology for Wafer Scale Integration (WSI), and
6. a ceramic chip carrier technology.

The first four technologies are chip level technologies and assume a lithography capability of about  $1\ \mu\text{m}$  linewidths. Aluminum metalization is assumed; but the exact metalization material is not crucial in this cursory examination. The last two technologies require some explanation.

While most wafer scale integration circuitry uses existing metalization for inter-chip communication, the WSI technology considered here uses a coarser interconnect metalization on top of the fine-lined chip interconnections. This enables faster communication between chips, since conventional IC lines exhibiting *RC* signal propagation are unacceptably slow over complete wafers. Bergendahl [27] has suggested placing eight thick-film layers on a wafer, two layers for power, two for  $x$  and  $y$  direction wiring and four for vias between layers as shown in Figure 2-4. To achieve *LC* propagation, Bergendahl suggests a minimum metal thickness of  $10\ \mu\text{m}$  with an insulator thickness of  $5\text{-}10\ \mu\text{m}$ . Metal widths and spacings may be as low as  $5\ \mu\text{m}$ .

The ceramic chip carrier technology considered here is an advanced technology developed at IBM for mounting and interconnecting integrated circuits [31,32]. As shown in Figure 2-5, it has 33 metalization layers of which sixteen are signal distribution lines spaced  $0.5\text{mm}$  horizontally and between  $0.15\ \text{mm}$  and  $0.2\ \text{mm}$  vertically. The remaining layers are used for voltage reference planes, power distribution and local signal distribution.

Now we will match each interconnection technology to a propagation class. Table 2-1 lists approximate values for relevant technology parameters. Parameters for conductor-to-groundplane spacing,  $H$ , are illustrated in Figure 2-6 for the chip technologies. Notice that while the same conductor dimensions are assumed,  $H$  is quite different between silicon and both SOS and

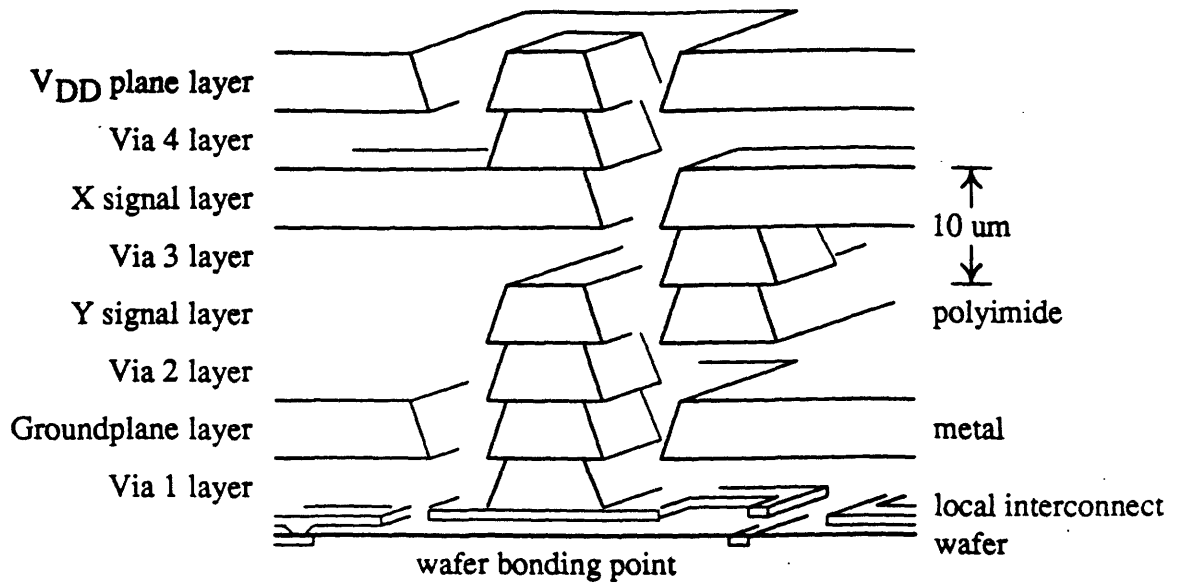


FIGURE 2-4: Eight layer Wafer Scale Integration interconnections.

There are eight planarized metal layers, four for interconnections and four for vias. This figure shows a wafer-to-*y*-signal connection, a *y*-signal-to-*x*-signal connection and a *x*-signal-to-surface connection.

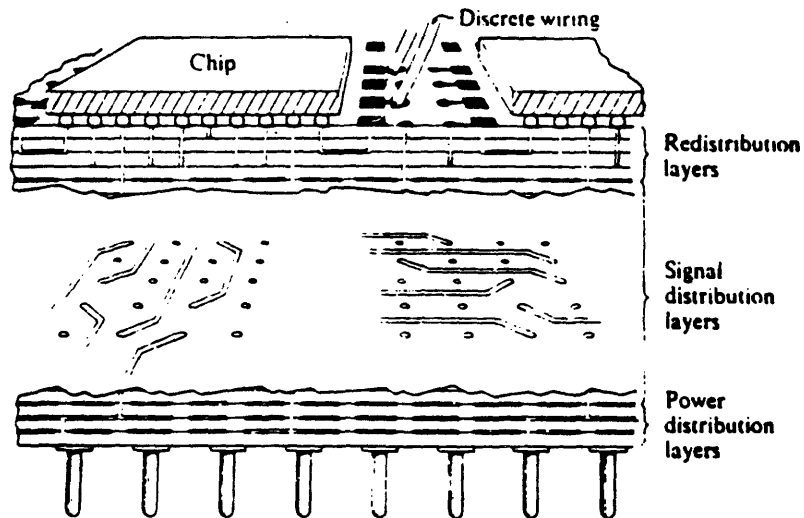


FIGURE 2-5: Multi-layer ceramic substrate technology.

From [31].

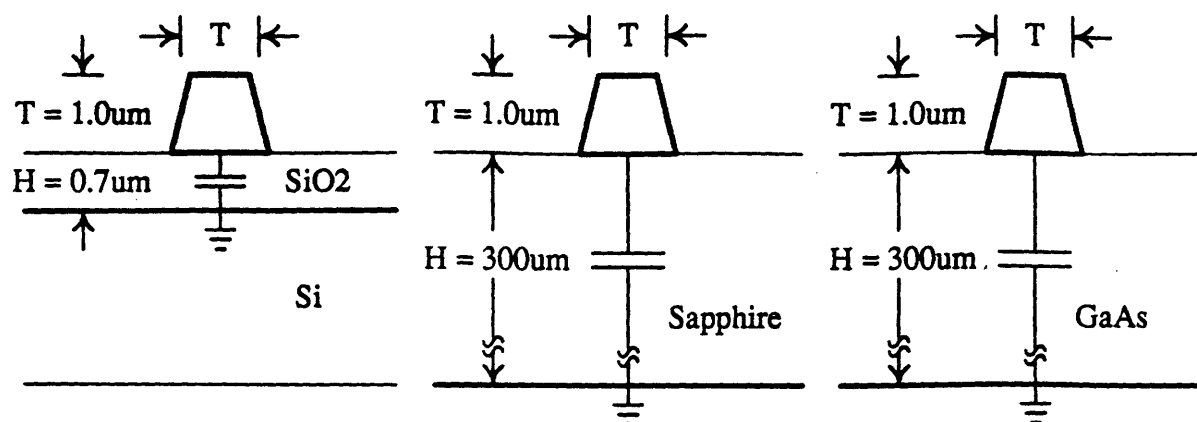


FIGURE 2-6: Integrated circuit groundplanes.

GaAs. This is due to the insulating properties of the sapphire and GaAs substrates. This analysis assumes that silicon forms a good groundplane at its top surface.<sup>1</sup> Values for transition time,  $\tau$ , are approximate values for a switching device driving a long interconnection. In some cases  $\tau$  ranges over an order of magnitude. This range is also reflected in the  $\omega L$  estimates. The bottom line of this table is the expected propagation class for each technology. It stems from a comparison of the series resistance and series inductive impedance estimates. We see with these estimated interconnection parameters that we can expect (1)  $RC$  propagation on silicon MOS interconnections, (2) transmission line or  $LC$  propagation on GaAs, wafer scale integration and ceramic chip carrier interconnections, and (3) a mixture or  $RLC$  propagation on SOS and silicon bipolar interconnections.

Because of the different nature of  $RC$  and  $LC$  interconnections, they are considered separately in the next sections.

## 2.4 Analysis of Silicon MOS Interconnections

To date, propagation on metal MOS interconnections has only marginally limited IC performance. This chapter shows that as MOS interconnection linewidths are scaled down, the intrinsic propagation delays along maximum length lines are becoming significant, and that the coupling between adjacent lines is also increasing to a noticeable level. Reasons for this are best understood through the scaling theory.

<sup>1</sup>Inductance values are also computed by assuming that the silicon surface is an ideal groundplane. Evidence suggests that this is not strictly true, and that the magnetic field extends into the silicon substrate [33], effectively increasing the inductance of silicon MOS and bipolar technologies.

	Silicon MOS	Silicon bipolar	Silicon on Sapphire	GaAs	Wafer Scale Integration	Ceramic Chip Carrier
$H$ ( $\mu\text{m}$ )	0.7	0.7	300	300	10	200 - 1600
$T$ ( $\mu\text{m}$ )	1	1	1	1	10	70
$\tau$ (nsec)	1.0	0.1	1.0	0.01 - 0.1	0.1 - 1.0	0.01 - 1.0
$\epsilon_{R,insulator}$	3.9	3.9	10.5	12	3	9.4
$\mu_{R,insulator}$	1	1	1	1	1	1
$C$ (pF/m)	190	190	63	72	95	110
$L$ ( $\mu\text{H}/\text{m}$ )	0.23	0.23	1.9	1.9	0.35	0.93
$\omega L$ ( $\Omega/\text{m}$ )	1.5K	15K	12K	117K-1.17M	2.2K - 22K	5.9K - 590K
$R$ ( $\Omega/\text{m}$ )	19K	19K	19K	19K	260	25
Propagation class	<i>RC</i>	<i>RLC</i>	<i>RLC</i>	<i>LC</i>	<i>LC</i>	<i>LC</i>

- $H$  conductor to groundplane distance  
 $T$  conductor thickness dimension  
 $\tau$  approximate transition time  
 $\epsilon_{R,insulator}$  relative dielectric constant of insulating medium  
 $\mu_{R,insulator}$  relative permeability of insulating medium  
 $C$  conductor ground capacitance per unit length  
 $L$  series inductance per unit length  
 $\omega L$  series inductive impedance per unit length  
 $R$  series resistance per unit length

Table 2-1: Interconnection characteristics of sample technologies.

### 2.4.1 Effects of Classical Scaling on Interconnections

The classical MOS scaling theory proposed by Dennard [34] predicts how IC layouts and operating ranges will change as inevitable improvements in IC technology occur. The theory assumes a proportional reduction of all dimensions; the amount of reduction is denoted by  $\alpha > 1$  such that any dimension  $l \rightarrow l' = l/\alpha$ . (Values after scaling are identified with a prime.) It also assumes retention of constant electric fields, resulting in a voltage reduction by  $v' = v/\alpha$ . Classical scaling of geometries is illustrated in Figures 2-7(a) and 2-7(b).

With the scaling theory we can predict interconnection propagation times for a scaled design. By computing the change in line resistance and capacitance:

$$R' = \rho \frac{\ell'}{w't'} = \rho \frac{\ell}{wt} \alpha = \alpha R, \quad (2.5)$$

and

$$C' = \epsilon \frac{\ell'w'}{h'} = \epsilon \frac{\ell w}{h} \frac{1}{\alpha} = C/\alpha, \quad (2.6)$$

we see that interconnection propagation times determined by the  $RC$  product remain constant, since  $R'C' = RC$ . While the scaling theory correctly predicts that MOS transistors gain in speed, interconnections do not. Without altering the classical scaling scenario, interconnection delays will dominate at some point in the scaling process.

The classical theory states that interconnection line lengths reduce by  $\alpha$ . But, in practice, the larger number of circuit elements is usually accompanied by an increase in average interconnection length relative to the minimum feature size [35,36]; that is,  $l'_{ave} > l/\alpha$ .<sup>2</sup> Conceivably, some interconnections (clocks, busses, etc.) span the chip's entire length. Thus, for *worst case interconnections*,  $l_{max}$  does not scale with  $\alpha$ , but remains roughly constant.<sup>3</sup> Thus, the maximum delay for classically scaled interconnections now grows to approximately  $\alpha^2 RC$ . It is this worse case propagation time which is of most concern.

We now investigate the above effects in more detail by examining the sample circuit in Figure 2-8(a)—a typical circuit configuration for an interconnection. It contains a driving gate—modeled as in Figure 2-8(b) with a stepped voltage source, an equivalent drive resistance,  $R_d$ , and drive capacitance,  $C_d$ —a uniformly distributed  $RC$  interconnection and a load capacitance,  $C_{ld}$ . We can approximate the propagation time from the driving gate to load with the Elmore time, which for this circuit is

$$T_d = R_d (C_d + C_{ld} + Cl) + RlC_{ld} + \frac{1}{2} RCl^2. \quad (2.7)$$

<sup>2</sup>In [36] it is shown that in some circuits, the average interconnection length does not vary with a larger number of circuit elements, but in most cases, the average interconnection length is proportional to  $G^n$ , where  $G$  is the number of gates, and  $n$  is a circuit-dependent value larger than zero.

<sup>3</sup>This assumes constant overall chip dimensions. Actually, the maximum chip dimension has tended to increase making the situation worse.

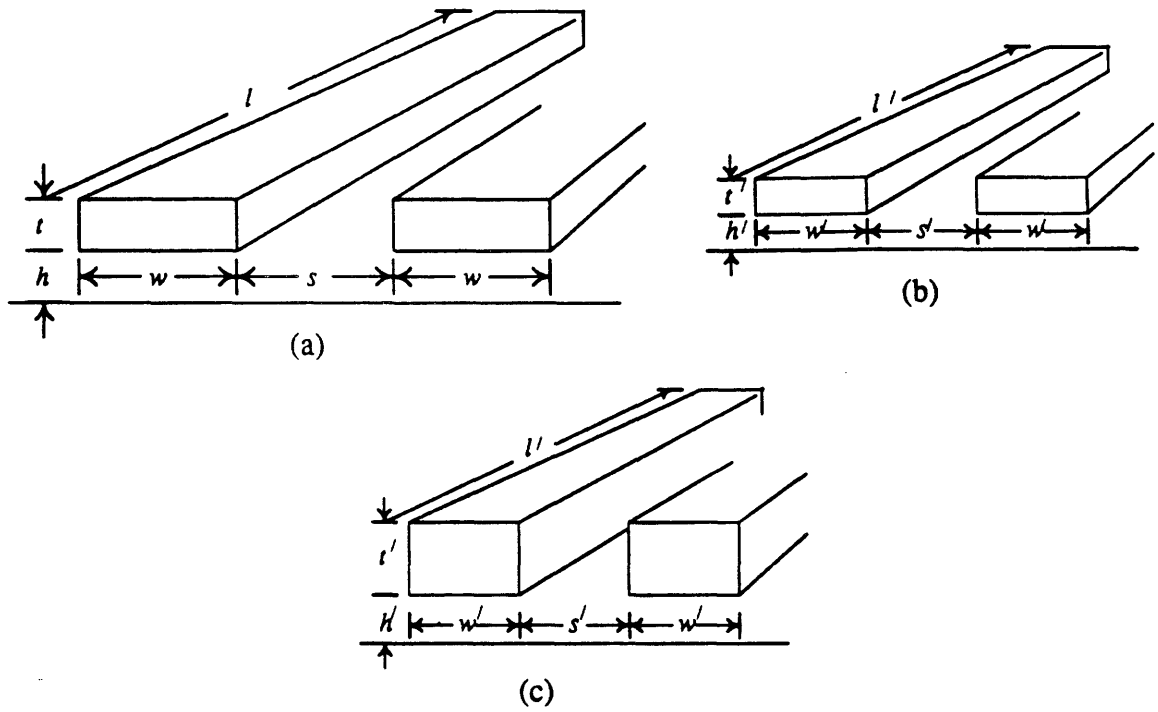


FIGURE 2-7: Scaling of IC interconnections.

(a) original structure, (b) classical scaling of the structure by the amount  $\alpha$ , and (c) a more realistically scaled structure with increased height-to-width ratio.

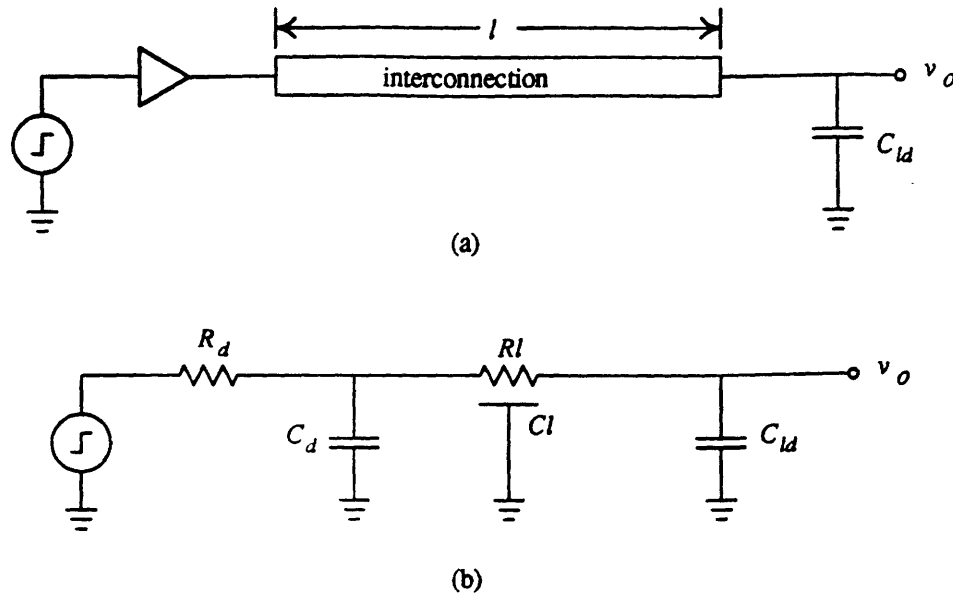


FIGURE 2-8: Sample circuit and model for studying delay.

The Elmore time is discussed in Sections 1.1.3 and 3.2. It suffices to say here that if the step response is sharp enough or symmetric enough,  $T_d$  closely matches the delay time when  $v_o(t)$  passes through the important, logic threshold range.

Figure 2-10 shows a plot of the five additive terms of Equation (2.7) versus interconnection length,  $l$ . Values are calculated from a representative minimum-width, first-level metal line of the  $1.5 \mu\text{m}$  process shown in Figure 2-9. The first and second terms of Equation (2.7) have no dependence on  $l$  and in fact by themselves represent the delay if no interconnection is present. These terms are lumped together into the *self and fanout delay* term. Both the third term (representing the charging of the line capacitance through the drive resistor) and the fourth term of Equation (2.7) (representing the charging of the load capacitance through the line resistance) have a linear dependence on line length. The fifth term is the intrinsic delay of the  $RC$  line. This delay cannot be reduced by increasing driver strength and only marginally by increasing linewidth. For long line lengths—above 6.5 mm for this example—the intrinsic delay dominates over all other delays. However, this length is close to the maximum line length defined by the maximum chip dimension.

Next, we look at the effects of scaling on Equation (2.7). Calculating new, scaled transistor drive resistances and transistor capacitances gives:

$$R'_d = \frac{V'_{ds}}{I'_{ds}} = \frac{V'_{ds}}{\frac{\mu\epsilon W'}{L'D'} (V'_{gs} - V'_{th}) V'_{ds}} = \frac{V_{ds}/\alpha}{I_{ds}/\alpha} = R_d,$$

$$C'_{ld} = \frac{\epsilon A'}{t'_{ox}} = C_{ld}/\alpha,$$

and likewise

$$C'_d = \frac{\epsilon A'}{t'_{ox}} = C_d/\alpha.$$



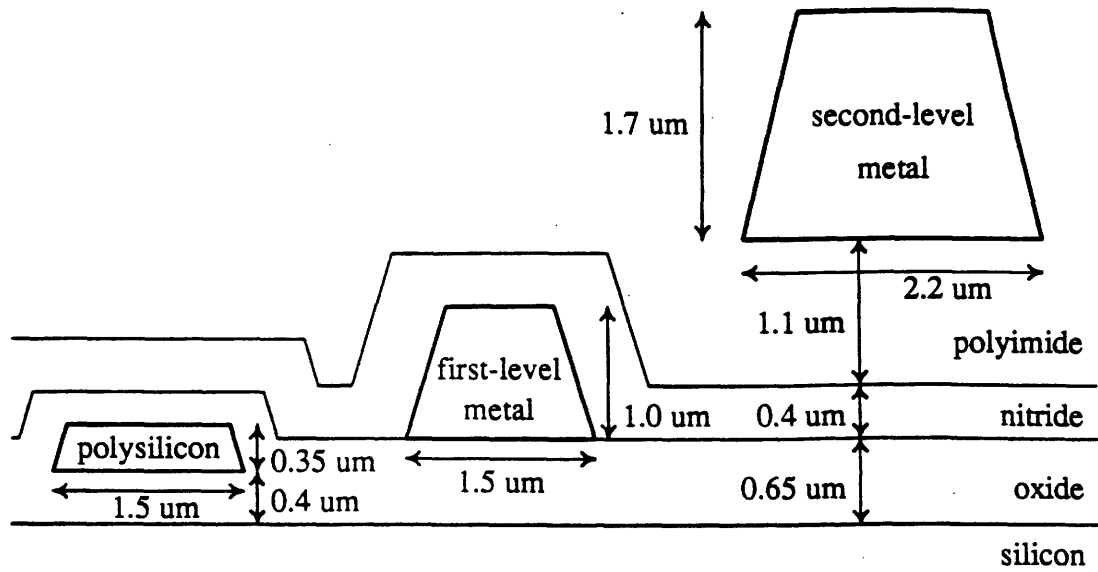


FIGURE 2-9: Interconnection dimensions of 1.5  $\mu\text{m}$  process.

These dimensions are close to those reported in [37] for a representative 1.5  $\mu\text{m}$  nMOS technology.

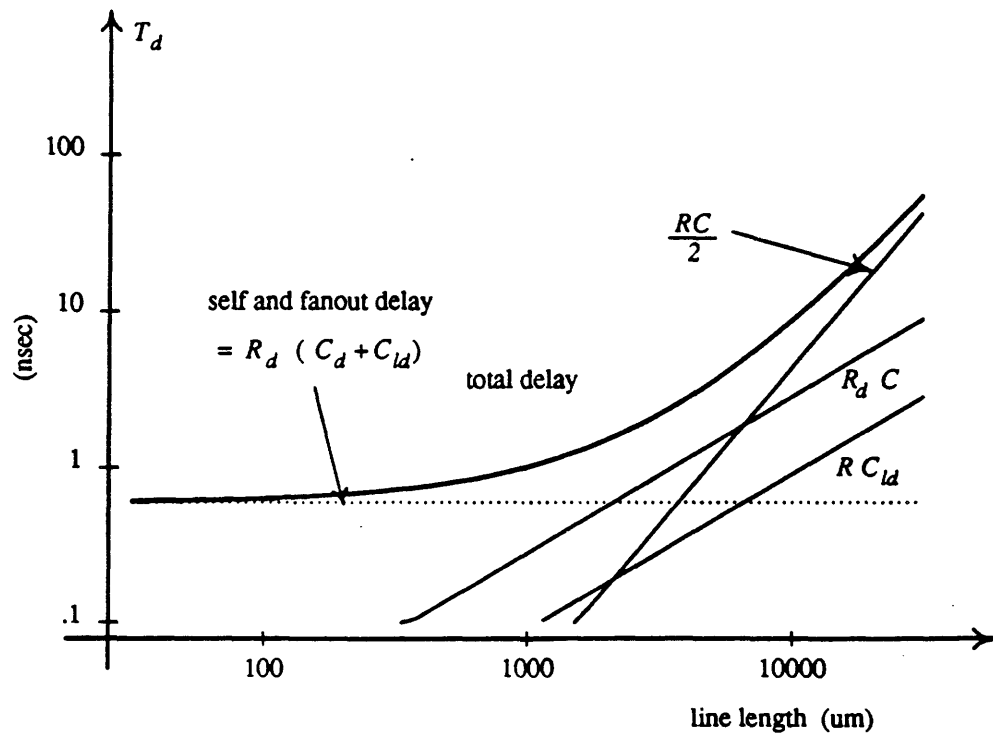


FIGURE 2-10: Delay vs. line length for 1.5  $\mu\text{m}$  process.

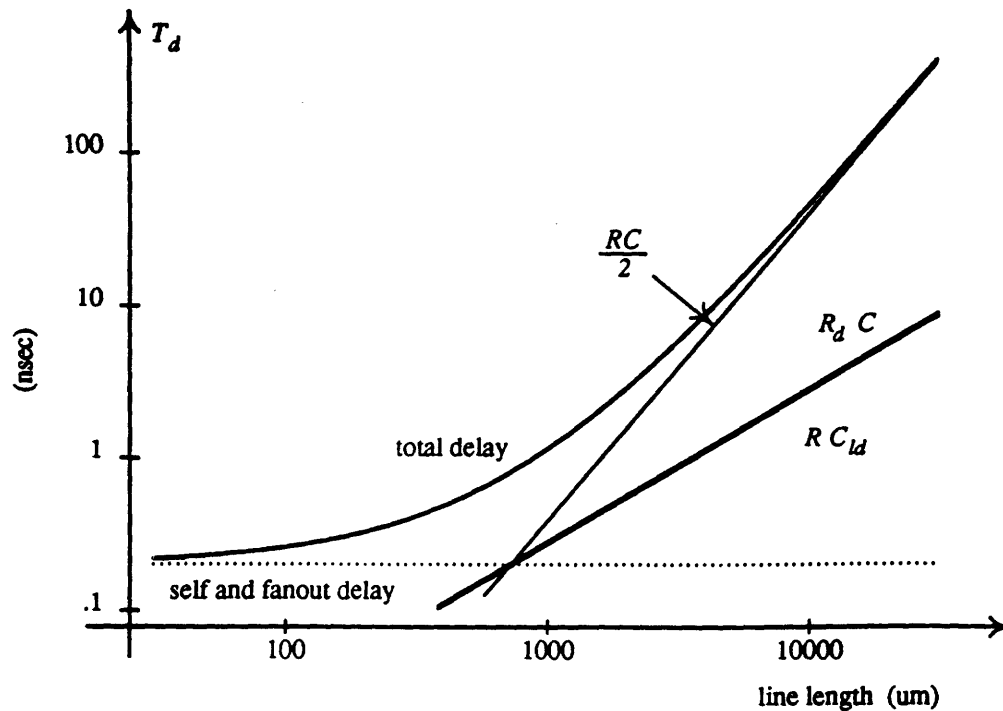


FIGURE 2-11: Delay vs. line length for 0.5  $\mu\text{m}$  process.

Figure 2-11 illustrates the effects of scaling down the 1.5  $\mu\text{m}$  process to a 0.5  $\mu\text{m}$  process ( $\alpha = 3$ ). The intrinsic  $RC$  interconnection delay begins to dominate at a shorter line length (0.7 mm)—at a line length well below the maximum line length.

#### 2.4.2 MOS Scaling in Practice

For reasons described above, conductor thickness and height are usually not shrunk as readily as the width and spacing dimensions as shown in Figure 2-7(c). One investigation [38] estimates that conductor thickness is scaled down by  $t' = t/\sqrt{\alpha}$ . Scaling the conductor thickness by a factor less than  $\alpha$ —i.e., increasing the thickness-to-width ratio—decreases the line resistance, thus reducing propagation delays. Increasing the height-to-width ratio has a smaller affect on ground capacitance. In the above example, improved scaling with  $t' = t/\sqrt{\alpha}$  and  $h' = h/\sqrt{\alpha}$  reduces resistance by 42% and ground capacitance of an isolated conductor by 25%.

This more realistic scaling of MOS circuits has, however, enlarged another problem. A greater amount of capacitance is observed between the sidewalls of adjacent conductors. Figure 2-12 plots the ground and coupling capacitance components of an interconnection parallel to other equi-spaced interconnections on both sides. It shows that as the spacing shrinks, the coupling capacitance increases, and the ground capacitance decreases. The total capacitance also increases at smaller spacings. The two plots show the relative ground and coupling capacitances

		1.5 $\mu\text{m}$	0.5 $\mu\text{m}$
poly-poly	separation ( $\mu\text{m}$ )	1.5	0.5
	Coupling ratio ( $2C_c/C_g$ )	19 %	33 %
metal1-metal1	separation ( $\mu\text{m}$ )	1.0	0.33
	Coupling ratio ( $2C_c/C_g$ )	58 %	79 %
	separation ( $\mu\text{m}$ )	1.8	0.6
	Coupling ratio ( $2C_c/C_g$ )	40 %	65 %
metal2-metal2	separation ( $\mu\text{m}$ )	1.8	0.6
	Coupling ratio ( $2C_c/C_g$ )	60 %	77 %
	separation ( $\mu\text{m}$ )	2.4	0.8
	Coupling ratio ( $2C_c/C_g$ )	52 %	72 %
metal1 over poly	Coupling ratio ( $C_c/C_{g,m1}$ )	54 %	43 %
metal2 over metal1	Coupling ratio ( $C_c/C_{g,m2}$ )	53 %	55 %

Table 2-2: Comparison of maximum coupling capacitance ratios of 1.5  $\mu\text{m}$  and 0.5  $\mu\text{m}$  process.

for the original 1.5  $\mu\text{m}$  and the scaled 0.5  $\mu\text{m}$  process with conductor thickness scaled only by  $\sqrt{\alpha}$ . After accounting for scaling, the  $x$ -axis covers the same separation range. This shows that considerably more coupling capacitance exists for the finer linewidth process. This is also shown in Table 2-2.

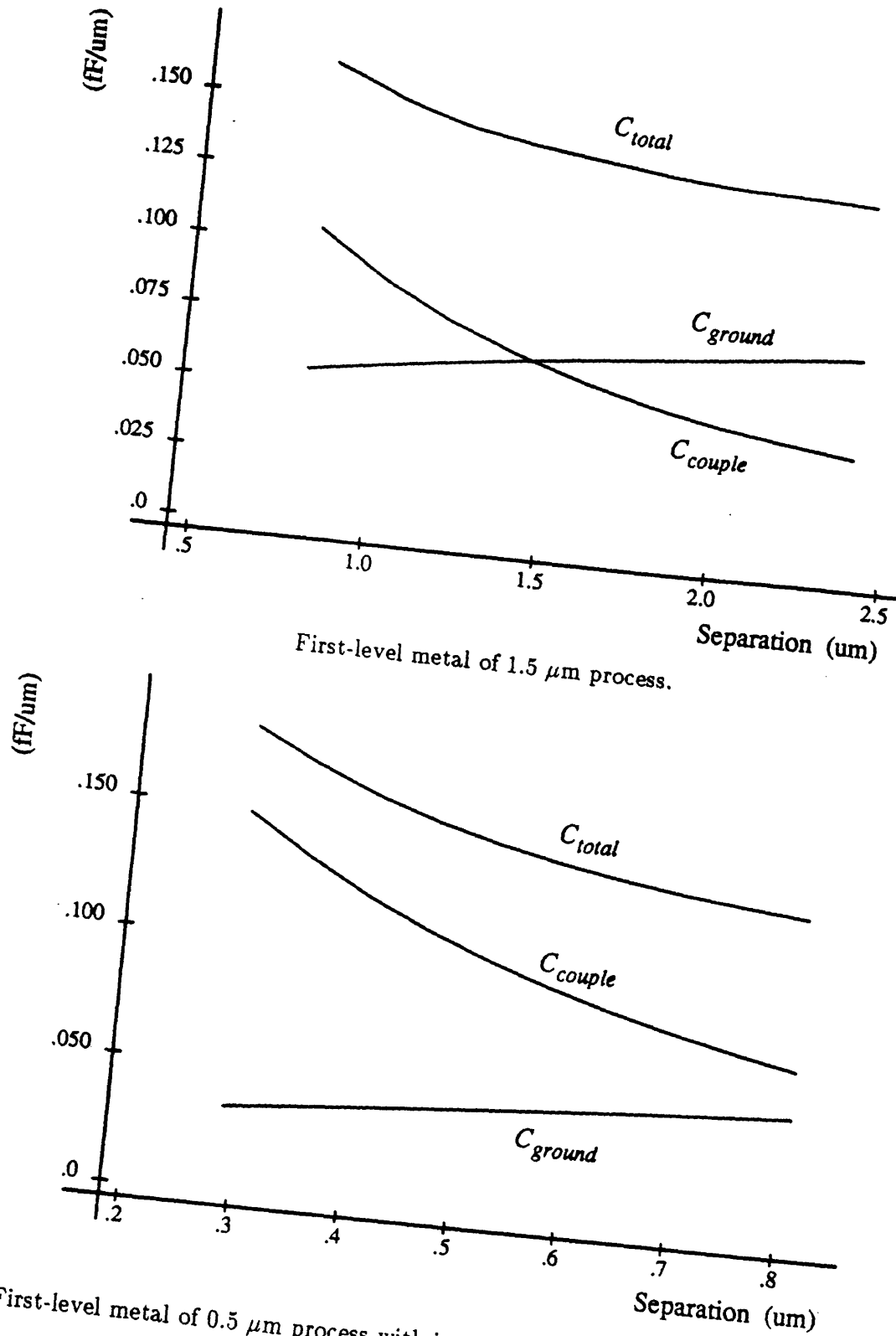
On an integrated circuit, increased coupling capacitance effects two performance measures: delay time and coupling noise. Increased total delay time arises primarily from increased total capacitance. As scaling progresses as described in this section, checks will be needed for a larger number of nodes—nodes which the designer may not suspect of having potential cross-talk problems. The luxury of simulating the noise with a time-consuming simulator cannot be afforded over the hundreds of thousands of nodes on a VLSI circuit.

## 2.5 Analysis of VLSI Transmission Line Interconnections

This section analyzes technologies which are dominated by  $LC$  propagation on interconnections. Silicon bipolar and SOS technologies are included in this discussion, but lossless transmission is assumed. We will see that interconnection propagation time over the length of a circuit is indeed a concern. We will also see that coupling effects of VLSI transmission lines is a major concern in the design of these systems.

### 2.5.1 Transmission Line Propagation

Table 2-3 shows simple calculations for interconnection propagation times and circuit propagation times for each technology. Comparing the two propagation time figures shows in all



First-level metal of 0.5 μm process with increased vertical-to-horizontal ratios.

FIGURE 2-12: Ground and coupling capacitance vs. conductor spacing.

	Silicon bipolar	Silicon on Sapphire	GaAs	Wafer Scale Integration	Ceramic Chip Carrier
$u$ (m/nsec)	0.151	0.091	0.085	0.173	0.099
$l_{max}$ (m)	0.01	0.01	0.005	0.1	0.15
$\tau_{interconnect}$ (nsec)	0.066	0.110	0.059	0.578	1.52
$\tau_{circuit}$ (nsec)	0.1	1.0	0.01 - 0.1	0.1 - 1.0	0.01 - 1.0

$u$  transmission line propagation velocity ( $1/\sqrt{LC}$ )  
 $l_{max}$  approximate maximum line length  
 $\tau_{interconnect}$  propagation time over  $l_{max}$   
 $\tau_{circuit}$  approximate circuit switching time

Table 2-3: Transmission line characteristics of sample technologies.

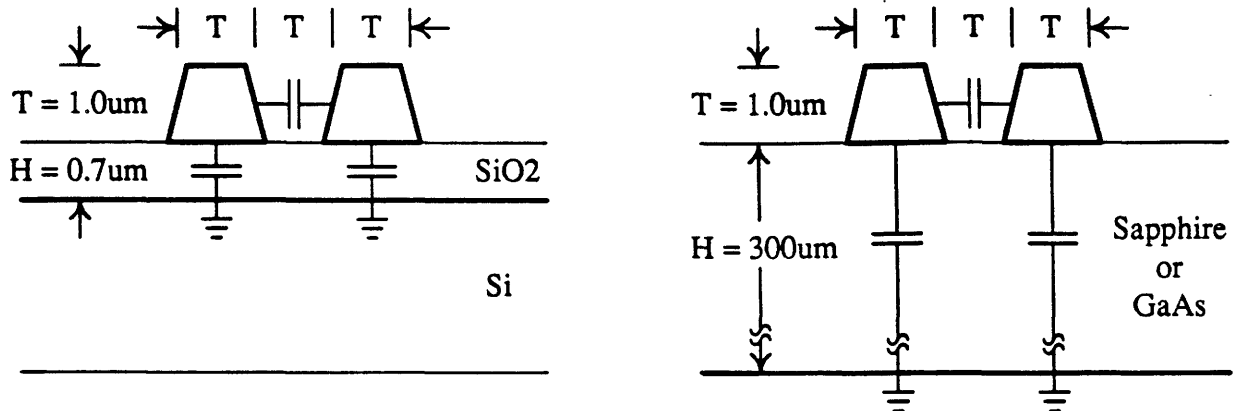


FIGURE 2-13: GaAs dimensions contrasted with silicon's.

cases, except for SOS, that the signal propagation time across the maximum dimension of the wiring surface is comparable to or exceeds the propagation time of the switching devices. For the ceramic chip carriers, interconnection propagation time can be quite significant when mounted with chips from a high-speed technology.

### 2.5.2 Transmission Line Coupling

Transmission line coupling has been the subject of much recent investigation. This is largely due to the increased interconnection density which is needed to support smaller circuitry.

In GaAs and SOS technologies, the fact that substrates are semi-insulating adds to the coupling capacitance. An examination of typical GaAs interconnection dimensions (Figure 2-13) shows that the line-to-line spacing is much less than the line-to-groundplane spacing. The reverse is true for silicon circuits, since the line-to-groundplane spacing is only the thickness of the field SiO<sub>2</sub>.

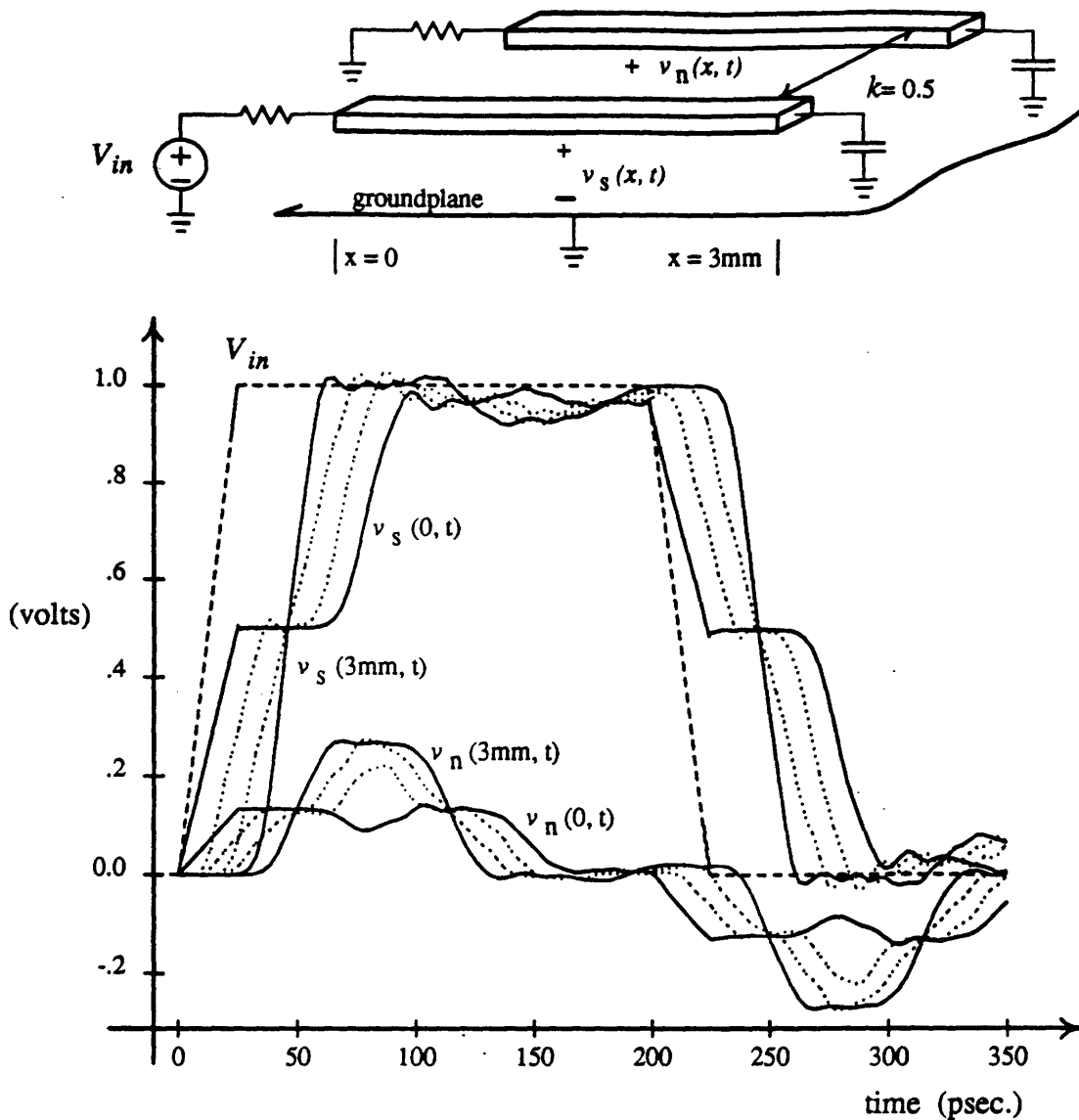


FIGURE 2-14: Transmission line coupling waveforms.

In a 1982 study [33], the coupling capacitance between similar adjacent lines on silicon, sapphire and GaAs substrates was measured. The lines were  $1 \mu\text{m}$  wide, spaced by  $1 \mu\text{m}$  and were only  $0.1 \mu\text{m}$  thick. The amount of coupling per unit distance between two adjacent lines was found to be

$$\frac{C_{\text{couple}}}{C_{\text{total}}} = \begin{cases} 5.6\% & \text{for Silicon,} \\ 45\% & \text{for Sapphire,} \\ 49\% & \text{for GaAs.} \end{cases}$$

Conductors with a larger thickness would have greater coupling.

Figure 2-14 shows the effects of transmission line coupling on two, 3 mm long GaAs interconnections with 50% coupling. The driving ends of the lines are ideally terminated in this example to eliminate reflections. The loading ends have a small capacitance. A substantial noise spike

is seen on the coupled line at the loading end. If the loading ends were ideally terminated, no noise would exist on the loading end, but voltage swings on the signal line would be halved. A noise spike would still exist at the driving end. The coupling effects of interconnections on GaAs substrates is further studied in [39,40,41].

Coupling effects on the IBM ceramic chip carrier has been the subject of several studies. Rubin [42] investigated the cross-talk noise in the presence of reference plane meshes with varying grid spacings. Gu [43] has investigated signal and coupling noise of interconnections in the presence of evenly-spaced crossing lines. The crossing lines add a large number of evenly spaced discontinuities. It is shown that for very sharp transitions ( $\tau < 50\text{psec.}$ ) the distortion on the line of 2 cm or more is too large.

## 2.6 Discussion

The major conclusion to be drawn from this chapter is that interconnections will play an increasingly dominant role in affecting the performance of digital VLSI circuits. The speed of switching circuits will no longer be the limiting factor for many circuits. As linewidths decrease on Silicon MOS IC's, interconnection resistance increases. Some of this can be traded off for increased inter-nodal capacitance. Either way, interconnection problems must be analyzed. Coupling effects on semi-insulating substrate technologies, SOS and GaAs, is larger than on silicon substrate technologies. Inter-chip communication is already a major factor in limiting system performance. Signal propagation and noise degradation on chip carriers or printed circuit boards now must be analyzed much more closely.





---

---

# The Moment Representation

The moment representation is used to describe both waveforms and circuits. In this chapter, we will look at the relationship between waveforms and the moment representation. In following chapters we look at the relationships between circuits and the moment representation.

The first section of this chapter describes the usage of waveforms in this thesis. The next section defines the moment representation. We will see that the moment representation forms a subspace of the Laplace transform. For purposes of simulation, the moment representation is preferred to the Laplace transform because of its simpler computer implementation. The moment representation also forms a superset of the earlier Elmore time representation and closely relates to the second order representation used by Horowitz in [18]. Thus, it characterizes responses more accurately than these earlier representations.

There exists a simple many-to-one projection of the Laplace transform to the moment representation. Thus, most moment representation properties are derived from known Laplace transform properties. These properties are covered in this chapter.

As illustrated in Figure 3-1, there is a closed form, one-to-one mapping between the time domain and the Laplace transform and a projection from the Laplace transform to the  $n^{\text{th}}$  order moment representation domain. Hence, it follows that there is an exact projection from the time domain to the moment representation domain. As we will see, this is through moment integrals. There is not, however, a direct mapping from the moment representation domain to the time domain. The last chapter showed that the simulation algorithms need such a translation. This chapter concludes with a presentation of fast heuristic algorithms that are possible by assuming a shape for the time domain waveform.

## 3.1 Node Voltage Transitions and Waveforms

The node voltage of a logic circuit is composed of a series of *transitions*. For instance, the representative logic circuit node voltage,  $\hat{v}(\hat{t})$ , in Figure 3-2 is composed of five separate transitions,  $v_1(t)$  through  $v_5(t)$ . (A hatted variable, as in  $\hat{t}$ , indicates a global quantity, whereas

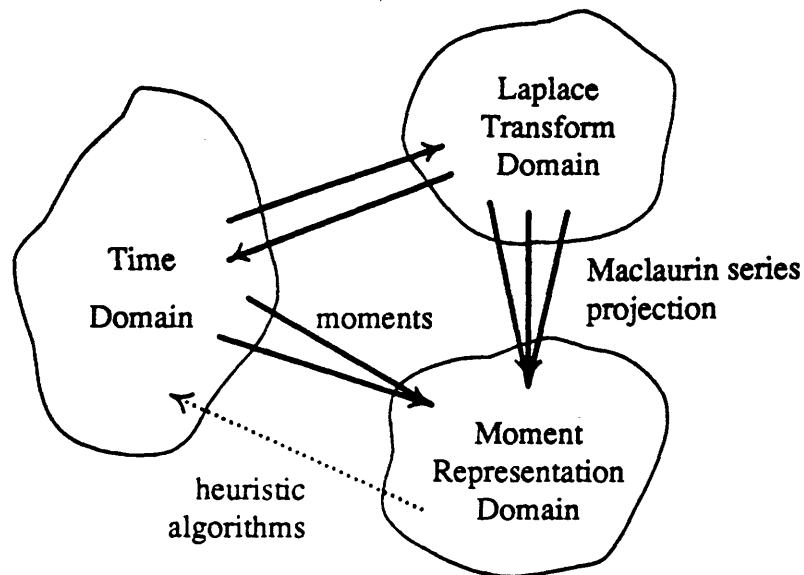


FIGURE 3-1: Interfaces between representation domains.

an unhatted variable indicates the quantity for an isolated transition.) Each of the transitions is caused by a change of logic state somewhere in the circuit. Transitions are classified and tagged at their creation as one of two types:

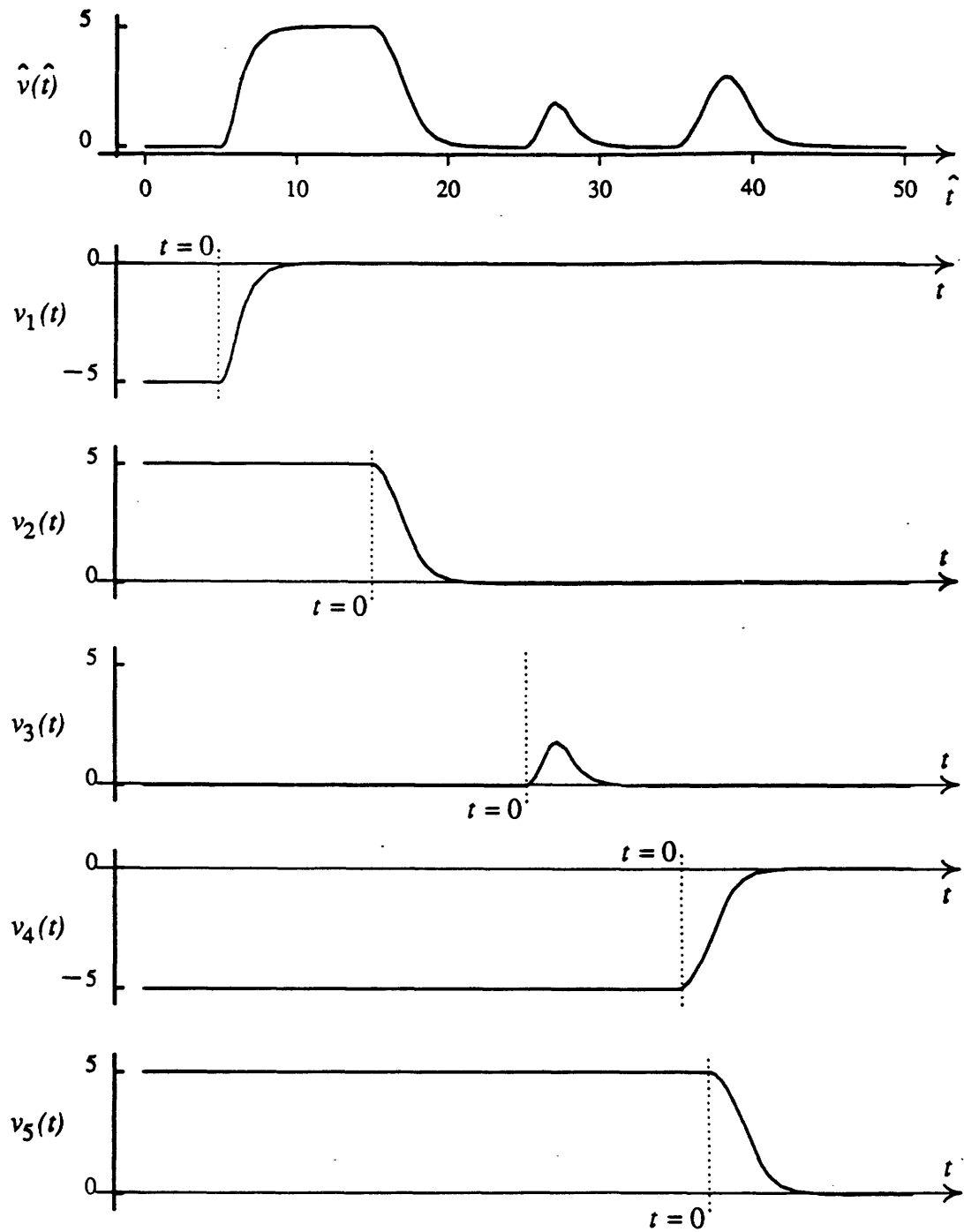
**signal transition:** Two conditions must exist for the simulator to declare a *signal transition*.

First, the transition must be the result of a change in input state on the circuitry that drives the node. Second, the transition must have sufficient d.c. change to cause a logic shift.

**noise transition:** A noise transition is defined as anything else. A noise transition may arise from coupling effects or from charge sharing. It is often characterized by a zero d.c. change, but in some cases may have a sufficiently large d.c. change to cause a logic change.

The motivation for tagging transitions into these types involves the actions taken for each type. Noise transitions cause warnings and indicate a potential circuit error. Signal transitions control the flow of the logical simulation, as their affects are propagated. The first two transitions in Figure 3-2 are normal logic state transitions. The third is a noise transition caused perhaps by a logic transition on a neighboring node. The last two transitions are, again, normal logic state transitions of this node; but the transitions are closely spaced in time such that they overlap. The net result of the last two transitions is a spike resembling the noise waveform of  $v_3(t)$ . The different modeling arises from different causes for the waveforms. As seen later in this chapter, transitions are also classified as *d.c.-shifted* or *non-d.c.-shifted* depending on whether or not the waveform initial and final voltages are equal.

The moment representation is a means of expressing the *waveform* for a single transition. Each moment representation waveform has its own time and voltage scale (denoted without

FIGURE 3-2: Individual transitions of node voltage,  $\hat{v}(t)$ .

hats) as shown in Figure 3-2, and described in the next two sections. Using the moment representation algorithms presented in this thesis, each transition waveform is computed individually. Then, the total waveform is constructed by summing individual transition waveforms. For the particular example in Figure 3-2, five moment representation waveforms are needed to construct the total waveform. In general, a node voltage in the moment representation simulator is a d.c. voltage interspersed by a chronological series of transitions. After each transition, the d.c. voltage is updated to reflect the d.c. voltage change of the transition.

Most discussion and examples in this thesis focus around single transitions. While it is often assumed that only one transition occurs between  $t = 0$  and  $t = \infty$ , all discussions can be generalized to include many transitions where the individual transitions are combined.

### 3.2 Representation Definition

We now assume that  $\hat{v}(\hat{t})$  has only one transition for all  $t > 0$ . This assumption holds for the remainder of this chapter so that we are not troubled by upper time bounds in the following analysis.

The moment representation of this waveform is defined through the Laplace transform of the waveform's derivative. The Laplace transform pair of its derivative,  $\hat{v}'(\hat{t})$ , is

$$\hat{v}'(\hat{t}) \longleftrightarrow V(s) = \int_0^{\infty} e^{-s\hat{t}} \hat{v}'(\hat{t}) d\hat{t} \quad (3.1)$$

Any initial delay between the global time origin,  $\hat{t} = 0$ , and the start of the transition is removed from the Laplace transform integral by defining a new time origin,  $t = 0$ , where the transition begins, as shown in each transition of Figure 3-2. The variable  $t_0$  denotes the delay from the global origin to the transform origin, or  $\hat{t} = t + t_0$ . With the new time origin, Equation (3.1) becomes

$$\hat{v}'(t + t_0) \longleftrightarrow V(s) = e^{s t_0} \left( \int_0^{\infty} e^{-st} \hat{v}'(t) dt \right). \quad (3.2)$$

In general, it is best to move the origin,  $t = 0$ , as close as possible to the start of the transition, in order to keep the best approximation in the next step. Expanding  $e^{-st}$  of Equation (3.2) into a Maclaurin series,

$$V(s) = e^{s t_0} \left( \int_0^{\infty} \hat{v}'(t) dt - s \int_0^{\infty} t \hat{v}'(t) dt + \frac{s^2}{2!} \int_0^{\infty} t^2 \hat{v}'(t) dt - \frac{s^3}{3!} \int_0^{\infty} t^3 \hat{v}'(t) dt + \dots \right), \quad (3.3)$$

and finally defining the integral quantities as  $M_0, M_1, M_2$ , etc., gives

$$V(s) = e^{s t_0} \left( M_0 - s M_1 + \frac{s^2}{2!} M_2 - \frac{s^3}{3!} M_3 + \dots \right). \quad (3.4)$$

The  $M_0, M_1, \dots$  terms are the *moments* of the  $\hat{v}'(t)$  function. It is also useful to talk about the normalized moments,  $\bar{M}_1 = M_1/M_0$ ,  $\bar{M}_2 = M_2/M_0$ , etc. The normalized moments are similar

to the moments of a probability density function—the first moment relates to the *mean* of a function, the second moment relates to the *mean* and *variance* of a function, and so forth.  $\bar{M}_1$  also equals the Elmore time constant presented in Section 1.1.3 and discussed in the *RC* tree literature [13,14,44].

For any initial delay,  $t_0$ , we find the  $n^{\text{th}}$  order projection of any Laplace transform,  $V(s)$ , into the moment representation domain by taking the Maclaurin series expansion (Taylor's series expansion around 0) of  $V(s)$  with respect to  $s$  and truncating the terms with polynomial order greater than  $n$ . That is,

$$V(s) \Rightarrow e^{-s t_0} \left( M_0 - M_1 s + M_2 \frac{s^2}{2!} - \cdots + M_n \frac{s^n}{n!} \right) \quad (3.5)$$

The  $n^{\text{th}}$  order *moment representation* is defined as the set of numbers,  $\{t_0, M_0, M_1, \dots, M_n\}$ , and  $n$  is the *approximation order*.

As shown later, a greater approximation order results in a more accurate waveform approximation, but requires more computation when performing operations on the waveform. An approximation order of three is typical for the simulation examples in this thesis, and gives close approximations to true circuit waveforms.

The polynomial portion of Equation (3.5) by itself is referred to as the *moment polynomial*.

From this point onward, a function of  $s$  with a tilde, as in  $\tilde{V}(s)$ , indicates a moment representation or moment polynomial, differentiating it from the Laplace transform,  $V(s)$ . A subscript, as in  $\tilde{V}(s)_n$ , indicates an  $n^{\text{th}}$  order moment representation.

### 3.2.1 Expressed in terms of $\hat{v}(t)$

It is useful to express the waveform representation in terms of the original waveform and not its derivative. To guarantee that moments of the original function are defined, we shift the function such that the final value asymptotically approaches zero, as in the transitions of Figure 3-2. The shifted function is denoted  $v(t)$ , where

$$v(t) = \hat{v}(t) - \hat{v}(\infty).$$

This does not change the analysis of the last section, since

$$\frac{d}{dt} \hat{v}(t) = \frac{d}{dt} v(t) = \hat{v}'(t).$$

Furthermore, we must ensure that the moment integrals are finite. For an order  $n$  approximation this requires  $\int_0^\infty t^n v(t) dt$  to be finite. All waveforms in this thesis meet this requirement.<sup>1</sup>

<sup>1</sup>Reasons for this statement being true may not be clear at this point, but, in Section 3.6, all assumed waveform shapes obey this requirement, and circuit waveforms are forced to this requirement by restriction 3 on Page 70.

term	$\hat{v}'(t)$ waveform	$-v(t)$ waveform
$t_0$	delay	delay
$M_0$	$\int_0^\infty \hat{v}'(t) dt$	$-v(0) = \hat{v}(\infty) - \hat{v}(0)$ d.c. transition
$M_1$	$\int_0^\infty t \hat{v}'(t) dt$ first moment	$\int_0^\infty [-v(t)] dt$ area under $-v(t)$ waveform Elmore time constant
$M_2$	$\int_0^\infty t^2 \hat{v}'(t) dt$ second moment	$2 \int_0^\infty t [-v(t)] dt$ twice first moment
$M_3$	$\int_0^\infty t^3 \hat{v}'(t) dt$ third moment	$3 \int_0^\infty t^2 [-v(t)] dt$ three times second moment

Table 3-1: Terms of moment representation for one transition.

By using the above restriction, it is easy to convert Equation (3.3) to a form with  $v(t)$ . Integrating by parts, we know that

$$\int_0^\infty t^n \frac{dv(t)}{dt} dt = [t^n v(t)]_0^\infty - n \int_0^\infty t^{n-1} v(t) dt.$$

The  $t^n v(t)$  term vanishes at both  $t = 0$  and  $t = \infty$  (using the restriction). Thus, Equation (3.3) becomes:

$$V(s) = e^{s t_0} \left( -v(0) + s \int_0^\infty v(t) dt - s^2 \int_0^\infty t v(t) dt + \frac{s^3}{2!} \int_0^\infty t^2 v(t) dt - \dots \right), \quad (3.6)$$

Equating the  $s^0, s^1, s^2, \dots$  terms of Equations (3.4), (3.3) and (3.6) gives

$$M_0 = \int_0^\infty \hat{v}'(t) dt = -v(0),$$

$$-M_1 = - \int_0^\infty t \hat{v}'(t) dt = \int_0^\infty v(t) dt,$$

$$\frac{1}{2!} M_2 = \frac{1}{2!} \int_0^\infty t^2 \hat{v}'(t) dt = - \int_0^\infty t v(t) dt,$$

and so forth.

The terms of the waveform representation are summarized in Table 3-1 and Figure 3-3. Here the moments are easier to see with the negative of the waveform,  $-v(t) = \hat{v}(\infty) - \hat{v}(t)$ .

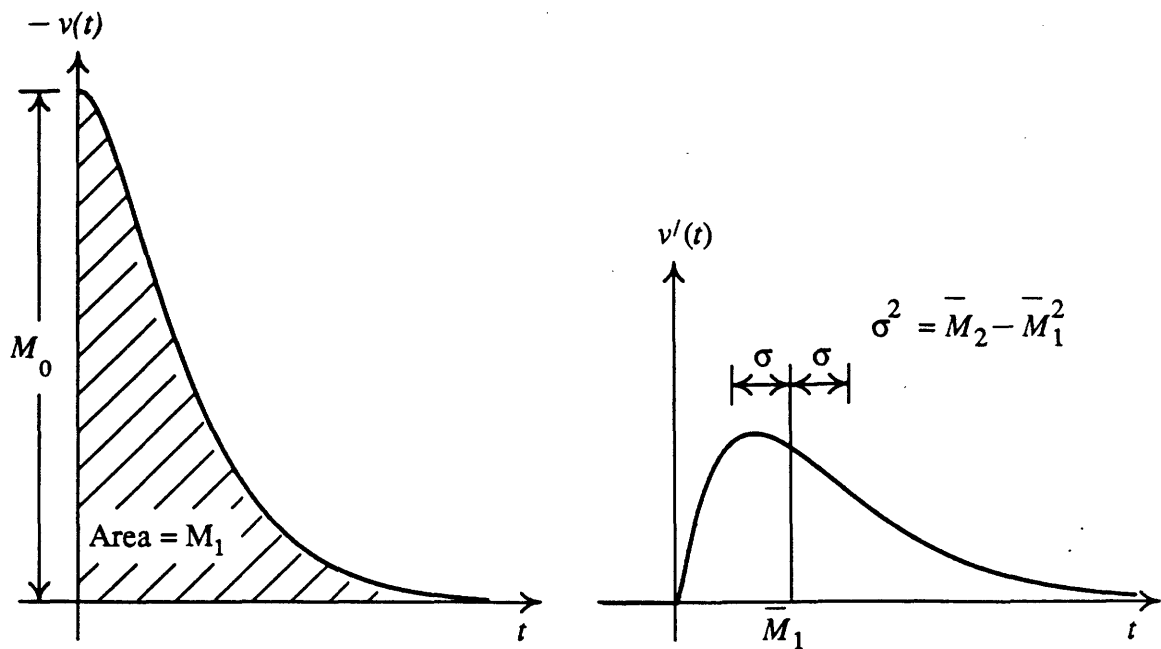


FIGURE 3-3: Graphical meaning of some moment representation terms.

### 3.2.2 Expressed in terms of the $V(s)$ rational function

It is also useful to express the moments in terms of the constants of the more common rational function form for the Laplace transform,

$$V(s) = \frac{a_0 + a_1 s + a_2 s^2 + a_3 s^3 + \dots}{1 + b_1 s + b_2 s^2 + b_3 s^3 + \dots} \quad (3.7)$$

The moments are easily found by taking terms from the Maclaurin series projection of Equation (3.7) into the moment representation domain:

$$\begin{aligned} M_0 &= [V(s)]_{s=0} = a_0 \\ M_1 &= - \left[ \frac{dV(s)}{ds} \right]_{s=0} = a_0 b_1 - a_1 \\ M_2 &= \left[ \frac{d^2 V(s)}{ds^2} \right]_{s=0} = 2(a_0 b_1^2 - a_0 b_2 - a_1 b_1 + a_2) \\ M_3 &= - \left[ \frac{d^3 V(s)}{ds^3} \right]_{s=0} = 6[a_0(b_3 - 2b_1 b_2 + b_1^3) + a_1(b_2 - b_1^2) + a_2 b_1 - a_3] \end{aligned}$$

and so forth.

## 3.3 Properties of the Moment Representation

Since the waveform representation is a projection from the Laplace transform, the well known Laplace transform properties can be used to derive a list of moment representation

properties. The Laplace transform properties are derived in most any circuit theory textbook, such as [45].

### 3.3.1 Shifting in Time

Shifting a waveform in time results in a multiplication of the Laplace transform by a constant:

$$v(t - \tau) \longleftrightarrow e^{-\tau s} V(s).$$

This fact was already used in going from Equation (3.1) to (3.2). From Equation (3.3) we see that it is easy to time shift a waveform representation by changing  $t_0$ :

$$t_{0,shifted} = t_0 + \tau.$$

Sometimes it becomes necessary to hold  $t_0$  constant and change the  $M$  terms. In these cases, the moment polynomial of Equation (3.3) is multiplied by the Maclaurin Series expansion of  $e^{-\tau s} = 1 - s\tau + \frac{(s\tau)^2}{2!} - \frac{(s\tau)^3}{3!} + \dots$ , and exact terms of  $\tilde{V}_{shifted}$  are

$$\begin{aligned} M_{0,V_{shifted}} &= M_0, \\ M_{1,V_{shifted}} &= M_0\tau + M_1, \\ M_{2,V_{shifted}} &= M_0\tau^2 + 2M_1\tau + M_2, \\ M_{3,V_{shifted}} &= M_0\tau^3 + 3M_1\tau^2 + 3M_2\tau + M_3. \end{aligned} \tag{3.8}$$

### 3.3.2 The Linearity Property

Laplace transforms exhibit the linearity property:

$$a v_1(t) + b v_2(t) \longleftrightarrow a V_1(s) + b V_2(s).$$

This property easily projects into the waveform representation, *provided the two  $t_0$  terms are equal*. If not, they must be equated with the second time shifting method described in the previous section, using Equation (3.9). (Usually, the waveform with greater  $t_0$  is shifted backward, so the resulting waveform's starting point is the earlier of the two original starting points.) With this provision, the new waveform terms of  $\tilde{V}_{total}(s)_n = a \tilde{V}_1(s)_n + b \tilde{V}_2(s)_n$  are:

$$\begin{aligned} t_{0,V_{total}} &= \min(t_{0,V_1}, t_{0,V_2}) \equiv t_{0,V_{min}} \\ &\quad \text{(The waveform with the largest } t_0 \text{ is time} \\ &\quad \text{shifted by } t_{0,V_{min}} - t_{0,V_{max}}) \\ \hat{v}_{total}(0) &= a \hat{v}_1(0) + b \hat{v}_2(0) \\ M_{0,V_{total}} &= a M_{0,V_1} + b M_{0,V_2} \\ M_{1,V_{total}} &= a M_{1,V_1} + b M_{1,V_2} \\ M_{2,V_{total}} &= a M_{2,V_1} + b M_{2,V_2} \\ M_{3,V_{total}} &= a M_{3,V_1} + b M_{3,V_2} \end{aligned} \tag{3.9}$$



This operation should be used with some caution. As we will see later, inverse transforms assume a single, smooth transition, so one cannot indiscriminately combine different waveforms with widely separated starting times.

### 3.3.3 Derivatives of Waveforms

Using the Laplace transform pair for the time derivative of a function,

$$\frac{d}{dt} v'(t) \longleftrightarrow s V(s) - v(0) = -v(0) + M_0 s - M_1 s^2 + \frac{M_2}{2!} s^3 - \dots \quad (3.10)$$

we see that the following substitutions can be made:

$$\begin{aligned} M_{0,V_{deriv}} &= -v(0) \\ M_{1,V_{deriv}} &= -M_0 \\ M_{2,V_{deriv}} &= -2 M_1 \\ M_{3,V_{deriv}} &= -3 M_2 \end{aligned} \quad (3.11)$$

This is in agreement with Table 3-1 of Section 3.2.1.

### 3.3.4 Linear Network Properties

The great advantage of using the Laplace transform in linear circuit analysis is that circuit elements and networks can be represented by Laplace transforms. Furthermore, circuit calculations with Laplace transforms are very simple—any independent linear circuit described by its impulse response transform  $H(s)$  responds to an input waveform described by transform  $V(s)$  with an output waveform described by transform  $Y(s) = V(s) \cdot H(s)$ .

This fact projects into the moment representation domain. The response of a linear network is easily found in terms of the moment representation parameters with

$$\begin{aligned} \tilde{Y}(s)_n = \tilde{V}(s)_n \cdot \tilde{H}(s)_n &= e^{s(t_0,V+t_0,H)} \left[ M_{0,V} - M_{1,V} s + M_{2,V} \frac{s^2}{2!} - \dots + M_{n,V} \frac{s^n}{n!} \right] \\ &\cdot \left[ M_{0,H} - M_{1,H} s + M_{2,H} \frac{s^2}{2!} - \dots + M_{n,H} \frac{s^n}{n!} \right]. \end{aligned} \quad (3.12)$$

The next chapter describes the link between circuits and the moment representation in much detail.

### 3.3.5 Non-Linear Network Properties

This is where the helpful list of Laplace transform properties ends. Analysis of non-linear networks with Laplace transforms is cumbersome, to say the least. However, in Chapter 6 a macromodeling method is presented for converting a non-linear digital circuit into a linear circuit equivalent. After this conversion, the moment representation again becomes beneficial.

### 3.4 Representations of Simple Waveform Functions

Table 3-2 shows some well-known Laplace transform pairs for simple waveform functions, as well as the unshifted waveform representation parameters. These transform pairs, particularly the exponentials, will be used in the next sections.

### 3.5 Time Domain to Moment Representation Conversion

A necessary step of the simulator is converting between the time domain representation,  $\hat{v}(t)$ , and the moment representation,  $\tilde{V}(s)$ . This section presents the time domain to moment representation conversion method, and the next section presents the moment representation to waveform conversion methods.

The ability to convert a time domain waveform to moment representation is needed (1) as an interface routine to the user's input should an input waveform be specified by an arbitrarily shaped waveform, and (2) in the macromodel parameter extract routines, where macromodel parameters are extracted from SPICE waveforms. Inner loops of the simulator program do not rely on this algorithm, so efficiency of this conversion is not crucial.

This conversion method is readily seen in the last column of Table 3-1. The  $t_0$  (defined where the waveform first deviates by more than a few percent from the initial voltage),  $\hat{v}(0)$  and  $M_0$  representation parameters are easily picked off of the waveform. Then, the waveform is shifted from  $\hat{v}(t)$  to  $v(t)$ , and the integrals for  $M_1, M_2, \dots, M_n$  are evaluated numerically.

### 3.6 Moment Representation to Time Domain Conversion

Conversion from the moment representation to time domain representation is needed substantially throughout the simulation program as demonstrated in Section 1.2. Unfortunately, this conversion is more arduous, since there is no unique direct technique to undo the projections shown in Figure 3-1. The overall accuracy and computation time of the simulator are greatly dependent on the accuracy and efficiency of this conversion method.

#### 3.6.1 Basic principles

While general inverse Laplace transform methods exist [46], they are not appropriate for the moment representation. Strictly speaking, the inverse Laplace transform of the moment polynomial would yield impulses, doublet (double impulses), etc. Too much information is lost in the Maclaurin series approximation to use the general inverse transform methods. Specifically, the general inverse transform methods need both  $P_n(s)$  and  $Q_m(s)$  of a Laplace transform,  $\frac{P_n(s)}{Q_m(s)}$ . The Maclaurin Series merges  $P$  and  $Q$  into a single polynomial.

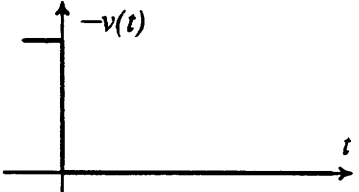
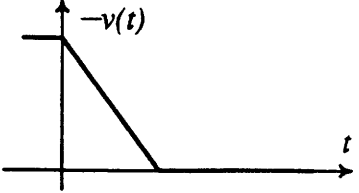
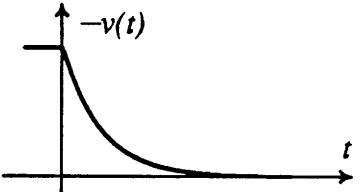
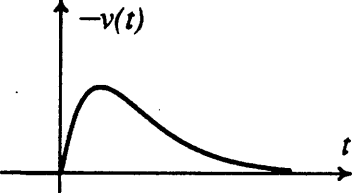
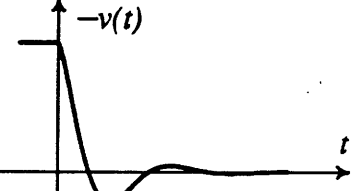
$-v(t)$	$v'(t)$	$V(s)$	$M_0$	$M_1$	$M_2$	$M_3$
	$\delta(t)$	1	1	0	0	0
$1 - u(t)$	$\frac{1}{\tau}[u(t) - u(t - \tau)]$	$\frac{e^{-s\tau} - 1}{s\tau}$	1	$\frac{1}{2}\tau$	$\frac{1}{3!}\tau^2$	$\frac{1}{4!}\tau^3$
	$e^{-\frac{t}{\tau}}$	$\frac{1}{1+s\tau}$	1	$\tau$	$2\tau^2$	$3!\tau^3$
$[1 - \frac{t}{\tau}] \cdot [u(t) - u(t - \tau)]$	$\frac{1}{\tau} e^{-\frac{t}{\tau}}$	$\frac{1}{1+s\tau}$	1	$\tau$	$2\tau^2$	$3!\tau^3$
	$t^n e^{-\frac{t}{\tau}}$	$-\frac{n! \tau^{n+1} s}{(1+s\tau)^{n+1}}$	0	$n! \tau^{n+1}$	$2(n+1)! \tau^{n+2}$	$3(n+2)! \tau^{n+3}$
$(\frac{t^n}{\tau} - n\tau^{n-1}) e^{-\frac{t}{\tau}}$	$e^{-\alpha t} \cos \beta t$	$\frac{\alpha^2 + \alpha s + \beta^2}{(s + \alpha)^2 + \beta^2}$	1	$\frac{\alpha}{\alpha^2 + \beta^2}$	$2 \frac{\alpha^2 - \beta^2}{(\alpha^2 + \beta^2)^2}$	$3! \frac{\alpha^3 - 3\alpha\beta^2}{(\alpha^2 + \beta^2)^3}$
	$e^{-\alpha t} (\alpha \cos \beta t + \beta \sin \beta t)$					
						

Table 3-2: Laplace transforms and waveform representation terms for simple functions.

Instead, a technique analogous to the method of undetermined coefficients is adopted. This technique (1) assumes a waveform shape,  $g(t)$ , which may be linear, exponentially decaying, underdamped decaying, etc., and (2) computes various coefficients and parameters of  $g(t)$  such that the first  $n$  moments of  $g(t)$  equal the first  $n$  moments of  $\tilde{V}(s)_n$ :

$$\begin{aligned} M_{1,g} &= M_{1,v}, \\ M_{2,g} &= M_{2,v}, \\ &\vdots \\ M_{n,g} &= M_{n,v}. \end{aligned}$$

$g(t)$  is said to be an *assumed waveform shape* of order  $n$ . A greater approximation order generally yields a better approximation to the true waveform, but generally requires more computation.

Ideally, the number of variable parameters in  $g(t)$  equals  $n$ , the number of constraints on the waveform. In some cases, this is not true, and heuristic methods must approximate some parameters.

The success of this method depends on how well the assumed waveform shape captures the form of all *real* waveform shapes that it models. It is found, for instance, that most logic transitions are modeled quite well by exponential functions.

Commonly used assumed waveform shapes are listed in Table 3-3. Most have a fixed order—only the last has an arbitrary order. Restrictions for some waveforms are noted in the fourth column. The restrictions apply to the d.c. change. (If the waveform's final value returns to the initial value, then  $M_0 = 0$ .) Conversion with some assumed waveform shapes—those requiring special algorithms—are described in the next sections of this chapter. The last section discusses how a good choice of  $g(t)$  can be made.

### 3.6.2 Polynomial Exponential

The polynomial exponential assumed waveform shape is the most general of the entries on Table 3-3. It has an arbitrary approximation order, and as  $n$  increases, the approximation becomes more exact. Because of its general nature and its ease in computation, this method will be discussed in more detail. This method does not, in general, give the most accurate match to true waveforms, so it is not always the preferred method.

The polynomial exponential waveform approximation, described by

$$g(t) = \left[ k_0 + k_1 \frac{t}{\tau_*} + k_2 \left( \frac{t}{\tau_*} \right)^2 + \cdots + k_n \left( \frac{t}{\tau_*} \right)^n \right] e^{-\frac{t}{\tau_*}}, \quad (3.13)$$

assumes a dominant real pole at  $-1/\tau_*$ .

If the true waveform being modeled has any perturbations away from a single pole (exponential) waveform, these are reflected in the polynomial terms with  $n > 0$ .

waveform shape	$g(t)$	d.c. restriction	approx. order	method
linear	$A[1 - \frac{t}{\tau}][u(t) - u(t - \tau)]$	$M_0 \neq 0$	1	$A = M_0$ $\tau = 2 M_1 / M_0$
single exponential	$A e^{-\frac{t}{\tau}}$	$M_0 \neq 0$	1	$A = M_0$ $\tau = M_1 / M_0$
erlang	$A t^{n-1} e^{-\frac{t}{\tau}}$	$M_0 = 0$	2+	$n = \text{round} \left( \frac{1}{\frac{4}{3} \frac{M_1 M_3}{M_2^2} - 1} \right)$ $\tau = \frac{M_2}{2^n M_1}$ $A = \frac{M_1}{(n-1)! \tau^n}$
underdamped signal exponential	$A e^{-\frac{t}{\tau}} \cos \beta t$	$M_0 \neq 0$	2	$A = M_0$ $\tau = \frac{2 M_1^2 - M_2}{M_1}$ $\beta = \frac{\sqrt{M_1^2 - M_2}}{2 M_1^2 - M_2}$
underdamped noise exponential	$A e^{-\frac{t}{\tau}} \sin \beta t$	$M_0 = 0$	3	$A = \frac{12 M_1^2}{\sqrt{27 M_2^2 - 24 M_1 M_3}}$ $\tau = \frac{M_1 \sqrt{27 M_2^2 - 24 M_1 M_3}}{2 M_1 M_3 - 3 M_2^2}$ $\beta = \frac{3 M_2^2 - 2 M_1 M_3}{3 M_1 M_2}$
double exponential	$(k_1 + k_2 t) e^{-\frac{t}{\tau_a}}$ $+ (k_3 + k_4 t) e^{-\frac{t}{\tau_b}}$		3	see Section 3.6.3
polynomial exponential	$[k_0 + k_1 \frac{t}{\tau} + k_2 (\frac{t}{\tau})^2 + \dots + k_n (\frac{t}{\tau})^n] e^{-\frac{t}{\tau}}$		$n$	see Section 3.6.2

Table 3-3: Common Assumed Waveform Shapes

### Estimating $\tau$

A value for  $\tau_*$  must be estimated from the moment representation parameters,  $M_0, M_1, \dots, M_n$ . For finding  $\tau_*$  we assume the *true* waveform is described by a series of exponentials,

$$V(s) = \frac{P}{(s - \tau_*)(s - \tau_2) \cdots (s - \tau_n)} \longleftrightarrow v(t) = c_* e^{-\frac{t}{\tau_*}} + c_2 e^{-\frac{t}{\tau_2}} + \cdots + c_n e^{-\frac{t}{\tau_n}}.$$

This assumption differs from the assumed waveform shape; we configure this part of the problem differently, since overall, there is one more parameter than the number of moment constraints. The  $i^{\text{th}}$  order moment of the above waveform is

$$M_i = \frac{1}{i!} (c_* \tau_*^i + c_2 \tau_2^i + \cdots + c_n \tau_n^i). \quad (3.14)$$

If  $\tau_*$  is the dominant time constant, then

$$\tau_*^i \gg \tau_2^i, \dots, \tau_n^i,$$

particularly for large  $i$ ,<sup>2</sup> and therefore

$$M_i \approx \frac{1}{i!} c_* \tau_*^i.$$

Thus, a good approximation for  $\tau_*$  from an  $n^{\text{th}}$  order moment representation is

$$\tau_* \approx \frac{1}{n} \frac{M_n}{M_{n-1}}. \quad (3.15)$$

In reality, our estimate of  $\tau$  is just a ratio of moments, but it serves as a very good time constant approximation for the true waveform's asymptotic decay.

### Computing the polynomial

Once an estimate of  $\tau_*$  is computed, the polynomial terms of Equation (3.13) are computed from a linear combination of a set of basis functions,  $\{f_0(t), f_1(t), \dots, f_n(t)\}$ , i.e.,

$$g(t) = a_0 f_0(t) + a_1 f_1(t) + \cdots + a_n f_n(t).$$

Figure 3-4 shows one possible set of basis functions.

The key to efficient computation with this method is for basis function  $f_i(t)$  to have all moments of order less than  $i$  be equal to zero. Then, the  $a$  coefficients can be computed in increasing order; each is computed without affecting the lower-ordered coefficients.

Suppose  $\tilde{V}_0(s) = M_0 - M_1 s + \frac{M_2}{2!} s^2 - \cdots + \frac{M_n}{n!} s^n$  is the initial, moment representation approximation. The conversion begins by setting

$$a_0 = M_0.$$

<sup>2</sup>The non-dominant  $\tau$ 's may be complex. In this case,  $\tau_*$  must be larger than the inverse of the complex pole's magnitude.

$i$	$f_i(t)$	$F(s)$				
		$M_0$	$M_1$	$M_2/2!$	$M_3/3!$	$M_4/4!$
0	$e^{-\frac{t}{\tau}}$	1	$\tau$	$\tau^2$	$\tau^3$	$\tau^4$
1	$\frac{t}{\tau} e^{-\frac{t}{\tau}}$	0	$\tau$	$2\tau^2$	$3\tau^3$	$4\tau^4$
2	$[(\frac{t}{\tau})^2 - 2\frac{t}{\tau}] e^{-\frac{t}{\tau}}$	0	0	$2\tau^2$	$6\tau^3$	$12\tau^4$
3	$[(\frac{t}{\tau})^3 - 6(\frac{t}{\tau})^2 + 6\frac{t}{\tau}] e^{-\frac{t}{\tau}}$	0	0	0	$6\tau^3$	$24\tau^4$
4	$[(\frac{t}{\tau})^4 - 12(\frac{t}{\tau})^3 + 36(\frac{t}{\tau})^2 - 24\frac{t}{\tau}] e^{-\frac{t}{\tau}}$	0	0	0	0	$24\tau^4$

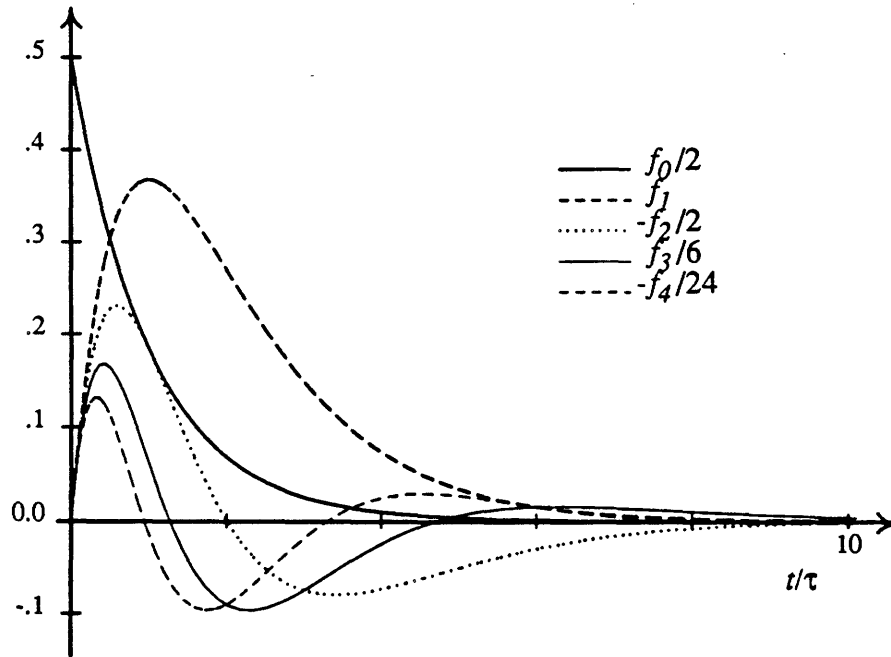


FIGURE 3-4: Basis functions for the polynomial exponential waveform approximation.

The  $i^{th}$  order basis function is  $f_i(t) = \frac{d^i}{dt^i} \left( \frac{t}{\tau} e^{-\frac{t}{\tau}} \right)$ . From (3.10) we know that if  $f_i(0) = 0$  then  $f_i(t) \rightarrow s^i \bar{F}(s)$ , thus guaranteeing that the first  $i$  moments are zero.

Then the transform of the first basis function,  $\bar{F}_0(s)$ , is subtracted, giving the transform of the unmatched waveform

$$\begin{aligned}\tilde{V}_1(s) &= \tilde{V}(s) - a_0 \bar{F}_0(s) \\ &= 0 - (M_1 - a_0 M_{1,F_0})s^1 + (M_2 - a_0 M_{2,F_0})\frac{s^2}{2!} - \cdots + (M_n - a_0 M_{n,F_0})\frac{s^n}{n!}\end{aligned}$$

Next the  $s^1$  term of  $\tilde{V}_1(s)$  is zeroed by subtracting

$$\tilde{V}_1(s) - a_1 \bar{F}_1(s) \equiv \tilde{V}_2,$$

where

$$a_1 = -\frac{M_{1,V_1}}{M_{1,F_1}}.$$

The process is continued where, in general,

$$a_i = -\frac{i^{\text{th}} \text{ term of } V^i(s)}{i^{\text{th}} \text{ term of } \bar{F}_i(s)}$$

and

$$\tilde{V}_{i+1}(s) = \tilde{V}_i(s) - a_i \bar{F}_i(s).$$

After  $n$  repetitions,  $\tilde{V}_n(s) = 0$ , and  $g(t)$  is found by

$$\begin{aligned}g(t) &= a_0 f_0(t) + a_1 f_1(t) + a_2 f_2(t) + \cdots + a_n f_n(t) \\ &= a_0 \left[ 1 \right] e^{-\frac{t}{\tau}} \\ &\quad + a_1 \left[ \frac{1}{\tau} t \right] e^{-\frac{t}{\tau}} \\ &\quad + a_2 \left[ -\frac{2}{\tau} t + \frac{1}{\tau^2} t^2 \right] e^{-\frac{t}{\tau}} \\ &\quad + a_3 \left[ \frac{6}{\tau} t - \frac{6}{\tau^2} t^2 + \frac{1}{\tau^3} t^3 \right] e^{-\frac{t}{\tau}} \\ &\quad \vdots \\ &= \left[ k_0 + k_1 t + k_2 t^2 + \cdots + k_n t^n \right] e^{-\frac{t}{\tau}}\end{aligned}$$

Figure 3-5 shows the effects of increasing the approximation orders of polynomial exponential approximations.

### 3.6.3 Double Exponential

The double exponential assumed waveform shape,

$$(k_1 + k_2 t)e^{-\frac{t}{\tau_a}} + (k_3 + k_4 t)e^{-\frac{t}{\tau_b}} \quad (3.16)$$

is harder to compute, but in many cases gives better results than the polynomial exponential shape with the same approximation order. The fit of the double exponential is better, in particular, if the true waveform exhibits a strong two-time-constant behavior as is seen in some  $RC$  circuits [26]. The double exponential waveform has not d.c. change restriction, since it may



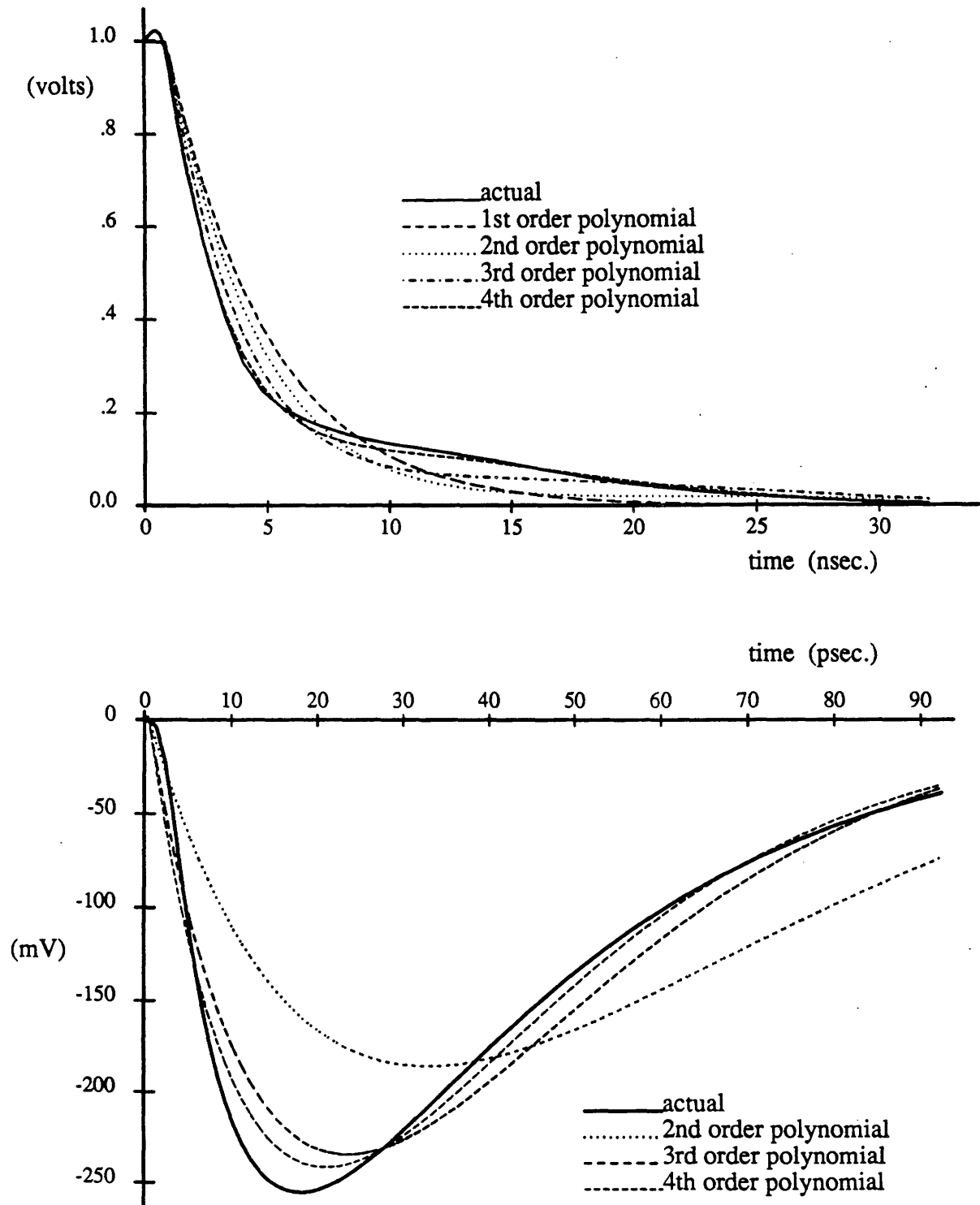


FIGURE 3-5: Polynomial exponential waveform approximations.

or may not be monotonic. For these reasons, it is the preferred waveform shape for most pure  $RC$  circuits.

This assumed waveform shape has two more parameters ( $k_1, k_2, k_3, k_4, \tau_a$  and  $\tau_b$ ) than constraints ( $M_0, M_1, M_2, M_3$ ), so two parameters,  $\tau_a$  and  $\tau_b$ , are estimated and the rest ( $k_1, \dots, k_4$ ) are fitted exactly to the moment constraints, based on the estimates for  $\tau_a$  and  $\tau_b$ .

### Estimating the time-constants

The method for estimating the waveform's time constants is similar to the one for estimating the polynomial exponential time constant (Section 3.6.2) where we assume the waveform is described by any series of exponentials as in Equation (3.14). But, now we need two dominant time-constants,  $\tau_a$  and  $\tau_b$ . Neglecting all but these two time constants, Equation (3.14) becomes

$$M_i \approx \frac{1}{i!} (c_a \tau_a^i + c_b \tau_b^i) \quad (3.17)$$

The second and third order moments are used to approximate  $\tau_a$ , the dominant pole:

$$\tau_a = \frac{M_3/3! - c_b \tau_b^3}{M_2/2! - c_b \tau_b^2}, \quad (3.18)$$

$$c_a = \frac{M_3/3! - c_b \tau_b^3}{\tau_a^3}, \quad (3.19)$$

and the zeroth and first order moments are used to approximate  $\tau_b$ :

$$\tau_b = \frac{M_1 - c_a \tau_a}{M_0 - c_a}, \quad (3.20)$$

$$c_b = \frac{M_1 - c_a \tau_a}{\tau_b}. \quad (3.21)$$

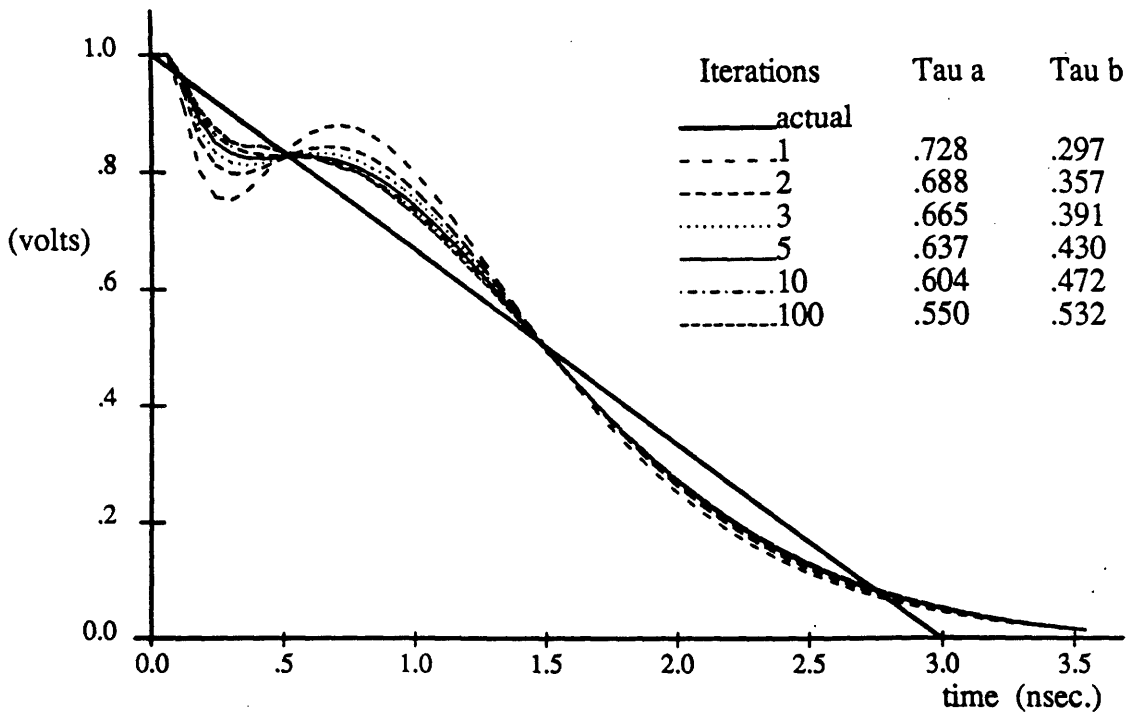
The calculation of each time constant requires a guess for the other, so an iterative process is used, where the above sequence of equations, (3.18) through (3.21), is followed with  $c_b = \tau_b = 0$ , initially. This iterative process attempts to find a solution to Equation (3.17) for the first four moments.

The values for  $\tau_a$  and  $\tau_b$  do not need to be exact! So, while one iteration suffices, a few iterations are better, and more than five only marginally improve results. Figure 3-6 shows these effects. Since the hard constraints are placed in the next step (computing the  $k$ 's), the pole locations can vary by as much as 20% without affecting the resultant waveform substantially.

### Computing the $k$ 's

Now we go back to assuming the double exponential waveform shape of Equation (3.16). The first four moments of the are:

$$\begin{aligned} M_0 &= k_1 && + && k_3 \\ M_1 &= k_1 \tau_a &+ & k_2 \tau_a^2 &+ & k_3 \tau_b &+ & k_4 \tau_b^2 \\ \frac{M_2}{2!} &= k_1 \tau_a^2 &+ & k_2 2\tau_a^3 &+ & k_3 \tau_b^2 &+ & k_4 2\tau_b^3 \\ \frac{M_3}{3!} &= k_1 \tau_a^3 &+ & k_2 3\tau_a^4 &+ & k_3 \tau_b^3 &+ & k_4 3\tau_b^4 \end{aligned}$$



This example demonstrates the iteration affects in finding time constants. The affects are more observable when the fit is harder to achieve, as in this example of fitting exponentials to a straight line. For real circuit-like waveforms, the solution converges very rapidly. Note, also that the waveform shape is rather insensitive to precise pole locations.

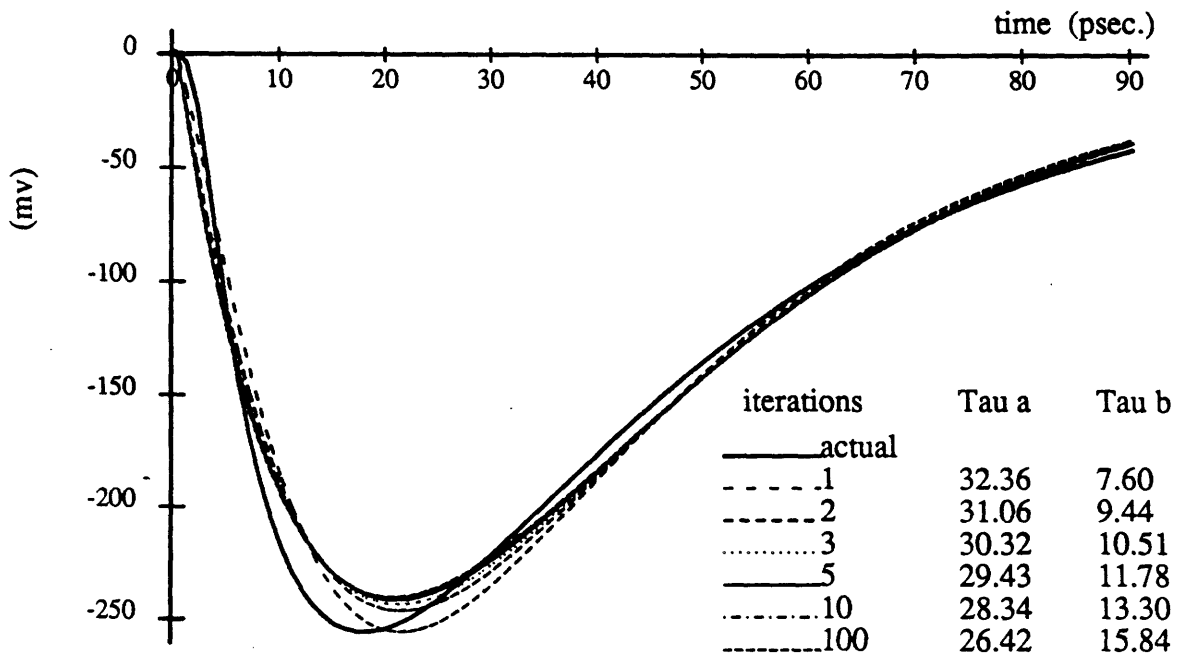


FIGURE 3-6: Effects of iterating to find double exponential poles.

These form a set of linear equations, which is solved for  $k_1$ ,  $k_2$ ,  $k_3$  and  $k_4$ .

### 3.6.4 Choice of $g(t)$

The choice of  $g(t)$  for any given waveform depends on these factors:

- *approximation order.* If the user wants quick but inexact results, a  $g(t)$  with small  $n$  is appropriate, whereas for accurate results, a larger  $n$  is appropriate.

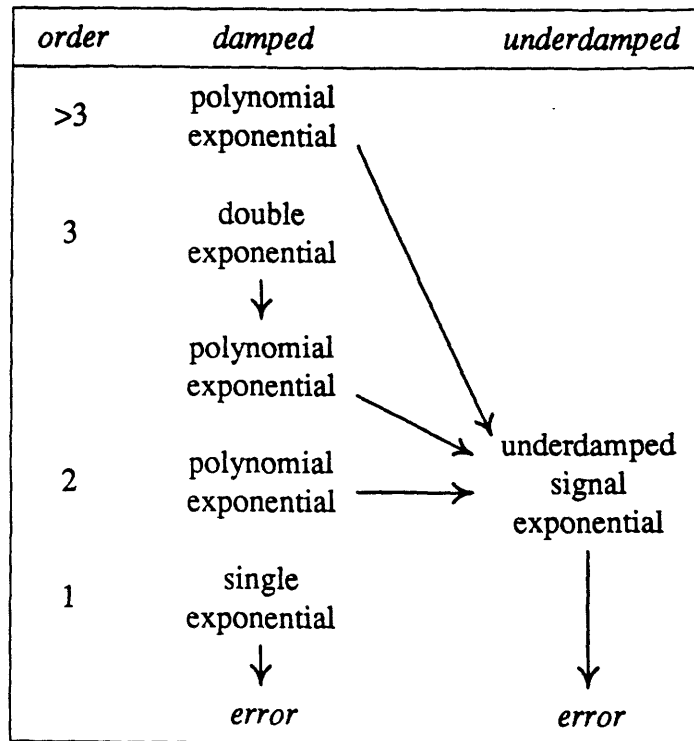
This factor must be decided by the user. An approximation order must be selected for the simulation. All subsequent moment polynomial calculations are made to that order. Different approximation orders can be given to particular circuits if differing degrees of accuracy are desired. Example circuits in this thesis all have approximation orders of three.

- *transition type.* As shown in Table 3-3, some assumed waveform shapes are applicable only to transitions with a non-zero d.c. change and others are applicable only to waveforms with the same initial and final value. Which waveform shape to apply is easily determined by examination of the  $M_0$  moment representation term.
- *circuit type.* An intelligent choice of  $g(t)$  based on the circuit type yields better results. For instance, a CMOS circuit with only  $RC$  interconnection will have no ringing, and its waveforms can be best approximated by real exponential function. A circuit expected to have ringing may be best approximated by complex exponential functions.

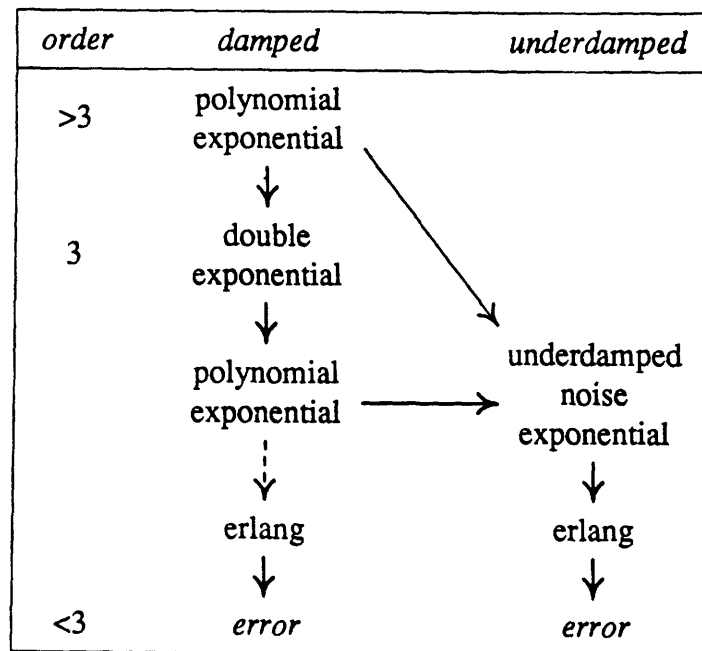
A general procedure has been developed for selecting a  $g(t)$  based on the first two factors. It is also based on observations of how well each of the assumed waveform shapes in Table 3-3 matches to real waveforms in MOS logic and interconnection circuits. Figure 3-7 illustrates the procedure for both transition types. The value for *order* on the left indicates the user-selected approximation order. Initially, the procedure assumes a damped waveform. The chart gives the preferred choice for  $g(t)$ —a third order preferred  $g(t)$  for a transition with a d.c. shift is the double exponential, for instance.

In some cases the initial  $g(t)$  choice fails to give a satisfactory approximation, because the time-constant waveform parameter,  $\tau$ , is negative. This condition indicates an underdamped waveform and requires an alternative assumed waveform shape. The arrows in Figure 3-7 show the path of alternatives to take in these conditions. The paths reflect these observations:

- When an error condition exists for a transition with a d.c. shift, little is gained by trying lower ordered, damped waveforms, and a direct jump to the underdamped waveform is prudent.
- An error condition for a non-d.c.-shifted transition is often handled by either an underdamped noise exponential or an erlang.



With d.c. shift ( $M_0 \neq 0$ )

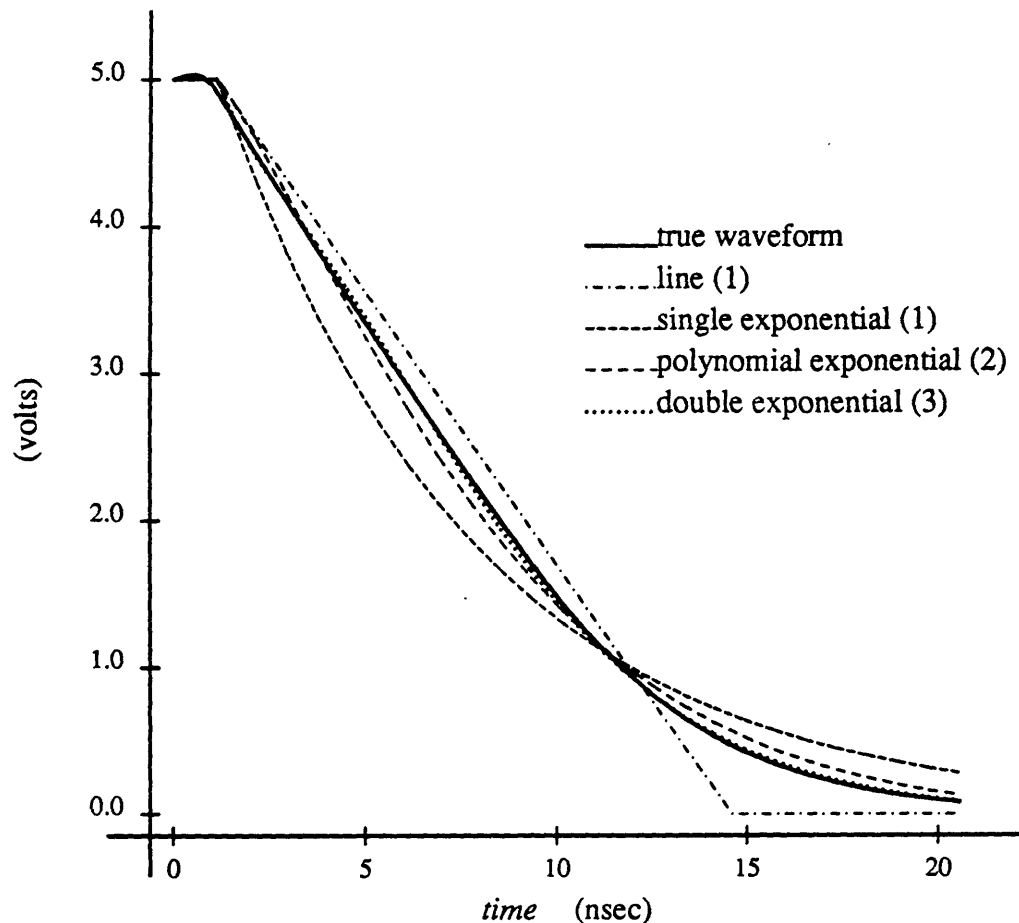


Without d.c. shift ( $M_0 = 0$ )

FIGURE 3-7: Procedure for selecting assumed waveform shape.

When the path of preferred  $g(t)$ 's ends in an error condition, it indicates that insufficient information is present in the moment parameters to reconstruct the waveform with the list given in Table 3-3. In general, this can be corrected by moving to a higher approximation order. Note for instance that a non-d.c.-shifted waveform always results in an error with an approximation order less than three. The error condition was never reached while correctly using the experimental simulator on real circuit waveforms—it was reached only while contriving waveform parameters which are known to be troublesome.

**Example 3.1** *Different waveform approximations for a sample logic transition waveform.*



### 3.7 Discussion

In this chapter we have seen that the moment representation has a link to both the time domain and the frequency domain Laplace transform. The moment representation is actually

a projection from the Laplace transform domain. This allows us to easily translate a number of Laplace transform properties to useful moment representation properties.

The link between the time domain and the moment representation domain is through moment integrals. The time domain to moment representation conversion domain is exact. The moment representation to time domain conversion is more relevant to the simulator operation and is done by heuristic approximations. A waveform shape is assumed, and then waveform parameters are computed such that its moments match the moment representation values.





---

---

## General Linear Network Solutions for the Moment Representation

In this chapter, we examine the first of two techniques for solving linear networks in the moment representation domain. It is capable of finding the moment representation solution to almost any linear, reciprocal circuit containing  $R$ 's,  $L$ 's and  $C$ 's, but is particularly useful in solving (1) circuits with coupled or uncoupled  $RC$  interconnections, (2) circuits with MOS transistor macromodels (described in Chapter 6), and (3) any  $RLC$  circuit with damped responses. The second linear network technique—discussed in Chapter 5—applies to coupled or uncoupled  $LC$  transmission line circuit. While the technique of this chapter is possible for transmission line circuits, the second technique offers a better solution.

The technique presented in this chapter is based on a new, Moment Polynomial Nodal Analysis (MPNA) method for computing the voltage response time moments of a linear circuit. The moment polynomial refers to just the time-moment terms,  $M_0, M_1, \dots$ , of the moment representation. In fact, this chapter deals only with these terms, and assumes a constant  $t_0$  and  $\hat{v}(0)$  between input and output transitions.

In a more limited sense, the problem addressed in this chapter has been the subject of much recent research. Penfield [47], Wyatt [15] and Lin and Mead [16] have shown methods for computing the normalized first moment (Elmore delay) only of  $RC$  trees,  $RC$  meshes and  $RC$  networks, respectively.<sup>1</sup> Horowitz [18] has developed a second order method which computes the first and second moments of  $RC$  trees. The trees in these cases could also contain switching elements, thus being able to model charge sharing circuits [48].

The method presented in this chapter is much more general. It is capable of finding the moment polynomial of the voltage on any node of any circuit containing linear resistors, capacitors, inductors (including coupled inductors) and independent sources, provided the circuit

---

<sup>1</sup>Penfield and Wyatt also compute bounds on the waveform, which is not done here.

satisfies the following minimal set of constraints:

1. The circuit forms a solvable, connected network.

By *solvable* we mean that sources do not impose an impossible constraint on the circuit—as two, series current sources with different values do.

By *connected* we mean that no two nodes are completely isolated from each other. There is always a path of circuit elements between any two nodes.

2. Independent voltage sources form part of a Thevenin equivalent (a voltage source in series with a  $R$ ,  $L$ , or  $C$  circuit element).
3. The *step* response node voltages have no impulses, and settle asymptotically to a final value.

This restriction precludes only some pathological circuits such as a switched current source connected in series with an inductor or a d.c. current source connected in series with a capacitor.

All circuits studied in previous  $RC$  modeling methods are a subset of allowable circuits for the MPNA method. In addition, the MPNA method allows sources to have any waveform shape that is representable by a moment representation. This is untrue of some other  $RC$  circuit methods with moment orders larger than one.

The MPNA method presented in this chapter computes the moment polynomial to any arbitrarily large order,  $p$ , without algorithmic change. Thus, if  $p = 0$ , then only the d.c. responses are computed, if  $p = 1$ , the d.c. responses and Elmore delays are computed, if  $p = 2$ , the d.c. responses, Elmore delays and second moment waveform information are computed, etc. The previous chapter demonstrated that as  $p$  increases, the true waveform can be reconstructed with greater accuracy.

The MPNA method solves a linear network with nodal analysis in the moment representation domain. From Chapter 3 we know that the moment representation domain is a subspace of the Laplace transform domain, but where a moment representation is a truncated Maclaurin series expansion of the Laplace transform. Nodal analysis of the circuit yields a matrix equation, which is then solved using gaussian elimination or  $LU$  decomposition. The matrix equations are set up and solved in the same fashion as in a standard, direct method circuit simulator. The difference, however, between standard simulations and MPNA is that the MPNA matrix elements are polynomials in  $s$ , or moment polynomials.

One note shall be made here on the use of the term *polynomial*. In the last chapter and in general usage, “polynomials” do not have negative ordered terms, but in this chapter negative ordered terms are allowed for moment representation admittance functions. Negative ordered polynomial terms are still not permitted for waveform representations, which in short, translates to circuit restriction 3 above.

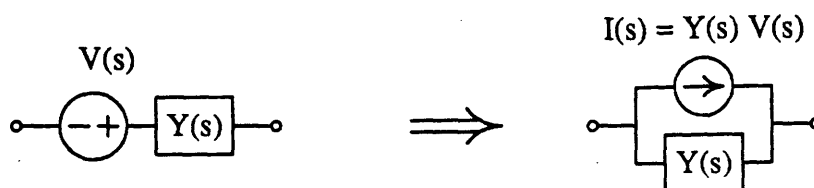


FIGURE 4-1: Thevenin and Norton equivalent circuits.

In comparing and contrasting MPNA methods with direct simulation methods, we make three observations:

- The similarity in formulating circuit equations accounts for the increased circuit flexibility of MPNA methods over previous  $RC$  circuit methods.
- Circuit restriction 2 on Page 70 is necessary since *modified nodal analysis*—common in standard circuit simulators—is not practical with the moment representation method.
- The computational savings with MPNA arises from the fact that only one matrix equation solution is necessary per input signal transition, whereas many are needed for standard circuit simulators—one at each time step.

The next two sections describe how MPNA matrix equations are set up and solved.

## 4.1 Formulating the Moment Polynomial Matrix Equations

The first step in formulating the nodal analysis equations is to convert all voltage sources from their Thevenin equivalent form to their Norton equivalent form (see Figure 4-1). This leaves current sources as the only source elements in the circuit.

Before writing the equations, one node is chosen as the datum (or ground) node. All node voltages are defined relative to the voltage on the datum node. Next, the Kirchoff's current law equations in the Laplace transform domain are written for each circuit node. For a circuit with  $n + 1$  nodes,  $n$  equations are written. (The datum node is ignored.) In matrix form,

$$Y(s)v(s) = i(s) \quad (4.1)$$

where  $Y(s)$  is the  $n \times n$  Laplace transform admittance matrix,  $v(s)$  is the  $n \times 1$  node voltage matrix, and  $i(s)$  is the  $n \times 1$  source current matrix. Element  $v_m(s)$  of  $v(s)$  is the Laplace transform of the node voltage at node  $m$ . Element  $i_m(s)$  of  $i(s)$  is the Laplace transform of the net source current flowing into node  $m$ .

Projecting Equation (4.1) into the moment representation domain, each of these matrices becomes a *polynomial matrix*—a matrix with polynomial elements. Polynomial matrices can

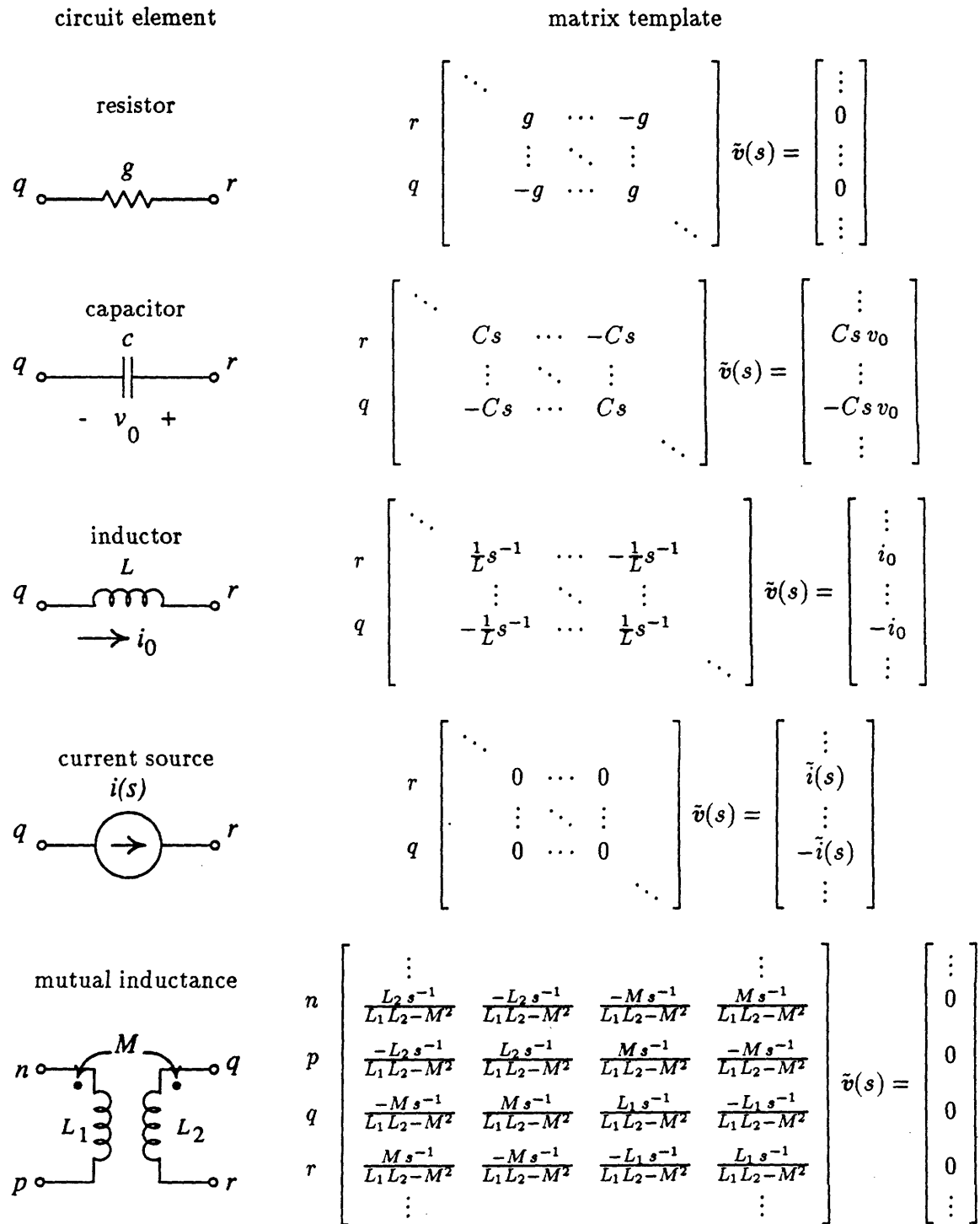


FIGURE 4-2: Circuit Element template patterns.

be split into scalar matrices. For instance, the admittance polynomial matrix can be split into three scalar matrices,

$$\tilde{Y}(s) = \Gamma s^{-1} + G + Cs.$$

The scalar matrices  $\Gamma$ ,  $G$  and  $C$  are the inductive, conductive and capacitive admittance matrices, respectively. ( $\Gamma$  has values of  $\frac{1}{L}$ , hence the inverted L.) An important advantage of the  $\tilde{Y}(s)$  matrix is that it can be inverted, even though neither  $\Gamma$ ,  $G$  nor  $C$  is invertible. More is said about this later.

The procedure for constructing  $\tilde{Y}(s)$  from knowledge of the circuit branch elements is identical to the procedure used in SPICE and described in [49]. It goes as follows. Initially, empty matrices are created for  $\tilde{Y}(s) = \mathbf{0}$  and  $\tilde{i}(s) = \mathbf{0}$ . Then, one by one the contributions of each branch element are added to the matrices. Each branch element type has a *template pattern* for its contributions into the matrices. These are illustrated in Figure 4-2 for discrete element types.

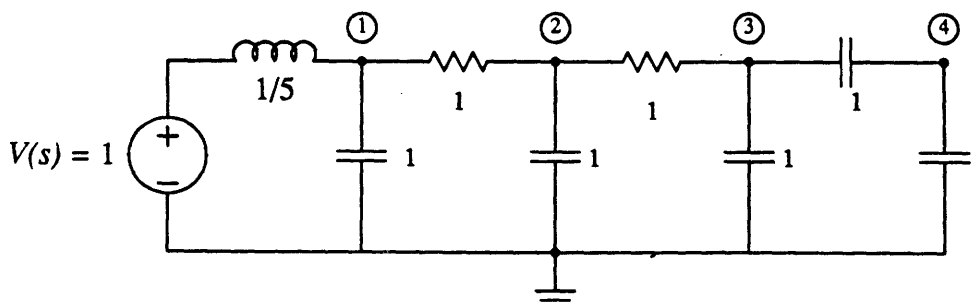
Notice that both inductors and capacitors may have non-zero initial conditions. Taking one of the element types—the inductor, for instance—we see that the inductor moment representation relation (equal to the Laplace transform relation),

$$i_r = -i_q = \frac{1}{Ls}(v_r - v_q) - i_0$$

is reflected in the template pattern. If one of the element nodes is the datum node, the matrix entries for that node are not included in the matrices, since  $v_{datum} = 0$ .

When all of the branch element contributions have been added to the matrices,  $\tilde{Y}(s)$  and  $\tilde{i}(s)$  are the desired result.

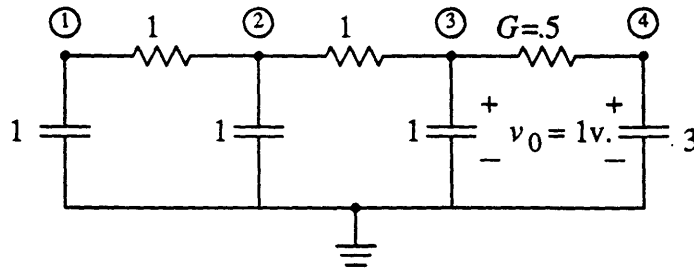
**Example 4.1** Write the Laplace transform nodal analysis matrix equations for the circuit below.



The Thevenin equivalent is converted into a Norton equivalent with  $\tilde{I}_{equiv}(s) = \tilde{V}(s)/Ls = 5s^{-1}$ .

$$\begin{bmatrix} 5s^{-1} & -1 & 0 & 0 \\ -1 & 2+s & -1 & 0 \\ 0 & -1 & 1+2s & -s \\ 0 & 0 & -s & 2s \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} 5s^{-1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (4.2)$$

**Example 4.2** Write the Laplace transform nodal analysis matrix equations for the circuit below.



$$\begin{bmatrix} 1+s & -1 & 0 & 0 \\ -1 & 2+s & -1 & 0 \\ 0 & -1 & 1.5+s & -0.5 \\ 0 & 0 & -0.5 & 0.5+3s \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ s \\ 3s \end{bmatrix} \quad (4.3)$$

#### 4.1.1 Distributed Circuit Elements

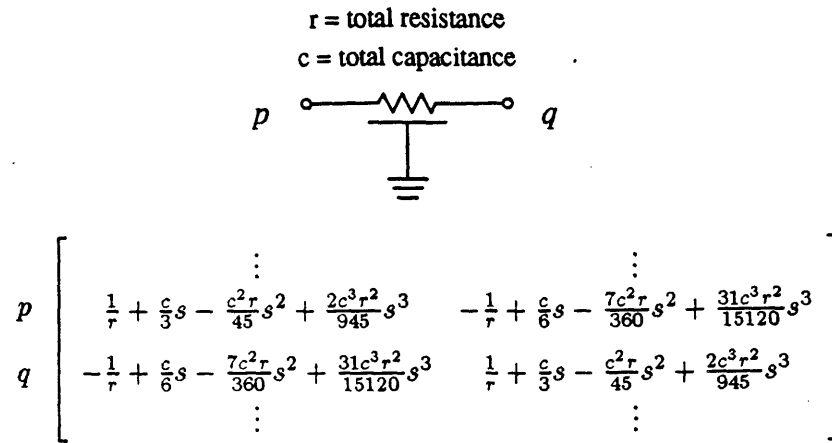
It is also possible to include distributed circuit elements in the MPNA equations, provided the elements can be described by a moment polynomial admittance matrix. This includes a large number of elements that cannot be included directly into time-domain circuit simulators like SPICE. For instance, the distributed  $RC$  element of Figure 4-3 is described by Laplace transform equation, [50]

$$\begin{bmatrix} I_1(s) \\ I_2(s) \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{cs}{r}} \coth(\sqrt{rcs}) & -\sqrt{\frac{cs}{r}} \operatorname{csch}(\sqrt{rcs}) \\ -\sqrt{\frac{cs}{r}} \operatorname{csch}(\sqrt{rcs}) & \sqrt{\frac{cs}{r}} \coth(\sqrt{rcs}) \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_2(s) \end{bmatrix}$$

Taking the Maclaurin Series gives the moment representation matrix entries shown in Figure 4-3.

#### 4.1.2 Moment Polynomial Matrix Properties

At this point we want to consider how to solve the MPNA equations. But, first, we must take a little detour to examine some matrix properties. All of these properties have been developed

FIGURE 4-3: Distributed  $RC$  element template

in linear algebra texts (such as [51]), so only the interesting results are presented here. First we start with a definition.

A *field* is a numbering system in which the operations of addition, subtraction, multiplication and division (except by 0) are performed without restriction. More precisely:

A field is a set of numbers,  $\mathcal{F}$ , and dyadic operations  $+$  and  $\cdot$ , such that if  $a$ ,  $b$  and  $c$  are any elements of  $\mathcal{F}$ , then:

1. The  $+$  and  $\cdot$  operations are commutative and associative,
2.  $a + b$  and  $a \cdot b$  exist in  $\mathcal{F}$ ,
3. there exists a unique  $0 \in \mathcal{F}$  such that  $0 + a = a + 0 = 0$ ,
4. there exists a unique  $(-a) \in \mathcal{F}$  such that  $a + (-a) = (-a) + a = 0$ ,
5.  $a \cdot (b + c) = a \cdot b + a \cdot c$  and  $(b + c) \cdot a = b \cdot a + c \cdot a$ ,
6. there exists a unique  $1$  such that  $1 \neq 0$  and  $a \cdot 1 = 1 \cdot a = a$ ,
7. if  $a \neq 0$  then there exists a unique  $(a^{-1}) \in \mathcal{F}$  such that  $a \cdot a^{-1} = a^{-1} \cdot a = 1$ .

Obviously, the sets of real numbers and complex numbers form fields. So does the set of infinite degree moment polynomials, given that we allow negative-ordered polynomial terms. Appendix A shows details of moment polynomial operations. The set of moment polynomials truncated to a fixed order does not form a field, since one can easily violate condition 2.

The notation  $\mathcal{F}_{m \times n}$  is used for a matrix with elements from  $\mathcal{F}$  arranged in  $m$  rows and  $n$  columns. For these matrices, we shall define addition, multiplication, inverses and determinants in the standard fashion in which we are familiar with for real number matrices. The following theorems are valid for matrices of any field.

- If  $A \in \mathcal{F}_{n \times n}$  and  $b$  and  $x \in \mathcal{F}_{n \times 1}$ , then the solution to  $Ax = b$  is  $x = A^{-1}b$ .

- The solution to  $A\mathbf{x} = \mathbf{b}$  is unique if the following equivalent statements are true:
  - $A$  has an inverse,
  - $\det(A) \neq 0$ ,
  - $A\mathbf{x} = \mathbf{0}$  has only the trivial solution.

Three elementary row operations are defined for  $\mathcal{F}_{m \times n}$ :

**Row operation 1:** Two rows are interchanged.

**Row operation 2:** A row is multiplied by  $c \in \mathcal{F}$  where  $c \neq 0$ .

**Row operation 3:** A row is replaced by itself plus  $k$  times another row, where  $k \in \mathcal{F}$ .

One last useful theorem involving these row operations on matrices of  $\mathcal{F}$  is:

- Any elementary row operation on the  $n \times n + 1$  matrix  $[A|\mathbf{b}]$  will not affect the solution of  $A\mathbf{x} = \mathbf{b}$ .

## 4.2 Solution of the Moment Polynomial Matrix Equation

The motivation of the preceding section was to show that matrix solution techniques using row reduction are not limited just to matrices of real numbers. The same row reduction techniques can solve any matrix with elements from a field number set—including the infinite-order moment polynomial number set. We are now prepared to look at methods for solving Equation (4.1).

### 4.2.1 Gaussian Elimination

Gaussian elimination is an efficient row reduction algorithm for solving a matrix equation like  $\mathbf{Y} \cdot \mathbf{v} = \mathbf{i}$ . It is a well-known algorithm discussed in any linear algebra text on in [49]. It will be presented here only with an example. A dual purpose of the example is to demonstrate the applicability of matrix algorithms to the infinite-order moment polynomial number field.

**Example 4.3** Solve the matrix Equation (4.2) of Example 4.1 using gaussian elimination.

**Step 1—forward elimination:** Reduce  $\tilde{\mathbf{Y}}(s)$  to upper triangular form by successively zeroing out the lower triangular terms by row operations

$$\tilde{\mathbf{Y}}'_{\text{row}_k}(s) \leftarrow \tilde{\mathbf{Y}}_{\text{row}_k}(s) - \left( \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \right) \tilde{\mathbf{Y}}_{\text{row}_j}(s), \quad \text{for } k > j$$



or

$$\tilde{y}'_{km}(s) \leftarrow \tilde{y}_{km}(s) - \left( \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \right) \tilde{y}_{jm}(s), \quad \text{for } k, m > j \quad (4.4)$$

and

$$\tilde{i}'_k(s) \leftarrow \tilde{i}_k(s) - \left( \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \right) \tilde{i}_j(s), \quad \text{for } k > j. \quad (4.5)$$

To zero the first column's lower triangular elements,

$$\text{row}_2 \leftarrow \text{row}_2 - \left( \frac{-1}{5s^{-1} + 1} \right) \text{row}_1 = \text{row}_2 - \left( -\frac{1}{5}s + \frac{1}{25}s^2 - \frac{1}{125}s^3 \dots \right) \text{row}_1$$

giving

$$\begin{bmatrix} 5s^{-1} & -1 & 0 & 0 \\ 0 & 2 + \frac{4}{5}s + \frac{1}{25}s^2 \dots & -1 & 0 \\ 0 & -1 & 1 + 2s & -s \\ 0 & 0 & -s & 2s \end{bmatrix} \tilde{v}(s) = \begin{bmatrix} 5s^{-1} \\ 1 - \frac{1}{5}s + \frac{1}{25}s^2 \dots \\ 0 \\ 0 \end{bmatrix}$$

Row 3 and 4 were initially zeroed and thus are left unchanged. To zero the second column:

$$\text{row}_3 \leftarrow \text{row}_3 - \left( \frac{-1}{2 + \frac{4}{5}s + \frac{1}{25}s^2 \dots} \right) \text{row}_2 = \text{row}_3 - \left( -\frac{1}{2} + \frac{1}{5}s - \frac{7}{100}s^2 \dots \right) \text{row}_2$$

giving

$$\begin{bmatrix} 5s^{-1} & -1 & 0 & 0 \\ 0 & 2 + \frac{4}{5}s + \frac{1}{25}s^2 \dots & -1 & 0 \\ 0 & 0 & \frac{1}{2} + \frac{11}{5}s - \frac{7}{100}s^2 \dots & -s \\ 0 & 0 & -s & 2s \end{bmatrix} \tilde{v}(s) = \begin{bmatrix} 5s^{-1} \\ 1 - \frac{1}{5}s + \frac{1}{25}s^2 \dots \\ \frac{1}{2} - \frac{3}{10}s + \frac{13}{100}s^2 \dots \\ 0 \end{bmatrix}$$

Lastly, column 3 is zeroed in row 4 by

$$\text{row}_4 \leftarrow \text{row}_4 - \left( \frac{-s}{\frac{1}{2} + \frac{11}{5}s - \frac{7}{100}s^2 \dots} \right) \text{row}_3 = \text{row}_4 - \left( -2s + \frac{44}{5}s^2 \dots \right) \text{row}_3$$

giving

$$\begin{bmatrix} 5s^{-1} & -1 & 0 & 0 \\ 0 & 2 + \frac{4}{5}s + \frac{1}{25}s^2 \dots & -1 & 0 \\ 0 & 0 & \frac{1}{2} + \frac{11}{5}s - \frac{7}{100}s^2 \dots & -s \\ 0 & 0 & 0 & 2s - 2s^2 + \frac{22}{25}s^3 \dots \end{bmatrix} \tilde{v}(s) = \begin{bmatrix} 5s^{-1} \\ 1 - \frac{1}{5}s + \frac{1}{25}s^2 \dots \\ \frac{1}{2} - \frac{3}{10}s + \frac{13}{100}s^2 \dots \\ s - 5s^2 + \frac{112}{5}s^3 \dots \end{bmatrix} \quad (4.6)$$

**Step 2—backward elimination:** Solve for  $\tilde{v}(s)$  terms starting with the last and working upward, with

$$\tilde{v}_k(s) = \frac{\tilde{i}_k(s) - \sum_{m=k+1}^n \tilde{v}_m(s) \tilde{y}_{km}(s)}{\tilde{y}_{kk}(s)}. \quad (4.7)$$

Thus,  $v_4$  is calculated first:

$$v_4 = \frac{s - 5s^2 + \frac{112}{5}s^3 + \dots}{2s - 2s^2 + \frac{144}{5}s^3 + \dots} = \frac{1}{2} - 2s + 7s^2 + \dots$$

Then,

$$v_3 = \frac{(\frac{1}{2} - \frac{3}{10}s + \frac{13}{100}s^2 + \dots) - (-s)(\frac{1}{2} - 2s + 7s^2 + \dots)}{\frac{1}{2} + \frac{11}{5}s - \frac{7}{100}s^2 + \dots} = 1 - 4s + 14s^2 \dots$$

and so forth, giving a final solution

$$\tilde{v}(s) = \begin{bmatrix} 1 - 2s^2 + \dots \\ 1 - \frac{5}{2}s + 8s^2 + \dots \\ 1 - 4s + 14s^2 + \dots \\ \frac{1}{2} - 2s + 7s^2 + \dots \end{bmatrix}$$

### 4.2.2 LU Decomposition

In many standard circuit simulation programs, (like SPICE and ASTAP) *LU* decomposition is implemented for matrix solutions [49]. *LU* decomposition is a slight modification of gaussian elimination. It progresses through the same sequence of elementary row operations, the difference being that intermediate row multipliers are preserved during the forward elimination process. Rather than placing unnecessary zeros in the lower triangular section of  $\tilde{Y}(s)$ , the row multiplier,  $(\frac{y_{kl}}{y_{jj}})$  in Equation (4.4), is placed in  $y_{kj}$ . The net effect is to generate the sum of a lower triangular matrix (without the diagonal 1's) and an upper triangular matrix

$$[L \setminus U] = L - I + U \quad (4.8)$$

where

$$A = LU.$$

*LU* decomposition is advantageous if the same linear circuit ( $\tilde{Y}(s)$ ) is solved for many different source currents and initial conditions ( $\tilde{i}(s)$ ). The circuit matrix,  $\tilde{Y}(s)$ , is *LU* decomposed once, thereafter containing all of the information needed to solve  $\tilde{Y}(s)\tilde{v}(s) = \tilde{i}(s)$  for any number of  $\tilde{i}(s)$ 's with Equations (4.5) and (4.7).

In considering *LU* decomposition for the infinite polynomial number field, the relevant point of this section was that *LU* decomposition of the matrix is subjected to the same elementary row operations. Thus, *LU* decomposition is perfectly suitable for infinite-ordered moment polynomials.

**Example 4.4** *What is the  $LU$  decomposed matrix for  $\tilde{Y}(s)$  in Equation (4.2) of Example 4.1?*

By putting the row multipliers found during forward elimination in Example 4.3 into the appropriate locations of the matrix in Equation (4.6), we have

$$\begin{bmatrix} 5s^{-1} & & -1 & & 0 & & 0 \\ -\frac{1}{5}s + \frac{1}{25}s^2 \dots & 2 + \frac{4}{5}s + \frac{1}{25}s^2 \dots & & & -1 & & 0 \\ 0 & -\frac{1}{2} + \frac{1}{5}s - \frac{7}{100}s^2 \dots & \frac{1}{2} + \frac{11}{5}s - \frac{7}{100}s^2 \dots & & & & -s \\ 0 & 0 & -2s + \frac{44}{5}s^2 \dots & 2s - 2s^2 + \frac{22}{25}s^3 \dots & & & \end{bmatrix}$$

### 4.2.3 Finite Truncation in Moment Polynomial Gaussian Elimination

The previous section demonstrates that gaussian elimination and  $LU$  decomposition can solve MPNA equations if no polynomial truncation is performed. Obviously, this is impossible in practice, for just about any moment polynomial division operation potentially yields an infinite degree polynomial. And yet, it is not sufficient to truncate all polynomials at the same order of  $s$ . We can easily see this by considering the zero moment solution (d.c. solution) to Example 4.2. If all polynomials are truncated past the  $s^0$  terms, this is equivalent to solving the resistive admittance matrix,  $G$ , only. Clearly, this will not do, since the capacitances play a significant role in determining the d.c. response.

This section shows that if we chose the truncation order wisely during gaussian elimination, the final solution for  $\tilde{v}(s)$  will have moment polynomials guaranteed to be accurate to a prespecified order, and computed with a minimal number of floating point operations.

To start, a few definitions are made. All *orders* are moment polynomial orders of  $s$ .

- The *minimum order* of a polynomial, denoted by  $[\tilde{a}(s)]$  for polynomial  $\tilde{a}(s)$ , is the order for the lowest ordered, non-zero term of  $\tilde{a}(s)$ . If  $\tilde{a}(s) = 0$ , the minimum order is undefined.
- *Result order*,  $p$ , is the highest desired moment order of the terms in  $\tilde{v}(s)$ .
- An *operation truncation order*,  $T[\tilde{a}(s) \text{ op } \tilde{b}(s)]$ , is the highest necessary order of the calculated result of  $\tilde{a}(s) \text{ op } \tilde{b}(s)$ . *Truncation rules* are given in Appendix A for truncation orders for addition, subtraction, multiplication and division of moment polynomials.
- A *circuit element order* is the minimum Laplace transform order of the circuit element's admittance. The circuit element order of an inductor is  $-1$ , of a resistor,  $0$ , and of a capacitor,  $+1$ . It is also defined for a connection of elements as the minimum moment polynomial order of the combined admittance function.

- The *node order* of node  $j$ ,  $\mathcal{B}_j$ , is the minimum over all paths between node  $j$  and ground of the maximum circuit element order along the path. In forming the paths, independent sources are zeroed, i.e., voltage sources are shorted and current sources are opened.
- 

**Example 4.5** *What are the node orders for the nodes in the circuit of Example 4.1?*

$$\mathcal{B}_1 = -1, \mathcal{B}_2 = \mathcal{B}_3 = 0, \mathcal{B}_4 = 1$$

**Example 4.6** *What are the node orders for the nodes in the circuit of Example 4.2?*

$$\mathcal{B}_1 = \mathcal{B}_2 = \mathcal{B}_3 = \mathcal{B}_4 = 1$$

---

The truncation orders for Gaussian Elimination are based on a circuit's node orders. Basically, when calculating any new term of row  $j$  of  $\tilde{\mathbf{Y}}(s)$  or  $\tilde{\mathbf{i}}(s)$ , it must be calculated to order  $p$ , the desired result order, plus the node order of node  $j$ . By definition of  $p$ , any new term of  $\tilde{\mathbf{v}}(s)$  must be calculated to order  $p$ .

The following important theorem states this more precisely—in terms of the gaussian elimination equations:

---

**Theorem 4.1** To guarantee exact  $\tilde{v}(s)$  moment terms to order  $p$ , it is sufficient to calculate the intermediate results during gaussian elimination to these orders:

- during forward elimination with equations:

$$\tilde{y}'_{km}(s) = \tilde{y}_{km}(s) - \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \tilde{y}_{jm}(s)$$

and

$$\tilde{i}'_k(s) = \tilde{i}_k(s) - \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \tilde{i}_j(s)$$

the truncation order for

$$\begin{cases} \text{subtraction} & = p + \mathcal{B}_k \\ \text{multiplication} & = p + \mathcal{B}_k \\ \text{division} & = p + \mathcal{B}_k - \lfloor y_{kk} \rfloor \end{cases}$$

- during backward elimination with equation:

$$\tilde{v}_k(s) = \frac{\tilde{i}_k(s) - \sum_{m=k+1}^n \tilde{v}_m(s) \tilde{y}_{km}(s)}{\tilde{y}_{kk}(s)}$$

the truncation order for:

$$\begin{cases} \text{division} & = p \\ \text{summation and subtraction} & = p + \mathcal{B}_k \\ \text{multiplication} & = p + \mathcal{B}_k \end{cases}$$

The proof for this is non-trivial, and is included in Appendix B. It is based on the properties of admittance matrices of reciprocal networks.

One can start to see a justification for this by understanding, for instance, what information is needed for computing just the d.c. response to a circuit ( $p = 0$ ). If a node is resistive ( $\mathcal{B}_j = 0$ ), only the inductance and resistance information is needed ( $s^{-1}$  and  $s^0$  terms), not capacitance. Thus, terms with order greater than 0 are not needed for its nodal analysis equation. If a node is capacitive ( $\mathcal{B}_j = 1$ ) then the d.c. response is determined by a ratio of capacitances; thus, a truncation order of 1.

In algorithmic form, the gaussian elimination solution for moment polynomials looks just like a gaussian elimination algorithm for real number matrices, with two differences. First, a few procedure calls are added to set a global truncation order variable. In the algorithms below, these statements are marked with a " $\Leftarrow$ ". Second, all arithmetic operations on matrix elements are polynomial operations. These are distinguished from scalar operations with a box.

Procedure LU\_DECOMPOSE converts the  $\tilde{Y}(s)$  matrix into an  $LU$  factorization as described in Equation (4.8). Procedure SOLVE\_LINEAR solves  $\tilde{Y}(s)\tilde{v}(s) = \tilde{i}(s)$  for  $\tilde{v}(s)$  from  $\tilde{i}(s)$  and the  $LU$  decomposition of  $\tilde{Y}(s)$ . The *node\_order* parameter to both routines is an integer array,

where the  $j^{\text{th}}$  element is  $B_j$ . The algorithm for computing *node\_order* is the topic of the next section.

---

**Algorithm 4.1**

pvector = array [polynomial]

pmatrix = array [array [polynomial]]

**procedure** LU\_DECOMPOSE (*Y*: pmatrix, *n*: integer, *node\_order*: array[integer],  
*p*: integer)

returns (pmatrix) begin

{ *Gaussian elimination of Y* }

for *j* from 1 to *n* - 1 do begin

*diagonal*: polynomial := *Y*[*j*, *j*];

  if IS\_ZERO\_POLYNOMIAL(*diagonal*) then error ("matrix is singular");

  for *k* from *j* + 1 to *n* do begin

    SET\_TRUNCATION\_ORDER(*p* + *node\_order*[*k*] -  
                           POLYNOMIAL\_LOW\_ORDER(*diagonal*)); ←

*mult*: polynomial := *Y*[*k*, *j*]  $\boxed{/}$  *diagonal*;

*Y*[*k*, *j*] := *mult*;

    if not(IS\_ZERO\_POLYNOMIAL(*mult*)) then begin

      SET\_TRUNCATION\_ORDER(*p* + *node\_order*[*j*]); ←

      for *m* from *j* + 1 to *n* do begin

*Y*[*k*, *m*] := *Y*[*k*, *m*]  $\boxed{-}$  *mult*  $\boxed{*}$  *Y*[*j*, *m*];

      end

    end

  end

end

if IS\_ZERO\_POLYNOMIAL(*Y*[*n*, *n*]) then error ("matrix is singular");

return (*Y*);

end

---

**Algorithm 4.2**

{ *solve a system of equations from the LU decomposed array and vector i.* }

**procedure** SOLVE\_LINEAR (*Y*: pmatrix, *i*: pvector, *n*: integer,

*node\_order*: array[integer], *p*: integer)

returns (pvector) begin

{ *create v* }

*v*: pmatrix := create\_pvector(*n*);

{ *account for the forward elimination steps on i;*

for *j* from 2 to *n* do begin

```

SET_TRUNCATION_ORDER( $p + \text{node\_order}[j]$ );  $\Leftarrow$ 
sum: polynomial := ZERO_POLYNOMIAL();
for  $k$  from 1 to  $j - 1$  do begin
    sum := sum  $\boxed{+}$   $Y[j, k]$   $\boxed{*}$   $v[k]$ ;
end
 $v[j] := i[j]$   $\boxed{-}$  sum;
end

{ back substitution }
for  $j$  from  $n$  to 1 by  $-1$  do begin
    SET_TRUNCATION_ORDER( $p + \text{node\_order}[j]$ );  $\Leftarrow$ 
    sum: polynomial := ZERO_POLYNOMIAL();
    for  $k$  from  $j + 1$  to  $n$  do begin
        sum := sum  $\boxed{+}$   $Y[j, k]$   $\boxed{*}$   $v[k]$ ;
    end
    sum :=  $v[j]$   $\boxed{-}$  sum;
    SET_TRUNCATION_ORDER( $p$ );  $\Leftarrow$ 
     $v[j] := \text{sum}$   $\boxed{/}$   $Y[j, j]$ ;
end

return ( $v$ );
end

```

---

#### 4.2.4 Computing Node Orders

The node order of each node is computed prior to solution with gaussian elimination or *LU* decomposition, as it is needed in setting the truncation orders. Conceptually, node orders of a circuit are found by first constructing an undirected graph. There is a vertex (or node) in the graph which corresponds to each node of the circuit. Also, an edge is placed in the graph between a pair of vertices corresponding to each circuit element. An edge is labeled with a “-1”, “0” or “+1” depending on its corresponding circuit element’s order.

Node orders are assigned by tracing, first, all nodes connected to the ground node through -1 edges. These nodes are marked with node order “-1”. Then all previously unmarked nodes connected to ground through any path of edges labeled 0 or less are traced. These nodes are marked “0”. Lastly, all previously unmarked nodes connected to ground through any edge are marked with node order “+1”. At this point, any unmarked node represents a floating node, and indicates an unsolvable network.

The node tracing algorithm is done with a modification of the standard depth-first search algorithm described in [52]. It is expressed algorithmically below, where  $\mathcal{G}$  represents the undirected graph of the circuit.

---

**Algorithm 4.3**

**procedure** COMPUTE\_NODE\_ORDERS ( $\mathcal{G}$ : graph) returns (array[integer]) **begin**

```

node_order: array[integer] := CREATE_ARRAY(n);
for each vertex  $v$  in graph  $\mathcal{G}$  do node_order[v] := "unknown";

SEARCH( $\mathcal{G}$ , node_order, "ground_vertex", -1); { inductive nodes }
SEARCH( $\mathcal{G}$ , node_order, "ground_vertex", 0); { resistive nodes }
SEARCH( $\mathcal{G}$ , node_order, "ground_vertex", 1); { capacitive nodes }

return (node_order);
end

```

---

**Algorithm 4.4**

**procedure** SEARCH ( $\mathcal{G}$ : graph, node\_order: array[integer],  
 $v$ : vertex, order: integer) **begin**

```

node_order[v] := order;
for each vertex  $w$  through edge  $x$  adjacent to  $v$  in graph  $\mathcal{G}$  do
  if (node_order[w] = "unmarked") and (LABEL[x] ≤ order)
    then SEARCH( $\mathcal{G}$ , node_order, w, order);
end
end

```

---

The undirected graph is constructed concurrently with the admittance matrix. As each element's template is added to  $\tilde{Y}(s)$ , a new edge is added to  $\mathcal{G}$ . While almost all of the information in  $\mathcal{G}$  is contained in  $\tilde{Y}(s)$  (all but the ground connections), a separate linked list structure is maintained for  $\mathcal{G}$  to minimize the node order computation time at the expense of some memory. The time required to compute node orders is insignificant in comparison to gaussian elimination time.

**4.2.5 Pivoting for Accuracy**

To avoid numerical roundoff problems found in stiff simulation problems, matrix solutions usually employ a pivoting algorithm where rows and/or columns are interchanged during gaussian elimination. The need for this is diminished with nodal analysis, since the *pivot*, or diagonal term is always the dominant element of any row or column.

There are cases, however, where finite precision arithmetic can affect the solution's accuracy. For instance, the solution to the circuit of Figure 4-4, described by

$$\begin{bmatrix} 1000001 + s & -1000000 \\ -1000000 & -1000000 + s \end{bmatrix} \tilde{v}(s) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



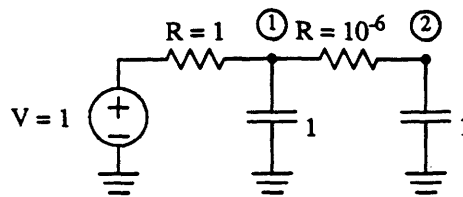


FIGURE 4-4: Circuit requiring node pivoting.

yields a solution,

$$\tilde{v}_1(s) = \tilde{v}_2(s) = 0.88889 - 1.5802 s + 2.8093 s^2 - \dots$$

with 22-bit mantissa arithmetic precision. Full pivoting or partial pivoting strategies cannot be used in the MPNA solution algorithms, since this can disrupt the diagonal-term minimum polynomial orders which are needed for Theorem 4.1 to operate correctly.

Instead, diagonal pivoting is used where both rows and their corresponding columns are swapped. This equates to simply renumbering nodes. With diagonal pivoting, during any stage of forward elimination the best choice for the next pivot is  $\tilde{y}_{jj}(s)$  with the maximum  $[\tilde{y}_{jj}(s)] \equiv b$  or when more than one node has the same  $b$ , to choose the node with minimum value of  $y_{jj,b}$ . Thus, in the above example, the second node is eliminated first, and a true solution of

$$\tilde{v}_1(s) = \tilde{v}_2(s) = 1 - 2s + 4s^2 - \dots$$

is computed.

#### 4.2.6 Pivoting for Sparsity

Matrix pivoting for purposes of achieving sparsity in the  $LU$  decomposed matrix is hindered for the same reasons as pivoting for accuracy. Once again, only rows and columns can be interchanged together. In some instances, it may be advantageous to move a heavily connected node to a lower position in a matrix. It may also be advantageous to number nodes of a ladder type network in consecutive fashion, since these form a band-matrix, where only the diagonal, and terms one away from the diagonal are non-zero.

### 4.3 Examples

On the following pages are several examples of results obtained from the algorithms described in this chapter. All are representative of circuit *topologies* which model VLSI circuits or interconnections. Namely, the topologies match models for

- connecting  $RC$  trees,
- charge sharing circuits,
- lumped  $RC$  trees,

- distributed *RC* trees,
- discrete coupled *RC* lines, and
- discrete coupled *RLC* lines (damped).

Each of the following examples shows a plot of the simulation results of SPICE (in solid lines), and the moment polynomial method calculated to order 3 and with a double exponential inverse (in dotted lines).

#### 4.4 Computational Requirements

In this section, the computational requirements for the MPNA algorithms are compared against the computational requirements for direct methods, and where applicable, against the computational needs of the *RC* tree algorithms of Penfield [47] and Horowitz [18].

Chapter 7 shows a method for moving linear network solutions out of the simulation loop and into the preprocessor step. So, this computation time is not a major concern. This discussion is included here for completeness.

The quantitative value for computation in this section is based on the number of floating point operations needed to compute one waveform or closed-form waveform expression. This figure of merit is chosen over CPU time for two reasons, (1) to first order, it compares the algorithms more directly, rather than the efficiency of compilers, and (2) some of the simulation times are extremely small and difficult to measure. To generate a single number for *equivalent computations*, each type of floating point operation is scaled by approximate relative computation times, i.e., the number of equivalent computations is

$$\begin{aligned}
 & (\# \text{ additions}) \\
 & + (\# \text{ subtractions}) \\
 & + 3 (\# \text{ multiplications}) \\
 & + 4 (\# \text{ divisions}) \\
 & + 10 (\# \text{ square roots})
 \end{aligned}$$

Table 4-1 shows the number of equivalent computations for each circuit. The three columns given for MPNA are computed with different result orders. Direct method estimates are computed as follows. The number of floating point operations for one matrix solution is determined using the same circuit matrix, the same set of matrix operations, and the same pivoting for both MPNA and direct methods. First order numerical integration is assumed. The number of time-steps is estimated by running SPICE simulations where the automatic time-step control keeps node voltages within 5-10 % accuracy at all times. These accuracies may conceivably be worse than the moment polynomial method. SPICE waveforms shown in the preceding examples used several times more time-steps than this number to give an accurate number.

From Table 4-1 we see that the special *RC* methods are more efficient computationally, but these are also very limited in ability. The *RC* tree method for instance applies to only

Example 4.7

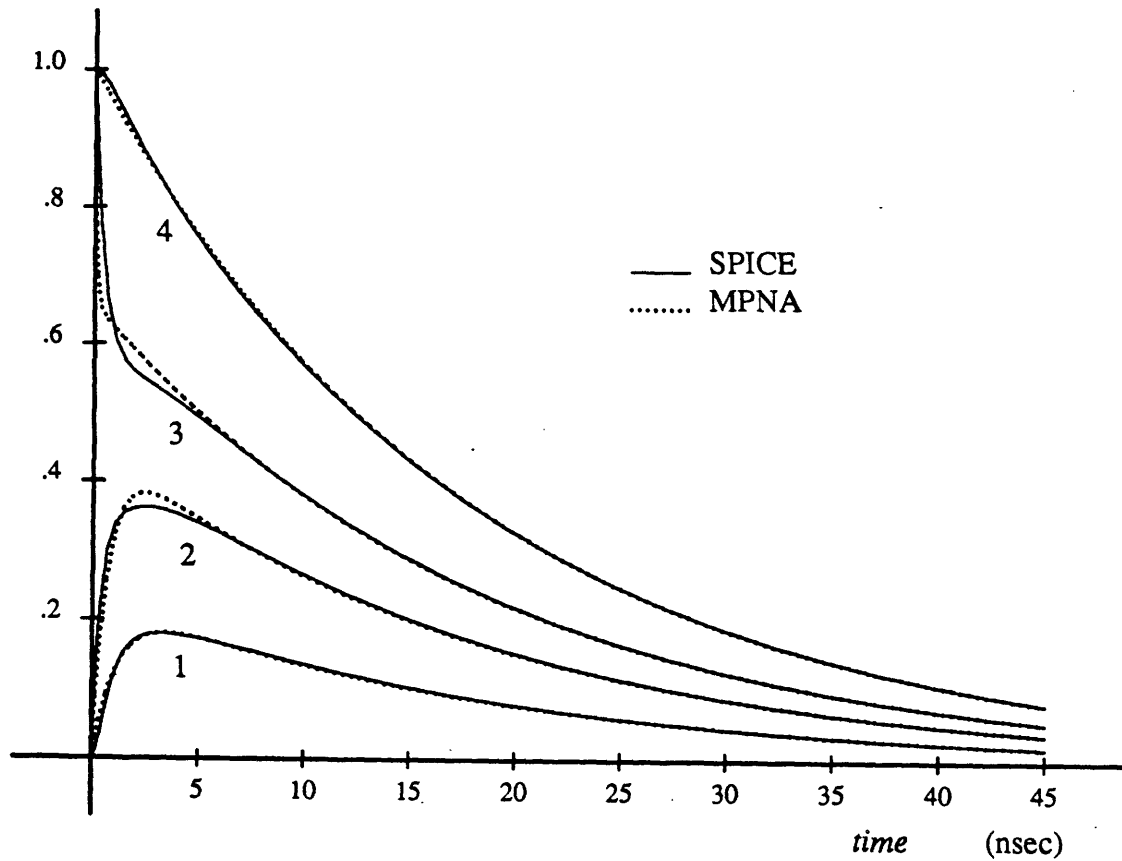
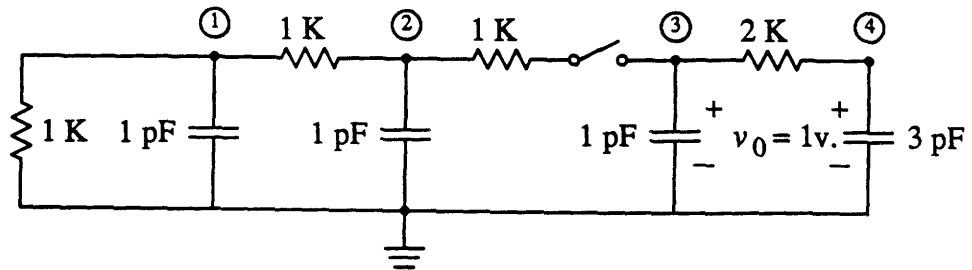


FIGURE 4-5: Two connecting *RC* trees

## Example 4.8

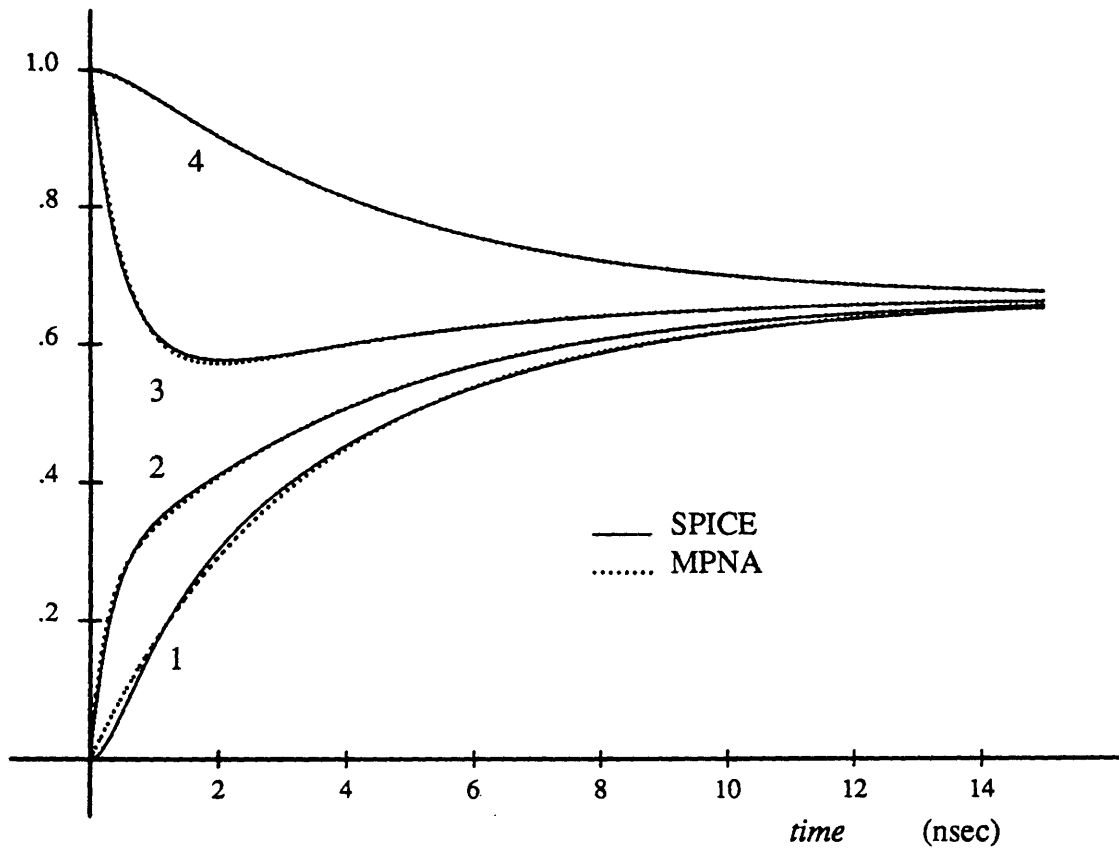
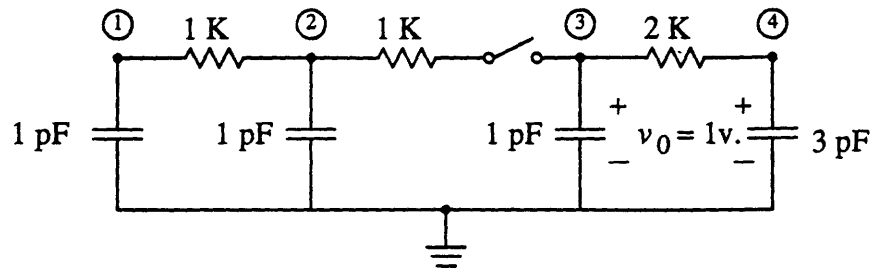


FIGURE 4-6: Charge sharing

Example 4.9

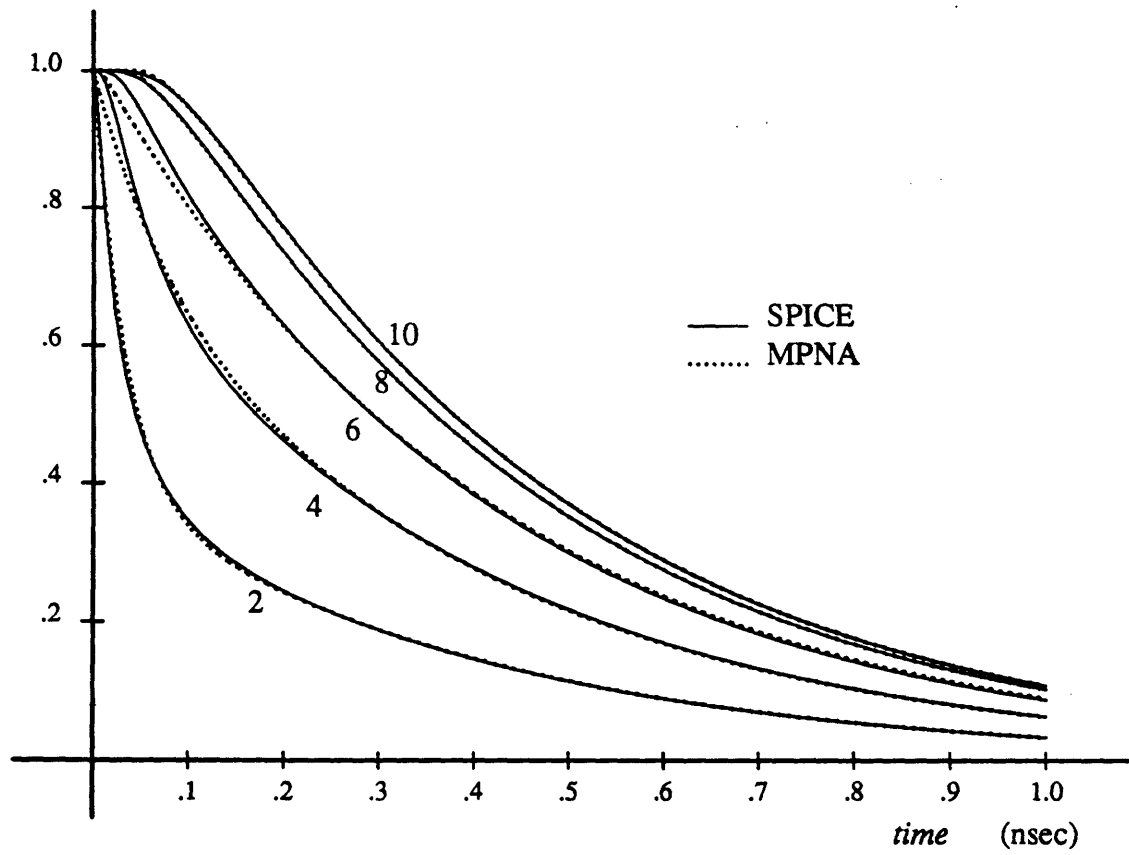
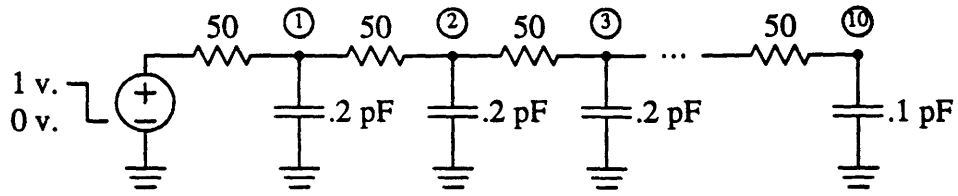


FIGURE 4-7: 10-stage RC line

Example 4.10

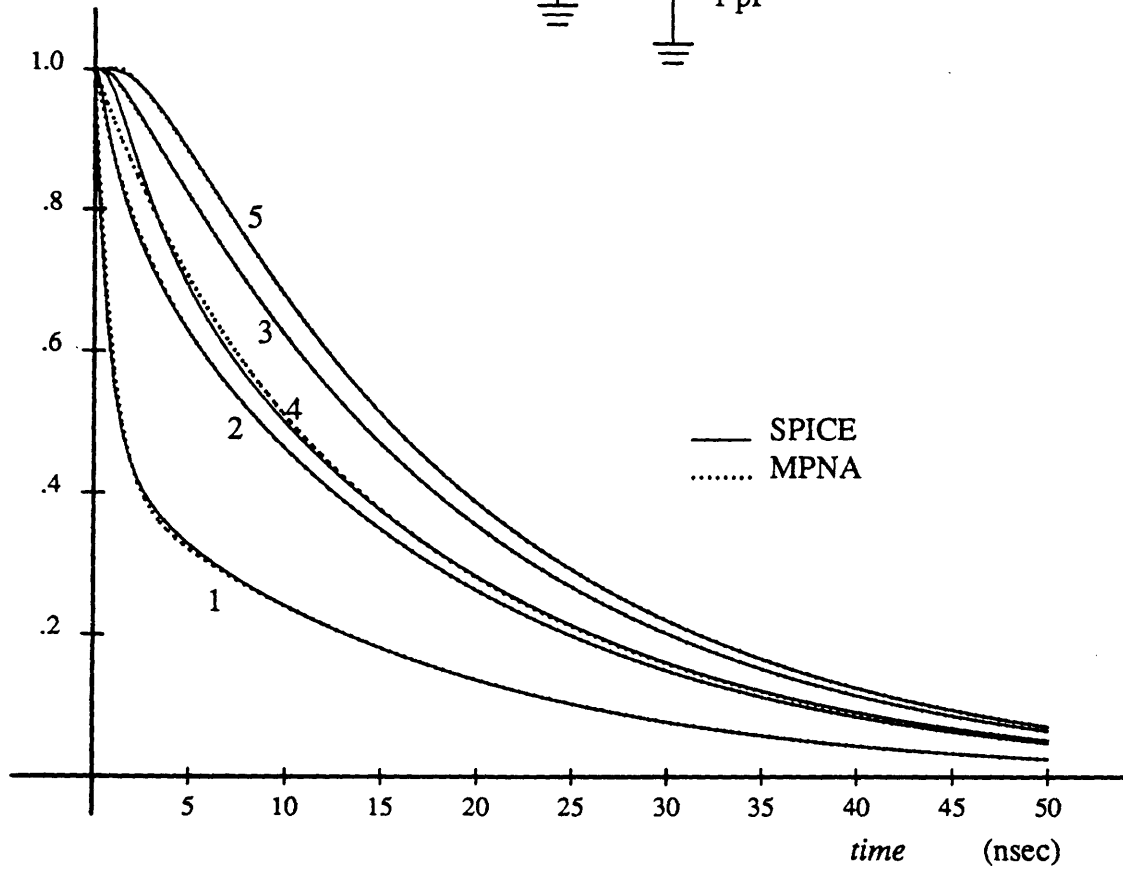
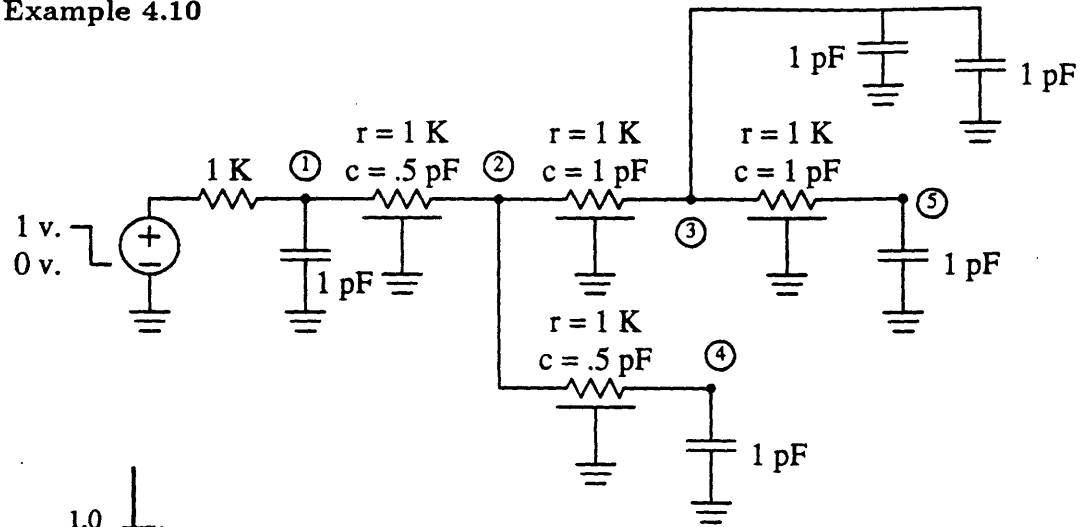


FIGURE 4-8: Distributed RC tree

Example 4.11

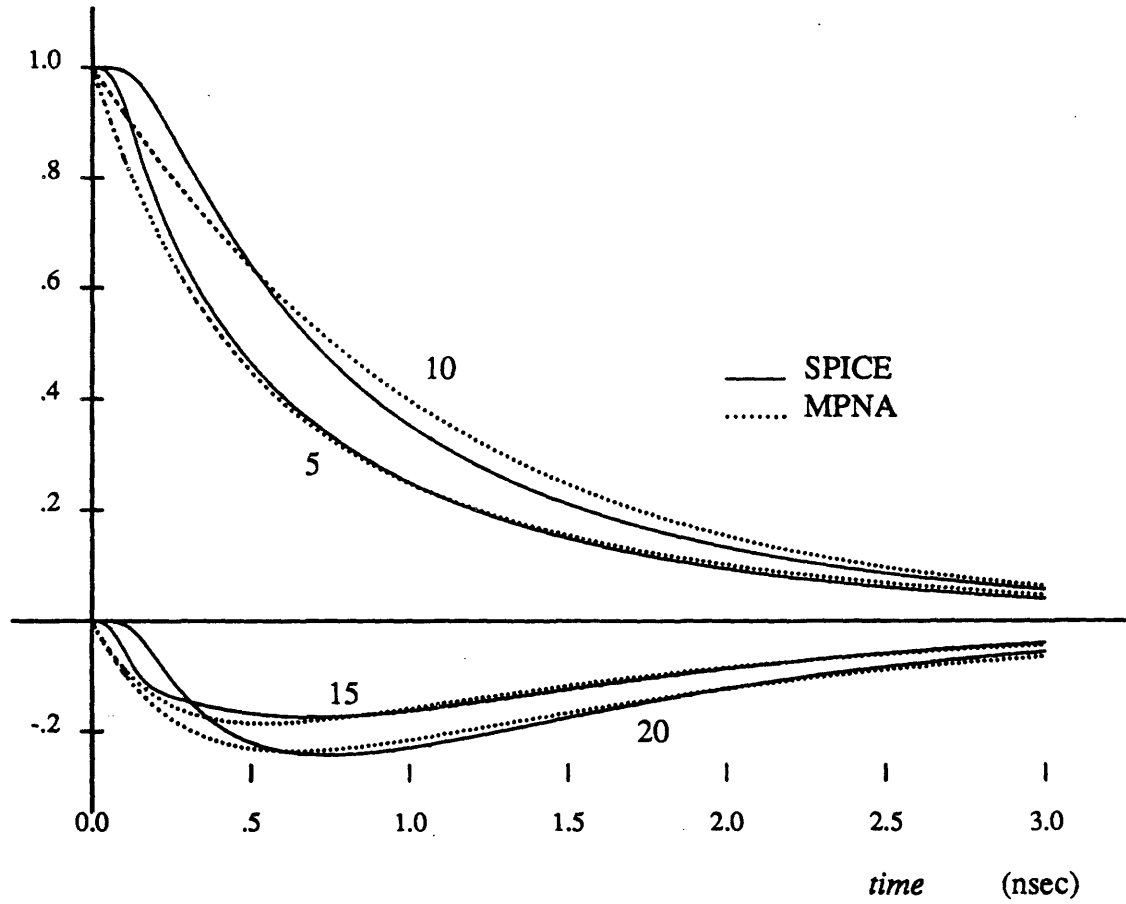
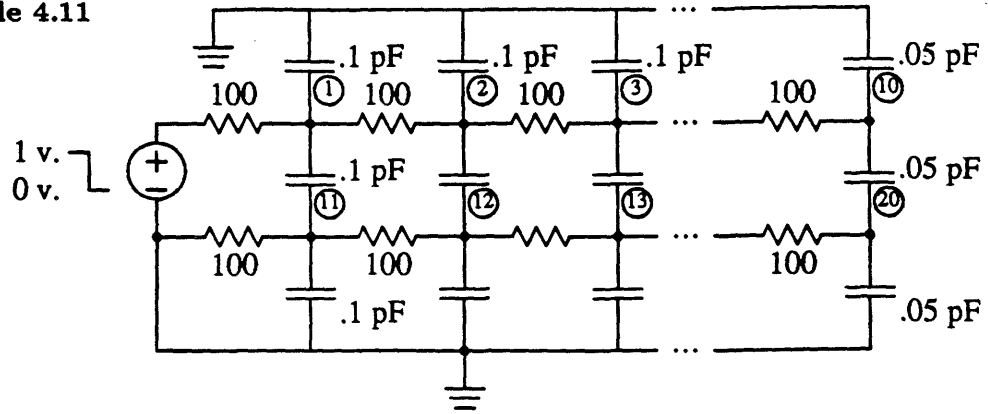
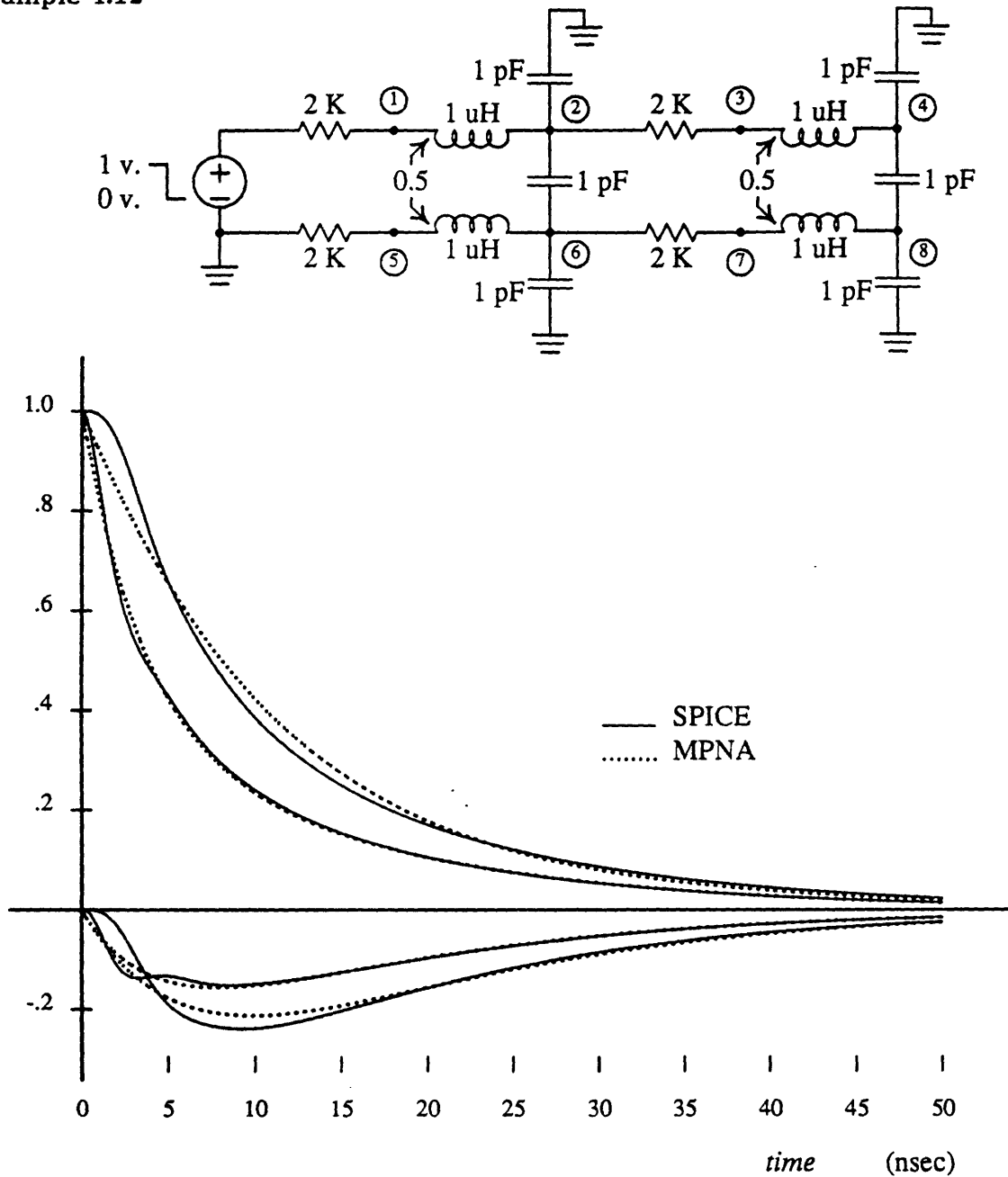


FIGURE 4-9: 10-stage coupled RC lines

## Example 4.12

FIGURE 4-10: Two-stage coupled *RLC* lines



	Penfield	Horowitz	MPNA $p = 1$	MPNA $p = 2$	MPNA <sup>2</sup> $p = 3$	direct methods
Example 4.7		100	83	200	370+400	1600
Example 4.8			150	310	510+400	1600
Example 4.9	50	180	500	1000	1700+400	5800
Example 4.10	45		440	940	1700+400	~9000 <sup>3</sup>
Example 4.11			1500	6000	15000+400	54000
Example 4.12			2200	4200	7400+400	13000

Table 4-1: Equivalent computations of examples for each method

two example circuits, and computes only first moment information. The *RC* tree models of Horowitz are equally limited, and only compute first and second moment information. Neither of these is capable of computing important coupling information.

Another thing to observe in Table 4-1 is that in all cases, the moment polynomial method requires less computation than direct methods. This is particularly true when modeling distributed elements, since these are treated as one element with MPNA equations. For instance, in Example 4.10 the computation requirements are 5–20 times less than direct methods, depending on result order.

Two final observations are made regarding the computational complexity of the MPNA algorithms. First, for a given circuit, each increase in order results in an increase of computation by two on average. Second, for a fixed result order and a variable number of nodes,  $n$ , the number of computations varies between  $\mathcal{O}(n)$  and  $\mathcal{O}(n^3)$ .  $\mathcal{O}(n^3)$  occurs only for a completely filled admittance matrix, which rarely happens with interconnection circuits. Figure 4-11 show more realistic complexity orders for interconnection circuits which vary between  $\mathcal{O}(n^{1.05})$  and  $\mathcal{O}(n^{1.90})$ .

## 4.5 Discussion

In this chapter we have seen that one can solve linear networks for the moment representation of all node voltage waveforms. By following the truncation order for intermediate calculations, the moment representations are guaranteed to be accurate to any desired order. Unlike other simulation algorithms, the type of circuit is not constrained to just *RC* trees, meshes, or to circuits with grounded capacitors at each node, etc.

<sup>2</sup>This figure is separated into two numbers, the first for the matrix solution time, the second for the number of operation to find one double exponential inverse, (about 400).

<sup>3</sup>Based on 3-stage lumped approximation for each distributed *RC* section.

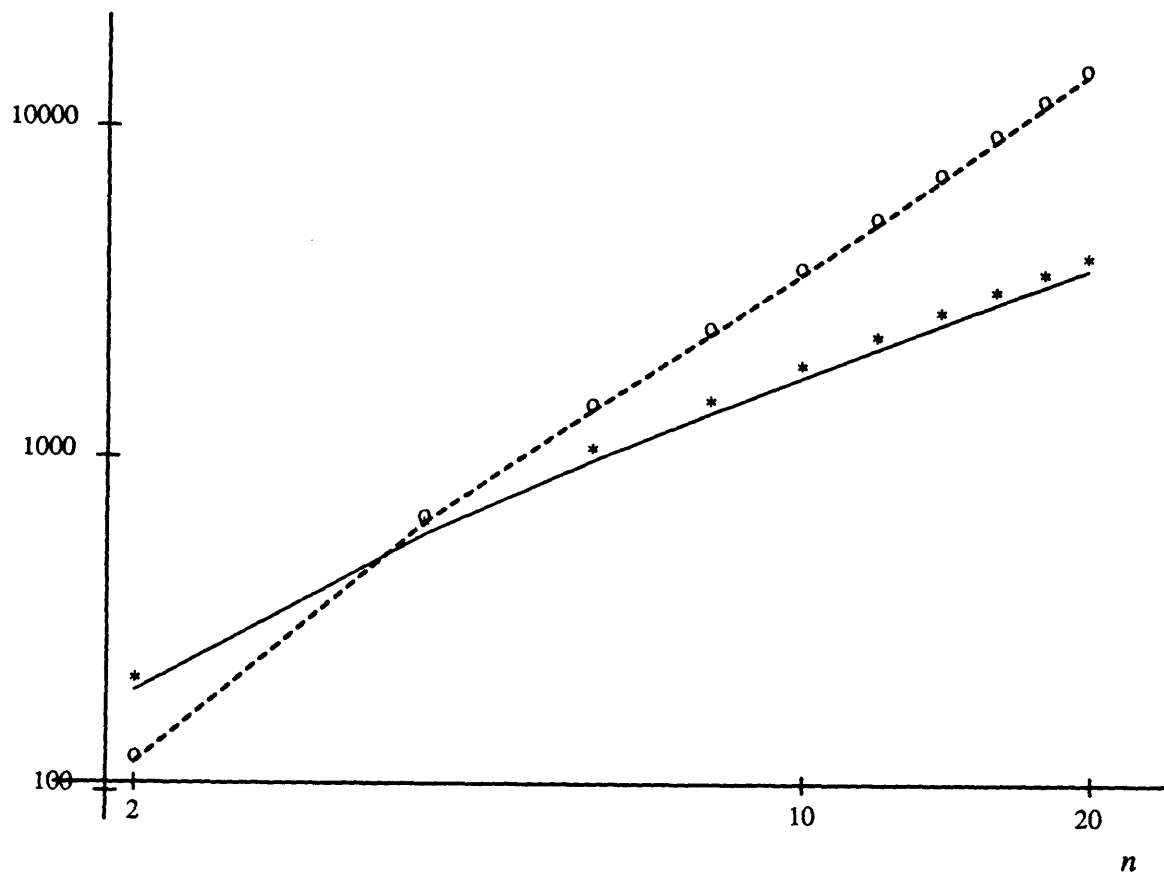


FIGURE 4-11: Computations vs. node count for  $n$ -stage  $RC$  lines (—) and  $\frac{n}{2}$ -stage coupled  $RC$  lines (- - -).

---



---

## Transmission Line Solutions for the Moment Representation

This chapter presents some special considerations for finding solutions to transmission line circuits. It relies on the previous chapter's methods, but improvements can be made for much better solutions. In this discussion, we will consider only lossless, coupled or uncoupled transmission lines which are in a homogeneous medium along their length. We also allow arbitrary networks at the transmission line ends. Abrupt changes in transmission line direction, coupling or dielectric surroundings can be modeled by two connected sections.

The advantage of the improved transmission line modeling is easily seen by considering the voltage response on the uncoupled, uniform  $LC$  line shown in Figure 5-1. The ends are ideally terminated to eliminate reflections. The voltage response at some distance,  $x_1$ , due to input  $V_{in}(s)$  is

$$V_1(s) = V_{in}(s) \cdot \frac{1}{2} e^{-s \frac{x_1}{u}} \quad (5.1)$$

where  $u = 1/\sqrt{LC}$  is the propagation velocity. We could model this distributed transmission line in the moment representation domain as described in Section 4.1.1 by taking the Maclaurin series of Equation (5.1):

$$\tilde{V}_1(s) = \tilde{V}_{in}(s) \cdot \frac{1}{2} \left( 1 - \frac{x_1}{u} s - \frac{x_1^2}{2u^2} s^2 + \dots \right).$$

But, since this response (aside from multiplying by the transmission coefficient of one-half) represents just a signal delay of  $\frac{x_1}{u}$ , an exact solution is easily computed by incrementing the  $t_0$  term of the moment representation,  $t_{0,V_1} = t_{0,V_{in}} + \frac{x_1}{u}$ , or

$$\tilde{V}_1(s) = \tilde{V}_{in}(s) \cdot e^{-s \frac{x_1}{u}} \left[ \frac{1}{2} s^0 \right].$$

The moment polynomial portion of the transfer function has one term,  $(\frac{1}{2}) s^0$ .

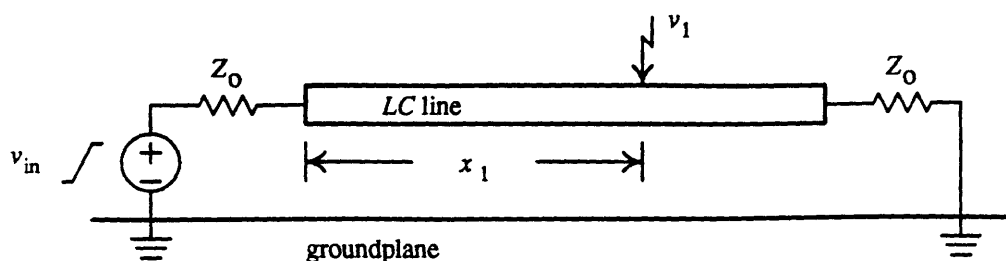


FIGURE 5-1: Response of single transmission line.

This solution is far easier to compute than a solution with MPNA, and is more accurate. Figure 5-2 shows an extreme example of what may occur with an MPNA solution. The waveform starting location is not updated, and the resulting double exponential waveform is far from the correct waveform that is obtained by increasing  $t_0$ . The above example demonstrates the basic technique for transmission line solutions with the moment representation—*increment  $t_0$  rather than the moment polynomial*. To handle circuits with reflections or coupling we turn to previous work on modal analysis of transmission lines.

## 5.1 Modal Analysis of Coupled Lossless Transmission Lines

In this section we consider a set of  $N$  coupled, lossless transmission lines. One common configuration is shown in Figure 5-3. The theoretical basis for this coupled transmission line modeling is well developed, and was introduced in Section 1.1.5. Here, we look at these existing methods more thoroughly.

Any excitation applied at one end of  $N$  coupled conductors activates a set of  $N$  *modes* which propagate to the opposite end at different velocities. In this thesis the two ends are designated as the *transmitting* and *receiving* ends, and variables are given  $t$  and  $r$  subscripts, respectively, to distinguish between quantities at the two ends. While this designation is arbitrary, we will assume that signals originate at the transmitting end. With the new method, mode strengths are computed for any specific input excitation, hence the name *modal analysis*. The modes propagate to the receiving end at their specific mode velocities; the mode strengths remain constant while making the end-to-end transition. At the receiving end, some energy reflects back to the transmitting end, depending on external circuitry connected to the transmission lines.

The telegrapher's equations (Equation (2.4)) with  $\mathbf{R}$  and  $\mathbf{G}$  absent reduce to an eigenvalue problem. Modal propagation velocities on coupled transmission lines are related to the eigenvalues of the matrix product,  $\mathbf{LC}$ , where  $\mathbf{L}$  is the inductance matrix per unit length of the conductors and  $\mathbf{C}$  is the capacitance matrix per unit length [53]. If the  $m^{\text{th}}$  eigenvalue of  $\mathbf{LC}$  is  $\lambda_m$ , then the propagation velocity of mode  $m$  is

$$u_m = 1/\sqrt{\lambda_m}.$$

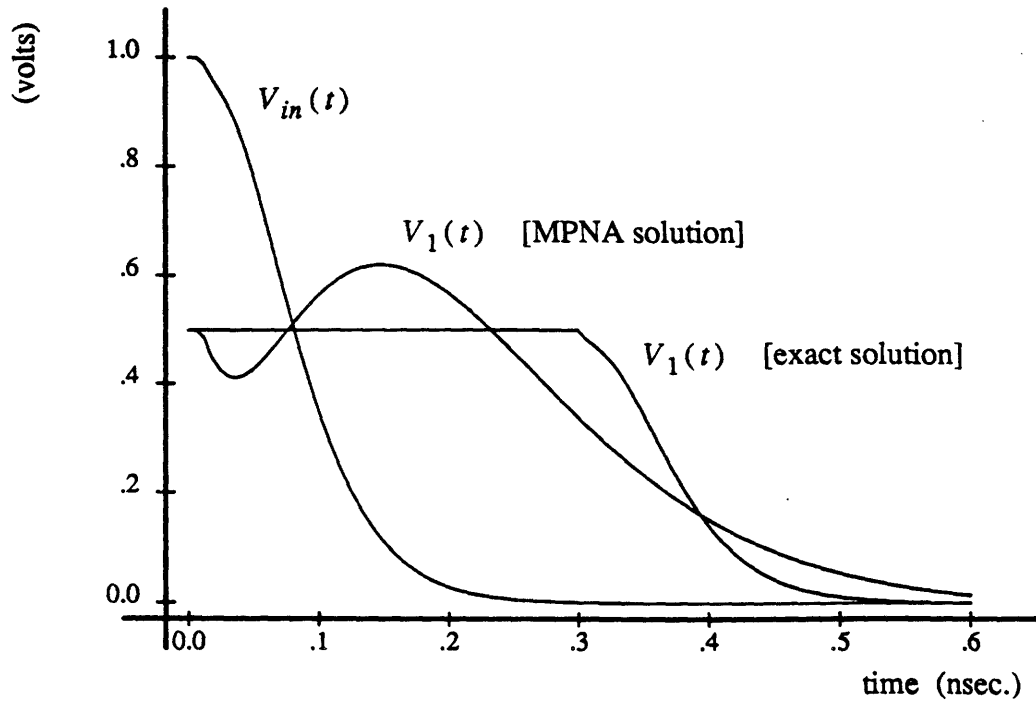


FIGURE 5-2: Waveforms from MPNA and modal analysis of transmission lines.

All waveforms are third order double exponential inverses of a sample output waveform.

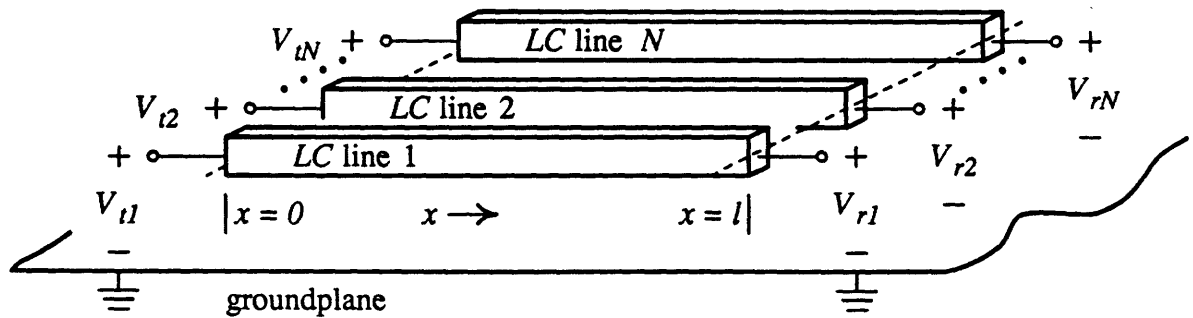


FIGURE 5-3: Coupled transmission line geometries.

The strength of mode  $m$  is designated  $h_{m\pm}(t)$ , where a “+” subscript indicates propagation in the  $+x$  direction and *vice versa*. At any  $x$  position, transmission line voltages and currents are related to mode strengths through the modal matrices,  $M_V$  and  $M_I$ , by

$$v(x, t) = M_V h(x, t) \quad (5.2)$$

$$i(x, t) = M_I h(x, t) \quad (5.3)$$

where  $h(x, t)$  is the vector of total mode strengths at  $x$ ,

$$h(x, t) = \begin{bmatrix} h_{1+}(t - \frac{x}{u_1}) \\ \vdots \\ h_{N+}(t - \frac{x}{u_N}) \end{bmatrix} + \begin{bmatrix} h_{1-}(t - \frac{l-x}{u_1}) \\ \vdots \\ h_{N-}(t - \frac{l-x}{u_N}) \end{bmatrix}.$$

The voltage modal matrix,  $M_V$  contains eigenvectors of  $LC$ , where column 1 contains the eigenvectors corresponding to  $\lambda_1$ , column 2 contains the eigenvectors corresponding to  $\lambda_2$ , and so forth. In this discussion, the eigenvectors may be scaled by any non-zero value. In cases where the  $N$  eigenvalues are not all distinct, care should be taken to ensure that the eigenvectors are linearly independent. The eigenvalues and eigenvectors may be found with any standard algorithm such as those described in [54]. The voltage and current modal matrices are related by

$$M_I = L^{-1} M_V \Lambda,$$

$$M_V = C^{-1} M_I \Lambda,$$

where

$$\Lambda = \begin{bmatrix} 1/u_1 & & & 0 \\ & 1/u_2 & & \\ & & \ddots & \\ 0 & & & 1/u_N \end{bmatrix}.$$

Figure 5-4 depicts the modal analysis method. First, a set of voltages applied to the transmitting end, is mapped into mode strengths with the voltage modal matrix. Next, the modes are propagated to the receiving end at their individual speeds. At time  $t = l/u_m$  mode  $m$  reaches the receiving end and is mapped back into transmission line voltages.

Thus far, we have not considered the effects of the transmitting or receiving circuitry. This circuitry and the transmission line characteristic impedances must be considered when calculating reflections and transmissions at endpoints. The common technique for computing interface behavior is to substitute the equivalent circuit of Figure 5-5 for each uncoupled mode transmission line in Figure 5-4. As shown in the equivalent circuit, each mode has a characteristic impedance,  $Z_m = L'_{mm}/u_m$  where  $L' = M_I^T L M_I$ . The sources in Figure 5-4 are activated when a propagating signal reaches the endpoint, i.e.,

$$j_{tm}(t) = 2 \frac{h_{rm}(t - l/u_m)}{Z_m} - j_{rm}(t - l/u_m),$$

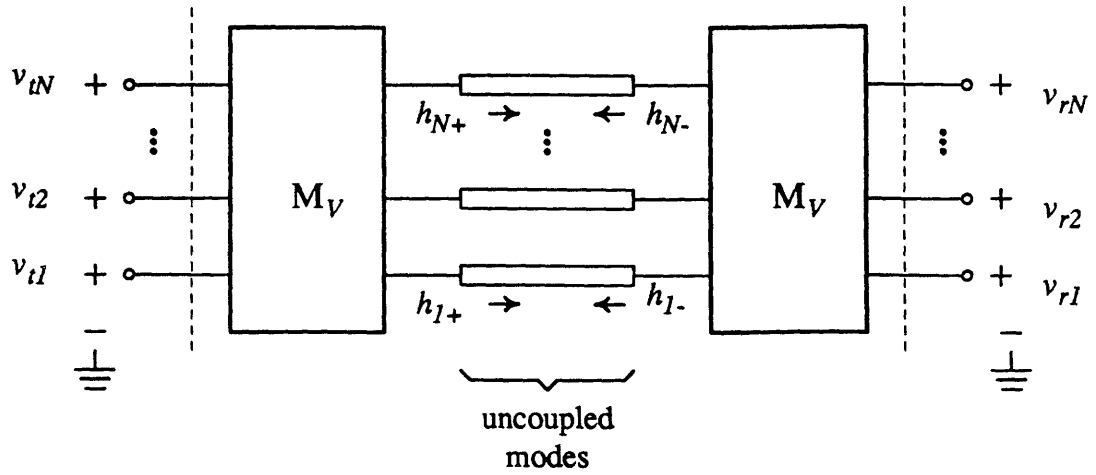


FIGURE 5-4: Modal analysis equivalent representation.

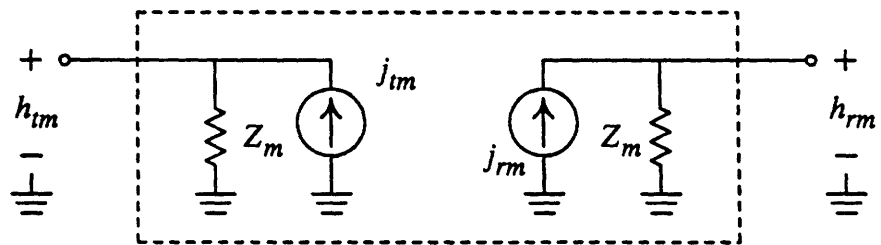


FIGURE 5-5: Method of characteristics equivalent circuit.

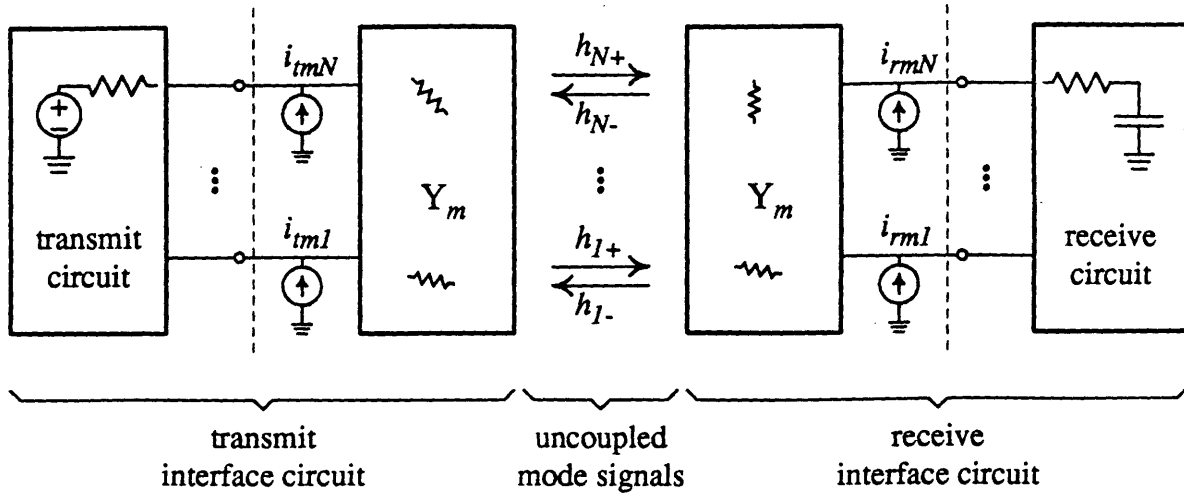


FIGURE 5-6: Modal analysis circuit.

$$j_{rm}(t) = 2 \frac{h_{tm}(t - l/u_m)}{Z_m} - j_{tm}(t - l/u_m).$$

This method, known as the *method of characteristics*, was first presented in [55] for a single transmission line.

The circuit that results from the above substitutions can be transformed into the circuit shown in Figure 5-6 by moving the transmission line impedances and sources outside of the modal matrices. The resulting transmission line impedances and sources are now represented by a characteristic admittance matrix,  $Y_0$ , and a pair of current source vectors,  $i_t(t)$  and  $i_r(t)$ . These connect directly to the external transmit and receive circuitry. The characteristic admittance matrix is found directly from the transmission line properties, e.g.,

$$\begin{aligned} Y_0 &= L^{-1}(LC)^{-\frac{1}{2}} \\ &= M_I \Lambda^{-1} M_I^{-1} C \\ &= L M_V \Lambda^{-1} M_V^{-1}. \end{aligned}$$

Signals still pass from end to end as mode strengths with  $h_+(t)$  and  $h_-(t)$ . The current source values are computed at the transmit and receive ends by

$$i_t(t) = 2 M_I h_-(t - l/u_m), \text{ and}$$

$$i_r(t) = 2 M_I h_+(t - l/u_m).$$

The reflected mode strength from each is computed from the voltage at the interface network and from the incoming mode strengths:

$$h_+(t) = M_V^{-1} v_t(t) - h_-(t - \frac{l}{u_m}), \quad (5.4)$$



$$\mathbf{h}_-(t) = \mathbf{M}_V^{-1} \mathbf{v}_r(t) - \mathbf{h}_+(t - \frac{l}{u_m}). \quad (5.5)$$

Considering two conditions of signal excitation on the transmitting end:

- when the excitation is caused entirely by the transmitter circuitry, only the first term of the right hand side of Equation (5.4) is non-zero,
- when the excitation is caused by a signal propagating from the receiving end, the reflected signal is computed by the difference between the first and second terms of Equation (5.4). When the first and second terms are equal, the transmitting end is ideally terminated since the two terms cancel, leaving no reflected signal.

## 5.2 Modal Analysis with the Moment Representation

Modal analysis methods translate easily into the moment representation domain. To develop the method, we will consider a signal initially generated by the transmit circuit with the transmission lines at rest. The moment representation simulator performs the following steps:

1. Solve the entire transmit interface circuit (see Figure 5-6) for  $\tilde{\mathbf{v}}(s)$  using the methods of Chapter 4. The voltage excitation comes from the transmit circuit, and since the transmission lines are at rest, the current source vector,  $\tilde{\mathbf{i}}_t(s) = \mathbf{0}$ . The transmission line admittance matrix is resistive, so in the moment representation domain of this portion of the circuit is  $\mathbf{Y}_0 s^0$ .
2. Find the moment representation mode strengths with

$$\tilde{\mathbf{h}}_+(s) = \mathbf{M}_V^{-1} s^0 \tilde{\mathbf{v}}_t(s), \quad (5.6)$$

which is derived by rewriting Equation (5.2) in the moment representation domain.

3. Compute each mode's arrival time at the receiving end,

$$t_m = t + l/u_m \quad \text{for mode } m,$$

and queue each mode's arrival on the other end as a separate transition in the simulator's global event queue. The individual mode signals on the opposite end are formed by zeroing all but one term of the  $\tilde{\mathbf{h}}_+(s)$  vector and then shifting it in time by incrementing the moment representation  $t_0$  term, i.e.,

$$\hat{\mathbf{h}}_{m+}(s) = e^{-s t_m} \begin{bmatrix} 0 \\ \vdots \\ \tilde{\mathbf{h}}_{m+}(s) \\ \vdots \\ 0 \end{bmatrix}$$

If the arrival times of different modes are equal or nearly equal, as is often the case, then their separate transitions can be combined into a single transition. We use the moment representation time shifting algorithm of Equation (3.9) to match all of the moments to a global starting time. To combine modes  $m$  and  $n$ , for instance,

$$\hat{h}_{m+,n+}(s) = e^{-s t_m} \left[ \hat{h}_{m+}(s) + \left\{ 1 - s(t_n - t_m) + \frac{s^2(t_n - t_m)^2}{2!} - \dots \right\} \hat{h}_{n+}(s) \right]$$

The notation will refer to a mode strength vector containing one or more individual modes all time shifted to a common starting point,  $t_m$ .

The advantage of combining transitions is to decrease the number of transitions which otherwise could be very numerous if signal reflections remain strong at both ends. Caution must be used in deciding which transitions to combine, since too much shifting can smear distinct signals into an inaccurate single waveform. It is usually safe if the difference between arrival times is small relative to the waveform durations, or if

$$|t_n - t_m| < M_{1,h_{m-}} / M_{0,h_{m-}}.$$

4. When a signal reaches the receiving end, convert the mode strength(s) into the current source waveforms

$$\tilde{i}_r(s) = 2 M_I \hat{h}_{\{m\}+}(s).$$

5. Solve the receive interface circuit, using the methods of Chapter 4.
6. At this stage we are repeating the cycle started in Step 1 above. Appropriate modifications to terminology are easily made to account for the fact that we are the receiving end, rather than the transmitting end. But first, one additional feature must be added to calculate the reflected signal strength at any interface. This is achieved by modifying Equation (5.6) in Step 2 to mirror Equation (5.4) in the moment representation domain, or

$$\tilde{h}_+(s) = M_V^{-1} s^0 \tilde{v}_i(s), - \hat{h}_{\{m\}-}(s).$$

After some number of reflections, the magnitudes of the mode signals decay to an insignificantly small value. Before any mode transition is added to the event queue, this condition is tested, and the signal is terminated if it is too small.

The algorithm for calculating the waveforms at either endpoint is stated concisely below.

**Algorithm 5.1**

Excitations can come from the external circuit or from within the transmission line or both. The procedure parameters are:

$\hat{h}_{\{m\}\mp}$ : Excitations from within transmission line or **0** if none.

$i_x$ : Excitations from external transmit or receive circuit or **0** if none.

$Y$ : Interface circuit admittance matrix, equal to  $Y_{external} + Y_0$ .

$M_I$ : Current modal matrix.

$M_v^{-1}$ : Inverse voltage modal matrix.

$t$ : Mode propagation times for length of transmission line.

All matrices are polynomial matrices.

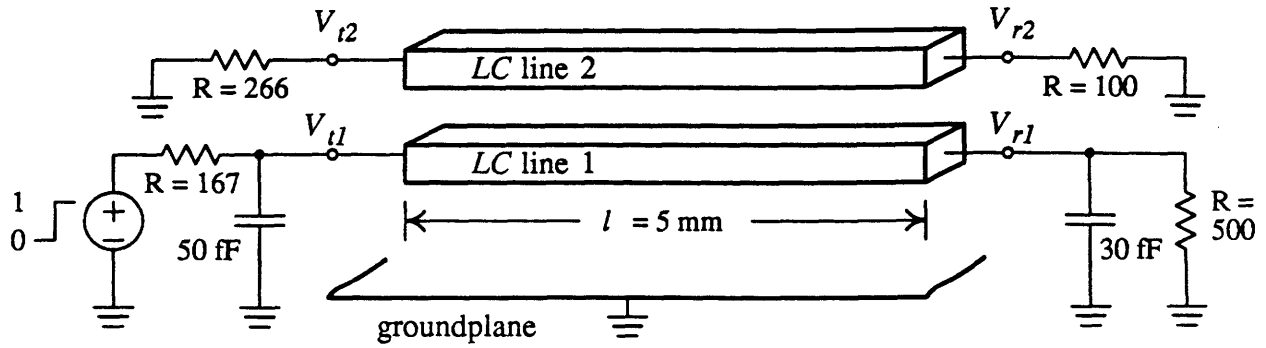
```

procedure LC_INTERFACE_SOLVE( $\hat{h}_{\{m\}\mp}$ ,  $i_x$ ,  $Y$ ,  $M_I$ ,  $M_v^{-1}$ ,  $t$ ) begin
{ Compute source currents as the sum of externally applied sources and
  the incident modes equivalent currents. }
   $i := i_x + 2M_I \hat{h}_{\{m\}\mp}$ ;
  { Solve the interface circuit. }
   $v := \text{SOLVE}(Y, i)$ ;
  { Compute new mode strengths. }
   $h_{\pm} := M_v^{-1} v - \hat{h}_{\{m\}\mp}$ ;
  { Delay modes by propagation times for length  $l$ . }
  for each mode,  $n$  do
     $h_{\pm}[n] := e^{-st[n]} h_{\pm}[n]$ ;
  for each set of closely spaced transitions,  $\hat{h}_{\{n\}\mp}$ , at time,  $t_n$  in  $h_{\pm}$  do begin
    if MAGNITUDE( $\hat{h}_{\{n\}\mp}$ ) > _threshold then
      QUEUE_TRANSITION( $t_n$ ,  $\hat{h}_{n\mp}$ );
    end
  end

```

**Example 5.1 (Two coupled transmission lines.)**

What is the moment representation solution to the circuit below?



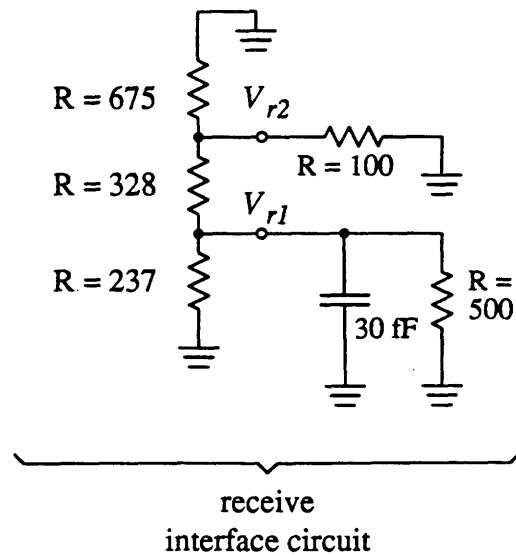
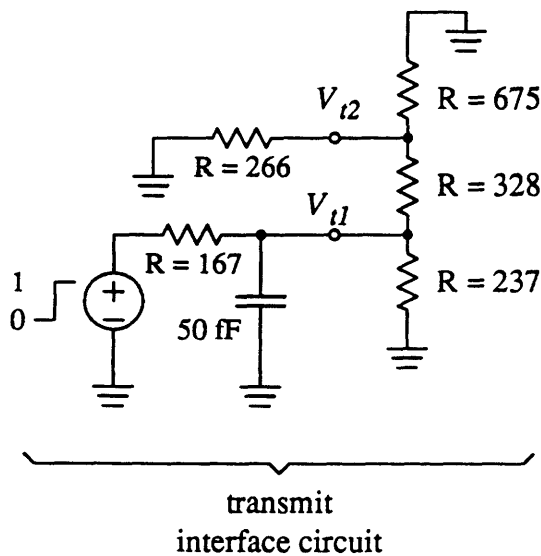
$$L = \begin{bmatrix} 2.0 & 1.5 \\ 1.5 & 3.6 \end{bmatrix} \mu\text{H/m}; \quad C = \begin{bmatrix} 72 & -30 \\ -30 & 50 \end{bmatrix} \text{pF/m}$$

For this configuration,

$$M_V = \begin{bmatrix} 1.0 & 0.4391 \\ -0.0439 & 1.0 \end{bmatrix},$$

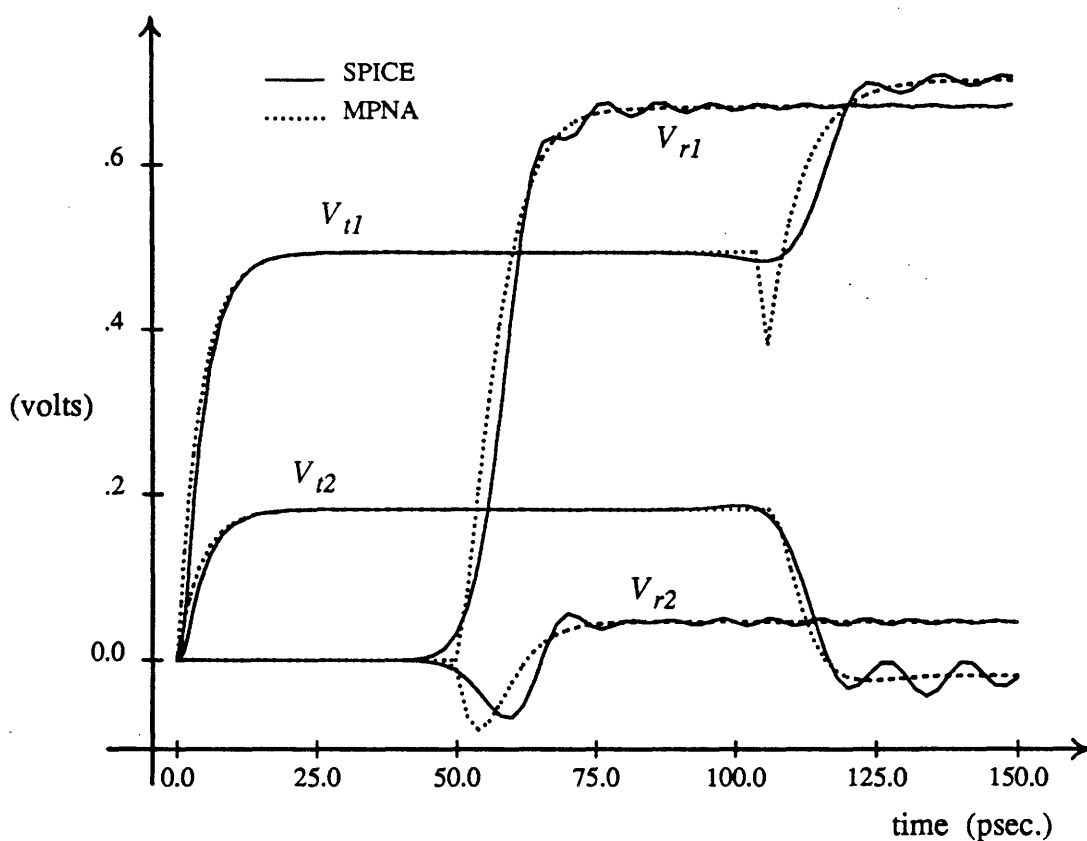
$$M_I = \begin{bmatrix} 7.3933 & 0.1401 \\ -3.2466 & 3.1913 \end{bmatrix},$$

and the interface circuits are as shown below.



Propagation times are 49.6 picoseconds for mode 1 and 57.7 picoseconds for mode 2. Since transition rise times are longer than the difference between propagation times, both modes are combined into one transition in this example.

The solution through the first internal reflection from the transmit end is shown below in dotted lines. The response from SPICE is shown in solid lines. The SPICE solution did not use modal analysis, and was done with 50 stages of coupled  $LC$  equivalent sections. The SPICE waveform ringing is an artifact of this, and does not reflect the true solution. For this example the moment representation solution is more correct than SPICE.



### 5.3 Discussion

This chapter has presented a modal analysis method for simulating lossless coupled transmission lines in the moment representation domain. This special modeling is needed because transmission line circuits tend to retain sharp transitions even through large delays. If the large delays are contained in the polynomial portion of the moment representation—as happens when we use the methods of Chapter 4—the time domain waveform approximations are

poor. The modal analysis methods operate by adjusting the  $t_0$  time delay term of the moment representation, rendering a much improved fit to the true solution.

Coupled transmission lines are simulated by modeling mode strengths in the moment representation domain. The interfaces at both ends of a set of coupled transmission lines is modeled by an equivalent resistive network, and is simulated using the linear circuit methods presented in Chapter 4. Each reflection is treated as a separate transition, or set of transitions in the event of widely varying mode velocities.

The computational requirements for moment representation modal analysis depend on several factors:

- the number of individual conductors contained in the coupled transmission lines,
- the number of reflections before a signal decays to an insignificant level,
- the closeness of the propagating mode velocities, and
- the complexity of the transmit and receive circuits.

During simulation the linear network solution time for the interface circuits is the most complex step. Chapter 7 shows a method for removing this computation time from the simulation steps.

One final note is made regarding lossy transmission lines. Lossy transmission lines have been modeled in the past by using an extension of time domain modal analysis [22] where many sections of lossless transmission line are connected in series with approximating resistors. The number of sections depends on the amount of attenuation; more attenuation requires more discrete sections. This type of modeling can be applied to moment representation simulation.

---

---

## Macromodels for Non-Linear Networks

Clearly, non-linear circuits, cannot be simulated with the linear network methods described in Chapter 4. Yet, non-linear transistor elements are a significant building block in digital circuits. In this chapter a macromodeling method is presented which translates a transistor circuit into an equivalent linear circuit that causes the same response. The linear circuit element values do not remain constant and are functions of the transistor circuit input waveform and output loading admittance. So, like SPICE, simulation time is increased for non-linear circuits. However, an advantage of combining macromodeling and the moment representation is that for a given circuit, the input-output characteristics can be precompiled into a small set of macromodel functions. Simulation speed is then considerably improved. This feature is demonstrated in the next chapter.

The macromodeling techniques in this thesis are based on the successful methods developed by Brocco [26,56]. The methods are modified to fit into the moment representation framework. Because of the more precise circuit and waveform specification ability of the moment representation, the macromodeling is improved even further over the previous methods.

A major issue in macromodeling is the number of macromodel function parameters. On the one hand, we desire as few as possible for efficiency reasons, but on the other hand, we desire enough to ensure an accurate performance characterization. In this work, two parameters describe the environment of an MOS transistor circuit with good accuracy. These are, roughly, the time for the input and time for the output waveforms to traverse through critical operating regions of the transistors. The output waveform parameter is mapped into an effective load capacitance parameter. The macromodeling converts a non-linear circuit into a sub-circuit of linear elements. The macromodel functions describe values for the circuit elements.

The above synopsis shows the steps taken during simulation: (1) a moment representation waveform is converted to the time domain, (2) time domain parameters are used to look up

macromodel function values, and (3) an equivalent linear circuit is formed using the macromodel function values.

Macromodel functions are constructed by extracting moments from real waveforms of the non-linear circuits. Most commonly, the waveforms are from SPICE simulations. Different circuit models are needed for the two types of transistor cells. First we look at MOS transistor circuit decoupling, and examine what constitutes a macromodel cell.

## 6.1 MOS Network Decoupling

Digital MOS circuits are divided into sub-circuits for different reasons. Two types of sub-circuits—cells and sub-networks—are discussed next.

### 6.1.1 Cells

A *cell* is a portion of a circuit which performs a well-characterized circuit function. A cell's boundary is usually defined by the circuit designer, and has input and output signals passing across its border. A typical cell may constitute an inverter, a NOR gate, an adder circuit, or an ALU. A cell may contain nothing more than interconnection lines, or may call other cells hierarchically.

Since cell hierarchy and cell structure are not primary subjects of this thesis, and since the subject could occupy an entire chapter, if not thesis, on its own, it will suffice to say that any MOS circuitry can be composed of the following elementary cell types.

**Interconnection cell:** A portion of a circuit representing one or more interconnections. It is composed of nodes connected by linear circuit elements. These cells do not contain non-linear circuits.

**Transistor driver cell:** An output node driven to a high or low voltage through a transistor path to  $V_{DD}$  or ground. The transistors are controlled by one or more controlling inputs to the cell. A logic gate output is modeled by a transistor driver cell.

**Transistor transmission cell:** Two nodes connected by a path through one or more transistors. The transistors are controlled by one or more controlling inputs to the cell. A CMOS transmission gate is, for instance, modeled by a transistor transmission cell.

### 6.1.2 Sub-networks

A *sub-network* is a portion of a circuit consisting of a set of nodes and elements which are connected through MOSFET sources and drains, resistors, or inductors. Basically, a sub-network is the smallest set of tightly connected or *conducting* nodes which must be solved as a unit if iteration is to be avoided. The boundary of a sub-network is computed by the preprocessor program with the algorithm described by Bryant in [52]. Bryant refers to sub-networks as



*transistor groups*—the different term is used here, since resistors and inductors also connect pieces of the sub-network.<sup>1</sup>

When circuit designers connect cells together, they almost certainly form sub-networks which are different from the cells. The notion of a sub-network is usually only useful in circuit simulation. Many simulators for MOS circuits [52,10] exploit the fact that a signal waveform can be closely determined by simulating sub-network circuits individually with known input signals applied to transistor gates in the sub-network.

The moment representation simulator exploits this same feature for calculating signal waveforms. The preprocessor program regroups circuit components into sub-networks. During simulation, each sub-network is solved independently; the only link between sub-networks is achieved by passing the output waveform of one sub-network to the input of another sub-network.

Coupling noise waveforms are computed with more than one sub-network, since capacitively coupled nodes are not typically in the same sub-network. A special simulator feature added for calculating capacitive coupling noise is outlined in this paragraph and is depicted in Figure 6-1. If the node of one sub-network undergoes a signal transition, and if this node is capacitively coupled to a node or nodes of another sub-network, then the circuit formed by the combination of the two sub-networks is solved for the noise waveform. All other capacitively coupled sub-networks are ignored (with the exception of extremely noise sensitive nodes) hence, their nodes are assumed to be held at constant voltages. If a sub-network is capacitively influenced by more than one sub-network at the same time, then the influences are calculated individually (i.e., still only two sub-networks are combined at once) and then the waveforms are summed. Only the negligible effect of double coupling (where one line couples to a second line which couples to a third line) is not treated.

### 6.1.3 Macromodel Cells

Circuits are macromodeled on cell boundaries. A macromodeled cell is best utilized when replicated many times since much preprocessing is needed for each macromodeled cell, and sharing this among many circuits makes the most efficient use of the preprocessing. The best boundary for a macromodel cell is at the corresponding boundaries of the circuit cell or layout cell. Of the cell types defined above, transistor driver cells and transistor transmission cells are macromodeled, while interconnections are not.

One restriction exists in forming macromodel cell boundaries and in connecting macromodel cells: *circuit feedback paths must not cross cell boundaries*. That is, a circuit feedback loop must be entirely contained within one macromodel cell. (See Figure 6-2.) This restriction does not

---

<sup>1</sup>The sub-network can also be defined by the terms given in Chapter 4. A sub-network is a set of nodes connected through circuit elements with order 0 or less. MOSFET source-to-drain connections have circuit element orders equal to 0 and MOSFET gate-to-source and gate-to-drain connections have element orders equal to +1.

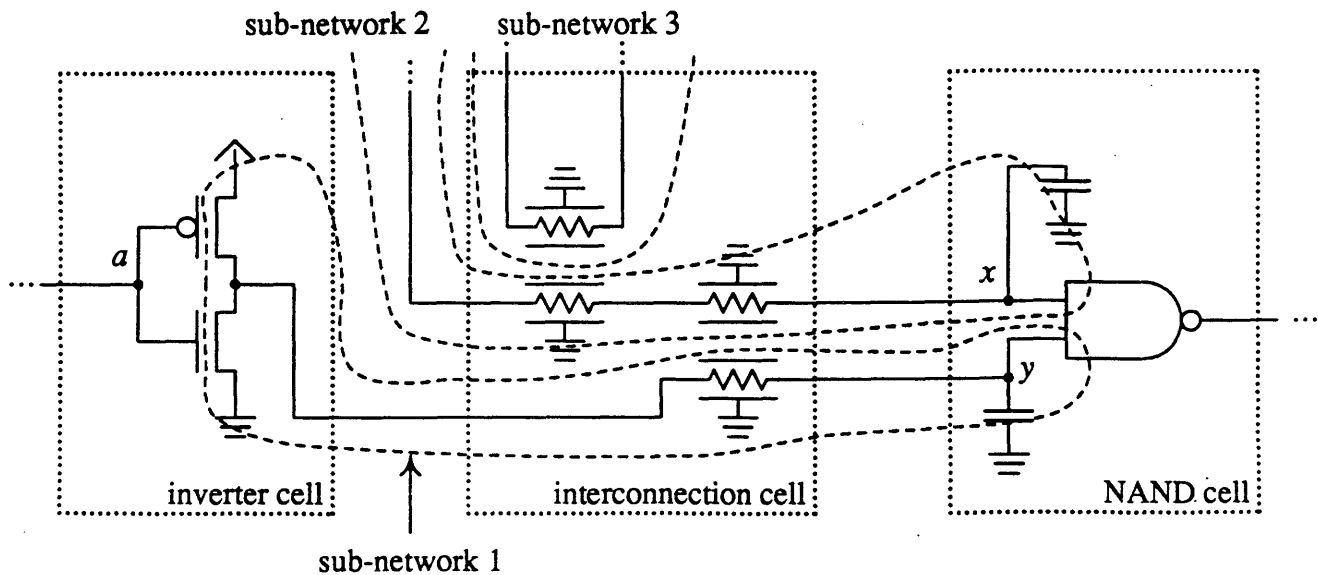


FIGURE 6-1: Sub-network boundaries.

To find the coupling noise caused at node  $x$  by a transition at node  $a$ , the circuit formed by combining sub-network 1 and 2 is solved independently of sub-network 3.

apply to clocked logic feedback, as in the case of a clocked finite-state machine.

## 6.2 Fundamentals of Macromodeling

The motivation behind macromodeling is to reduce a computationally expensive task to a simple table lookup task. Table values are computed once, by the expensive method, but then can be accessed quickly, any number of times.

A *macromodel function* is any function with respect to important, controlling *input parameters*,  $p_1, p_2, \dots, p_n$ . Sample values of an  $n$ -dimensional macromodel function,  $F(p_1, p_2, \dots, p_n)$ , are stored in an  $n$ -dimensional array, at discrete, closely-spaced increments of the input parameter *data points*. When a macromodel value is fetched from the tables with an input parameter located between two data points, some form of interpolation is used. In this thesis, linear interpolation is assumed, and only one- and two-dimensional functions are used.

Previous macromodeling efforts for circuit simulation have macromodeled

1. logic gate output waveform delay time and slope versus input waveform slope and output load capacitance [57,26,25],
2. MOSFET  $i_{ds}$  versus  $v_{ds}$  and  $v_{gs}$  [9,58],

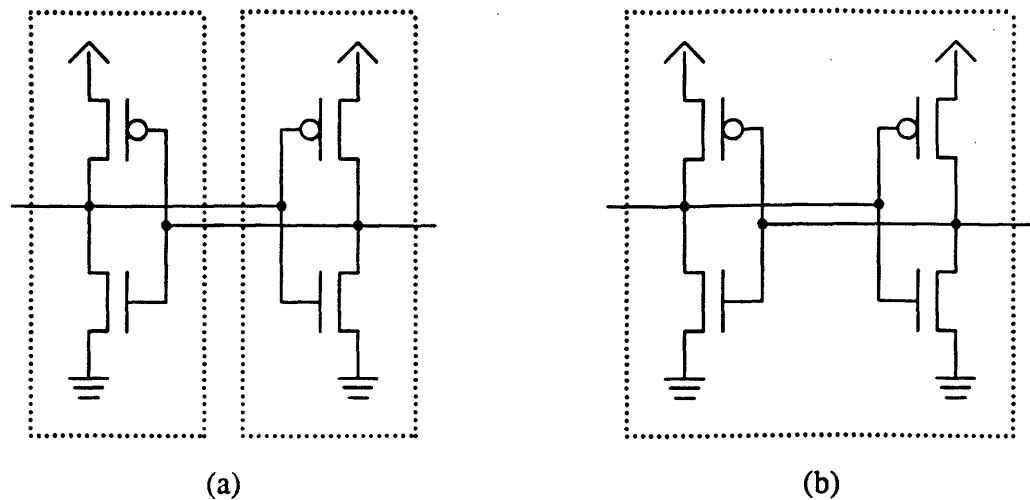


FIGURE 6-2: Illegal (a) and legal (b) macromodel cells for a cross-coupled inverter circuit.

### 3. ECL gate drive values versus effective load capacitance and input pulse duration [59].

In this thesis, macromodel functions describe moment representation values or circuit element values versus the input waveform slope and output driving-point (or load) admittance.

Transistor driver cells and transistor transmission cells are modeled slightly differently. Transistor driver cells are discussed in detail. In the next sections, the circuit model, input parameters, macromodel functions and macromodel extraction of driver cells are covered. Then, transistor transmission cells are covered briefly.

## 6.3 Transistor Driver Cell Model

Figure 6-3(a) depicts a general transistor driver cell connected to load admittance,  $\tilde{Y}_{load}(s)$ , which includes all circuit elements in the sub-network. It is modeled by the circuit of Figure 6-3(b). The elements surrounded by a dashed line are macromodeled. Transistor driver cells have distinct inputs and outputs, which are in separate sub-networks. The inputs' sub-networks are solved first, and then the output's sub-network. We assume that the output response has no effect on the input sub-network, which to a good approximation is true for CMOS gates. Miller feedback capacitance of single level logic gates is the only connection between inputs and outputs.

### 6.3.1 Input Capacitance

A capacitance exists on any cell input that connects to a MOSFET gate. The transistor capacitance is modeled by a single grounded capacitor,  $C_{in}$ , even though it actually connects to the source and drain. The  $C_{in}$  approximation completely decouples the input and output

sub-networks. For any input transition an equivalent capacitance,  $C_{in}$  is calculated by

$$C_{in} \equiv \frac{\Delta q_{in}}{\Delta V_{in}} = \frac{C_{gd}(\Delta V_{in} - \Delta V_d) + C_{gs}(\Delta V_{in} - \Delta V_s)}{\Delta V_{in}}$$

where the voltage and charge terms are defined in Figure 6-4.

The values of  $\Delta V_d$  and  $\Delta V_s$  depend on the input transistor's circuit configuration, so it's often best to determine  $C_{in}$  by measuring  $\Delta q_{in}$  directly while the macromodel extractions are made.

## 6.4 Macromodel Input Parameters

The transition behavior of a transistor driver cell is governed by two things: (1) the input waveform slope and (2) the loading circuit connected to the output. By carefully quantifying the input signal waveform,  $\tilde{V}_{in}(s)$ , and load admittance,  $\tilde{Y}_{load}(s)$  with a single value apiece, macromodel functions are kept compact and fast. Now, the question arises how we can accurately characterize a waveform representation ( $\tilde{V}_{in}(s)$  or  $\tilde{Y}_{load}(s)$ ) with a single value.

Let us first consider the possibility of using the lowest-ordered, unassumed term in the moment polynomial. By this, we mean  $M_1$  of the input signal waveform<sup>2</sup> and  $M_1$  of the driving-point admittance<sup>3</sup>. These equal the input waveform's Elmore time and the load's total ground capacitance, respectively. These values as macromodel parameters work well for some circuits, but, as shown in [56], can lead to errors of 100% or more for circuits with large line resistances. Thus,  $M_1$  of  $\tilde{V}_{in}(s)$  and  $\tilde{Y}_{load}(s)$  are not good macromodel parameters.

### 6.4.1 Input Waveform Parameter

The waveforms of Figure 6-5 demonstrate why  $M_1$  of  $\tilde{V}_{in}(s)$  is a poor parameter. Both waveforms are representative of MOSFET and/or  $RC$  interconnect waveforms, and both have identical  $M_1$ . However, if these waveforms drive an inverter input, for instance, the responses are noticeably different. The critical section of a voltage waveform which defines an inverter response is the section which passes through the inverter high-gain input voltage region. Thus, a better macromodel parameter for input waveform shape is the amount of time for the waveform to pass through the high-gain voltage region.

More precisely, the input waveform macromodel parameter is

$$t_{r,in} \equiv \left| \frac{t_{cr2} - t_{cr1}}{V_{cr2} - V_{cr1}} \right| \quad (6.1)$$

where  $V_{cr1}$ ,  $V_{cr2}$ ,  $t_{cr1}$  and  $t_{cr2}$  are the critical voltages and times defined in Figure 6-5.

<sup>2</sup> $M_0$  is the d.c. transition voltage and is always  $\pm(V_{high} - V_{low})$  for a logic transition.

<sup>3</sup>The load on the macromodel driver cell has no resistive or inductive paths to ground, and is only capacitive. From Theorem B.1 we know that  $M_1$  is the lowest ordered, non-zero term.

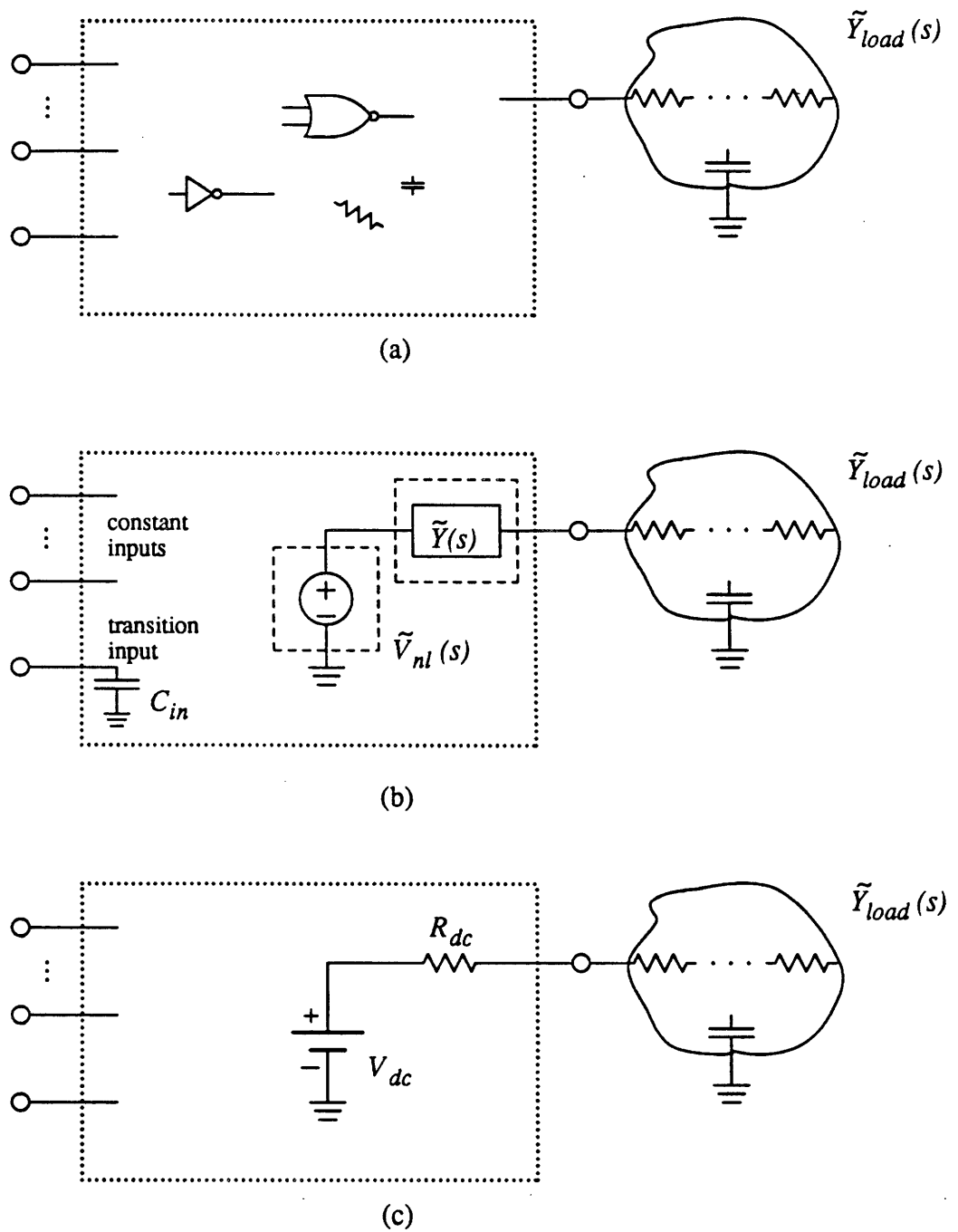


FIGURE 6-3: (a) General macromodel driver cell, (b) Linear network equivalent for a changing output, (c) Linear network equivalent for static output.

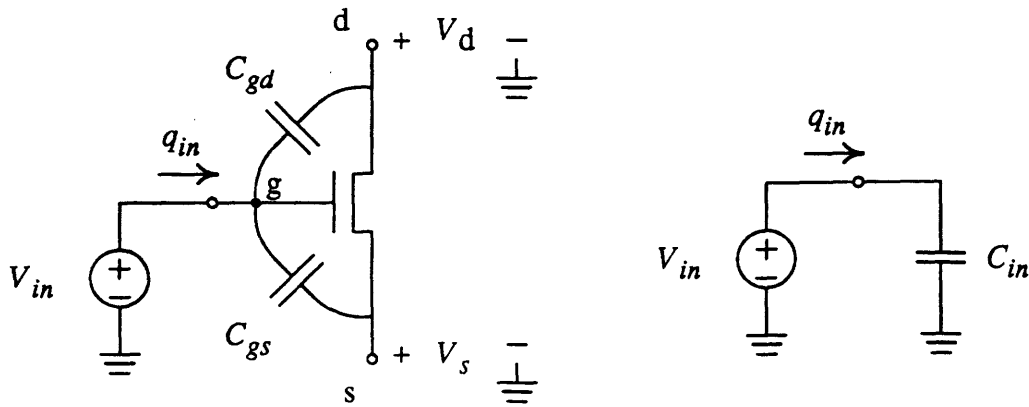


FIGURE 6-4: MOSFET input capacitance.

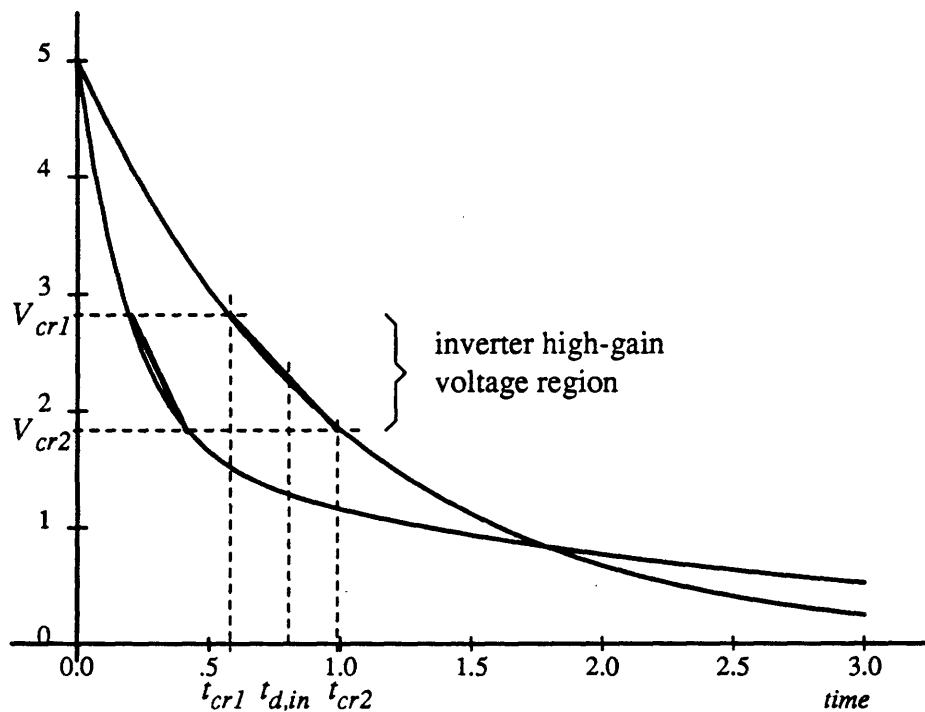


FIGURE 6-5: Representative input voltage waveforms with identical Elmore times,  $M_1$ .

This macromodel parameter was first used in [26] and is also used in this thesis. Two things are readily seen. First, the overall delay of the critical line segment can vary. Another input waveform parameter,  $t_{d,in}$ , is extracted from the input waveform, and the cell output delay is adjusted by adding  $t_{d,in}$  to correct for the overall shift in input time. Secondly, if the input waveform is in the moment representation, it must be converted to a time domain representation.

The possibility of using many moment representation terms (more than just  $M_1$ ) for macromodel input parameters has also been considered. An advantage of using  $M_1$ ,  $M_2$  and  $M_3$ , for instance, is that we can avoid the time consuming task of converting each moment representation waveform into the time domain with the double exponential assumed waveform shape. This idea was rejected, for three reasons: first, macromodel functions with third order moment representation terms as input parameters are not smooth. The accuracy of macromodeling is largely affected by the smoothness of the functions; it is difficult to interpolate accurately on macromodel functions that jump around alot, and hence function evaluations are inaccurate. Second, the size of macromodel function tables is extremely large, since (1) table dimensions must be increased to at least three for third order moment representation modeling, (2) to acheive any type of accuracy, very closely spaced macromodel data points are necessary, and (3) moment representations terms may span a broad range of values, even when intelligently "normalized". Third, it is hard to extract macromodel functions from circuit data. Appendix C discusses this topic in more detail.

#### 6.4.2 Output Load Parameter

Unlike other macromodeling efforts, the output load macromodel parameter for this thesis is also based on the time for a waveform—in this case the output waveform—to pass through a critical region. This critical region covers (1) the region where the driving transistors are conducting most, and (2) the critical regions for inputs to succeeding stages. The output voltage critical region is selected as the voltages between 0% and 75%, of the output voltage swing as shown in Figure 6-6.

To use a time,  $t_{75\%} - t_{0\%}$ , based on the output waveform as a macromodel parameter brings out two problems:

1. the output voltage waveform must be presupposed before the output voltage macromodel functions are known,
2. performing test simulations to compute macromodel functions at regularly spaced intervals of  $t_{75\%} - t_{0\%}$  is difficult, as this would require iteration to find the correct input to achieve a specific output transition time.

Hence, another macromodel parameter, effective load capacitance,  $C_{eff}$ , is chosen for the macromodel parameters.  $C_{eff}$  is, very approximately, the capacitance which can replace  $\tilde{Y}_{load}(s)$ ,

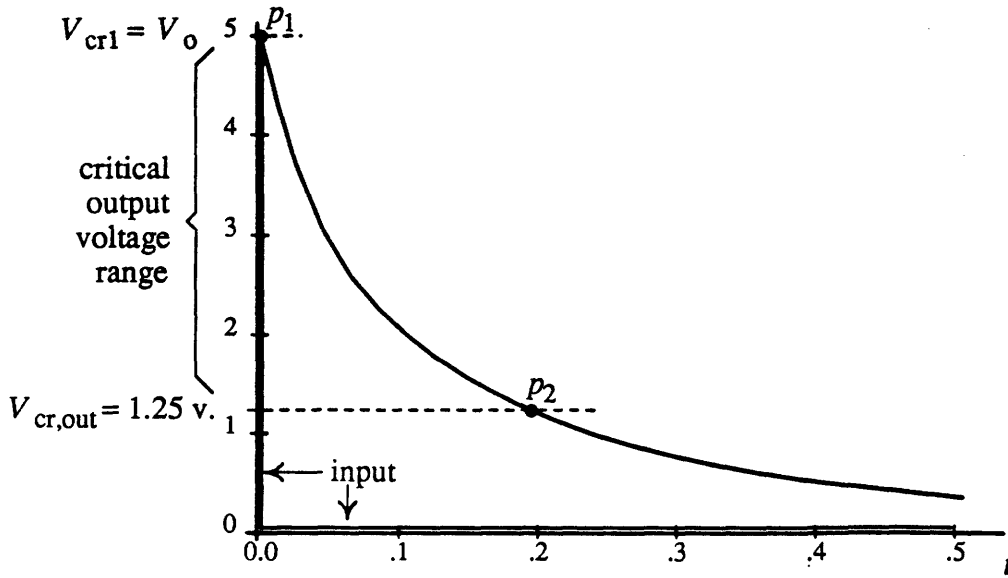


FIGURE 6-6: Critical range of output voltage.

such that the output voltage waveform of the cell passes through the same end points of the critical output voltage range as the real output—namely, the output voltage passes through  $p_1$  and  $p_2$  of Figure 6-6. During simulation, effective load capacitance is computed as follows:

1. For each occurrence of a macromodel driver cell, an approximate step response waveform,  $V_{step}(t)$ , is computed for the output node with a simple equivalent circuit substituted for the macromodel cell (Figure 6-7(a)).
2. The effective load for a step input is computed such that the output voltage of Fig-

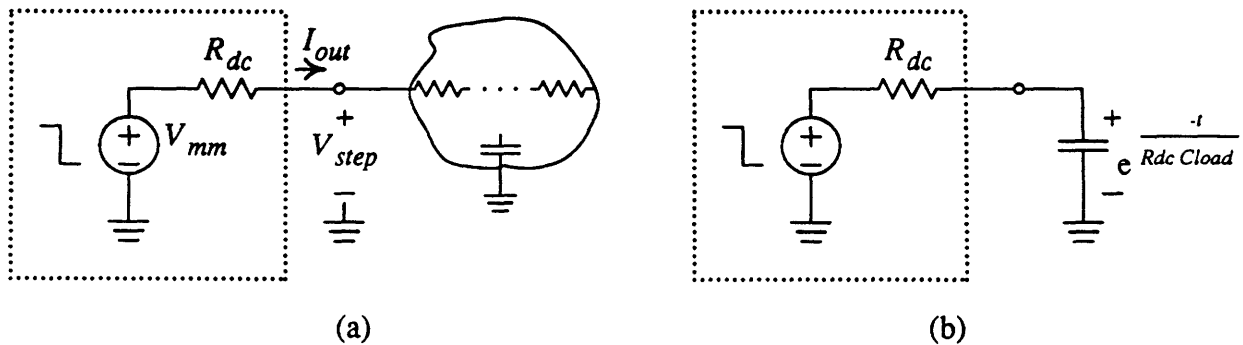


FIGURE 6-7: Approximating circuit for computing  $V_{step}(t)$ .



ures 6-7(a) and 6-7(b) are the same at  $t = 0$  and  $t = t_{cr,out}$ , i.e.

$$C_{step} = -\frac{t_{cr,out}}{R_{dc} \ln(V_{cr,out}/V_0)}$$

where  $R_{dc}$  is the static incremental output resistance

$$R_{dc} = \left. \frac{\partial V_{step}}{\partial I_{out}} \right|_{V_{step}=V_{mm}}$$

3.  $C_{step}$  equals  $C_{eff}$  only if the inverter response is fast compared to the interconnect step response,  $V_{step}(t)$ . If the inverter response is slow compared to  $V_{step}(t)$ , then  $Y_{load}$  is better approximated by  $C_{load} = M_{1,Y_{load}}$ , the total load capacitance to ground.

For each output transition during simulation, the effective load capacitance is computed in the range between  $C_{step}$  and  $C_{load}$ .<sup>4</sup> This is done by comparing the time constants of the macromodel voltage source,  $\tilde{V}_{mm}(s)$ , (the derivation of  $\tilde{V}_{mm}(s)$  is described in the next section) and the step response,  $\tilde{V}_{step}(s)$ .  $C_{eff}$  is approximated by

$$C_{eff} = C_{step} + (C_{load} - C_{step}) \frac{1}{1 + \tau_{step}/\tau_{mm}}$$

where

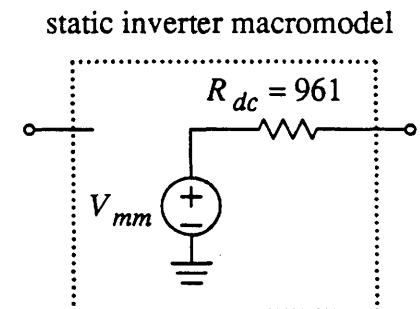
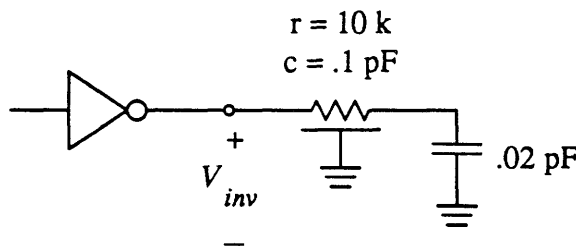
$$\tau_{step} \equiv M_{1,V_{step}}/M_{0,V_{step}}$$

and

$$\tau_{mm} \equiv M_{1,V_{mm}}/M_{0,V_{mm}}$$

Note that steps 1 and 2 require more computation, but only once per circuit, whereas step 3 requires little computation, but more often; once per circuit transition.

**Example 6.1** What is  $C_{eff}$  for the circuit below when the inverter output time constant,  $\tau_{mm}$ , is 0.01ns, 0.1ns and 1.0ns?

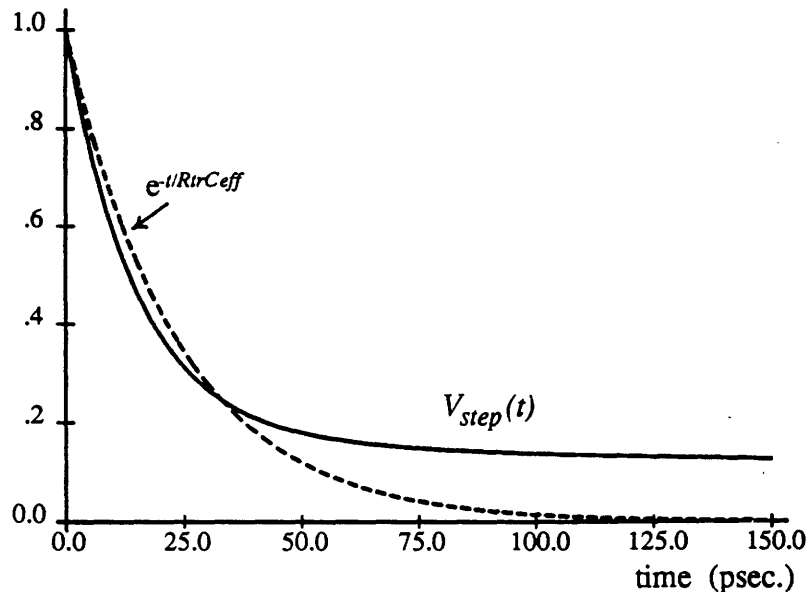


<sup>4</sup>For RC loads  $C_{step} \leq C_{load}$ , and if the load capacitance is a single capacitor to ground, then  $C_{step} = C_{load}$ .

Attach the inverter's load to the static inverter macromodel, and solve for  $\tilde{V}_{inv}(s)$  with  $\tilde{V}_{mm}(s) = 1$ , a step input. This gives

$$\tilde{V}_{step}(s) = 1 - 0.106 \times 10^{-9} s + 0.0535 \times 10^{-8} s^2 - 0.0305 \times 10^{-27} s^3 \quad (6.2)$$

which is plotted with a solid line below.



From the value of  $t_{cr}$ ,

$$C_{step} = -\frac{t_{cr}}{R_{dc} \ln(.25)} = 0.0245 \text{ pF}$$

which gives the approximate response plotted with a dashed line above. From inspection of the circuit, the total ground capacitance of the load is  $C_{load} = 0.12 \text{ pF}$ .

From (6.2) above,

$$\tilde{\tau}_{step}(s) = M_{1,V_{step}} = 0.106 \text{ ns},$$

and finally, from Equation (3)

$$\begin{aligned} C_{eff} &= 0.0245 + \frac{0.0955}{1 + (0.106/\tau_{mm})} \text{ pF} \\ &= \begin{cases} 0.0327 \text{ pF} & \text{for } \tau_{mm} = 0.01 \text{ ns} \\ 0.0709 \text{ pF} & \text{for } \tau_{mm} = 0.1 \text{ ns} \\ 0.1108 \text{ pF} & \text{for } \tau_{mm} = 1.0 \text{ ns} \end{cases} \end{aligned}$$

---

It is important to adjust the load capacitance parameter particularly for lines with much resistance. This is demonstrated in Example 6.6 at the end of this chapter, where the input voltage conditions set  $C_{eff}$  to be much different from  $C_{load}$ . Propagation delay errors are about five times less while using  $C_{eff}$  as the macromodel parameter instead of using  $C_{load}$  as the macromodel parameter.

## 6.5 Macromodel Output Functions

Figure 6-8 shows the linear circuit equivalents for a transistor driver cell in MOS technology. The value of a circuit element enclosed in a dashed box is a macromodel function. The complexity of the circuit differs depending on the order of the analysis. A higher ordered linear circuit equivalent yields a better fit to the true circuit response. Figure 6-9 demonstrates one instance of this by showing simulation waveforms of an inverter and of the three transistor circuits from Figure 6-8. The second order fit is very close in this example.

Circuit elements of the linear circuit equivalents are described below:

- The no-load voltage source,  $V_{nl}$ , equals the open-circuit cell output voltage.  $\tilde{V}_{nl}(s)$  is a moment representation with constant  $M_0$  and macromodeled  $t_0, M_1, M_2, M_3, \dots$ . These macromodels are one-dimensional functions of the input voltage slope,  $t_r, v_{in}$ . It is often the case with all but the fastest inputs that  $V_{nl}$  traces out the d.c. transfer function between input and output.
- The d.c. resistance,  $R_{dc}$ , is the incremental resistance observed at the output after the transition is completed. That is,

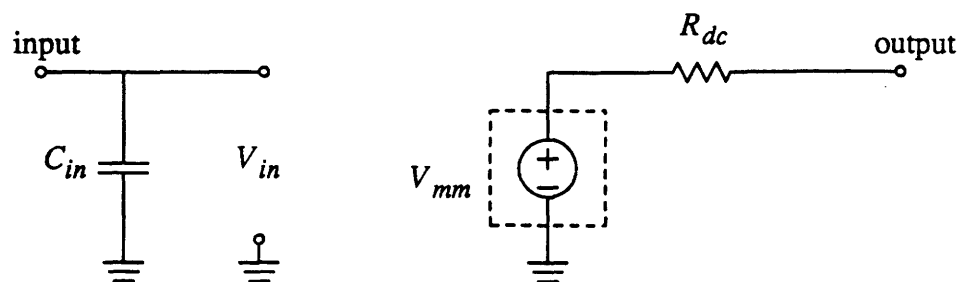
$$R_{dc} = \left. \frac{\partial V_{out}}{\partial I_{out}} \right|_{V_{out}}$$

where  $V_{out} = V_{low}$  for a falling transition and  $V_{out} = V_{high}$  for a rising transition.

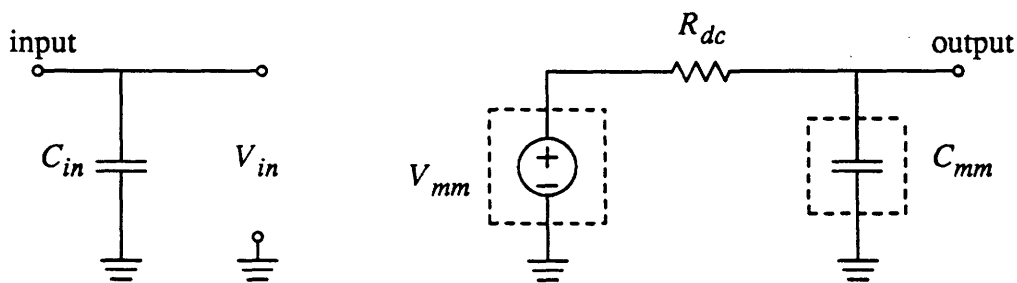
- The  $L_{mm}$  and  $C_{mm}$  circuit elements adjust the cell's output admittance during the transition. Their values are macromodeled functions of  $t_r, v_{in}$  and  $C_{eff}, Y_{load}$ .

These macromodel functions and scalar values model the cell output given any input or load condition in the desired range. These functions and scalars (summarized in Table 6-1) form a driver cell *macromodel set*. A different macromodel set is needed for each transition type caused by a different set of input conditions. For instance, an inverter has one set for a falling output transition and another set for a rising output transition. A full adder circuit may need as many as 24 macromodel sets for the *sum* output signal and 12 macromodel sets for the *carry* output signal. Figure 6-10 shows a full adder Karnaugh map, and marks each input transition that causes a different output transition. Depending on circuit implementation, some macromodel sets may be shared, if they are equal.

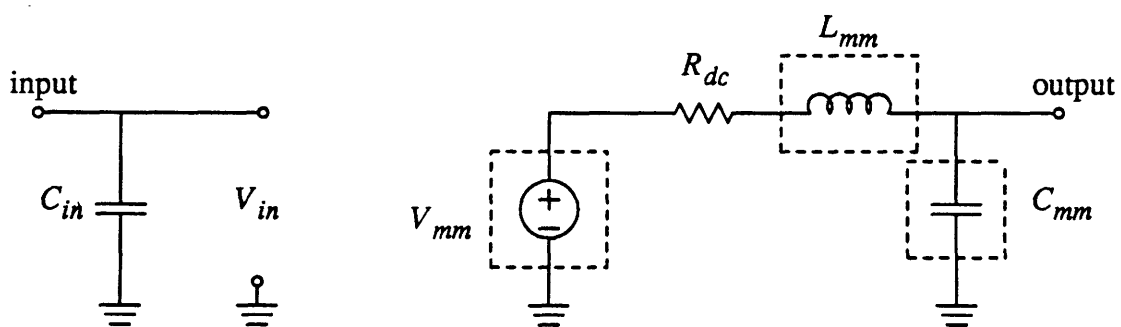
Embedded in the moment representation simulator is a logic simulator. Each cell has a procedure to calculate the output logic states. Any change in an output state causes the output transition waveform to be computed with a specific macromodel set. The operation of the logic simulator is described by Brocco [26], which also discusses the issues of conflicting and overlapping transitions.



(D.C. resistance approximation)



(First-order approximation)



(Second-order approximation)

FIGURE 6-8: Transistor driver cell linear circuit equivalents.

Elements in dashed boxes are not constant, and depend on input waveform and output load.

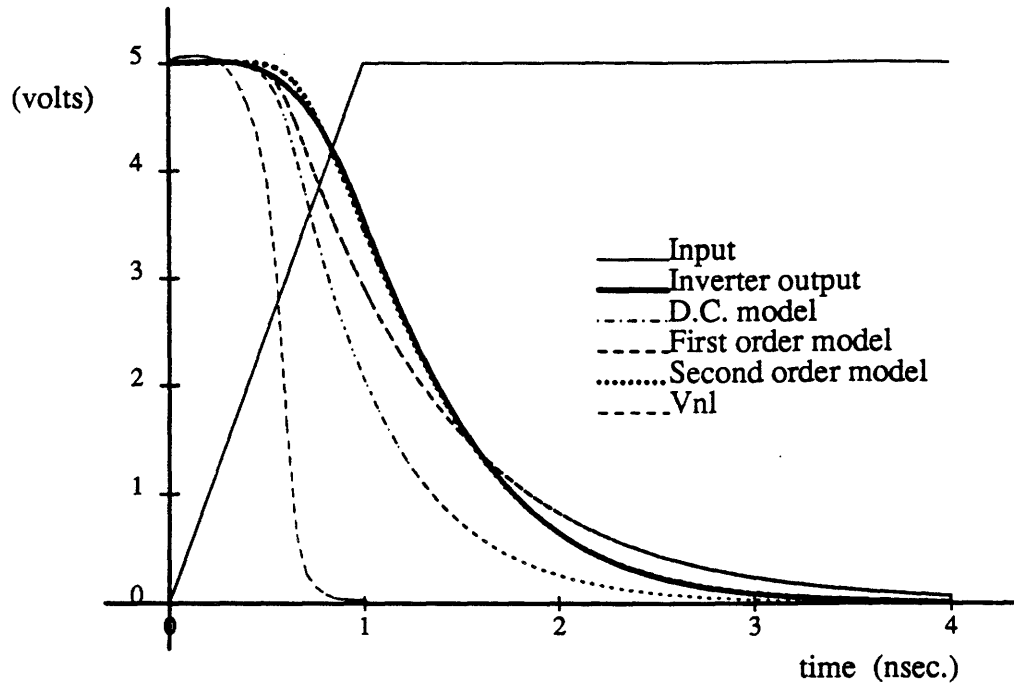


FIGURE 6-9: SPICE simulation of true inverter and linear circuit equivalents.

$t_{0,nl}(t_r, V_{in})$
$M_{0,nl}$
$M_{1,nl}(t_r, V_{in})$
$M_{2,nl}(t_r, V_{in})$
$M_{3,nl}(t_r, V_{in})$
$R_{dc}$
$C_{mm}(t_r, V_{in}, C_{eff}, Y_{load})$
$L_{mm}(t_r, V_{in}, C_{eff}, Y_{load})$

Table 6-1: Macromodel functions and scalar values of a third-order macromodel set.

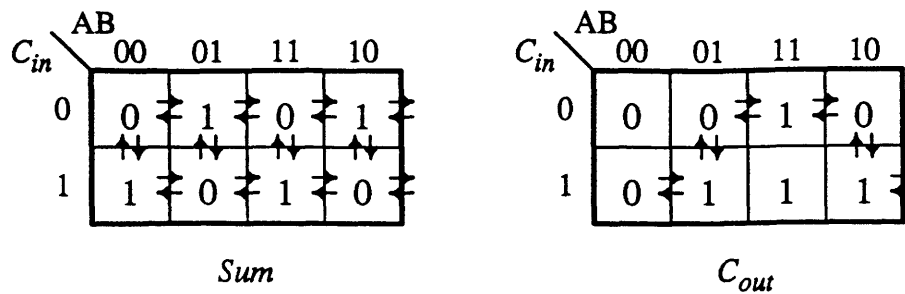


FIGURE 6-10: Full Adder Karnaugh map with output transition possibilities.

Each arrow represents a macromodel set for an output transition. The sum function has 24 macromodel sets—the maximum for a 3 input combinational function. The carry function has 12 macromodel sets.

## 6.6 Extracting Macromodel Function Values

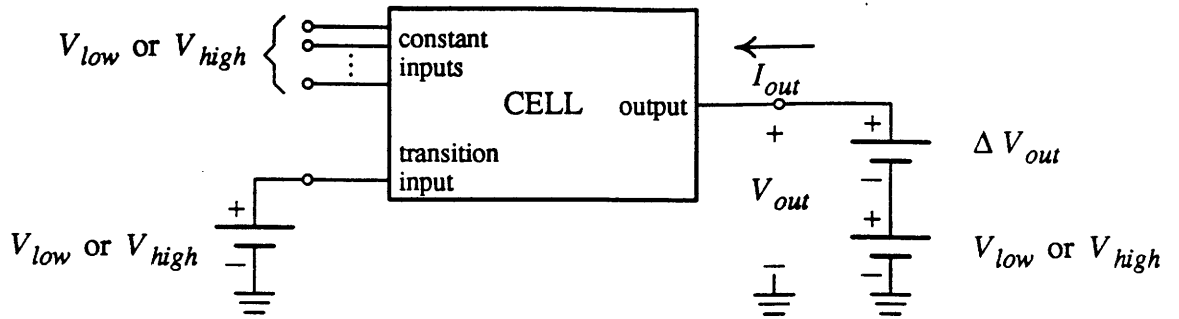
This section describes the procedure for calculating values in a driver macromodel set. Computing macromodel numbers is called *macromodel extraction*. The amount of computation needed for macromodel extraction is not a critical concern, since this is done once per cell. The computation may, in fact, be considerable. This will be detailed later.

Macromodel numbers are found from any source of transistor cell waveform information. Namely, the macromodels can be extracted from waveforms generated from SPICE simulations of the cell, or from waveforms measured on the real cell circuit, if possible. All macromodel functions presented in this thesis derive from SPICE simulations.

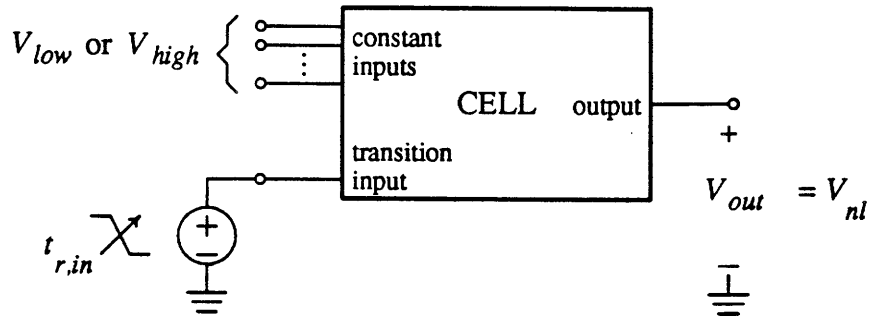
Test circuit (a) of Figure 6-11 extracts the scalar value for  $R_{dc} = \Delta V_{out} / \Delta I_{out}$ . This test circuit is done under static conditions.

The procedure for extracting the macromodel functions is to conduct *macromodel extraction tests* with closely spaced increments of macromodel parameter data points. Test circuit (b) of Figure 6-11 extracts  $V_{nl}(t)$  waveforms of a transistor driver cell. The response at  $V_{out}$  is measured when transitions with different  $t_{r,in}$  are applied at the input. It has been observed that the input waveform shape which gives the best overall matching to true waveforms is a combination of a ramp and exponential tail, as shown in Figure 6-12. The output waveform moments are extracted from the time domain waveforms as described in Section 3.5. The macromodel function  $t_{0,V_{nl}}(t_{r,in})$  is defined as the time delay between  $t_{d,in}$  of the input waveform and  $t_0$  of the output waveform.

Test circuit (c) of Figure 6-11 extracts the admittance macromodel parameters,  $C_{mm}$  and  $L_{mm}$ . These tests are performed with different combinations of input signal slope and load capacitance. The output load is a pure capacitance in the tests, so  $C_{eff} = C_{load}$ . For each test with specific load capacitance and input waveform slope, the measured output waveform

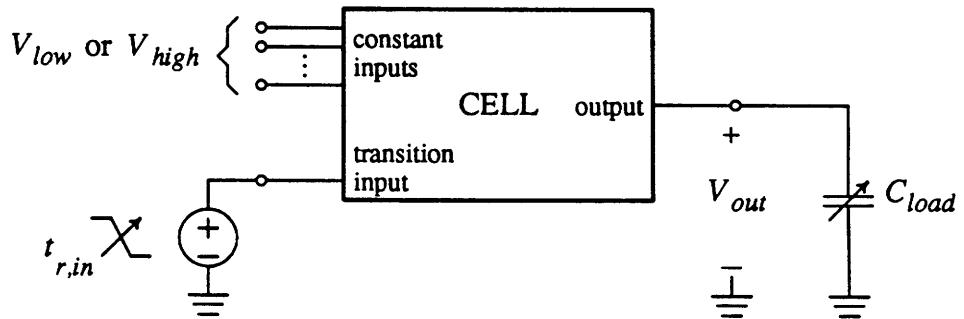


(a) Test circuit for extracting  $R_{dc}$ .



(b) Test circuit for extracting  $\tilde{V}_{nl}(s) =$

$$e^{t_0, nl(t_r, v_{in})} \left( M_{0, nl} - M_{1, nl}(t_r, v_{in}) s + M_{2, nl}(t_r, v_{in}) \frac{s^2}{2!} - \dots \right).$$



(c) Test circuit for extracting  $C_{mm}(t_r, v_{in}, C_{eff}, Y_{load})$  and  $L_{mm}(t_r, v_{in}, C_{eff}, Y_{load})$ .

FIGURE 6-11: Macromodel extraction test circuits.

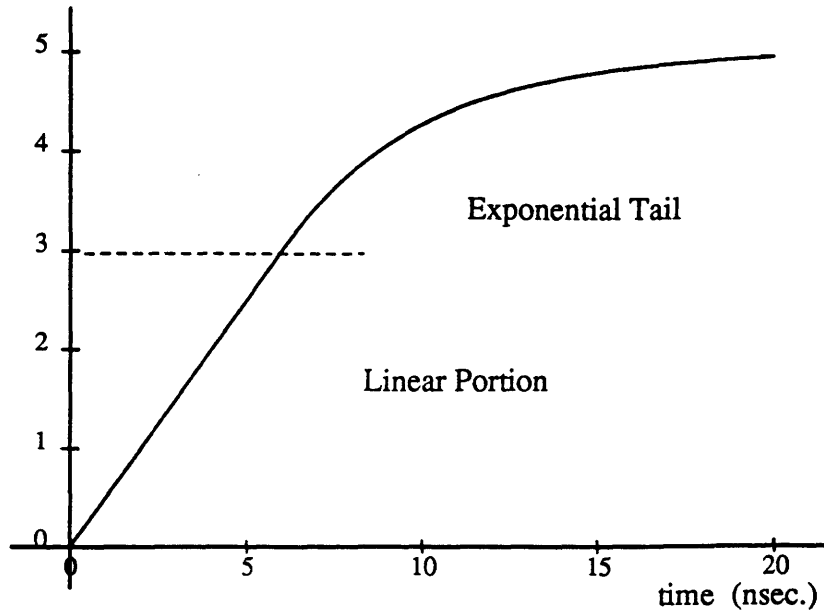


FIGURE 6-12: Test input waveform.

is converted from a the time domain into a moment representation. Values for  $C_{mm}$  and  $L_{mm}$  are found such that the second order moment representation of  $\tilde{V}_{out}(s)$  is the same for the cell test circuit and the second order linear equivalent circuit in Figure 6-8. For this to be true,

$$\tilde{V}_{out}(s) = \tilde{V}_{model}(s) = \frac{\tilde{V}_{ni}(s)}{1 + R_{dc}(C_{mm} + C_{load})s + L_{mm}(C_{mm} + C_{load})s^2}.$$

In terms of moment representations (with  $e^{-s t_0}$  suppressed),

$$M_{0,V_{out}} - M_{1,V_{out}}s + M_{2,V_{out}}\frac{s^2}{2} = \frac{M_{0,V_{ni}} - M_{1,V_{ni}}s + M_{2,V_{ni}}\frac{s^2}{2}}{1 + R_{dc}(C_{mm} + C_{load})s + L_{mm}(C_{mm} + C_{load})s^2}.$$

This equation necessarily means that  $M_{0,V_{out}} = M_{0,V_{ni}}$  as we expect. Solving for  $C_{mm}$  and  $L_{mm}$ ,

$$C_{mm} = \frac{\Delta M_1}{R_{dc}} - C_{load}, \quad (6.3)$$

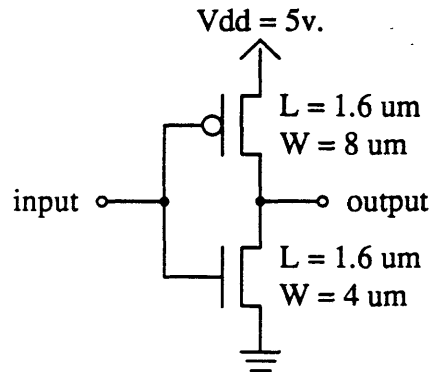
$$L_{mm} = \frac{M'_{1,out} - \Delta M_1 - \Delta M_2}{C_{mm}}, \quad (6.4)$$

where

$$\begin{aligned} M'_{1,out} &= M_{1,V_{out}}/M_0, \\ \Delta M_1 &= \frac{M_{1,V_{out}} - M_{1,V_{ni}}}{M_0}, \text{ and} \\ \Delta M_2 &= \frac{M_{2,V_{out}} - M_{2,V_{ni}}}{2M_0}. \end{aligned}$$



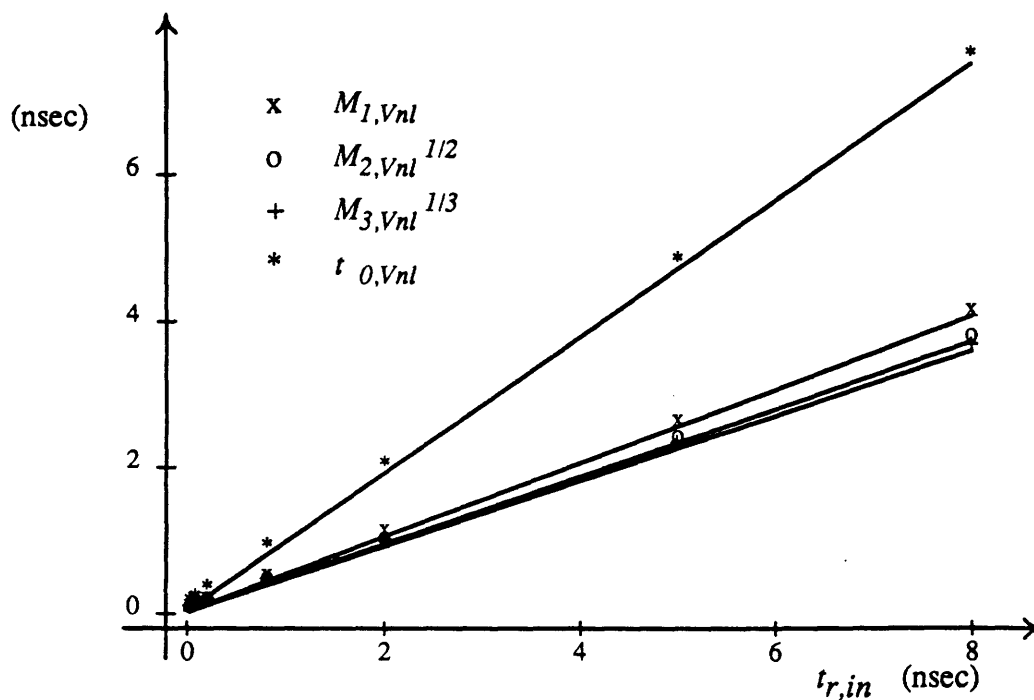
**Example 6.2** Plot macromodel functions for the following CMOS inverter for a falling output.



Using SPICE<sup>5</sup> for values of  $C_{eff} = 0.1, 0.2, 0.5, 1.0,$  and  $2.0$  pF, and  $t_{r,in} = 0.02, 0.08, 0.2, 0.8,$   $2.0, 5.0$  and  $8.0$  sec/v., a total of 35 simulations, the following macromodel set is derived. The scalar values are:  $M_0 = -5v., R_{dc} = 961\Omega.$

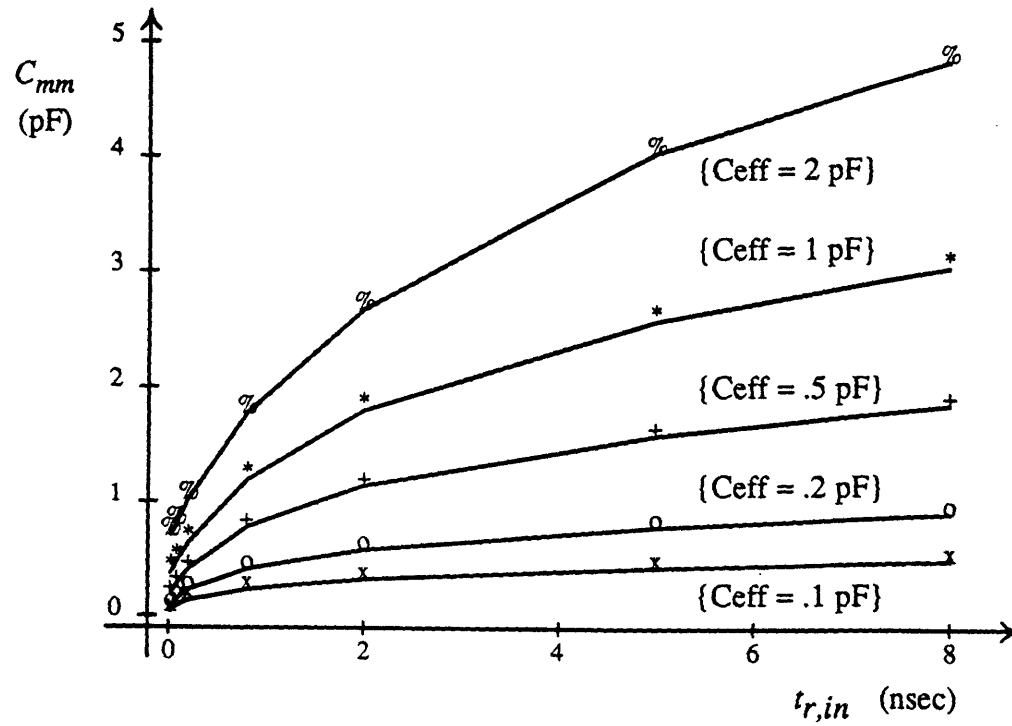
<sup>5</sup>n-channel SPICE parameters:  $V_{t0} = 0.75v., \mu_0 = 520 \text{ cm}^2/v \cdot \text{sec}, \gamma = 0.16 \text{ v}^{1/2}, t_{ox} = 250\text{\AA}, X_j = 0.2\mu\text{m},$   
 $L_d = 0.16\mu\text{m}, N_{sub} = 1 \times 10^{16} \text{ cm}^{-3}, C_j = 0.23 \times 10^{-3} \text{ pF}/\mu\text{m}^2, C_{jsw} = 0.2 \times 10^{-3} \text{ pF}/\mu\text{m}, C_{gs0} = C_{gd0} =$   
 $2.2 \times 10^{-4} \text{ pF}/\mu\text{m}, R_{sh} = 52.5\Omega.$

p-channel SPICE parameters:  $V_{t0} = -0.75v., \mu_0 = 190 \text{ cm}^2/v \cdot \text{sec}, \gamma = 0.55 \text{ v}^{1/2}, t_{ox} = 250\text{\AA}, X_j = 0.25\mu\text{m},$   
 $L_d = 0.2\mu\text{m}, N_{sub} = 1 \times 10^{16} \text{ cm}^{-3}, C_j = 0.67 \times 10^{-3} \text{ pF}/\mu\text{m}^2, C_{jsw} = 0.6 \times 10^{-3} \text{ pF}/\mu\text{m}, C_{gs0} = C_{gd0} =$   
 $2.8 \times 10^{-4} \text{ pF}/\mu\text{m}, R_{sh} = 120\Omega.$

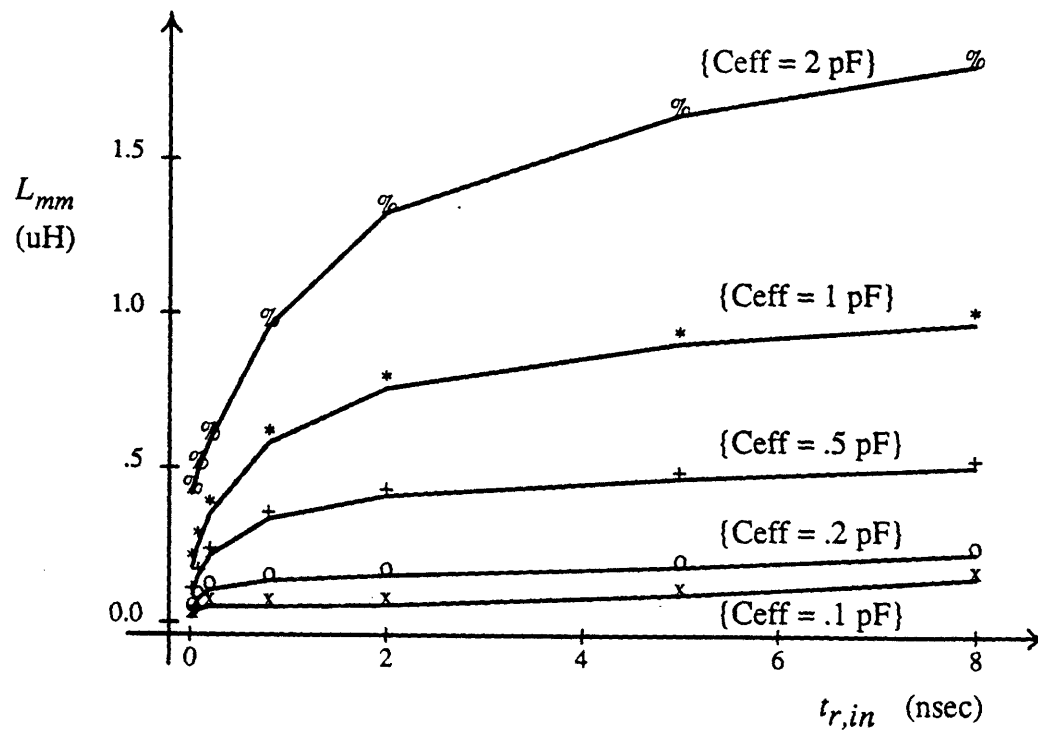


$V_{nl}$  macromodel functions vs. input transition time.

The square root and cube root of  $M_2$  and  $M_3$  are stored as the macromodel functions respectively. This makes all units be time, and keeps the macromodel functions closer to linear.

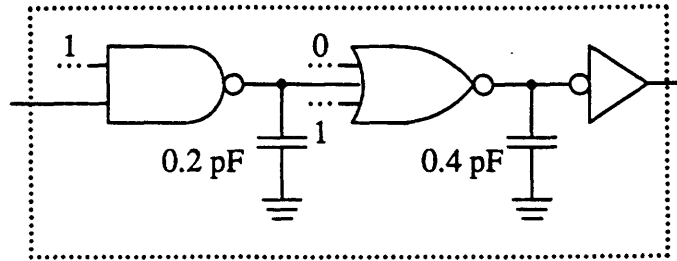


$C_{mm}$  vs. input transition time and  $C_{eff}$ .

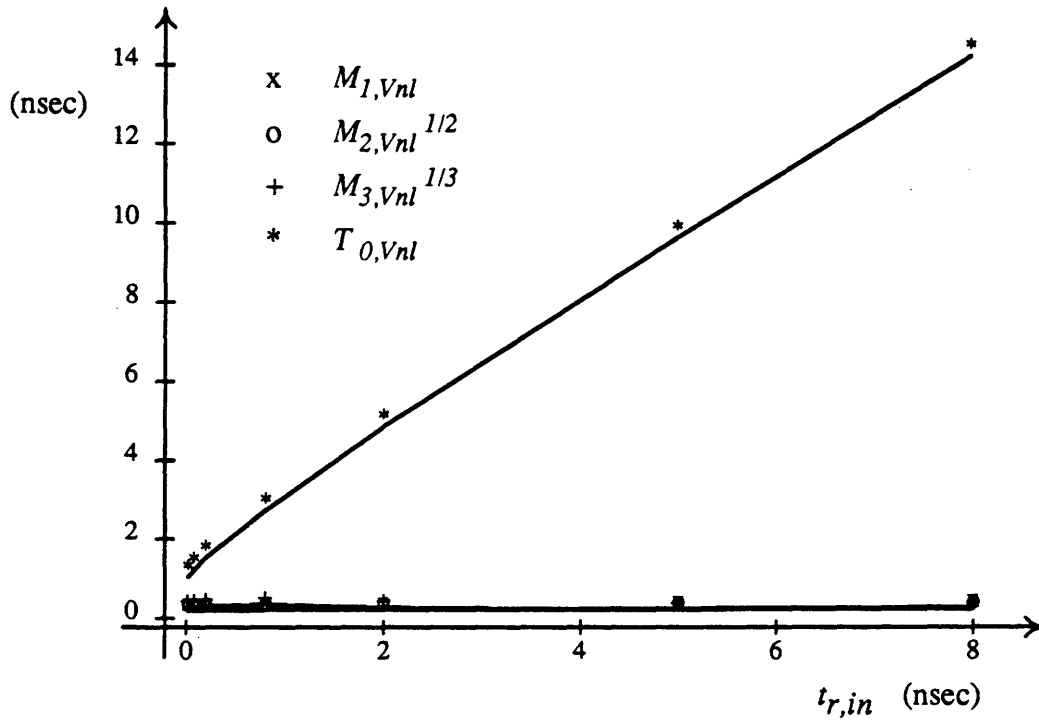


$L_{mm}$  vs. input transition time and  $C_{eff}$ .

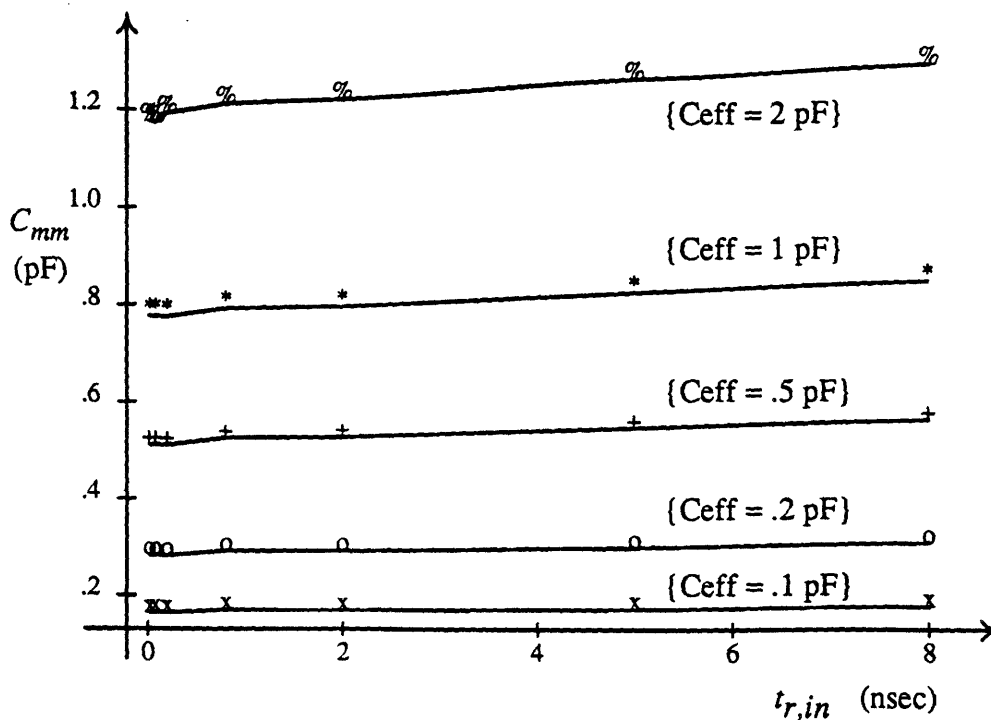
**Example 6.3** Plot macromodel functions for the shown path of the following CMOS circuit.



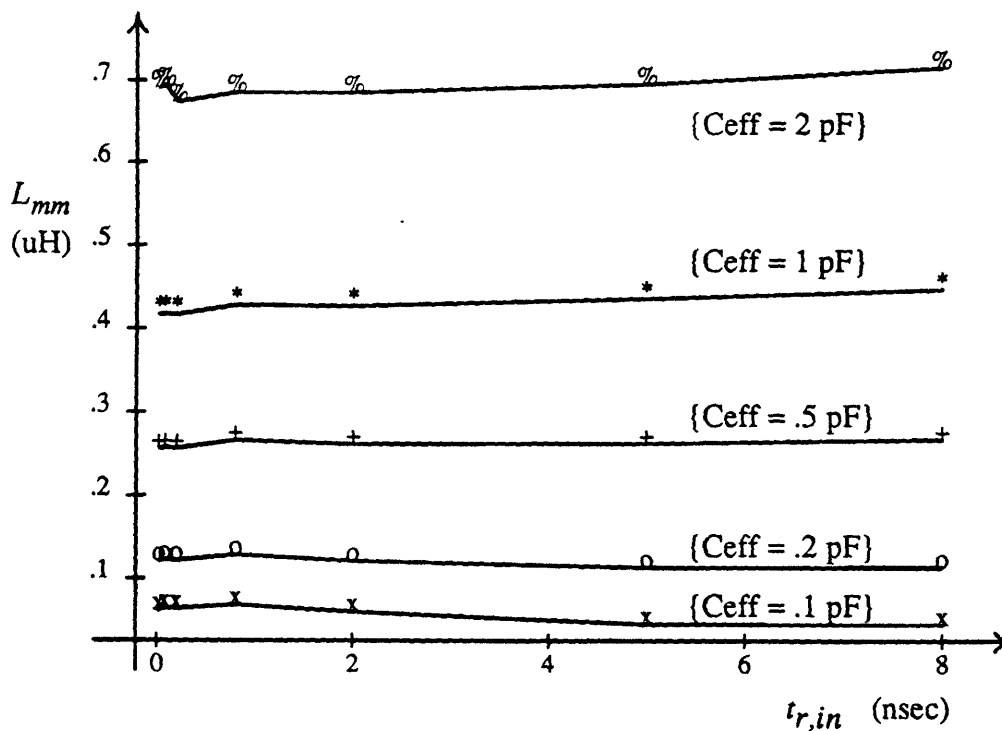
SPICE simulations using the same transistor models and macromodel input parameter data points give the following functions. The scalar values are:  $M_0 = -5v.$ ,  $R_{dc} = 961\Omega$ .



$V_{nl}$  macromodel functions vs. input transition time.



$C_{mm}$  vs. input transition time and  $C_{eff}$ .



$L_{mm}$  vs. input transition time and  $C_{eff}$ .

<i>order</i>	<i>maximum floating-point numbers</i>	<i>minimum floating-point numbers</i>
1	77	18
2	140	30
3 <sup>7</sup>	150	34

Table 6-2: Number of single precision floating point numbers for a macromodel set.

From the above two examples, we can make these interesting observations:

- In the first example, the  $\tilde{V}_{nl}(s)$  macromodel functions are very close to linear functions of time. This is to be expected since  $V_{nl}(t)$  traces out the same d.c. transfer function at linearly increasing rates.
- In the second example, the  $M_{1,V_{nl}}$ ,  $M_{2,V_{nl}}$ , ... macromodel function values are small, indicating  $V_{nl}(t)$  is approximately a voltage step. Also,  $t_{0,V_{nl}}$  is much larger for the multi-stage macromodel than for the inverter macromodel.
- In the second example, admittance macromodel functions are independent of the input slope. This observation and the last observation are true because there is significantly more restoring logic between the input and output.

### 6.6.1 Macromodel Extraction CPU Time

The CPU time required to extract a complete macromodel set depends heavily on the cell's circuit complexity. The macromodel examples in this chapter require approximately 5–40 minutes for SPICE simulation on an HP 9000/350 workstation<sup>6</sup> per macromodel set. The same workstation required less than one-half minute to extract the macromodel functions for one macromodel set from the simulation waveforms. To extract all macromodel sets for a cell may require several hours of CPU time. The SPICE simulation and macromodel extraction is done automatically, and is controlled by UNIX's "make" program facility.

### 6.6.2 Macromodel Memory Requirements

Macromodel functions are stored as one-dimensional and two-dimensional arrays. parameter data point values are stored in one-dimensional arrays, and index the macromodel function tables. Each third-order macromodel set takes no more than about 150 floating point numbers for all macromodel functions and data points (see Table 6-2).

<sup>6</sup>The HP 9000/350 has approximately a 3 MIPS processor.

<sup>7</sup>With 2<sup>nd</sup> order admittance model.

For very linear functions, the size of a macromodel set can be reduced automatically with a macromodel trimming program. The trimming program tries to locate two macromodel values that are separated by as much as possible, provided all intermediate macromodel points are offset from the line between the end-points by no more than a prespecified fraction (e.g., 1%) of the total macromodel function range. This type of trimming can considerably reduce the macromodel function array size in Example 6.3, for instance.

The size of a macromodel set is further reduced in some cases by raising the macromodel parameters to a power. This linearizes some macromodel functions that are not originally linear. For instance, the  $C_{mm}$  function of Example 6.2 can be approximately linearized as shown in the three-dimensional plots of Figure 6-13. The procedure for this is automatic, and is outlined below:

1. Find logarithms of macromodel scale values.
2. Perform a linear regression on the macromodel functions with the logarithm scale.
3. Find function slope with linear regression, and raise scale values by the slope.
4. store the slope values in the macromodel set, so future macromodel functions calls use the same parameter.

For the inverter of Example 6.2, two macromodel sets are needed, totalling to about 220 floating point numbers or 0.86 KByte. In comparison, a three input, full adder needs about 3300 floating point numbers or about 3.2 Kbyte. While it is true, in general, that as the number of cell inputs and internal logic states increases, the internal logic complexity increases and the macromodel sets become simpler; it is also true that the number of macromodel sets increases exponentially, effectively limiting the number of cell inputs to around a dozen.

## 6.7 Transistor Transmission Cells

Transistor transmission cells are used to model the non-linear effects of isolated transmission gates or pass transistors. Their prevalence in MOS circuits makes this an important cell type. Brocco [26] was the first to incorporate transmission gate macromodeling into  $RC$  tree solutions. Once again, these methods are joined into the moment representation simulation methods. The macromodel parameters are basically the same. The principle difference is in the waveform and circuit representation. The moment representation allows for a more accurate solution, since the methods in [26] are effectively a first-order macromodel and a second order linear network solution.

A large number of issues must be addressed to correctly model MOS transmission gates properly. The reader should consult [26] or [56] for a thorough understanding of these. These

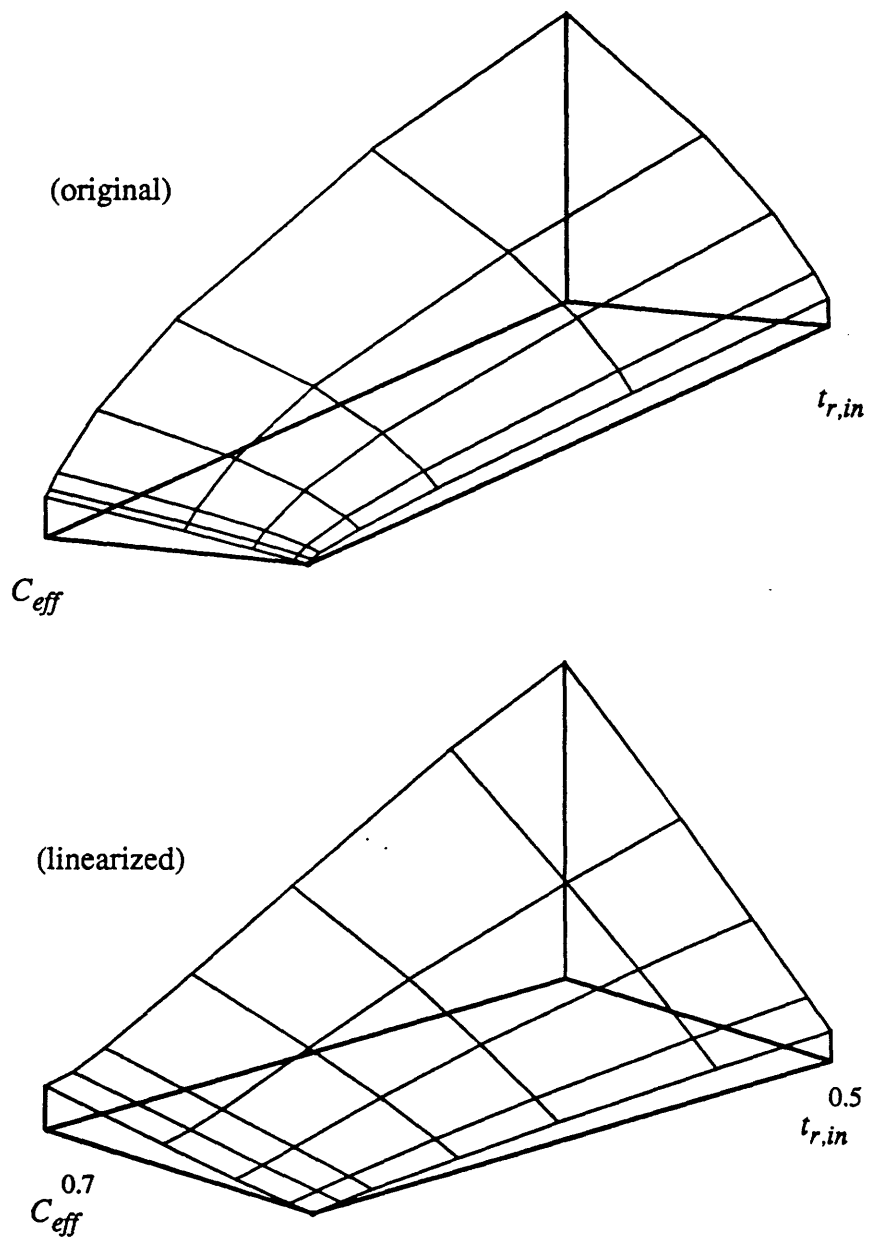


FIGURE 6-13: Approximate linearization of a macromodel function.



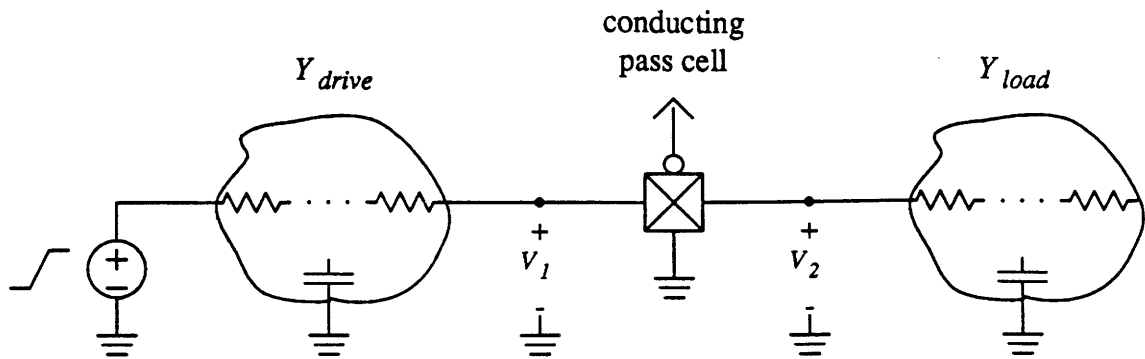


FIGURE 6-14: Conducting transmission cell

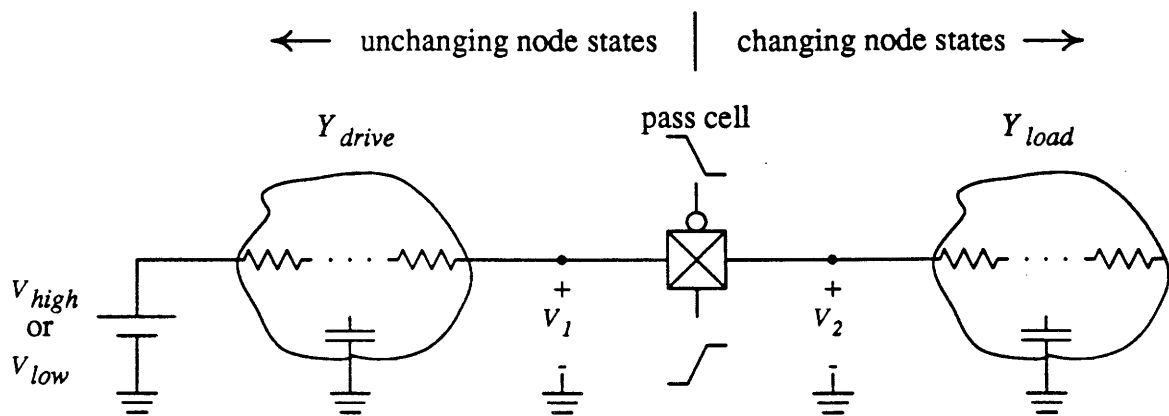


FIGURE 6-15: Switching transmission cell

issues are discussed briefly here, except where significant differences arise in using the moment representation.

Solutions for transistor transmission cell simulations separate into two distinct problem types depending on where the input signal transition is applied to the circuit. In a *conducting transmission cell* problem the transition is applied to the driving gate as shown in Figure 6-14; in a *switching transmission cell* problem the transition is applied to the controlling inputs of the transmission gate as shown in Figure 6-15. The two problem types are discussed separately below.

### 6.7.1 Conducting Transmission Cell

The d.c., first and second order linear circuit equivalent models for a conducting transmission cell are shown in Figure 6-16. They are, in short, just the admittance portion of the linear circuit equivalents for a transistor driver cell.

The macromodel parameters for the linear circuit elements are, once again, a signal tran-

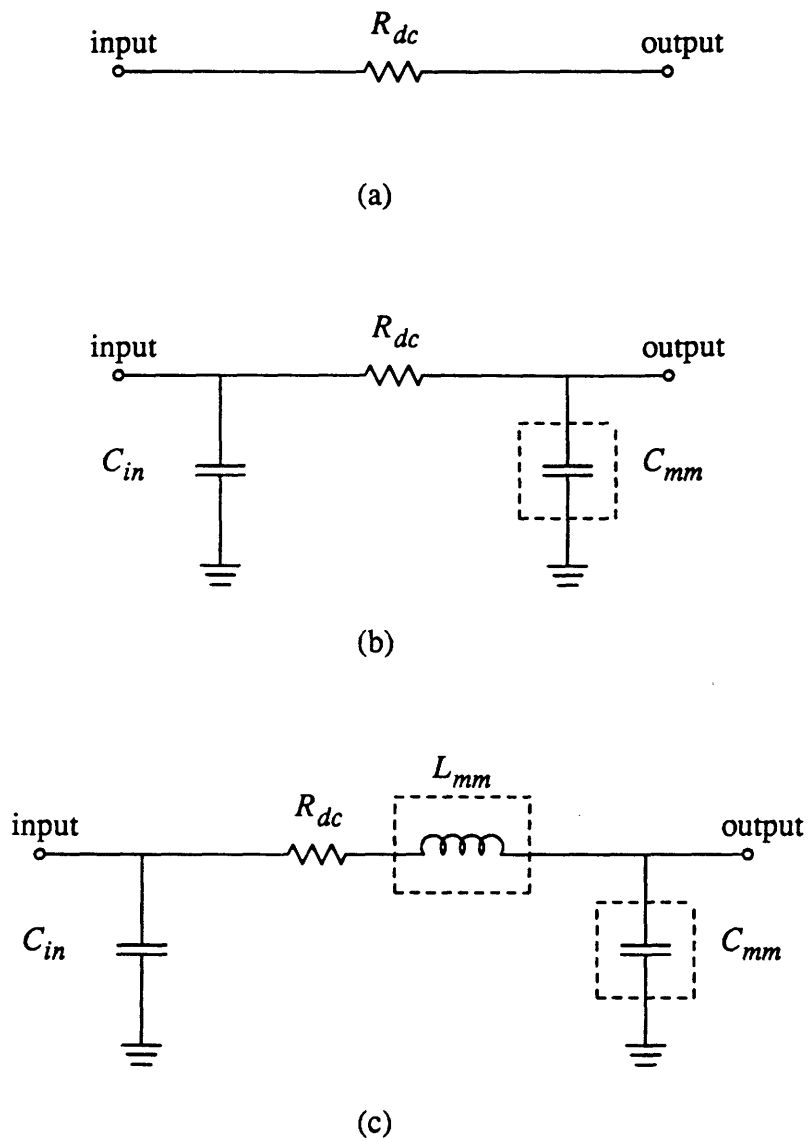


FIGURE 6-16: Linear circuit equivalents for a conducting transmission cell

Elements in dashed boxes are not constant, and depend on input waveform and output load.

sition time and an effective load capacitance. The procedure for computing the parameters is described below.

- A precursory, first order simulation is performed on the sub-network with  $R_{dc}$  substituted for the transmission cell (Figure 6-16(a)). This gives approximate waveforms at both terminals of the transmission cell.
- The input transition time is the time required for the signal,  $V_1(t)$ , on the driving end of the transmission cell to traverse through a critical region. The actual parameter is defined, as in (6.1) and Figure 6-5 as

$$t_r = \left| \frac{t_{cr2} - t_{cr1}}{V_{cr2} - V_{cr1}} \right|$$

- The effective load capacitance is defined for  $Y_{load}$  in the same way as for an driver gate, and as described in Section 6.4. The behavior through the critical region of  $V_2(t)$  determines the value of  $C_{eff}$ .
- For both parameters, critical voltages at 0% and 75% of the total transition are used.

Fewer macromodel functions are needed for transmission cells than for driver cells. For a conducting transmission cell, these are:

$$\begin{array}{c} R_{dc} \\ C_{mm}(t_r, V_1, C_{eff}) \\ L_{mm}(t_r, V_1, C_{eff}) \end{array}$$

To determine the macromodel functions, experiments are performed on the circuit in Figure 6-17. As with transistor driver cells, the experiments can be done with SPICE or with real transmission gate circuits, and the input signal slope and load capacitance are stepped through the range of expected values for circuit operation. The method for computing the macromodel function values from experimental waveforms is the same as for a transistor driver cell, and is described in Section 6.6.

### 6.7.2 Switching Transmission Cell

Switching transmission cells can be difficult to model. Single transistor transmission cells which are typically found in NMOS circuits are more easily modeled than the more complex two transistor cells of CMOS circuits. Single transistor transmission cells are modeled with the linear circuit equivalent models shown in Figure 6-18. The switch closes at time  $t_0$ , where  $t_0$  depends on the input waveform,

$$t_0 = t_{0,mm}(t_r, V_c) + t_{d,V_c}.$$

A non-zero d.c. voltage on  $V_{th}$  delineates a threshold drop on the pass transistor for the particular transition. The other circuit elements model the admittance properties of the pass transistor

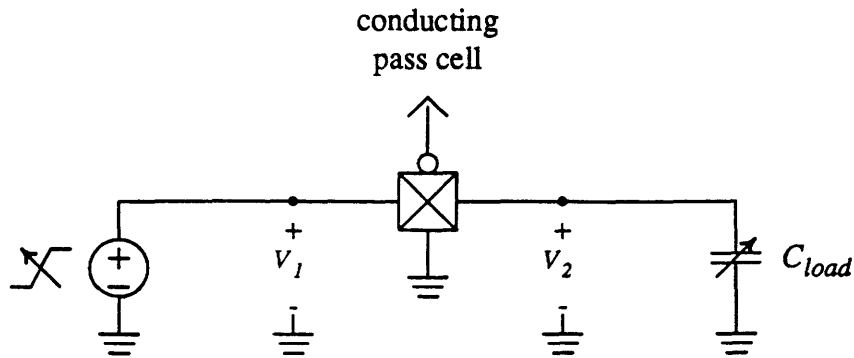


FIGURE 6-17: Conducting transmission cell macromodel extraction circuit.

and are macromodeled depending on the input waveform and the effective capacitance of the load.

The two transistor transmission gates common in CMOS circuits have two independent, controlling inputs, adding to the complexity of their models. If both inputs arrive simultaneously, concerns would be minimized, but this does not usually happen. Three approaches can be taken to model these cells:

1. Use the same linear models for the one transistor transmission cell, and increase the number of macromodel parameters to four:  $C_{eff}$ ,  $t_{r,V_{c,n}}$ ,  $t_{r,V_{c,p}}$  and  $t_{d,V_{c,n}} - t_{d,V_{c,p}}$ .
2. Compute approximate circuit values for the single transistor models based on the easier-to-calculate response that the transmission cell would have if it were driving just a single load capacitor. (This approach is used in [26].)
3. Solve the entire network for both combinations of switch positions, and combine the waveforms.

The first method is rejected because of the large macromodel function size and the difficult macromodel extraction which would be needed. The second approach is rejected because the time domain responses are difficult to obtain with the more complex, higher-order moment representation circuits. The third approach has been selected for this thesis. It operates with the following sequence:

- When the first signal arrives (at  $t = t_1$ ), solve the network with one path conducting for the given transition, as shown in Figure 6-19(a). Find waveforms for the desired output nodes. We will designate one such waveform as  $V_{sw,1}(t)$ .
- When the second signal arrives (at  $t = t_2$ ), solve the network with both paths conducting, as shown in Figure 6-19(b). The resulting waveforms are designated as  $V_{sw,2}(t)$ .

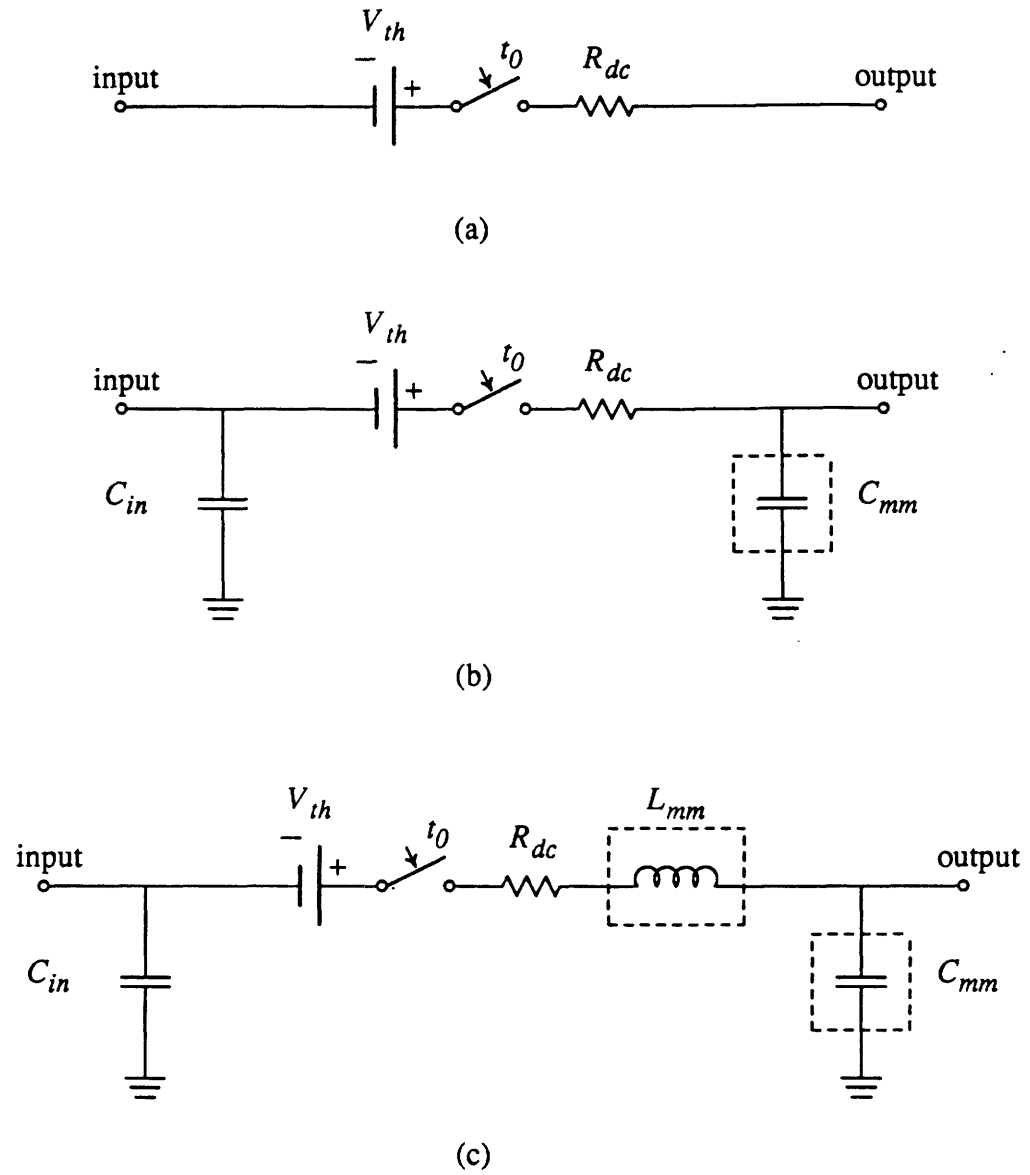


FIGURE 6-18: Linear circuit equivalents for a one-transistor switching transmission cell.

Elements in dashed boxes are not constant, and depend on input waveform and output load.

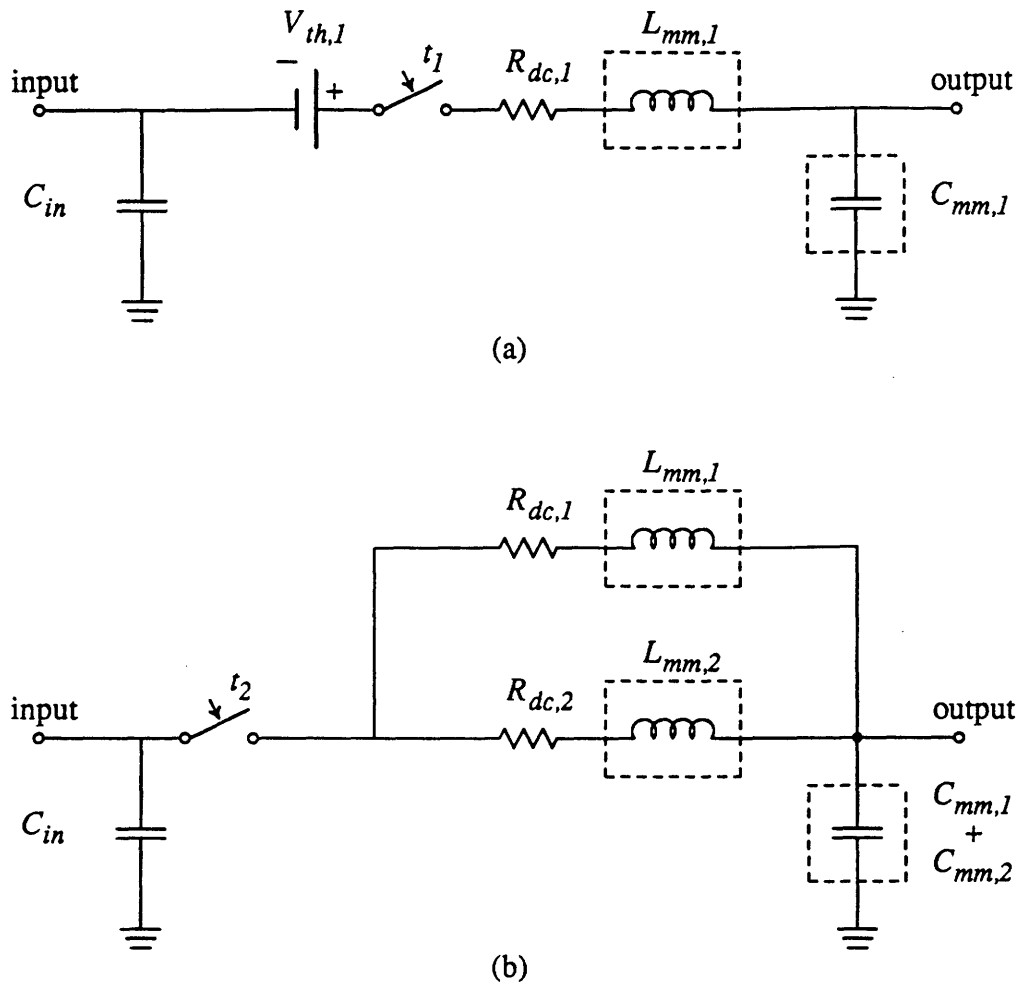


FIGURE 6-19: Second order linear circuit equivalents for a two-transistor switching transmission cell.

- Combine the two waveforms:

$$V_{total}(t) = \begin{cases} V_{sw,1}(t) & \text{for } t_1 \leq t \leq t_2, \\ V_{sw,1}(t_2)V_{sw,2}(t) & \text{for } t \geq t_2 \end{cases}$$

The linear circuit element macromodel parameters are the signal transition time of the transmission cell's controlling inputs,  $T_r, V_c$ , and the effective load capacitance,  $C_{eff}, Y_{load}$ .

The following macromodel functions are needed for each switching transmission cell macromodel set.

$$\begin{aligned} & t_0(t_r, V_c) \\ & R_{dc} \\ & C_{mm}(t_r, V_c, C_{eff}) \\ & L_{mm}(t_r, V_c, C_{eff}) \end{aligned}$$

These macromodel function values are determined with the experimental test circuit shown in Figure 6-20. In all extraction tests the input transition turns the transmission cell "on". For

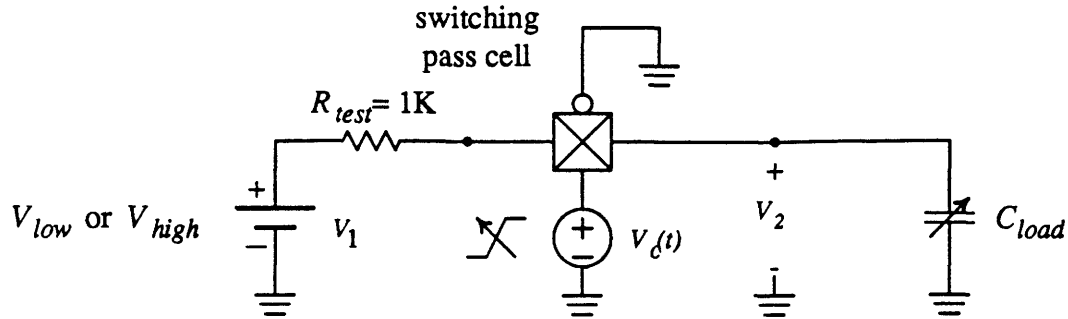
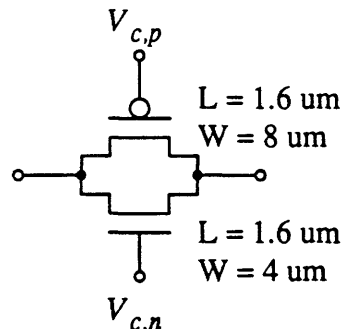


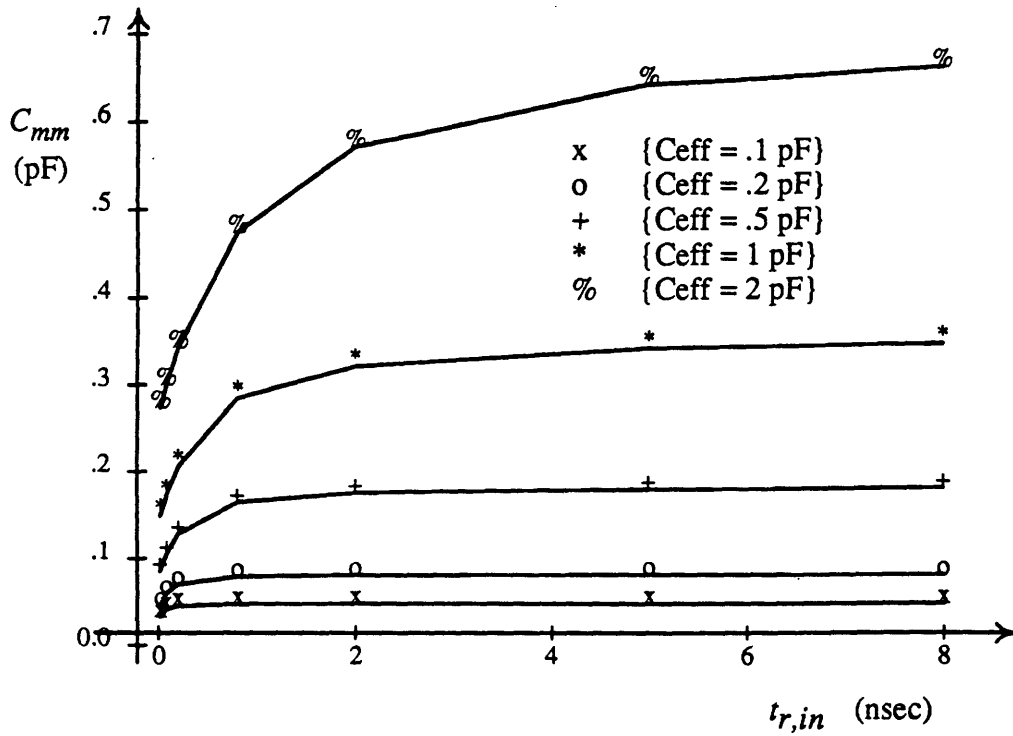
FIGURE 6-20: Test circuit for extracting switching transmission cell macromodel values.

circuits with two controlling inputs, like the one shown, the path not being tested is “off”. For a one-transistor transmission cell, two macromodel sets are needed to cover the two driving logic state possibilities. For a two-transistor transmission cell, four macromodel sets are needed to cover all combinations of driving logic state, and transistor path.  $R_{drive}$  exists in the circuit to approximate the effects of the driving circuit. It’s value has little effect on the macromodel function, and is not critical.

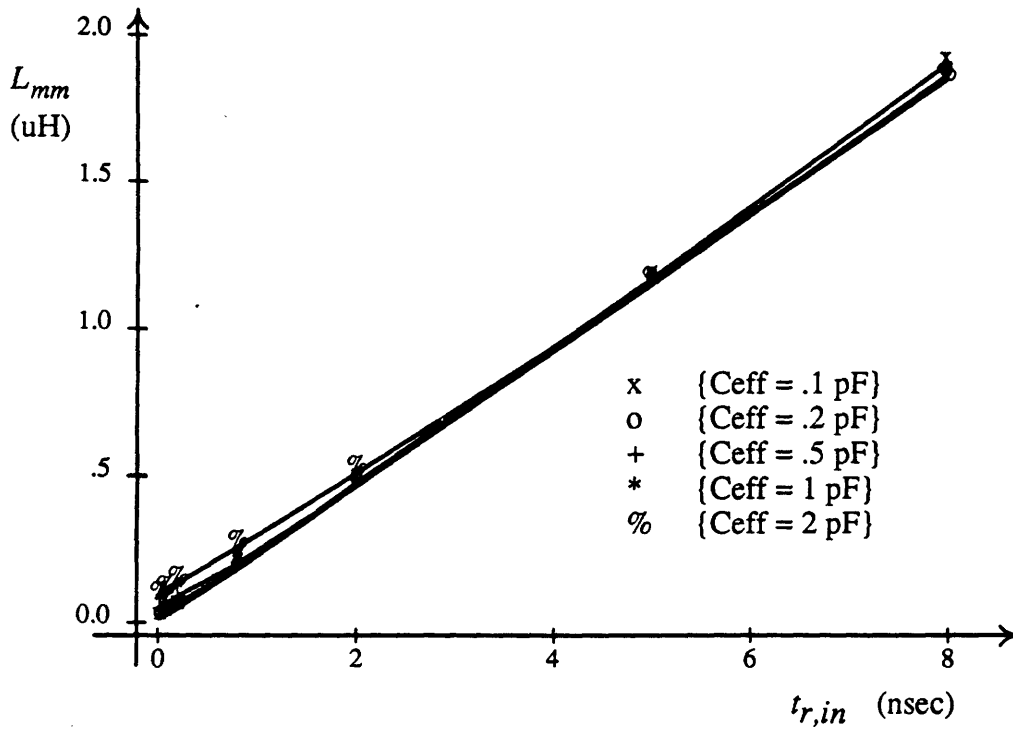
During extraction, this circuit is treated as a regular driver cell, where  $V_c(t)$  is the transition input. Macromodel function values are extracted as described in Section 6.4. Only  $t_{0,V_{nl}}(t_r, V_C)$  is retained from the  $V_{nl}$  functions.

**Example 6.4** Plot macromodel functions for the following CMOS transmission gate for the conducting case using the SPICE transistor parameters listed in Example 6.2.





$C_{mm}$  vs. input transition time and  $C_{eff}$ .



$L_{mm}$  vs. input transition time and  $C_{eff}$ .



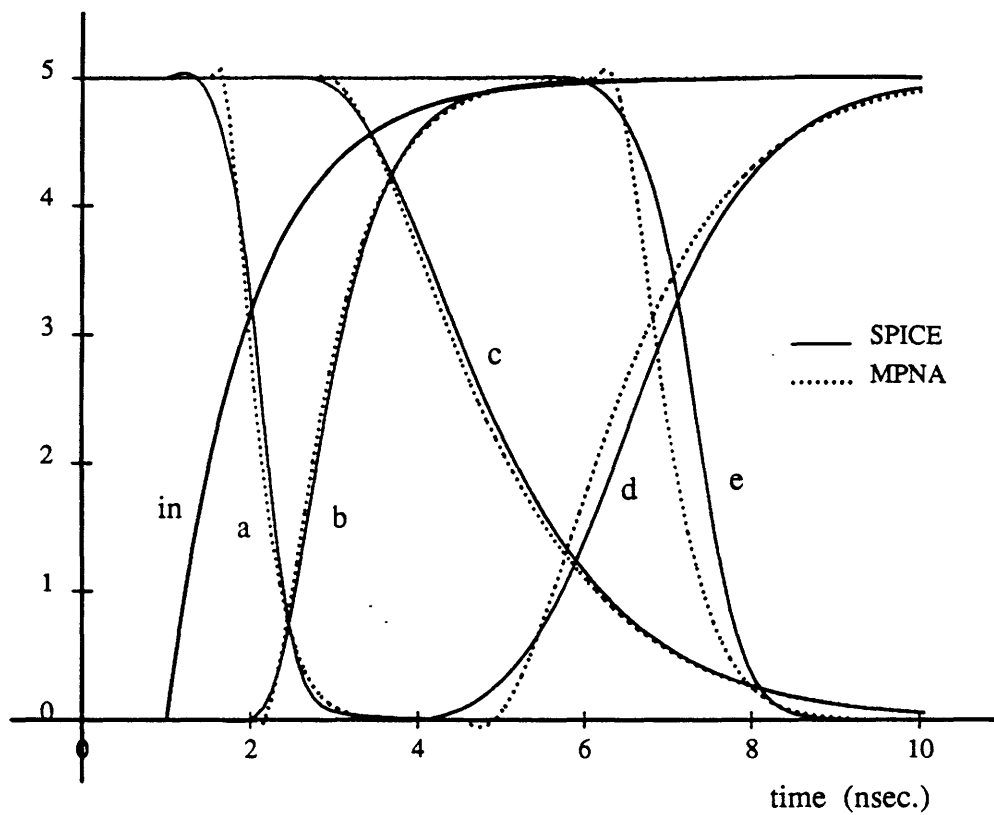
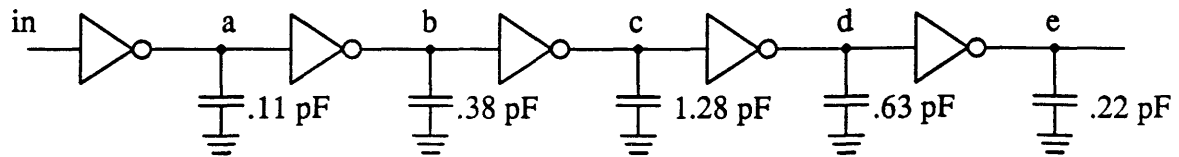
## 6.8 Examples

This section contains several examples, showing results of the macromodeling methods described in this chapter. These examples are designed to test all important aspects of macromodeling for moment representation simulation. Each of the examples shows a plot of the simulation results of SPICE (in solid lines) and the third order moment polynomial method calculated with a double exponential inverse (in dotted lines). All macromodel parameters are extracted from SPICE simulations.

Each example also includes a table showing numerical differences between waveforms derived from the two methods. Delay time is defined by the time difference in waveform crossings at  $V = 2.5\text{volts}$ . For some waveforms, such delay times are very small and percentage errors appear large, when in fact, the global delay errors are small.

**Example 6.5 (Inverter chain)**

This example tests the inverter macromodeling methods. Load capacitances are selected to fall between parameter data points and to test all regions of inverter operation.

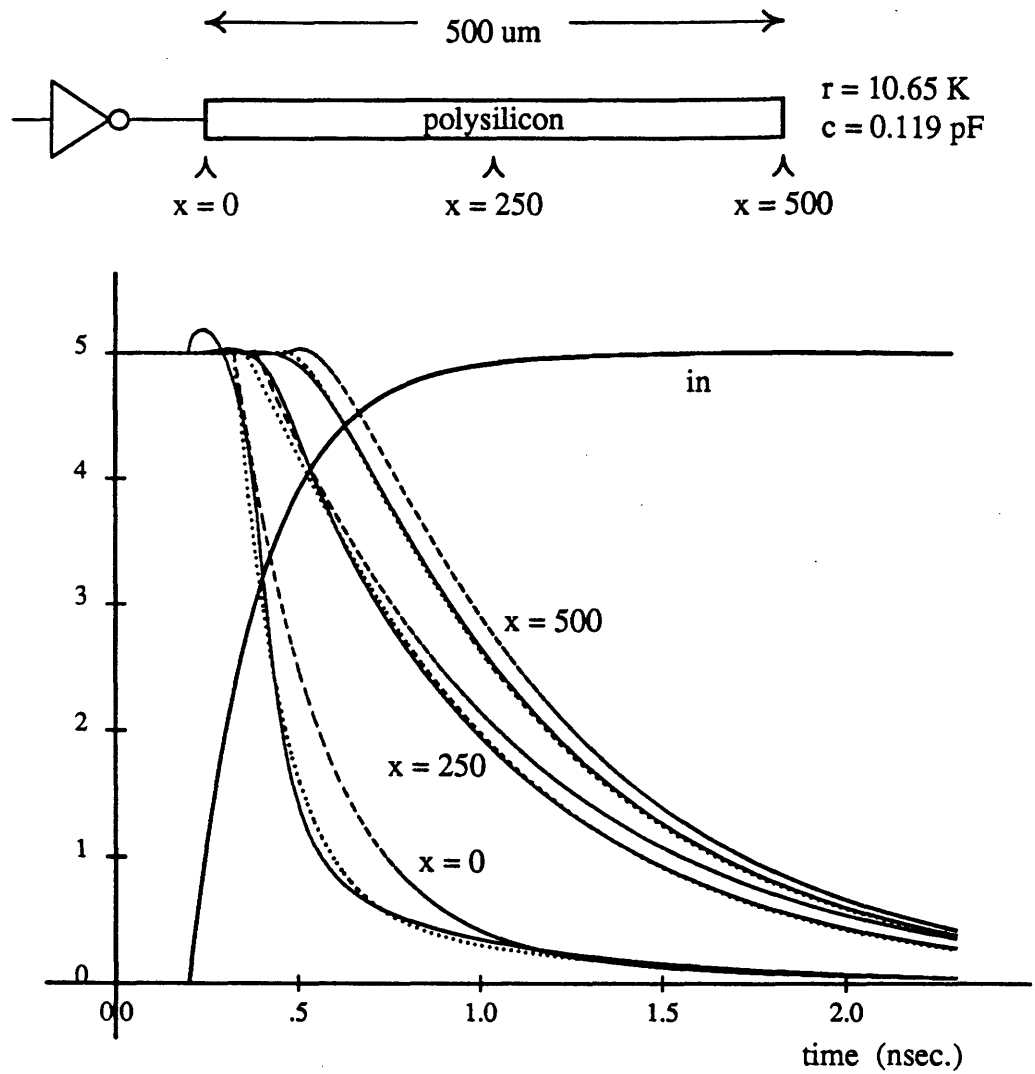


node	$t_d$		$t_d$ % diff	slope		slope % diff
	SPICE	mom. rep.		SPICE	mom. rep.	
a	.443	.366	19.1	-6.31	-6.59	4.24
b	1.24	1.19	4.26	3.42	3.37	1.60
c	3.12	3.03	2.98	-1.46	-1.46	.0667
d	4.97	4.73	4.86	1.72	1.71	.704
e	5.63	5.26	6.47	-4.39	-4.39	.0011

Waveform comparison for inverter chain example.

**Example 6.6 (Inverter driving interconnection)**

This example demonstrates the effects of combining macromodeled and linear circuits. It shows the effects of using  $C_{eff}$  for a macromodel parameter ( $\cdots$ ) versus the total load capacitance ( $---$ ).

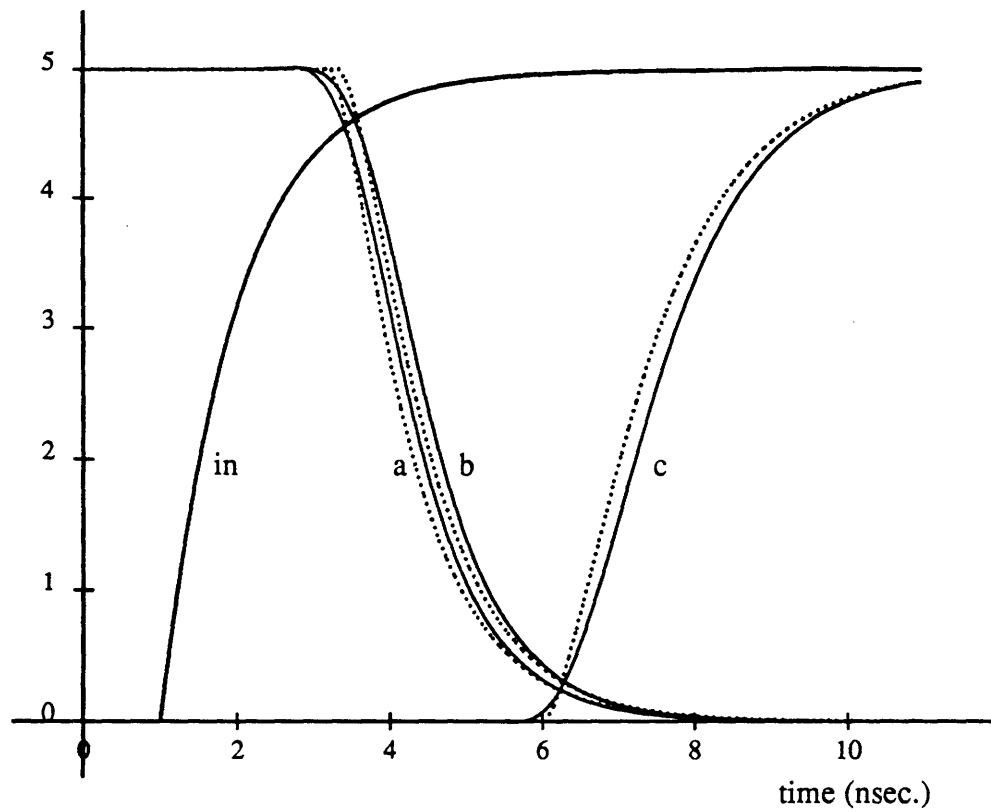
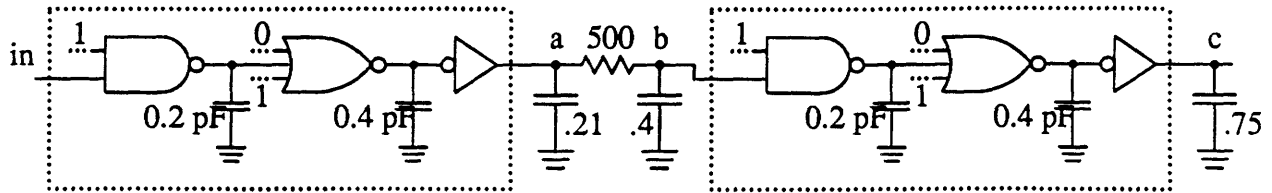


node	$t_d$		$t_d$ % diff	slope		slope % diff
	SPICE	mom. rep.		SPICE	mom. rep.	
$x = 000 \mu\text{m}$	.319	.350	9.51	-4.69	-4.73	.789
$x = 250 \mu\text{m}$	1.21	1.20	.241	-1.49	-1.53	2.68
$x = 500 \mu\text{m}$	1.42	1.41	.742	-1.49	-1.50	.156

Waveform comparison for inverter driving interconnection example.

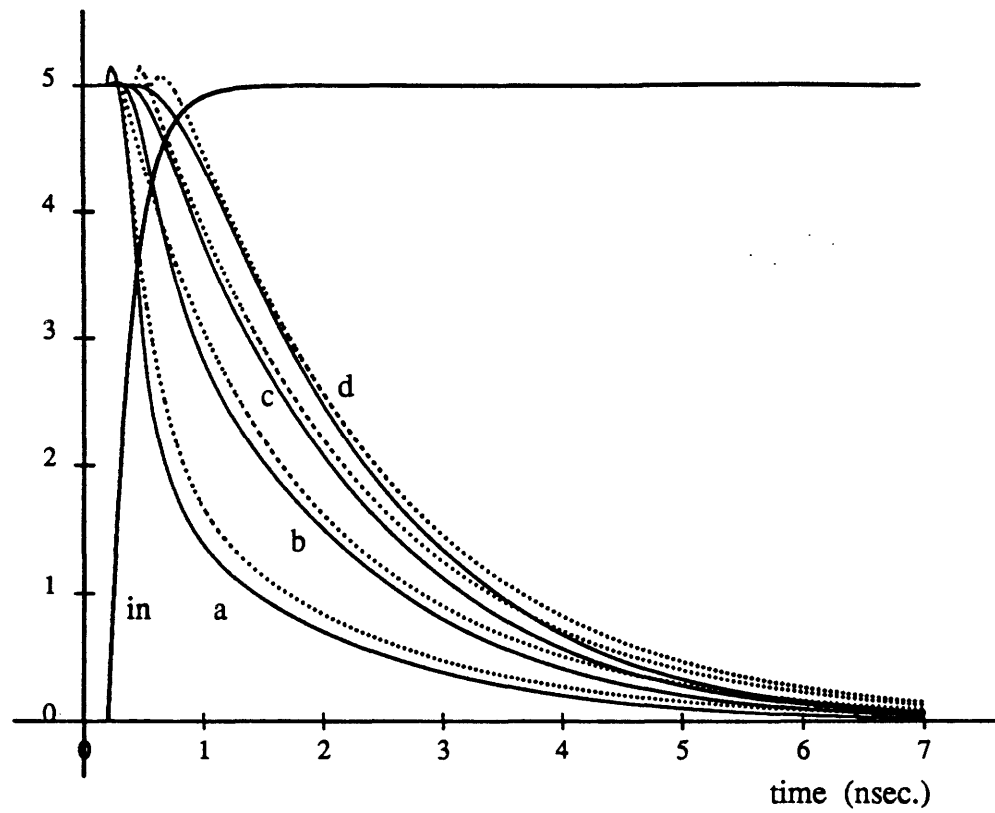
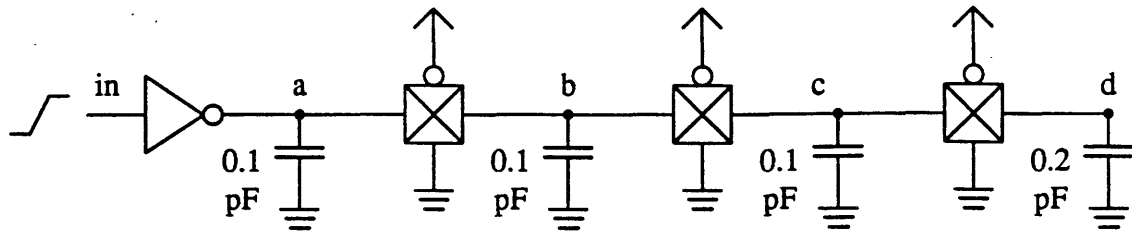
**Example 6.7 (Multi-stage logic)**

In this example, more complex macromodel circuits are tested.



node	$t_d$		$t_d$ % diff	slope		slope % diff
	SPICE	mom. rep.		SPICE	mom. rep.	
a	2.55	2.41	5.43	-2.57	-2.65	3.03
b	2.77	2.64	4.75	-2.45	-2.50	2.14
c	5.79	5.58	3.74	1.86	1.90	2.46

Waveform comparison for multi-stage logic example.

**Example 6.8 (Conducting CMOS transmission gate)**


node	$t_d$	$t_d$	$t_d$	slope	slope	slope
	SPICE	mom. rep.		% diff	SPICE	
a	.218	.332	41.6	-6.10	-3.94	43.1
b	.823	.962	15.5	-1.72	-1.63	5.21
c	1.35	1.44	6.37	-1.44	-1.41	1.76
d	1.64	1.70	3.75	-1.52	-1.42	7.43

Waveform comparison for conducting CMOS transmission gate example.

Example	SPICE transient analysis	First Order Moment Rep. Macromodeling	Third Order Moment Rep. Macromodeling
Inverter chain	23.4	0.080	0.223
Inverter driving interconnection	6.50	0.245	0.729
Multi-stage logic	48.3	0.039	0.178
Conducting CMOS transmission gate	19.07	0.172	0.401

Table 6-3: Simulation time in CPU seconds.

## 6.9 Computational Requirements

In Table 6-3 the computational requirements of macromodeled moment representation simulation and SPICE simulation are compared for the above examples. Only pertinent, transient simulation time is included in the values. The table shows that speedups of one to two orders of magnitude are possible. For larger circuits, even better improvements are expected.

The largest portion of the moment representation simulation time is spent solving linear networks and converting moment representations to time-domain waveforms. The ratio depends heavily on the circuit. Of the above examples, the extremes lie with the inverter chain example on the one hand, where computing time-domain waveforms and slopes takes 60% of CPU time, to the inverter driving interconnection example on the other hand, where solving the linear network takes 92% of the CPU time. Macromodel lookup occupies an insignificant fraction of the CPU time.

## 6.10 Discussion

This chapter has shown a macromodeling method which enables non-linear circuits to be solved rapidly. Linear circuits are formed which match the behavior at the edge of a cell when connected to other cell circuits. The linear elements are macromodeled functions which depend on what is connected on either side. For the typical driver cell type, the macromodel parameters are input waveform slope and effective output load capacitance. Once the non-linear circuit has been transformed into a linear circuit, it is solved with the methods described in Chapter 4.

The macromodeling methods require more computation, because both the input and output parameters depend on waveform information that has to be converted from moment representations. Nonetheless, these methods are significantly faster than non-linear simulation methods included in SPICE. The next chapter shows that improvements to the simulation algorithm amass even further improvements to simulation speed.

A negative aspect of this macromodeling method is the computation time needed to compute the macromodel function values. This may be hours of CPU time per macromodel cell. Yet, if

this simulation method is applied to a cell library, the utilization of the macromodels is very high, making them extremely efficient. This issue is also addressed in the next chapter.





---

---

# Circuit Model Compilation

We have now seen all of the basic simulation and modeling techniques for the moment representation and are ready to address some efficiency issues for moment representation methods.

This chapter presents a simulation method for compiling an entire sub-network into a few simple macromodel functions. The benefit of this method is greatly improved simulation speed. First, the method is described; then the method is evaluated with some estimated CPU run-time costs.

## 7.1 Circuit Model Levels

We begin this discussion by looking at the different levels of circuit modeling that are possible with the moment representation. Here we are interested in which circuit level exists during simulation. The levels are shown in Figure 7-1 for a sample sub-network containing an inverter driving an interconnection leading to the input of another inverter. The interconnection is capacitively coupled to another sub-network.

The first two circuit model levels are familiar. The first level (circuit (a)) contains non-linear circuit elements and is represented by a non-linear matrix equation. The circuit can be solved by direct methods methods (Newton-Raphson iteration and numerical integration).

The second level (circuit (b)) is the moment representation level formulated in previous chapters. It contains a macromodeled linear network. To simulate one cycle at this level, the simulator (1) looks up macromodel values for the driving inverter's linear equivalent circuit based on the input signal slope and load, (2) solves the linear network for output node moment representations, and (3) converts output node moment representations to time domain equations, if needed.

The next two circuits, in (c) and (d), depict new, simple circuit model levels described in the next two sections.

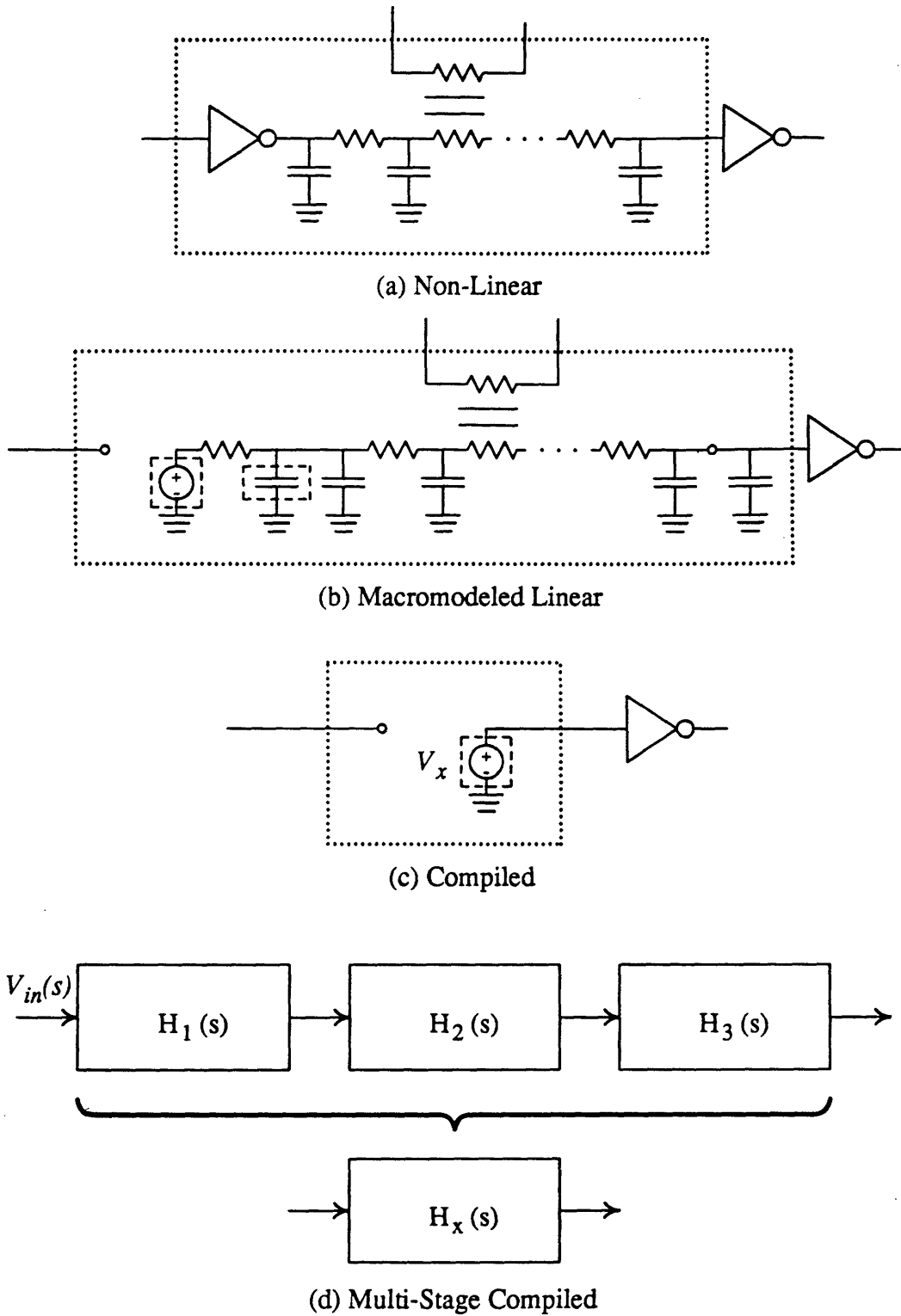


FIGURE 7-1: Circuit model levels.

### 7.1.1 Compiled Circuit Model

Once complete sub-network details are known, a sub-network circuit can be *compiled* into the circuit model shown in Figure 7-1(c). It contains only a macromodeled voltage source.

The compiled circuit model is equivalent to a circuit transfer function. If the entire sub-network was linear, the voltage source,  $\tilde{V}_x(s)$ , of the compiled circuit model would equal the circuit transfer function multiplied by the input waveform transform. However, since the circuit is non-linear,  $\tilde{V}_x(s)$  is a macromodeled function with input voltage slope the only parameter. These macromodels do not depend on load, since this information is compiled into the  $\tilde{V}_x(s)$  macromodel functions. This fact decreases the size of a complete macromodel set for a compiled circuit model, since only the following information is needed for a third-order model:

$$\begin{aligned} & t_{0,x}(t_{r,in}) \\ & M_{0,x} \\ & M_{1,x}(t_{r,in}) \\ & M_{2,x}(t_{r,in}) \\ & M_{3,x}(t_{r,in}) \end{aligned}$$

The size of this macromodel set typically varies between 20 and 50 floating point numbers.

The method for converting a network from the macromodeled linear circuit model to a compiled circuit model is simple, as shown in Algorithm 7.1. Basically, the linear network is solved for each  $t_{r,in}$  data point of the original network. Then voltage moment values of relevant output nodes are stored in new macromodel functions.

**Algorithm 7.1**

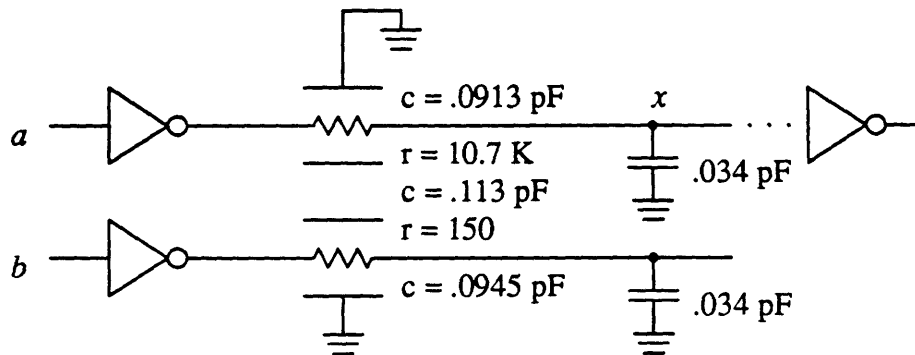
{  $\mathcal{L}$  is the original, macromodeled linear circuit sub-network,  $\mathcal{C}$  is the new, compiled sub-network. The macromodel functions of  $\mathcal{L}$  are originally unfilled. The notation  $\mathcal{L}[T_{in}]$  indicates the macromodel set in  $\mathcal{L}$  for the input transition,  $T_{in}$ , where  $T_{in}$  refers to a specific combination of input state and input transition.  $\mathcal{C}[T_{in}, T_{out}]$  indicates the macromodel set in  $\mathcal{C}$  for input transition,  $T_{in}$ , and output transition,  $T_{out}$ . }

```

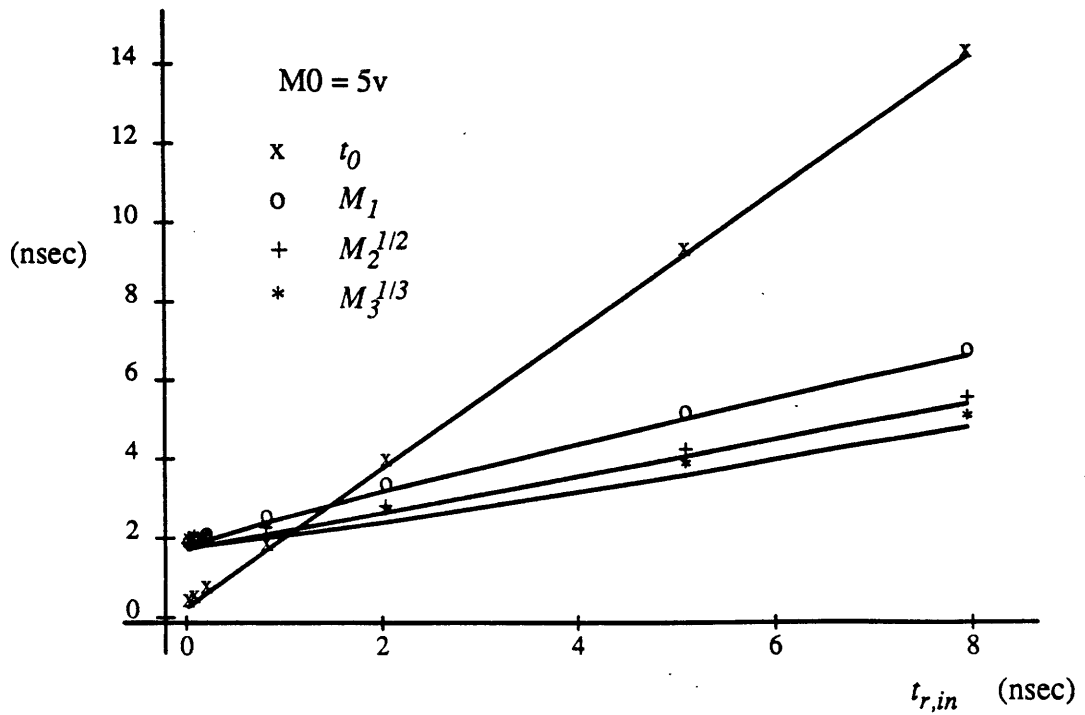
procedure SUBNETWORK_COMPILE ( $\mathcal{L}$ ,  $\mathcal{C}$ : subnetwork) begin
  SET_STATIC_CONDITIONS( $\mathcal{L}$ );
  for each input transition,  $T_{in}$  in  $\mathcal{L}$  do begin
    for each macromodel data point,  $t_r$  in  $\mathcal{L}[T_{in}]$  do begin
       $C_{eff} :=$  COMPUTE_EFFECTIVE_LOAD( $\mathcal{L}$ ,  $T_{in}$ ,  $t_r$ );
      UPDATE_MACROMODEL_ELEMENTS( $\mathcal{L}$ ,  $T_{in}$ ,  $t_r$ ,  $C_{eff}$ );
       $v :=$  SOLVE_NETWORK( $\mathcal{L}$ );
      for each output transition,  $T_{out}$  in  $\mathcal{L}$  caused by  $T_{in}$  do begin
         $V_x := v[T_{out}]$ ;
        { Get the macromodel set for this pair of input and output transitions. }
         $m := \mathcal{C}[T_{in}, T_{out}]$ ;
        { Set the macromodel functions to  $V_x$  moment representation values. }
         $m.t_0(t_r) := t_{0,V_x}$ ;
         $m.M_0 := M_{0,V_x}$ ;
         $m.M_1(t_r) := M_{1,V_x}$ ;
         $m.M_2(t_r) := M_{2,V_x}$ ;
         $m.M_3(t_r) := M_{3,V_x}$ ;
      end
    end
  end
end

```

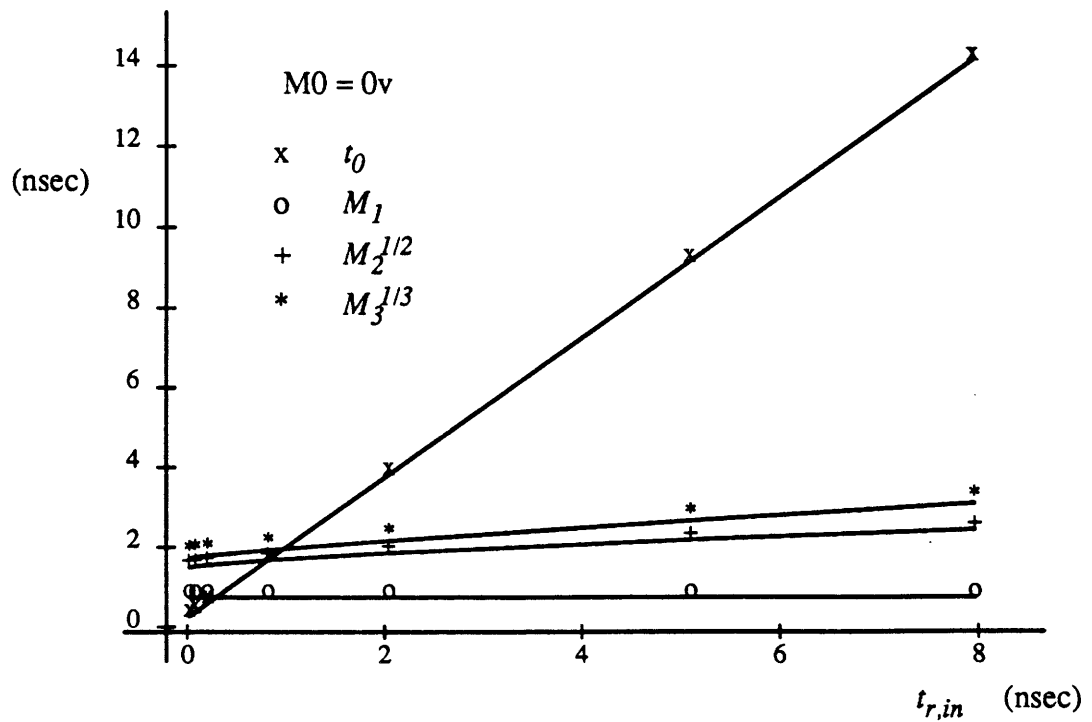
**Example 7.1** Plot compiled circuit model macromodel functions for node  $x$  in the circuit below.



There are four macromodel sets for this circuit—one for each combination of input signal level and transition on nodes  $a$  and  $b$ . Two are plotted below.



Macromodel functions vs.  $t_{r,V_a}$  for  $V_a = 0 \rightarrow 5v$ ,  $V_b = 5v$ .



Macromodel functions vs.  $t_{r,V_a}$  for  $V_a = 5v$ ,  $V_b = 0 \rightarrow 5v$ .  
 These macromodel functions define a noise spike.

### 7.1.2 Multi-Stage Compiled Circuit Model

The last circuit level model depicted in Figure 7-1(d) is a *multi-stage compiled* circuit model. Each block in the circuit represents a sub-network. The idea here is that several interconnected sub-network stages,  $H_1$ ,  $H_2$  and  $H_3$ , can be combined into one equivalent compiled circuit model,  $H_x$ .

This idea is analogous to computing a combined transfer function of a linear system. If all of the sub-networks were linear then we could find  $\tilde{H}_x(s) = \tilde{H}_1(s) \cdot \tilde{H}_2(s) \cdot \tilde{H}_3(s)$ . Since the networks are not linear, it suffices to model  $\tilde{H}_x(s)$  with

$$\tilde{H}_x(s) = e^{-s t_0(t_r, V_{in})} (M_0 + M_1(t_r, V_{in}) + M_2(t_r, V_{in}) + M_3(t_r, V_{in}) + \dots)$$

where  $t_0(t_r, V_{in})$ ,  $M_0$ ,  $M_1(t_r, V_{in})$ ,  $M_2(t_r, V_{in})$ ,  $\dots$  form a macromodel set. Each combination of input logic levels and transition has a different macromodel set.

The number of sub-network enclosed in a multi-stage compiled circuit model is limited by the number of input logic combinations and internal logic states, since, the number of output macromodel sets may grow exponentially with increasing size. Currently, sub-networks are combined only at the request of the user.

Algorithm 7.1 also computes the macromodel functions for a multi-stage circuit model. The difference here is that routine `SOLVE_NETWORK` computes the output response of the cascaded sub-networks.

## 7.2 Computation Requirements

The big advantage of circuit model compilation is simulation efficiency. Circuit model compilation represents the final improvement in simulation efficiency, so we will look into computational requirements more thoroughly, now.

During simulation, each input transition causes the following computation steps for a single, compiled circuit model: (1) macromodel function lookup for the output voltage moment representation values, (2) computation of the time domain waveform at relevant outputs. The time consuming task of solving linear circuits is not done during simulation.

First, consider the CPU time required to do a complete chip simulation. Simulation time decreases as simpler circuit models are used. In this section we consider what the computation is needed to simulate a chip using each circuit level shown in Figure 7-1.

Simulation time estimates in this section are based on the parameters listed in Table 7-1. These parameters approximate values for a chip using a standard cell approach where interconnections are carefully modeled. The estimated number of linear circuit nodes per sub-network assumes that (1) interconnections are modeled with distributed circuit elements, as in Example 4.10, and (2) noise waveforms are computed for one coupled line. These parameters also

Number of sub-networks	$10^4$
Simulation input transitions per sub-network	$10^3$
The average sub-network has:	
Number of transistors	8
Number of linear circuit nodes	10
Number of macromodel sets	15
Number of data points per macromodel sets	6
Number of connected output nodes	3
Macromodel cells in circuit library	300
Average number of sub-networks per multi-stage circuit model	2.5

Table 7-1: Parameters for complete system CPU time estimates.

	<i>first order</i>	<i>third order</i>
Non-linear circuit model		55,500
Macromodeled linear circuit model	1106	1725
Compiled circuit model	3.8	97
Multi-stage compiled circuit model	1.5	39

Table 7-2: Simulation times for different circuit model levels in hours/MIP.

assume a small macromodel cell size, as indicated in the average number of transistors per sub-network. Larger macromodel cells would further improve moment representation simulation times.

Based on these parameters and on careful simulation time measurements of average sized cells, simulation times are estimated. Table 7-2 lists simulation time estimates for each circuit model level. Two run-times are given for moment representation methods—for first-order and third-order simulations. The non-linear times are based on SPICE simulations, but the total time is estimated by assuming that each sub-network is simulated separately. Thus, this time reflects, more accurately, the CPU run-time of a chip-wide simulation with SPLICE [10] or with waveform relaxation methods [11]. At the level of circuit complexity prescribed by the chip parameters, the macromodeled linear circuit model simulations consume most of its CPU time solving linear networks. The compiled circuit model eliminates this burden; saving at least an order of magnitude in simulation run-time. Some additional benefit is gained by combining sub-networks into multi-stage models.

Translating circuits from a complex circuit model to a simpler circuit model is not without a cost. Some set of simulations must be done with the expensive method, and model parameters must be extracted before one can simulate with the simpler model. We have already seen that converting a cell from a non-linear circuit model to a macromodeled linear circuit model

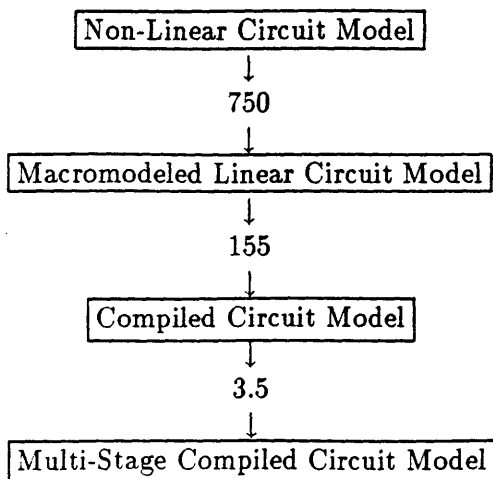


Table 7-3: Third-order circuit model translation times in hours/MIP.

(macromodel extraction) is time-consuming. Table 7-3 shows the CPU time for converting from a higher level circuit model to a lower level circuit model.

We clearly see, even when translation time from the top level circuit model is added to simulation time, that it is still advantageous to compile circuits to the simplest level and to simulate at this level. This is true in general when many simulation cycles are needed for each circuit. If cell libraries are shared among several chip designs, the macromodel extraction time can also be shared. At the macromodeled linear circuit level and below, exact interconnectivity and circuit usage is assumed. Hence, translation below this level is done only on a per circuit basis, i.e., in the preprocessor.



---

---

## Conclusions and Future Work

This thesis has presented the moment representation as a means for simulating digital circuits containing interconnections and non-linear active devices. The methods are optimized for simulating interconnection networks. Chapter 2 showed a strong motivation for this, namely, it is because VLSI circuits are being increasingly effected by interconnection performance.

In Chapter 3 the moment representation was defined. It demonstrated that the moment representation has two views: in one view it is a series of moment parameters for a time domain waveform, in the other view it is a projection from the Laplace transform. The first view allows us to covert between the moment representation and the time domain with a set of special algorithms, described in Chapter 3. The number of moment representation parameters is variable as specified by the representation "order". The user can prescribe a simulation order—as the order increases, accuracy increases at the expense of computation complexity.

Using the second view of the moment representation, i.e., as a mapping from the Laplace transform, Chapter 4 showed a method for solving any linear circuit configuration with independent current sources, resistors, capacitors, inductors (including mutual inductors) and distributed combinations of the above, with a very minimal set of restrictions. This circuit flexibility is achieved by using nodal analysis matrix equations. Transmission line circuits can be treated specially with modal analysis, as Chapter 5 describes, to give an exact value for propagation delay time, and hence a better waveform solution.

Chapter 6 described a very accurate macromodeling method that enables moment representation simulation of non-linear drivers and admittances. Non-linear elements are converted into linear circuit equivalents which may have many, variable circuit elements. Circuit element values are macromodeled as functions of input signal slope and output load. We saw that this method is favorable for very rapid simulation of combined linear and non-linear circuits.

Lastly, in Chapter 7 a circuit model compilation method was presented which can drastically enhance simulation speed. The chapter illustrated how a circuit's output waveform moment representation can be compiled into a small set of macromodel functions which depend on the

input waveform. This method can be viewed as using the circuit's transfer function to compute the response, but the method also models the behavior of non-linear circuit elements.

Direct simulation methods are far too slow to consider for the complete chip simulation which is needed for adequate circuit performance verification. As a result, many specialized methods have tackled interconnection simulation, as we saw in Chapter 1. Moment representation methods offer many advantages over these.

The moment representation simulator is capable of simulating all configurations of interconnection simulation. It can simulate  $LC$  and  $RC$  interconnections with or without coupling, circuits with charge sharing, single-ended drivers, pass gates, etc. The moment representation is very suited to interconnection problems of digital circuits, since it is able to (1) simulate an entire transition in one matrix solution (2) directly include distributed circuit elements (3) responses to any non-linear circuit can be "compiled" by a preprocessor into a very compact moment representation form.

Moment representation methods can simulate non-linear circuits with a high degree of accuracy. On a number of non-linear test examples, after several stages of logic all third-order moment representation simulated waveforms had delay times that were within 10% of those computed by SPICE. Noise waveforms had peak magnitudes within 18% on all tested circuits. Typically, accuracies were far better than these worse case results.

## 8.1 Future Work

This thesis has presented fundamentals which were necessary to make the first successful version of the moment representation simulator. A number of unresolved issues and questions still remain. Below is a partial list of these questions for future investigation.

- Can the moment representation be replaced by other field number system during network solution? Different representations may make some parts of the problem easier and others more difficult. One possible representation may be the set of rational polynomials of  $s$ , the Laplace Transform variable. This form is more easily converted to a time-domain waveform, but is more difficult to do be derived from a time-domain waveform.
- Is it possible to automatically make a more intelligent selection of inverse Laplace Transform assumed waveform shape?
- Is it possible to extend circuit flexibility to include dependent source elements? Improved Miller capacitance modeling of MOS loads, accurate modeling of bipolar devices and circuit feedback modeling are just a few potential uses of dependent sources.

Circuit matrices with dependent sources are not always symmetric and diagonally dominant. Truncation algorithms presented in this thesis depend on this fact. Yet, several examples have shown that circuits with voltage dependent current-sources can be solved

by selectively increasing the truncation orders of  $b$  during forward elimination. A general algorithm for this has not been determined.

- Can we solve non-linear networks without knowing actual macromodel parameters, and instead leave them as symbolic variables? Rather than resolving the network for each different input waveform, a symbolic function is evaluated. It is quite possible to redefine mathematical procedures with symbolic numbers and to incorporate these into the matrix solution routine. But, whether this would reduce computation or whether the functions would become too complex for moderately sized circuits are unknown.
- Can lossy transmission lines be simulated?
- Lastly, is it possible to use these models in the reverse direction? This thesis has assumed circuit *analysis* through simulation, but one should be able to use the moment representation as a *synthesis* tool. For instance, if we have a compiled circuit model, it is possible possible to determine an input waveform shape to the circuit that guarantees a performance specification on the output—be that specification speed or noise margins. We could, for instance, determine the minimum transition time that guarantees a network with no noise margin violations.

A more difficult problem to consider is whether the moment representation can be used in the reverse direction to find a critical circuit element value that meets a specification—for instance, a minimum drive resistance that guarantees a network with no noise margin violations. Unlike variable waveforms, variable circuit elements require changes to the matrix circuit solution methods. It may be that symbolic, variable circuit values will answer this need.



---



---

## Moment Polynomial Operations

Basic polynomial operations form the backbone of the circuit simulation techniques of this thesis. Polynomial operations are needed for addition, subtraction, multiplication and division of polynomials with the form

$$\tilde{f}(s) = f_m s^m + f_{m+1} s^{m+1} + \cdots + f_{n-1} s^{n-1} + f_n s^n \quad (\text{A.1})$$

where  $m$  is the polynomial's base order, denoted by  $\lfloor \tilde{f}(s) \rfloor$ , and  $n$  is the polynomial's order. The result of each operation is truncated to a maximum order  $p$ , the *truncation order*. The truncation order is always specified in advance of the operation to minimize the actual number of floating point operations.

### A.1 Definitions

The operations for addition, subtraction and multiplication are common algebraic calculations. Presented in a mathematical form most like its true computer implementation:

- addition of  $\tilde{a}(s) + \tilde{b}(s) = \tilde{c}(s)$  is

$$c_i = a_i + b_i, \quad \text{for all } \min\{\lfloor a \rfloor, \lfloor b \rfloor\} \leq i \leq p,$$

- subtraction of  $\tilde{a}(s) - \tilde{b}(s) = \tilde{c}(s)$  is

$$c_i = a_i - b_i, \quad \text{for all } \min\{\lfloor a \rfloor, \lfloor b \rfloor\} \leq i \leq p,$$

- and multiplication of  $\tilde{a}(s) \cdot \tilde{b}(s) = \tilde{c}(s)$  is

$$c_i = \sum_{m=0}^i a_m \cdot b_{i-m}, \quad \text{for all } (\lfloor a \rfloor + \lfloor b \rfloor) \leq i \leq p,$$

Algebraically, division of polynomials does not yield an answer in the form of Equation (A.1). The Maclaurin Series of the quotient is taken, however, without loss of accuracy in the first  $p$

terms of  $\bar{a}(s)/\bar{b}(s)$ . First, the dividend and divisor terms are normalized with respect to the lowest-ordered term of the denominator:

$$\begin{aligned} a_0 &= \frac{a_{[a]}}{b_{[b]}} \\ a_1 &= \frac{a_{[a]+1}}{b_{[b]}} \\ &\vdots \end{aligned}$$

and

$$\begin{aligned} b_0 &= 1 \\ b_1 &= \frac{b_{[b]+1}}{b_{[b]}} \\ b_2 &= \frac{b_{[b]+2}}{b_{[b]}} \\ &\vdots \end{aligned}$$

Then, quotient terms are found as needed:

$$\begin{aligned} [c] &= [a] - [b] \\ c_{[c]} &= a_0 \\ c_{[c]+1} &= a_1 - a_0 b_1 \\ c_{[c]+2} &= a_0 b_1 b_1 - a_0 b_2 + a_2 - a_1 b_1 \\ c_{[c]+3} &= -(a_0 b_3 + b_2 (a_1 - 2 a_0 b_1) + b_1 (a_2 - b_1 (a_1 - b_1 a_0)) - a_3) \\ c_{[c]+4} &= -(b_1 (a_3 - b_1 (a_2 - b_1 (a_1 - b_1 a_0))) + b_2 (a_2 - b_1 (2 a_1 - b_1 3 a_0)) \\ &\quad + b_3 (a_1 - b_1 2 a_0) + a_0 (b_4 - b_2 b_2) - a_4) \\ &\vdots \\ c_p &= \dots \end{aligned}$$

## A.2 Minimum Polynomial Orders

From the above definitions, the minimum polynomial orders which result from each operation are:

$$[\bar{a}(s) \pm \bar{b}(s)] = \min\{[\bar{a}(s)], [\bar{b}(s)]\} \quad (\text{A.2})$$

$$[\bar{a}(s) \cdot \bar{b}(s)] = [\bar{a}(s)] + [\bar{b}(s)] \quad (\text{A.3})$$

$$[\bar{a}(s)/\bar{b}(s)] = [\bar{a}(s)] - [\bar{b}(s)] \quad (\text{A.4})$$

### A.3 Truncation Order Rules

With the above definitions, we can define a set of operations truncation rules. These rules take the form of, "assuming the result of  $\tilde{a}(s)$  op  $\tilde{b}(s)$  has a truncation order of  $x$ , to guarantee an exact result to order  $x$  the truncation orders of the two operands must be..."

The first two rules are easy:

$$\begin{aligned} \text{If } \mathcal{T} [\tilde{a}(s) \pm \tilde{b}(s)] = x, \text{ then } \mathcal{T} [\tilde{a}(s)] = x, \\ \text{and } \mathcal{T} [\tilde{b}(s)] = x. \end{aligned}$$

$$\begin{aligned} \text{If } \mathcal{T} [\tilde{a}(s) \cdot \tilde{b}(s)] = x, \text{ then } \mathcal{T} [\tilde{a}(s)] = x - [\tilde{b}(s)], \\ \text{and } \mathcal{T} [\tilde{b}(s)] = x - [\tilde{a}(s)]. \end{aligned}$$

From the division equations above, we see that the number of terms needed in the numerator and denominator equals  $x$ , or the number of terms in all of the bracketed areas of

$$\frac{\overbrace{a_{[a]} s^{[a]} + \dots + a_{\mathcal{T}[a]} s^{\mathcal{T}[a]}}}{\overbrace{b_{[b]} s^{[b]} + \dots + a_{\mathcal{T}[b]} s^{\mathcal{T}[b]}}} = \overbrace{c_{[a]-[b]} s^{[a]-[b]} + \dots + c_x s^x}$$

are equal. Thus,

$$\begin{aligned} \text{If } \mathcal{T} [\tilde{a}(s)/\tilde{b}(s)] = x, \text{ then } \mathcal{T} [\tilde{a}(s)] = x - [\tilde{b}(s)], \\ \text{and } \mathcal{T} [\tilde{b}(s)] = x - [\tilde{a}(s)] + 2 [\tilde{b}(s)]. \end{aligned}$$

### A.4 Floating Point Operation Counts

Table A-1 shows the number of floating point operations needed for polynomial operations of varying truncation orders.

<i>operation</i>	<i>p</i>	<i>floating point operations</i>		
		adds and subtracts	multiplies	divides
addition or subtraction	0	1	0	0
	1	2	0	0
	2	3	0	0
	3	4	0	0
	4	5	0	0
multiplication	0	0	1	0
	1	1	3	0
	2	3	6	0
	3	6	10	0
	4	10	15	0
division	0	0	0	1
	1	1	1	3
	2	4	5	5
	3	10	12	7
	4	21	26	9

Table A-1: Maximum number of floating-point operations for polynomial operations.



---



---

## Truncation Orders During Gaussian Elimination

This appendix section develops a proof that the truncation orders shown in Theorem 4.1 guarantee an exact result for  $\tilde{v}(s)$  to order  $p$  if the linear circuit obeys the MPNA restrictions listed on Page 70.

### B.1 Moment Representation Admittance Properties

First, we consider the moment representation admittance function of a combination of elements.

**Theorem B.1** *The minimum order of  $\tilde{Y}_{ab}(s)$ , the combined admittance between two nodes,  $a$  and  $b$ , of any connection of  $R$ ,  $L$  and  $C$  elements, is  $x$ , where  $x$  is the minimum over all paths between the two nodes of the maximum circuit element order along the path. That is,*

$$\min_{\text{all paths}} \left[ \max_{\text{all path elements}} \{[\tilde{y}_{\text{element}}(s)]\} \right] = [\tilde{Y}_{ab}(s)] \equiv x. \quad (\text{B.1})$$

**Proof:** If  $\tilde{I}_{total}(s)$  is the current that flows between nodes  $a$  and  $b$  when  $\tilde{V}_{ab}(s)$  is applied across  $a$ - $b$ , then

$$\tilde{I}_{total}(s) = \tilde{Y}_{ab}(s) \cdot \tilde{V}_{ab}(s).$$

Assume a voltage,  $\tilde{V}_{ab}(s) = 1$ , is applied across  $a$ - $b$ , then  $\tilde{I}_{total}(s) = \tilde{Y}_{ab}(s)$  and from Equation (A.3),

$$[\tilde{I}_{total}(s)] = [\tilde{Y}_{ab}(s)] = x. \quad (\text{B.2})$$

Now define any set of unique paths between node  $a$  and  $b$  such that at least one path goes through all elements, and such that one path, *minpath*, is the minimum path in defined in Equation (B.1). From KCL,

$$\tilde{I}_{total}(s) = \sum_{\text{all paths}} \tilde{I}_{\text{path}}(s). \quad (\text{B.3})$$

From Equations (B.2) and (B.3) and the addition minimum order equation, (A.2),

$$\begin{cases} [\tilde{I}_{path}(s)] = x, & \text{for } \underline{minpath}, \\ [\tilde{I}_{path}(s)] \geq x, & \text{for any other path.} \end{cases}$$

Since any element current is the sum of one or more path currents,

$$\begin{cases} [\tilde{I}_{elt}(s)] = x, & \text{for any element along } \underline{minpath}, \\ [\tilde{I}_{elt}(s)] \geq x, & \text{for any other element.} \end{cases} \quad (\text{B.4})$$

Applying KVL along any path, we know that the sums of voltage drops equals  $\tilde{V}_{ab}(s)$ , or

$$\tilde{V}_{ab}(s) = \sum_{\text{all path elements}} \frac{\tilde{I}_{path}(s)}{\tilde{y}_{element}(s)}, \quad \text{for any path.}$$

Using minimum order rules:

$$[\tilde{V}_{ab}(s)] = \min_{\text{all path elements}} \{[\tilde{I}_{elt}(s)] - [\tilde{y}_{elt}(s)]\}, \quad \text{for any path,} \quad (\text{B.5})$$

Since  $[\tilde{V}_{ab}(s)] = 0$ , Equation B.5 can be rewritten

$$\max_{\text{all path elements}} \{[\tilde{y}_{elt}(s)] - [\tilde{I}_{elt}(s)]\} = 0, \quad \text{for any path,}$$

and from Equation B.4,

$$\begin{cases} \max_{\text{all path elements}} \{[\tilde{y}_{elt}(s)]\} = x, & \text{for any element along } \underline{minpath}, \\ \max_{\text{all path elements}} \{[\tilde{y}_{elt}(s)]\} \geq x, & \text{for any other element.} \end{cases}$$

This is equivalent to the theorem statement.

## B.2 MPNA matrix properties

Because we have restricted the set of permissible circuits to reciprocal *RLC* circuits, there are several things we can say about the minimum orders of MPNA matrix elements. Each of the following theorems is based on the fact that the nodal analysis equations of *RLC* circuits have

1. at a diagonal term, the positive sum of admittances of all circuit elements connected to the node for that equation,

$$\tilde{y}_{kk}(s) = \tilde{y}_{k1}(s) + \cdots + \tilde{y}_{kk-1}(s) + \tilde{y}_{k0}(s) + \tilde{y}_{kk+1}(s) + \cdots + \tilde{y}_{kn}(s). \quad (\text{B.6})$$

(For a mutual inductance, the diagonal admittance term, is scaled by  $L_{cpl}/M$ , but, is always positive and non-zero.)

2. at an off-diagonal term,  $\tilde{y}_{jk}(s)$ , the admittance of a circuit element connected between nodes  $j$  and  $k$  only.

**Theorem B.2** *The diagonal terms of admittance matrix,  $\tilde{Y}(s)$ , have minimum order*

$$[\tilde{y}_{kk}(s)] \leq B_k$$

for any node  $k$ .

**Proof:** From the definition of  $B_k$ , there is a path to ground with a maximum circuit element order of  $B_k$ . Either that path is to ground directly, in which case, from Theorem B.1,  $[\tilde{y}_{k0}(s)] = B_k$ , or it is through another node. If it is through another node, the element between it and node  $k$  must have  $[\tilde{y}_{km}(s)] \leq B_k$ . Furthermore, since all admittance terms are positive, when the admittances are summed as in Equation (B.6), there is a positive term of order less than or equal to  $B_k$ .

**Theorem B.3** *The minimum order of any off-diagonal term, where defined, is never less than the order of the diagonal term in its row or column, or*

$$[\tilde{y}_{jk}(s)] \geq [\tilde{y}_{jj}(s)] \text{ and } [\tilde{y}_{jk}(s)] \geq [\tilde{y}_{kk}(s)], \text{ for any } j, k.$$

**Proof:** In a reciprocal network,  $y_{jk} = y_{kj}$ , and from Equation (B.6), and since the positive-only terms cannot cancel in Equation (B.6), if there is a contribution of any minimum order in an off-diagonal term, then there must also be a contribution in the diagonal term of the same order.

**Theorem B.4** *The minimum order of any current term of  $\tilde{i}(s)$  is never less than the order of the diagonal term in its row, given that circuit Restriction 3 on Page 70 is satisfied, or*

$$[\tilde{i}_k(s)] \geq [\tilde{y}_{kk}(s)], \text{ for any } k.$$

**Proof:** From the nodal analysis equation of node  $k$ ,

$$\tilde{i}_k(s) = \tilde{y}_{k1}(s) \tilde{v}_1(s) + \tilde{y}_{k2}(s) \tilde{v}_2(s) + \cdots + \tilde{y}_{kn}(s) \tilde{v}_n(s)$$

we see that none of the individual terms of the summation can have order less than  $[\tilde{y}_{kk}(s)]$ , since, from Theorem B.3,  $[\tilde{y}_{kx}(s)] \geq [\tilde{y}_{kk}(s)]$  for any  $x$ , and from Restriction 3,  $[\tilde{v}_x(s)] \geq 0$ .

### B.2.1 Gaussian Elimination of Nodal Analysis Matrices

In this section we examine what happens to the circuit described by  $\tilde{Y}(s)\tilde{v}(s) = \tilde{i}(s)$  during the forward elimination steps of gaussian elimination. We start by looking at what happens after the first column is zeroed below  $\tilde{y}_{11}(s)$ . Suppose Node 1 is connected to three other nodes as shown in Figure B-1(a). Since Row 1 (Equation 1) is unchanged, Node 1 sees the same impedances and voltages, but now none of the remaining equations depends on  $v_1$ . This can be exactly represented in a *circuit* sense by decoupling Node 1 as shown in Figure B-1(b).

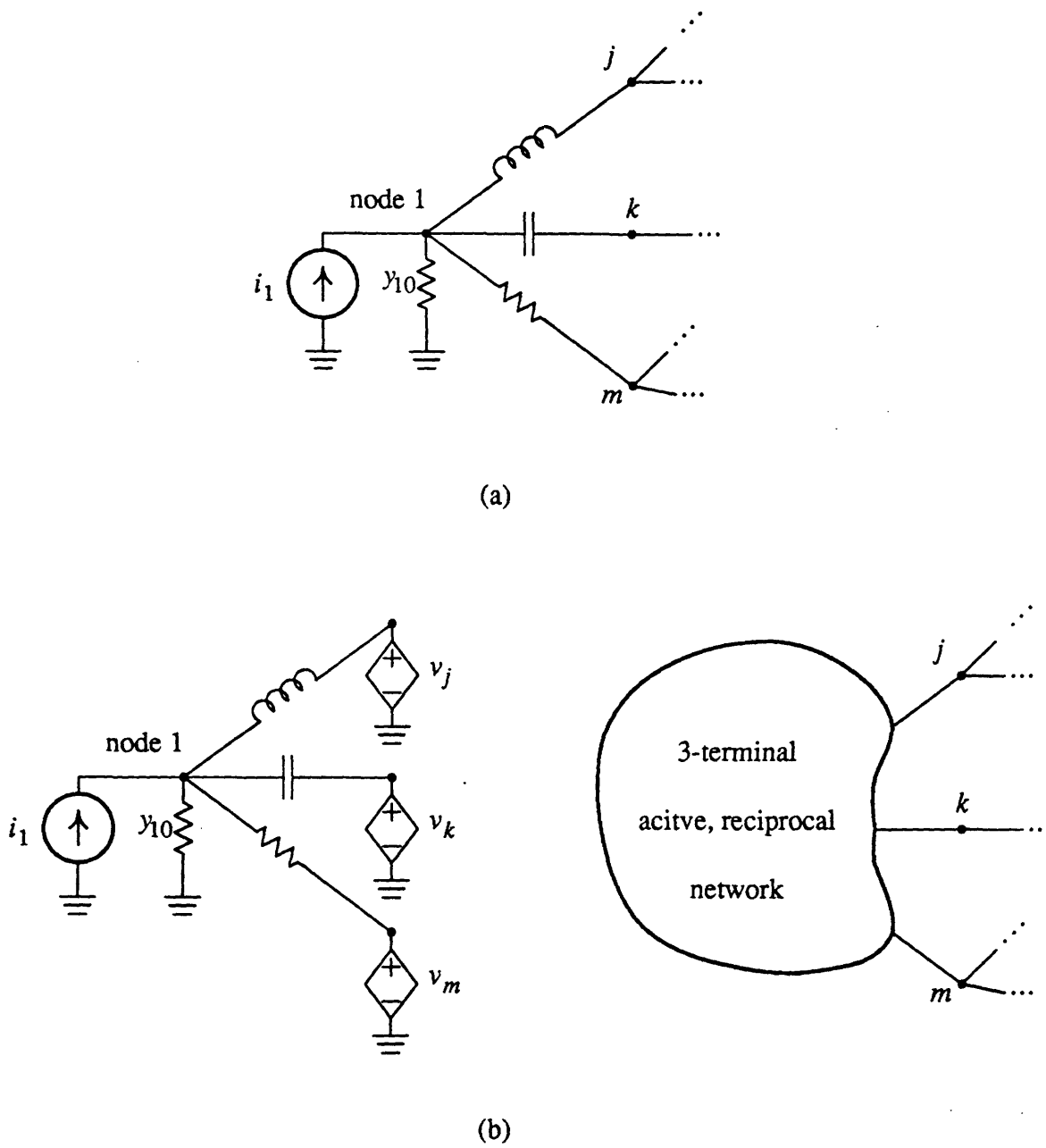


FIGURE B-1: Circuit representation of Node 1 before (a) and after (b) zeroing the first column.

The dependent source paths are reflected in the non-symmetric first row and column. The remaining  $n - 1$  rows and columns of  $\tilde{Y}(s)$  and  $n - 1$  rows of  $\tilde{i}(s)$  contain the sum of the original circuit and the 3-terminal active, reciprocal network which models Node 1 and its interconnections. We can see the original network terms and 3-terminal terms directly in the forward elimination equations.

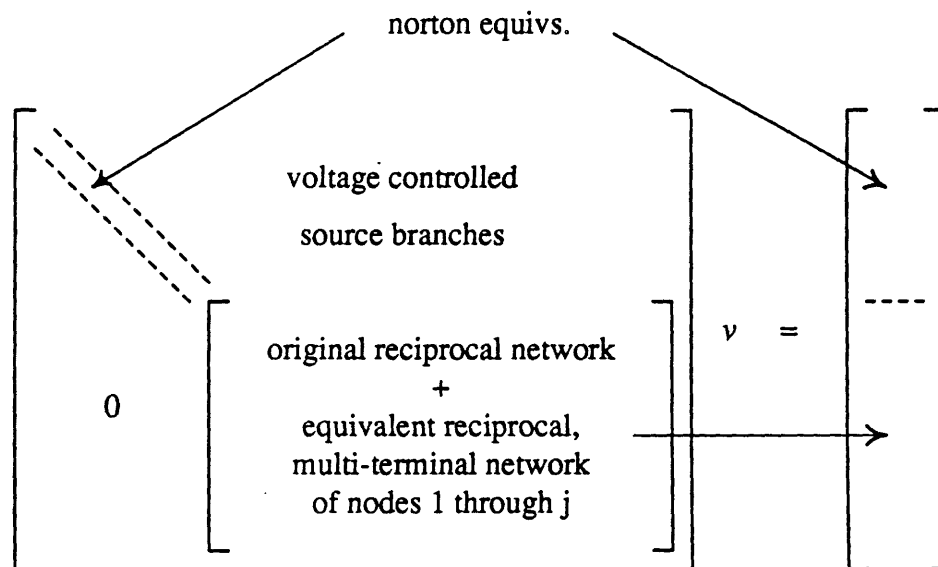
$$\begin{aligned} \tilde{y}'_{km}(s) &= \underbrace{\tilde{y}_{km}(s)}_{\text{original network}} - \underbrace{\frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \tilde{y}_{jm}(s)}_{\text{3-terminal equivalent network}} \\ \tilde{i}'_k(s) &= \underbrace{\tilde{i}_k(s)}_{\text{original network}} - \underbrace{\frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \tilde{i}_j(s)}_{\text{3-terminal equivalent network}} \end{aligned}$$

As expected, the admittance matrix for the 3-terminal network,

$$\frac{-1}{\tilde{y}_{11}(s)} \begin{bmatrix} \tilde{y}_{j1}(s) \tilde{y}_{1j}(s) & \tilde{y}_{j1}(s) \tilde{y}_{1k}(s) & \tilde{y}_{j1}(s) \tilde{y}_{1m}(s) \\ \tilde{y}_{k1}(s) \tilde{y}_{1j}(s) & \tilde{y}_{k1}(s) \tilde{y}_{1k}(s) & \tilde{y}_{k1}(s) \tilde{y}_{1m}(s) \\ \tilde{y}_{m1}(s) \tilde{y}_{1j}(s) & \tilde{y}_{m1}(s) \tilde{y}_{1k}(s) & \tilde{y}_{m1}(s) \tilde{y}_{1m}(s) \end{bmatrix}$$

is symmetric (reciprocal) if the original network is symmetric (reciprocal).

Extending this to any step of the forward elimination process, we have a matrix equation of the form:



This continues until the last node is represented by only a current source driving a single admittance. During the backward elimination steps, the node voltages are easily evaluated by their equivalent, single-node circuits.

From the above discussion, we can extend the theorems of the previous section to the nodal analysis matrix equation during any step of the gaussian elimination process. Theorems B.2

and B.4 still hold at any time, since their proofs depend only on a node's equivalent admittance to ground and any other admittances between it and other node voltages. Theorem B.3 also holds for off-diagonal terms *where defined*, since an off-diagonal term still represents an admittance connected between two valid node voltages, and hence, Equation (B.6) still holds.

### B.3 Truncation Orders for Gaussian Elimination

We are now ready to compute the necessary truncation orders for gaussian elimination operations involving moment representation polynomials. This is done by systematically applying the truncation rules found in Appendix Section A.3 to the gaussian elimination equations.

#### B.3.1 Backward elimination

This will be done backwards, by first considering the backward elimination equation,

$$\tilde{v}_k(s) = \frac{\tilde{i}_k(s) - \sum_{m=k+1}^n \tilde{v}_m(s) \tilde{y}_{km}(s)}{\tilde{y}_{kk}(s)}. \quad (\text{B.7})$$

We want a final answer to order  $p$ , thus

$$\mathcal{T}[\tilde{v}_k(s)] = p.$$

By recursively applying truncation rules:

$$\mathcal{T} \left[ \frac{\tilde{i}_k(s) - \sum_{m=k+1}^n \tilde{v}_m(s) \tilde{y}_{km}(s)}{\tilde{y}_{kk}(s)} \right] = p \quad (\text{B.8})$$

$$1. \mathcal{T} \left[ \tilde{i}_k(s) - \sum_{m=k+1}^n \tilde{v}_m(s) \tilde{y}_{km}(s) \right] = p + [\tilde{y}_{kk}(s)], \quad (\text{B.9})$$

which, from Theorem B.2, gives

$$\mathcal{T} \left[ \tilde{i}_k(s) - \sum_{m=k+1}^n \tilde{v}_m(s) \tilde{y}_{km}(s) \right] = p + [\tilde{y}_{kk}(s)] \leq p + \mathcal{B}_k. \quad (\text{B.10})$$

$$(a) \mathcal{T}[\tilde{i}_k(s)] = p + [\tilde{y}_{kk}(s)] \leq p + \mathcal{B}_k, \quad \text{for } m > k \quad (\text{B.11})$$

$$(b) \mathcal{T}[\tilde{v}_m(s) \tilde{y}_{km}(s)] = p + [\tilde{y}_{kk}(s)] \leq p + \mathcal{B}_k, \quad \text{for } m > k \quad (\text{B.12})$$

$$i. \mathcal{T}[\tilde{y}_{km}(s)] \leq p + \mathcal{B}_k - [\tilde{v}_m(s)], \quad \text{for } m > k \quad (\text{B.13})$$

From circuit restriction 3,  $[\tilde{v}_m(s)] \geq 0$ , and

$$\mathcal{T}[\tilde{y}_{km}(s)] \leq p + \mathcal{B}_k, \quad \text{for } m > k \quad (\text{B.14})$$

$$ii. \mathcal{T}[\tilde{v}_m(s)] = p + [\tilde{y}_{kk}(s)] - [\tilde{y}_{km}(s)], \quad \text{for } m > k \quad (\text{B.15})$$

But, from Theorem B.3,  $[\tilde{y}_{kk}(s)] - [\tilde{y}_{km}(s)] \leq 0$ , so

$$\mathcal{T}[\tilde{v}_m(s)] = p, \quad \text{for } m > k \quad (\text{B.16})$$

which matches Equation (B.8).

$$2. \mathcal{T} [\tilde{y}_{kk}(s)] = p - \left[ \sum_{m=k+1}^n \tilde{v}_m(s) \tilde{y}_{km}(s) \right] + 2 [\tilde{y}_{kk}(s)] \quad (\text{B.17})$$

From Theorem B.4, we know that  $[\tilde{i}_k(s)] \geq [\tilde{y}_{kk}(s)]$  and from Theorem B.3 and circuit restriction 3,  $[\tilde{v}_m(s) \tilde{y}_{km}(s)] \geq [\tilde{y}_{kk}(s)]$  for any  $m > k$ . So,

$$\mathcal{T} [\tilde{y}_{kk}(s)] \leq p + [\tilde{y}_{kk}(s)] \leq p + \mathcal{B}_k \quad (\text{B.18})$$

In summary, by the definition of  $p$ , the result minimum order, we can say

$$\mathcal{T} [\tilde{v}_k(s)] = p \quad \text{for any } k. \quad (\text{B.19})$$

In doing the operations of Equation (B.7), the truncation order for:

$$\begin{cases} \text{division} & = p & (\text{from Equation (B.8)}) \\ \text{summation and subtraction} & = p + \mathcal{B}_k & (\text{from Equation (B.10)}) \\ \text{multiplication} & = p + \mathcal{B}_k & (\text{from Equation (B.12)}) \end{cases}$$

For backward elimination to work properly, forward elimination must give the following truncation orders. From Equation (B.11),

$$\mathcal{T} [i_k] = p + \mathcal{B}_k \quad \text{for any } k, \quad (\text{B.20})$$

and from Equations (B.14) and (B.18),

$$\mathcal{T} [\tilde{y}_{km}(s)] = p + \mathcal{B}_k \quad \text{for any } k \text{ and } m \geq k. \quad (\text{B.21})$$

### B.3.2 Forward Elimination

In the forward elimination phase, we use

$$\tilde{y}'_{km}(s) = \tilde{y}_{km}(s) - \left( \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \right) \tilde{y}_{jm}(s) \quad (\text{B.22})$$

to calculate new terms of the  $Y(s)$  matrix. The starting point for this,

$$\mathcal{T} [\tilde{y}'_{km}(s)] = p + \mathcal{B}_k \quad (\text{B.23})$$

is from Equation (B.21).

Applying truncation rules to this:

$$\mathcal{T} \left[ \tilde{y}_{km}(s) - \left( \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \right) \tilde{y}_{jm}(s) \right] = p + \mathcal{B}_k \quad (\text{B.24})$$

$$1. \mathcal{T} [\tilde{y}_{km}(s)] = p + \mathcal{B}_k \quad (\text{B.25})$$

$$2. \mathcal{T} \left[ \left( \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \right) \tilde{y}_{jm}(s) \right] = p + \mathcal{B}_k \quad (\text{B.26})$$

$$(a) \mathcal{T} [\tilde{y}_{jm}(s)] = p + \mathcal{B}_k - \left\lfloor \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \right\rfloor \quad (\text{B.27})$$

$$\mathcal{T} [\tilde{y}_{jm}(s)] = p + \mathcal{B}_k - \lfloor \tilde{y}_{kj}(s) \rfloor + \lfloor \tilde{y}_{jj}(s) \rfloor \quad (\text{B.28})$$

We now wish to show that this equals  $p + \mathcal{B}_j$ . This is done by considering two cases:

**Case 1**  $\mathcal{B}_j \geq \mathcal{B}_k$

Since  $\lfloor \tilde{y}_{jj}(s) \rfloor - \lfloor \tilde{y}_{kj}(s) \rfloor \leq 0$  for all reciprocal *RLC* circuits (Theorem B.3), Equation (B.28) becomes

$$\mathcal{T} [\tilde{y}_{jm}(s)] \leq p + \mathcal{B}_k \quad (\text{B.29})$$

and by definition of Case 1,

$$\mathcal{T} [\tilde{y}_{jm}(s)] \leq p + \mathcal{B}_j. \quad (\text{B.30})$$

**case 2**  $\mathcal{B}_j < \mathcal{B}_k$

If this case is true, then by the definition of  $\mathcal{B}_k$ , there must not be an element connected to node  $k$  with order less than  $\mathcal{B}_k$ , or  $\lfloor \tilde{y}_{kj}(s) \rfloor \geq \mathcal{B}_k$ . Thus Equation (B.28) becomes

$$\mathcal{T} [\tilde{y}_{jm}(s)] \leq p + \lfloor \tilde{y}_{jj}(s) \rfloor \leq p + \mathcal{B}_j. \quad (\text{B.31})$$

$$(b) \mathcal{T} \left[ \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \right] = p + \mathcal{B}_k - \lfloor \tilde{y}_{jm}(s) \rfloor \quad (\text{B.32})$$

From Theorem B.3,

$$\mathcal{T} \left[ \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \right] = p + \mathcal{B}_k - \lfloor \tilde{y}_{jj}(s) \rfloor. \quad (\text{B.33})$$

$$i. \mathcal{T} [\tilde{y}_{kj}(s)] = p + \mathcal{B}_k - \lfloor \tilde{y}_{jm}(s) \rfloor + \lfloor \tilde{y}_{jj}(s) \rfloor \quad (\text{B.34})$$

From Theorem (B.3),

$$\mathcal{T} [\tilde{y}_{kj}(s)] = p + \mathcal{B}_k \quad (\text{B.35})$$

$$ii. \mathcal{T} [\tilde{y}_{jj}(s)] = p + \mathcal{B}_k - \lfloor \tilde{y}_{jm}(s) \rfloor - \lfloor \tilde{y}_{kj}(s) \rfloor + 2 \lfloor \tilde{y}_{jj}(s) \rfloor \quad (\text{B.36})$$

Since  $\lfloor \tilde{y}_{jm}(s) \rfloor \geq \lfloor \tilde{y}_{jj}(s) \rfloor$ ,

$$\mathcal{T} [\tilde{y}_{jj}(s)] = p + \mathcal{B}_k - \lfloor \tilde{y}_{kj}(s) \rfloor + \lfloor \tilde{y}_{jj}(s) \rfloor \quad (\text{B.37})$$

and since this is similar to Equation (B.28), we use the same reasoning to conclude

$$\mathcal{T} [\tilde{y}_{jj}(s)] = p + \mathcal{B}_j \quad (\text{B.38})$$

In summary of the forward elimination section, we can say that when calculating any term of  $Y(s)$  during forward elimination, it is sufficient (from Equations (B.25), (B.30), (B.31), (B.35) and (B.38)) to truncate at

$$\mathcal{T} [\tilde{y}_{ky}(s)] = p + \mathcal{B}_k.$$



In performing the actual calculation of Equation (B.22) on row  $k$ , the truncation order for

$$\begin{cases} \text{subtraction} & = p + \mathcal{B}_k & (\text{from Equation (B.24)}) \\ \text{multiplication} & = p + \mathcal{B}_k & (\text{from Equation (B.26)}) \\ \text{division} & = p + \mathcal{B}_k - \lfloor y_{kk} \rfloor & (\text{from Equation (B.33)}) \end{cases}$$

Following the same forward elimination steps one can also show that the same truncation orders are needed for calculating the terms of  $\tilde{i}(s)$  during forward elimination with

$$\tilde{i}'_k(s) = \tilde{i}_k(s) - \left( \frac{\tilde{y}_{kj}(s)}{\tilde{y}_{jj}(s)} \right) \tilde{i}_j(s).$$



---

---

## Using Moments as Macromodel Parameters

This appendix examines the feasibility of using moment representation terms directly as macromodel parameters. Advantages for doing this are apparent for large approximation orders, since the time-consuming moment representation to time domain conversion could be avoided. It is shown that this is not a practical alternative to doing the conversion from moment representation to time domain.

The moment representation is a global descriptor of waveforms—meaning that the moment terms are used to describe the *overall* waveform shape. Furthermore, the moment terms are not independent—a minor change in one without changing the other often results in an extreme change in the calculated waveform. (This demonstrates the importance of developing the truncation rules in Chapter 4.) Yet, when looking at the effects of input waveforms on non-linear digital circuits, it is often only a local time fragment of the waveform that is important. Time fragments are not readily extracted from the moments.

The possibility of macromodeling with moment representation terms is not an important issue for small approximation orders, like  $n = 1$  or  $n = 2$ . At these orders, macromodel functions are smooth, but as one can see in Table 3-3, the computations to convert a moment representation to time domain are already trivial. The big computational savings are seen with approximation orders of three or more—in the rest of this discussion, we consider only the possibility of macromodeling the double exponential assumed waveform shape.

We start by considering only a subset of the desired problem—we will attempt to macromodel the slope of a waveform through a fixed critical region with moment terms as parameters. It is possible to reduce the number of macromodel parameters to two for third-order monotonic signal waveforms (or more generally, for third-order waveforms with positive non-zero  $M_0$  and  $M_1$  terms.) This reduction in number of variables is achieved by normalizing voltages by  $M_0$

and times by  $M_1/M_0$ . This leaves two variables:

$$X_2 = \frac{M_2 M_0}{M_1^2}$$

and

$$X_3 = \frac{M_3 M_0^2}{M_1^3}.$$

We can undo this normalization at the end by multiplying the normalized slope by  $M_1/M_0$ .

A plot of normalized waveform slope through a 40%–60% critical region versus  $X_2$  and  $X_3$  is shown in Figure C-1. A large number of function values are not shown—these are at points where  $X_2$  and  $X_3$  combinations have complex poles or are otherwise impossible combinations. In general, the plot shows macromodel functions which jump around in value. The functions clustered towards the middle of the plot are from waveforms exhibiting single time constant behavior. Figure C-1 also shows the specific locations of a few simple waveforms. Note that two of the specific waveforms are close to function discontinuities.

We expect macromodel functions for nonlinear elements to be equally or more irregular, since these are originally defined in terms of the slope through a critical region.

The above example does not deal with a number of important conditions, namely (1) non-monotonic waveforms, (2) monotonic waveforms with parameters falling outside the range of  $0.1 \leq X_2 \leq 3.0$  and  $0.2 \leq X_3 \leq 3.2$ , (3) different critical voltage ranges, and (4) different assumed waveform types. A change in any of the above conditions would necessitate a new macromodel table or an additional macromodel parameter.

The above example demonstrates reasons for not using this type of macromodeling. First, macromodel functions with third order moment terms for parameters jump around substantially. The accuracy of macromodel function evaluation is poor, since accurate interpolation is difficult at discontinuities. Second, large macromodel function tables are needed to cope with the large number of variables, and large number of data points. To counteract function irregularity, a large number of points is required.

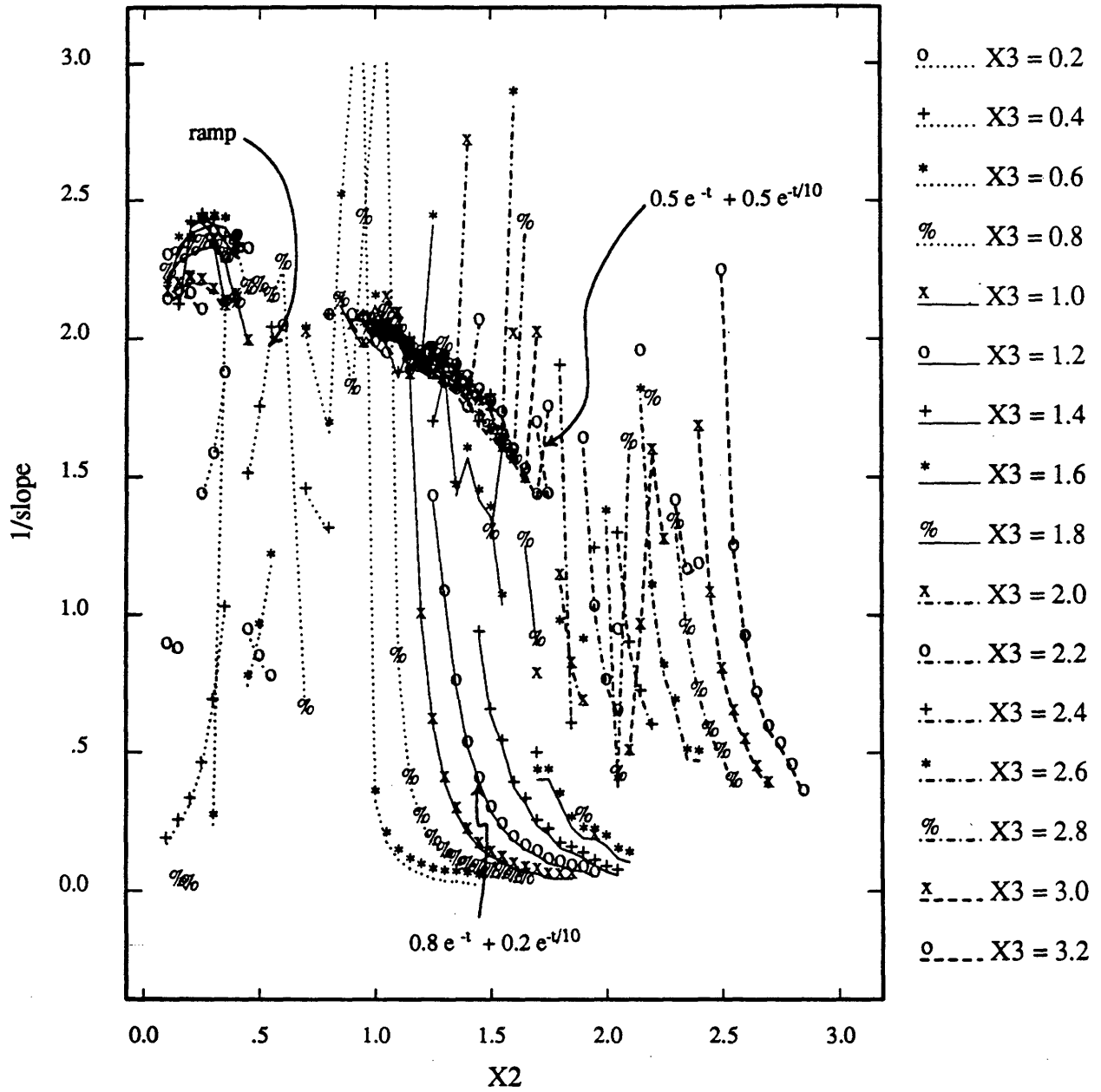


FIGURE C-1: Macromodel plot of critical region waveform slope vs. X<sub>2</sub> and X<sub>3</sub>.

Values are computed from the double exponential waveform shape.



---

---

## Bibliography

- [1] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Tech. Rep. ERL-M520, University of California, Berkeley, May 1975.
- [2] A. E. Ruehli, "Survey of computer-aided electrical analysis of integrated circuit interconnections," *IBM Journal of Research and Development*, vol. 23, pp. 626-639, November 1979.
- [3] J. D. Bastian *et al.*, "Symbolic parasitic extractor for circuit simulation (SPECS)," in *Proceedings of the 20<sup>th</sup> Design Automation Conference*, (San Diego), pp. 346-352, June 1983.
- [4] S. P. McCormick, *Automated Circuit Extraction from Mask Descriptions of MOS Networks*. Master's thesis, MIT Department of Electrical Engineering and Computer Science, February 1984.
- [5] S. P. McCormick, "EXCL: a circuit extractor for IC designs," in *Proceedings of the 21<sup>st</sup> Design Automation Conference*, (Albuquerque, New Mexico), pp. 624-628, June 1984.
- [6] C. Baker, *Artwork Analysis Tools for VLSI Circuits*. Master's thesis, M.I.T., June 1980.
- [7] W. S. Scott, "Magic's circuit extractor," in *Proceedings of the 22<sup>nd</sup> Design Automation Conference*, (Las Vegas), pp. 286-292, June 1985.
- [8] W. T. Weeks, A. J. Jininez, G. W. Mahoney, D. Mehta, H. Qassemzadeh, and T. R. Scott, "Algorithms for ASTAP—A network analysis program," *IEEE Transactions on Circuit Theory*, vol. CT-20, pp. 628-634, November 1973.
- [9] B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS—An MOS timing simulator," *IEEE Transactions on Circuits and Systems*, vol. CAS-22, pp. 901-910, December 1975.
- [10] A. R. Newton, "Techniques for the simulation of large-scale integrated circuits," *IEEE Transactions on Circuits and Systems*, vol. CAS-26, pp. 741-749, September 1979.
- [11] J. White and A. L. Sangiovanni-Vincetelli, "RELAX2.1: A waveform relaxation based circuit simulation program," in *Proceedings of the 1984 Custom Integrated Circuits Conference*, pp. 232-236, May 1984.
- [12] C. J. Terman, *Simulation Tool for Digital LSI Design*. PhD thesis, Massachusetts Institute of Technology, September 1983.

- [13] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *Journal of Applied Physics*, vol. 19, pp. 55-63, January 1948.
- [14] P. Penfield, Jr. and J. Rubinstein, "Signal delay in RC tree networks," in *Proceedings of the Second Caltech Conference on VLSI*, (Pasadena, CA), p. , January 1981.
- [15] J. L. Wyatt, Jr., "Signal delay in RC mesh networks," *IEEE Transaction on Circuits and Systems*, vol. CAS-32, pp. 28-33, May 1985.
- [16] T. Lin and C. A. Meade, "Signal delay in general RC networks," *IEEE Transaction on Computer-Aided Design*, vol. CAD-3, pp. 331-349, October 1984.
- [17] Q. Yu, J. L. Wyatt, Jr., C. Zukowski, H. N. Tan, and P. O'Brien, "Improved bounds on signal delay in linear RC models for MOS interconnect," in *Proc. 1985 Int'l. Symposium on Circuits and Systems*, (Kyoto, Japan), pp. 903-906, June 1985.
- [18] M. A. Horowitz, *Timing Models for MOS Circuits*. PhD thesis, Stanford University, December 1983.
- [19] A. R. Djordjević, T. K. Sarkar, and R. F. Harrington, "Time-domain response of multi-conductor transmission lines," *Proceedings of the IEEE*, vol. 75, pp. 743-764, June 1987.
- [20] J. E. Adair and G. I. Haddad, "Coupled-mode analysis of nonuniform coupled transmission lines," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-17, pp. 746-752, October 1969.
- [21] Y. E. Yang, J. A. Kong, and Q. Gu, "Time-domain perturbational analysis of nonuniformly coupled transmission lines," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-33, pp. 1120-1130, November 1985.
- [22] A. J. Gruodis, "Transient analysis of uniform resistive transmission lines in a homogeneous medium," *IBM Journal of Research and Development*, vol. 23, pp. 675-681, November 1979.
- [23] K. Kundert, A. L. Sangiovanni-Vincentelli, and J. White, "A mixed frequency time approach for finding the steady-state solution of clocked analog circuits," in *Proceedings of the 1988 Custom Integrated Circuits Conference*, pp. 6.2.1-6.2.4, May 1988.
- [24] R. E. Bryant, *A Switch-Level Simulation Model for Integrated Logic Circuits*. PhD thesis, Massachusetts Institute of Technology, June 1981.
- [25] S. R. Nassif and S. W. Director, "WASIM: a waveform based simulator for VLSIC's," in *Proceedings of the ICCAD*, (Santa Clara, Calif.), pp. 29-31, IEEE, November 1985.
- [26] L. M. Brocco, *Macromodelling CMOS Circuits for Timing Simulation*. Master's thesis, M.I.T., June 1987.
- [27] A. S. Bergendahl, C. B. J. Donlan, J. F. McDonald, R. H. Steinworth, and G. F. Taylor, "A thick-film lift-off technique for high frequency interconnection in wafer scale integration," in *Proc. of the VLSI Interconnection Conference*, pp. 154-162, IEEE, June 25-26 1985.
- [28] A. Deutsch and C. W. Ho, "Thin film interconnection lines for VLSI packaging," in *International Conference on Computer Design*, (Port Chester, New York), pp. 222-226, IEEE, October 1983.



- [29] H. Hasegawa, M. Furukawa, and H. Yanai, "Properties of microstrip line on Si-SiO<sub>2</sub> system," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-19, pp. 869-881, November 1971.
- [30] S. Seki and H. Hasegawa, "Pico-second pulse response of interconnections of high-speed GaAs and Si LSI's," in *GaAs IC Symposium*, pp. 119-122, ??? 1982.
- [31] A. J. Blodgett and D. R. Barbour, "Thermal conduction module: a high-performance multilayer ceramic package," *IBM Journal of Research and Development*, vol. 26, pp. 30-36, January 1982.
- [32] D. P. Seraphim, "A new set of printed-circuit technologies for the IBM 3081 processor unit," *IBM Journal of Research and Development*, vol. 26, pp. 37-44, January 1982.
- [33] H. Yuan, Y. Lin, and S. Chiang, "Properties of interconnection on silicon, sapphire, and semi-insulating gallium arsenide substrates," *IEEE Transactions on Electron Devices*, vol. ED-29, pp. 639-644, April 1982.
- [34] R. H. Dennard, F. H. Gaensslen, H. N. Yu, N. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE J. Solid State Circuits*, vol. SC-9, pp. 256-268, October 1974.
- [35] W. E. Donath, "Placement and average interconnect lengths of computer logic," *IEEE Trans. Circuits and Systems*, vol. CAS-26, pp. 272-277, April 1979.
- [36] D. K. Ferry, "Interconnection lengths and VLSI," *IEEE Circuits and Devices Magazine*, vol. 1, pp. 39-42, July 1985.
- [37] P. E. Cottrell and E. M. Burtula, "VLSI wiring capacitance," *IBM Journal of Research and Development*, vol. 29, pp. 277-287, May 1985.
- [38] S. Wong, private communication.
- [39] K. Yoshihara, T. Sudo, A. Iida, T. Miyagi, S. Ooe, and T. Saito, "Cross-talk predictions and reducing techniques for high speed GaAs digital IC's," in *GaAs IC Symposium Proceedings*, (Boston), pp. 23-26, IEEE, October 1984.
- [40] Q. Gu and J. A. Kong, "Transient analysis of single and coupled lines with capacitively-loaded junctions," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-34, pp. 952-964, September 1986.
- [41] W. R. Smith and D. E. Snyder, "Circuit loading and crosstalk signals from capacitance in SOS and bulk-silicon interconnect channels," in *VLSI Multilevel Interconnection Conference*, (New Orleans), pp. 218-227, IEEE, June 1984.
- [42] B. J. Rubin, "The propagation characteristics of signal lines in a mesh-plane environment," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-32, pp. 522-531, May 1984.
- [43] Q. Gu and J. A. Kong, "Transient analysis of single and coupled lines with capacitively-loaded junctions," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-34, pp. 952-964, September 1986.

- [44] J. L. Wyatt, Jr., *Signal Propagation Delay in RC Models for Interconnect*, ch. 11.2, pp. 254–290. *Advances in CAD for VLSI, Volume 3 - Part 2*, North Holland: Elsevier Science Publishers B. V., 1987.
- [45] W. M. Siebert, *Circuits, Signals and Systems*. New York: McGraw-Hill, 1986.
- [46] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*. New York: van Nostrand Reinhold Company, 1983.
- [47] P. Penfield, Jr., "Signal delay in RC tree networks," in *Proceedings of the 18<sup>th</sup> Design Automation Conference*, p. 613, June 1981.
- [48] C. Chu and M. A. Horowitz, "Charge sharing models for MOS circuits," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 274–277, November 1986.
- [49] A. L. Sangiovanni-Vincentelli, *Computer Design Aids for VLSI Circuits*, pp. 19–112. The Hague, Netherlands: Martinus Nijhoff, 1984.
- [50] W. W. Happ, "Time-domain analysis and measurement techniques for distributed RC structures. I. analysis in the reciprocal time domain," *Journal of Applied Physics*, vol. 40, pp. 109–117, January 1969.
- [51] C. G. Cullen, *Matricies and Linear Transformations*. Reading, Mass.: Addison-Wesley, 1972.
- [52] R. E. Bryant, "An algorithm for MOS logic simulation," *Lambda*, pp. 46–53, Fourth Quarter 1980.
- [53] C. W. Ho, "Theory and computer-aided analysis of lossless transmission lines," *IBM Journal of Research and Development*, vol. 17, pp. 249–255, May 1973.
- [54] L. G. Kelly, *Handbook of Numerical Methods and Applications*, ch. 9. Reading, Mass.: Addison-Wesley, 1967.
- [55] F. H. Branin, Jr., "Transient analysis of lossless transmission lines," *Proceedings of the IEEE*, vol. 55, pp. 2012–2013, November 1967.
- [56] L. M. Brocco, S. P. McCormick, and J. Allen, "Macromodelling CMOS circuits for timing simulation," *IEEE Transaction on Computer-Aided Design*, vol. 7, pp. 1237–1249, December 1988.
- [57] M. D. Matson, *Macromodelling and Optimization of Digital MOS VLSI Circuits*. PhD thesis, Massachusetts Institute of Technology, February 1985.
- [58] D. Overhauser and I. Hajj, "A tabular approach to fast timing simulation including parasitics," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 70–73, November 1988.
- [59] P. R. O'Brien, J. L. Wyatt, Jr., T. L. Savarino, and J. M. Pierce, "Fast on-chip delay estimation for cell-based emitter coupled logic," in *Proc. IEEE Int. Symp. on Circuits and Systems*, (Helsinki, Finland), pp. 1357–1360, June 1988.