Macromodeling CMOS Circuits for Timing Simulation

RLE Technical Report No. 529

June 1987

Lynne Michelle Brocco

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

This work was supported in part by the U.S. Air Force (Grant AFOSR-86-0164).

Macromodeling CMOS Circuits for Timing Simulation

by

Lynne Michelle Brocco

B.S. Electrical Engineering
University of Akron
(1984)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

at the

Massachusetts Institute of Technology
June 1987

Signature of Author Lynne Michelle Brocco
Department of Electrical Engineering and Computer Science
May 9, 1987

Certified by Jonathan Allen
Jonathan Allen
Thesis Supervisor

Accepted by ______________________
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Macromodeling CMOS Circuits for Timing Simulation

by

Lynne Michelle Brocco

Submitted to the
Department of Electrical Engineering and Computer Science
on May 13, 1987 in partial fulfillment of the requirements
for the Degree of Master of Science

Abstract

A macromodeling and timing simulation technique is presented that allows fast, accurate delay calculations for CMOS circuits. This method is well suited for delay calculations of regular structure VLSI circuits, as well as circuits designed from standard cell libraries. Timing models for both logic gate and transmission gate circuit forms are developed. For logic gates, output transition time and delay time are functions of input transition time and load impedance. Effective resistances for conducting transmission gates and switching transmission gates are functions of input transition time and load capacitance. Transmission gate circuits are then modeled as equivalent RC circuits. Separate waveform models and delay calculation methods exist for both types of circuit forms, with an interface to enable the use of both methods in the same simulation. An experimental event-driven simulator was developed to test the accuracy of the macromodels and to estimate improvements in execution time with respect to SPICE. Typical delay times were within 5% for logic gate circuits and 10% for transmission gate circuits when compared with SPICE. The execution time of the experimental simulator was over two orders of magnitude faster than SPICE.

Thesis Supervisor: Jonathan Allen

Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to thank my thesis supervisor, Jonathan Allen, for his guidance and vision throughout the course of my work.

Thanks to Bob Armstrong, who makes himself invaluable on the "eighth floor" with his many hacking skills, and to Don Baltus, who pointed me in the right direction.

I also appreciate the general support I received from all the people in the VLSI group—Lance Glasser, John Wyatt, Adam Malamy, Charles Selvidge, Cyrus Bamji, Barry Thompson, Mark Reichelt, Peter O'Brien, Dave Standley, and others.

Most of all, I thank Steven McCormick, my fiancé, for his valuable technical advice and, even more importantly, his love and emotional support.

This research was supported in part by a GE Foundation Fellowship and in part by Air Force Office of Scientific Research Grant AFOSR-86-0164.

Table of Contents

List of Figures

List of Tables

CHAPTER ONE

Introduction

As VLSI circuits grow larger and the design task becomes more complex, effective and efficient computer-aided design tools are becoming more of a necessity. This is true for many aspects of the design and implementation of a VLSI chip. One important aspect is timing of VLSI circuits.

Timing information is necessary not only in the evaluation stage of VLSI circuit development, but in the design stage as well. Designers want quick, accurate timing results in order to adequately explore all the topological possibilities for given circuit blocks. Thus, there is a necessity for design tools that allow alternate architecture exploration during the design phase of a VLSI project.

A macromodeling and timing simulation technique is developed in this paper that provides fast, accurate timing results for CMOS circuits. This method is well suited for evaluating regular structure circuits, in which a circuit is composed of orderly connections of many instances of the same cell type. It also works well for CMOS circuits designed using standard cell libraries, as delay information could be incorporated as part of the library.

The goals of this research include the development of a macromodeling technique for logic gate and transmission gate structures. The technology used in this thesis is a 2 micron CMOS process given in Glasser and Dobberpuhl's *The Analysis and Design of VLSI Circuits*. Sample macromodels representing several logic gate and transmission gate circuits are implemented in an experimental timing simulator. These macromodels represent a fixed technology, that is, the two-micron technology described above. Simulation of circuits using different CMOS technologies requires the derivation of new macromodel parameter sets. This experimental simulator will be used to evaluate delay accuracy of the models and improvements in execution time compared to SPICE.

1.1 Previous Work

The development of timing evaluation techniques for MOS circuits has propagated in many directions. There are many modeling methods and many simulation techniques. The following paragraphs summarize the major thrusts of this very active research area and to justify the approach taken in this thesis.

1.1.1 Modeling MOS Circuits

There are different ways to classify timing models. There are models used for *analysis* and models used for *simulation. Circuit analysis* uses models which are believed to be mathematically accurate. *Simulation models* are more approximate, usually geared toward calculating the output information desired. Timing simulation models are also defined at several levels, such as the transistor level, the gate level, or a sub-circuit level.

1.1.1.1 Circuit Analysis Models

Circuit analyzers such as SPICE [1] and ASTAP [2] are based on mathematical equations modeling the devices in the circuit. These more general circuit simulators form network equations for the circuit and use general integration techniques. This type of analysis requires much storage and computation time. Circuit size may be restricted to no more than a few hundred nodes. Thus, analysis may also be restricted to only select paths of the circuit. However, these programs are capable of 95 to 99% accuracy when compared to actual results [3].

1.1.1.2 Switch Level Models

On the other extreme, in terms of model complexity, are switch level models. In general, circuits are modeled as connections of bi-directional switches representing transistors. Bryant [4] developed the switch level model initially for a logic simulation program, MOSSIM. In this modeling scheme, a network is a set of nodes connected by transistor switches. Each node has a state 0, 1, or X and each transistor has a state *open, closed*, or *indeterminate*. This was an improvement over gate-type logic simulators as simulation of more general circuit connections became possible.

RSIM [5] incorporated delay calculation information in the switch-level model. A transistor is modeled as a switch in series with a resistor. Transistor groups, or closely connected sub-networks of transistors, are analyzed statically for output states. The total delay of the sub-network is then calculated from the RC time constant of the node. The

RC circuit models may be inaccurate, since only the resistance and capacitance of the node in question is considered, ignoring other nodes in the RC network.

Crystal [6, 7], another switch level timing simulator, is used for timing verification. Crystal locates slowest paths with a value independent, switch-level approach. Tables are then used to calculate series resistances, based on input rise time, transistor type and size, and output load. Intrinsic (step-input) rise time estimates are used, resulting in an average error in rise times of 30%. Typical delay time errors of less than 10% are claimed when compared to SPICE.

Rao, Trick, and Hajj [8] derived a switch-level simulator that uses table-driven delay operators for timing information. After circuit partitioning into sources, logic gate blocks, and pass transistor blocks, a switch-level simulation is performed to evaluate transition sequences on output nodes. The transition sequences, along with circuit parameters, such as load capacitance and transistor length and width, are passed to the delay operator. The delay operator yields the delay of the transitions by table lookup. No characterizations of RC circuits were evident.

Sundblad and Svensson [9] proposed an extended local relaxation algorithm (ELR) in order to make switch level timing simulation fully dynamic. While other switch-level simulators use the closely-connected transistor group as the basic element to be analyzed, the ELR algorithm treats a transistor or bi-directional element as any other element. The relaxation process follows the natural transient behavior of the circuit and is used as the simulation inside the transistor groups. Simultaneous propagation of several signals inside the transistor group is allowed. This method, when combined with a local timing algorithm for simulation between the groups, results in a fully dynamic switch-level simulation.

A different type of model was proposed for a switch-level simulator by Ruan and Vlach [10]. Each transistor is replaced by a model equivalent to a constant current source instead of a resistor. Neither numerical integration nor transistor model evaluation is needed. Output time responses are represented as piece-wise linear segments.

ELOGIC [11] is an electrical logic simulator which uses switch-level modeling and analysis techniques. Each switch-level element is modeled by a state transition table mapping the input voltages to an output Norton equivalent circuit. A circuit node model transforms the Norton equivalent circuit into a node voltage. ELOGIC is different from many simulators because it uses voltage as the independent variable and time as the dependent variable.

In general, switch-level models require little information to model transistors, usually only a switch and a resistor. However, since a flat transistor level representation of a circuit is used, the representation a large circuit can be very cumbersome. There appears to be no exploitation of regularity using these models. Also, errors in delay results can be quite large for this simple model.

1.1.1.3 RC Tree Modeling

Signal delay through RC circuits has been explored with emphasis on delay through RC tree models for interconnect and CMOS circuits. Penfield, Rubinstein, and Horowitz [12, 13] found a computationally simple technique for finding upper and lower bounds on delay in RC trees. This method is useful for MOS interconnect lines with fanout. Resistances and capacitances are assumed to be linear. Results may be used to (1) bound the delay, (2) bound the signal voltage, given a delay time, or (3) verify that a circuit performs faster than a given maximum delay.

Horowitz [14, 15] developed expressions for second order waveform approximations in conducting and switching RC trees. He also developed timing models for pass transistors, thus expanding the RC delay work to handle non-linear resistors. This is done by transforming the voltage to make a linear problem. In addition, Horowitz included methods to analyze slow inputs into inverters.

Lin and Mead [16, 17, 18] use RC circuits to model delay in MOS circuits. They developed signal delay calculation methods for general RC meshes, which is a more general circuit form than RC trees. A simulator was developed in which semantic cell representations of sub-circuits, characterized by a series resistance, a loading capacitance, and internal delay, may be composed hierarchically.

Calculation of signal delay through an RC circuit is only part of the task. An accurate RC representation for a circuit must first be found. RC modeling is convenient for modeling interconnect and also transmission gates, due to the bi-directional properties of both, as long as accurate values for resistance and capacitance are found.

1.1.1.4 Macromodeling

Macromodeling is an approach which models a piece of a circuit, or sub-circuit, by reducing the total amount of information representing that sub-circuit and keeping only the information needed to calculate the desired output variables. Macromodeling is a modular method which exploits repetitive sub-circuits. The use of macromodeling assumes circuits may be partitioned into sub-circuits, which can be represented separately while maintaining sufficient accuracy. Also, to be efficient, one must assume that circuits simulated with this method have a large number of repetitive blocks.

Macromodeling exploits approximation and simplification techniques. Circuit parameters which are not useful in calculating the desired output variables are discarded. Thus, macromodels may only be used to calculate the output variables for which they were derived. Accuracy of timing simulation using macromodels generally depends on delay data of macromodels in the cell library [3].

MOTIS [19, 20] is one of the earlier timing simulators developed that use macromodels. Improvements have been made over the years up to the present day. In general, MOTIS simulates circuits at the device level but uses methods found in logic simulators to propagate voltage signals between nodes. Table lookup methods are used to find device currents. Incremental voltages are found at each time point using nodal analysis.

A second generation MOTIS timing simulator [21] implements new methods that allow simulation of more general circuits. The new version of MOTIS also has improved speed and accuracy. Circuits are partitioned and decoupled into several small uni-directional sub-circuits. Active sub-circuits are then scheduled for simulation, which involves node voltage computation with local time step control.

MOTIS3 [22] is a mixed level, mixed mode timing simulator. Mixed level implies using models at different levels of circuit abstraction. Mixed mode means several types of simulation techniques are used in the same program. Models are developed at the behavioral level, the register transfer level, gate level, and transistor level. Modes of simulation include the unit delay mode, which is logic verification at the switch level, timing mode, which uses a simplified form of a conventional circuit simulator, and multiple delay mode, which is a logic simulation with precalculated rise and fall delays.

NEWTON [23] is a gate level simulator that uses macromodeling techniques. Delay is calculated as a function of transition times, gain and output loading. The logic models include multiple paths and are used for gate-level type circuits. Accuracies of 95 to 99% are claimed.

WASIM [24] is a waveform simulator using macromodels. The behavior of a sub-circuit is composed of static (logic) and dynamic (output response) part. Dynamic performance is done by modeling only the output stage of the macromodel. The waveform is a sum of exponentials and are curve fitted to results of a circuit simulation of the cell. The accuracy of this program was not explicitly mentioned.

Matson [25, 26] uses macromodeling as part of a circuit optimization package. Logic gate behavior is described by simple formulas based on device equations. The model formulas are curve fitted to SPICE results of the sub-circuit.

AUTODELAY [27, 28, 29] is an automatic delay calculator developed by Putatunda. It was designed for calculation of signal propagation delays along selected paths in standard cell and gate array designs. Delay and transition time of output waveforms are modeled as linear functions of load capacitance, load time constant, and input transition time. Circuit simulation of sub-circuits is used to derive parameters needed for the delay and transition time functions. AUTODELAY uses four basic algorithms: RC network synthesis to convert artwork to RC trees, RC network reduction to reduce complex RC networks to a simple network, gate delay computation, and signal path delay computation. Errors were found to be within 25% of conventional circuit simulators using AUTODELAY.

Fyson and Nichols [30] developed MASCOT, a program for verification of static and transient electrical characteristics of a network. Sub-circuits are uni-directional elements and are represented by eleven parameters derived from circuit simulations. One parameter describes the logical function of the sub-circuits, six parameters describe a 4-segment transfer function from the sub-circuit's dominant input to the output, and four parameters are used for calculation of propagation delays and rising and falling transition times of the waveforms. In the one example that was described, a 1-bit binary adder/subtractor was simulated and results differed from those of conventional circuit simulators by approximately 7%.

Most macromodeling techniques assume uni-directionality of the sub-circuits. That is, waveforms propagate from input nodes to output nodes only. Logic gates, ignoring Miller effect, fit into this category. Thus, most work has been done using logic gates and avoiding bi-directional devices such as transmission gates. Higher accuracies are also much easier to achieve when treating only logic gate forms.

1.1.2 Simulation Methods

Just as there are different modeling approaches, there are also different simulation techniques. Of course, often the selection of a particular modeling method requires a specific simulation technique.

1.1.2.1 Time Step Simulation

The time step, or incremental, simulation technique is generally used for circuit analyzers. Circuit analysis is partitioned into individual time steps. The entire circuit is analyzed during each time step. The step size is chosen to accurately model the fastest signal transitions occurring at that point in the simulation. Obviously, this technique requires a lot of computation. Much of this computation may be deemed unnecessary when considering variations in circuit activity. While a small part of the circuit may be active and signals are changing rapidly, the rest of the circuit may be very inactive but is analyzed using the same methods and step size as the active part of the circuit.

1.1.2.2 Waveform Relaxation

Waveform relaxation [31, 32] is an iterative method for circuit analysis. Each circuit is partitioned into strongly connected blocks. Each block is analyzed independently over the entire time interval using standard simulation techniques. This decomposition allows latency, or the variation of degree of activity in a sub-circuit, to be exploited. Computation using waveform relaxation tends to grow linearly with circuit complexity. Circuit size is still a major limitation, although not to the degree in incremental techniques.

1.1.2.3 Event Driven Simulation

Event driven simulation is a technique that performs delay calculation on a sub-circuit only when an input signal is applied to the sub-circuit. This type of simulation exploits the latency, or temporal sparsity, of a circuit by performing delay calculation on only those parts of the circuit that are changing state. Evaluation of inactive sub-circuits is bypassed without effecting the overall solution. Events, or pending signal transitions, are scheduled using an event queue. The amount of computation required for this method is related to how busy the event queue is. While most circuit analysis programs use the incremental approach, most simulators using macromodels use the event driven approach.

Event driven simulation, when used in conjunction with sub-division of circuits, exploits structural sparsity. The advantages of event driven simulation are maximized by sub-dividing the circuit as much as possible. MOS circuits are amenable to sub-division, due to the fact that, ignoring Miller effect, the gate of an MOS transistor is electrically isolated from the drain node, or output, of the transistor. This method is used in many modeling techniques such as macromodeling, described above. Fanout of most circuit structures is small, leading to a sparse connection matrix. Even though many incremental simulators, like SPICE, use sparse matrix techniques, computation still increases more than linearly with an increase in matrix (or circuit) size, since the entire matrix must be considered at any particular time. At any one time, event driven simulation using network sub-circuits effectively uses only a small portion of this connection matrix. This part of the matrix corresponds to the connections of the sub-circuit being simulated.

SAMSON [33] is an event driven circuit simulator that is said to have comparable waveform accuracy to SPICE while performing an order of magnitude faster due to its event driven nature. SAMSON uses two circuit models—a dormant model and an active, or alert, model. The dormant model decouples an inactive sub-circuit from the rest of the circuit. The alert model, for active sub-networks, is modeled by non-linear algebraic-differential system of subnet equations. Each sub-circuit has its own individual step size.

1.2 Overview of Thesis

The summary of timing simulation approaches in Section 1.1 supports the following conclusion. In order to quickly receive reasonably accurate results of large, fairly regular circuits, a macromodeling approach using event driven simulation should be used. Macromodeling, as stated previously, is good for accurate, reduced-information models of uni-directional logic gates.

It was also said that RC delay calculation techniques have been well developed and are useful for bi-directional networks that may be modeled as RC circuits. Interconnect and transmission gates fall into this category. Many efforts at macromodeling logic gates circuits have been done, with varying degrees of success. However, there has not been a successful effort, especially in terms of accuracy, involving macromodeling transmission gate and RC circuits, as well as logic gate circuits. In this thesis, macromodeling transmission gate circuits is done by linking accurate RC delay models to a macromodeling event driven simulation method. The rest of this document describes methods for developing delay models and delay calculation techniques for logic gate structures and bi-directional structures such as transmission gates and RC trees.

The development of macromodels is discussed in Chapter 2. This includes waveform models, logic gate models, transmission gate models, and delay calculations for the models. Chapter 3 presents the simulation methods used. Representation of sub-circuits, issues in event scheduling, and general program organization are a part of that chapter. Experimental results are the subject of Chapter 4. Comparisons of delay times and execution times are made. Finally, conclusions and possible future work on this topic are presented in Chapter 5.

CHAPTER TWO

Macromodeling

A macromodel is a *reduced data abstraction* of a circuit. This means only information necessary to calculate desired output variables is retained and the rest of the data is eliminated. In this case, the desired information is timing and logic of the cell. The timing of the cell is given in the form of output transitions, which contain delay and waveform information, including a rising/falling flag. The external variables required for calculating transitions are load impedance and input transition. Cell parameters dictate the exact function to be performed on input variables in order to yield output transitions. All circuits that are modeled in this thesis are CMOS.

Before plunging into the discussion on macromodeling, a few terms will be defined. A *cell*, also referred to as a *sub-circuit*, is part of a circuit. Macromodeling is performed upon these cells. In Chapter 3, cells will be further classified as modules and instances. The *input impedance* of a cell input is the impedance looking into that input. This impedance is generally used as part of the load impedance for another cell. The input impedance is usually purely capacitive for most logic gate structures, consisting of gate capacitance and other parasitic capacitances, and will be called the *input capacitance* of the cell. The *load impedance* of a cell is the resistive and capacitive loading on the outputs of a cell from interconnect, other cell inputs connected to the cell outputs, and other parasitics external to the cell. In general, the *output impedance* of a cell, which is the impedance looking into a cell output, is not used as part of the load impedance as its effect is accounted for in the cell parameters. This output impedance may consist of a resistive component and a capacitive component. An exception to this is made when the *effective driving impedance* of a cell is needed to create an RC tree. Again, the effective driving impedance may be composed of resistance and capacitance. The effective driving resistance is the average value of resistance seen looking into a cell output during a specific cell input transition. The effective driving resistance is usually a function of input transition. In general, any effective resistance is a particular average resistance seen by a given node for specific input transition and loading conditions.

The reduced information delay model is based on empirical results of SPICE simulations of the cells. In general, the delay model is a piece-wise linear function of input transition time and a linear function of load capacitance. The drawback of this method is there may be lack of circuit insight into the timing behavior of various cells. Also, empirical models may not model all situations that may arise as well as a physical model since those situations must be explicitly modeled in the empirical approach [34]. However, an empirical model is well suited to a fast, efficient computer simulation, as the data is well structured. Empirical models provide greater degrees of freedom for modeling delay accurately and efficiently without being constrained by the behavior of an equivalent circuit model.

Timing simulation may or may not use the logical values of circuit nodes, called *node values*, to determine timing information. Node value dependent simulation uses information about node values to calculate delay. In other types of simulation, timing calculations are done without regard to node values of the circuits. Differences in rising/falling transitions are compensated by keeping track of the number of inversions. Timing verification is also node value independent, and is usually used to locate and analyze the slowest paths [6]. Of course, it is not always the slowest paths that are of interest. At times a designer may be interested in the delay of a quicker path as a "worst case". The method used in this work is node value dependent simulation, where the value of each node is important to the simulation. Obviously, little data reduction is gained if all node values are stored. Usually, besides input and output nodes to a module, only those internal nodes necessary for correct logic and delay calculation are stored. Thus, each transition may be evaluated only in terms of local inputs to the cell and stored node values of the cell. There are separate evaluations for rising and falling transitions. This type of simulation may be somewhat slower, but is more accurate than simulations that do not use this information, plus logic information is evaluated and produced.

The method for calculating the delay of a cell, for a given input signal on a given cell input node, is

1. Find load impedance of each cell output node.
2. Calculate output transition time and output delay time for each cell output as a function of the load impedance of the output, input signal and node values of the cell.
3. Determine whether output waveforms are rising or falling.
4. Update node values of the cell.

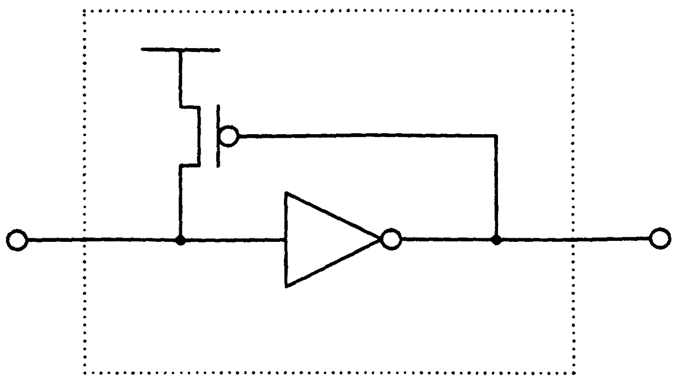
2.1 Circuit Partitions

A question of circuit partitioning arises when developing macromodels. The main issues are size of the cell and placement of cell boundaries.

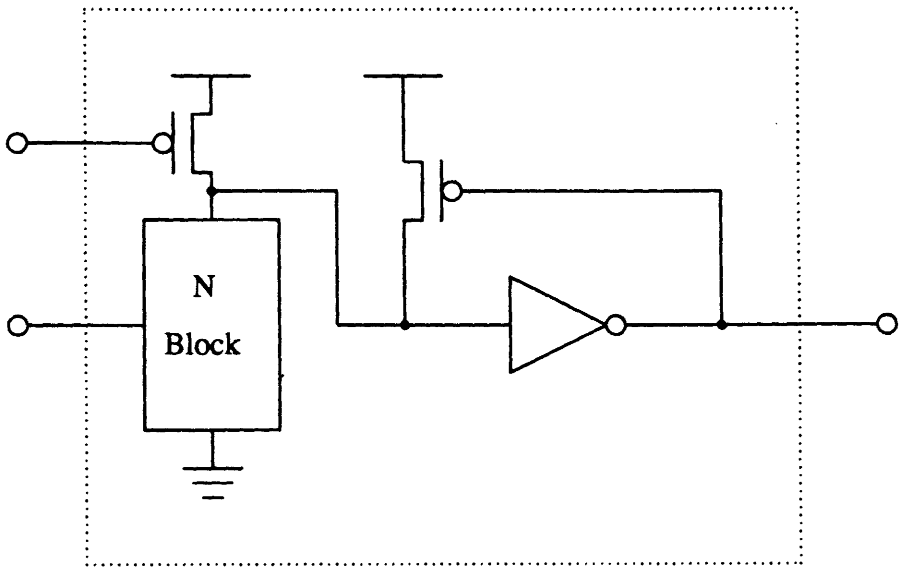
The size of a cell may vary. A cell may be as small as a single inverter or transmission gate, or as large as several logic gates. The trade-off is total number of cells versus cell complexity. A maximum of two sets of parameters are needed for each input/output pair, one for rising waveforms and one for falling waveforms. Thus, a three input, two output cell will require a maximum of 12 sets of parameters. Also, since the simulations are node value dependent, as the cell becomes larger, the number of node values to examine becomes larger. Another factor to consider, however, is that the number of SPICE simulations needed to generate a set of parameters may be less for a larger cell, since the outputs will not be as sensitive to the input waveforms.

Most cells can be defined on functional borders. This means that a circuit may be partitioned in such a way that portions of the circuit that perform a specific function may constitute a cell. Examples of these types of cells are register cells and adder cells. The major limitation on placement of a cell boundary is feedback. Feedback where the output waveform affects the input waveform transition will be referred to as dynamic feedback. Any tight, dynamic feedback loops must be internal to the cell, although any node that is fed back to an internal node may be a cell output. Figure 2-1 shows the type of feedback that would affect definition of cell boundaries. Figure 2-1 a) illustrates feedback via the p-transistor that aids rising waveforms of the input. However, the input to this cell has previously been calculated by the driving cell of the node and cannot be changed. Thus, effects of the p-transistor cannot be modeled. However, if the point of feedback is included internal to the cell such that it is isolated from the input, as in Figure 2-1 b), then macromodeling via SPICE simulations will account for the feedback effect.

Feedback plays an important role in the cross-coupled inverter circuit, shown in Figure 2-2. This is a difficult cell to model for two reasons. First, of course, are feedback effects encountered at the input node. Also, the load on the input node is not purely capacitive, since driving transistors of the feedback inverter effectively look like resistances to a voltage source. Unfortunately, with the assumption of unidirectionality of a cell, feedback cannot be adequately modeled as long as the feedback node is not internal to the cell. If one chooses to ignore effects of feedback, errors may be minimized by including loading effects of the feedback inverter. Thus, the input impedance of the cross-coupled inverter cell would be an RC circuit. By not explicitly



a)



b)

Figure 2-1: Modeling feedback in cell macromodels.

a) This type of feedback cannot be modeled as the output affects the input signal. b) Correct way of modeling feedback such that the effects are isolated from the input.

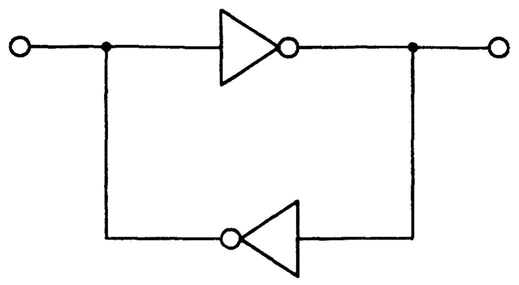


Figure 2-2: Cross-coupled inverter circuit.

modeling feedback, one must assume that the feedback inverter will never override the driven input signal. In fact, the assumption is made that *there will never be any waveform action at the input node initiated by the output node, as follows from unidirectionality.* Ruehli [35] gives a rigorous approach for decomposition of a circuit into sub-circuits such that there will never be any feedback between sub-circuits.

There are two basic types of delay calculation. One type of calculation is used when the cell structure is of logic gate form and load impedances on the cell outputs are purely capacitive. This method will be called logic gate delay calculation. A logic gate cell may be an inverter, NAND, NOR, or combinations of these and other similar logic forms. The other type of calculation, called RC circuit calculation, is used when the cell to be analyzed is a transmission gate or a logic gate cell driving a load that is not purely capacitive. This load may consist of conducting transmission gates or RC trees. A method is needed for interfacing results of the two types of delay calculation. In this way, cells that function differently in terms of delay may be modeled in a different way as long as an interface is found to existing methods of delay calculation. This allows the most efficient and accurate method of delay calculation without having to generalize over all circuit types.

The method of calculation of delay for cells containing both logic gate forms and RC circuits or transmission gates depends on the configuration of the cell. If all source/drain nodes of transmission gates are internal to the cell, then logic gate delay calculation may be used as long as the load impedance on the cell output nodes is purely capacitive. If the cell contains both a logic gate followed by a transmission gate or similar structure, and the load impedance is purely capacitive, then logic gate delay calculation is performed. If the load is not purely capacitive, then the effective driving resistance and capacitance will include effects of both the logic gate and transmission gate in the cell. If the cell is a transmission gate followed by a logic gate, the modeling is not so easy. The input impedance obviously contains some resistance when the transmission gate is conducting. However, calculating the effective resistance of the transmission gate in this configuration is not straightforward. Thus, the transmission gate should be modeled as a separate cell.

2.2 Waveform Models

Typically, waveforms of MOS circuits are fairly well behaved and easily characterized. This is true mostly for the case of a logic gate circuit driving a purely capacitive load. Output waveforms of logic gates may be separated into two basic categories: output waveforms caused by fast inputs and output waveforms caused by slow inputs [36].

A waveform caused by a fast input consists of an initial curve as the output begins to respond, a linear region, and an exponential region. The linear region is caused by driving transistors passing through saturation, and may be modeled by a current source driving a capacitor. When the driving transistor turning on is in the non-saturation region, the logic gate is better modeled as a resistor, and the exponential tail of the waveform results. Figure 2-3 shows an output waveform of an inverter.

Output waveforms caused by very slow inputs basically follow the circuit's D.C. transfer function. However, the transfer function is similar to the fast input case as it has a linear region and a decaying tail. Instead of load capacitance being the major determining factor in waveform shape, the shape of the transfer function is dominant. Figure 2-4 shows the D.C. transfer function of a CMOS inverter.

The logic gate waveform model uses two points on the waveform to define the model. These points correspond to times where the waveform completes 20% and 80% of the transition, in terms of signal voltage. The waveform is fairly linear in this region, thus two points should be sufficient to model the waveform action. Most of the critical part of the signal, as far as the circuit is concerned, also occurs in the region, so the location of the selected points is reasonable. In CMOS, these voltages are 1V and 4V for rising waveforms, or vice versa in the falling case, since CMOS is rail-to-rail.

These time values are interpreted as *delay time*, *Td*, and *transition time*, *Tr*, where

$$Td = t_{20\%}$$
$$Tr = t_{80\%} - t_{20\%}\,.$$

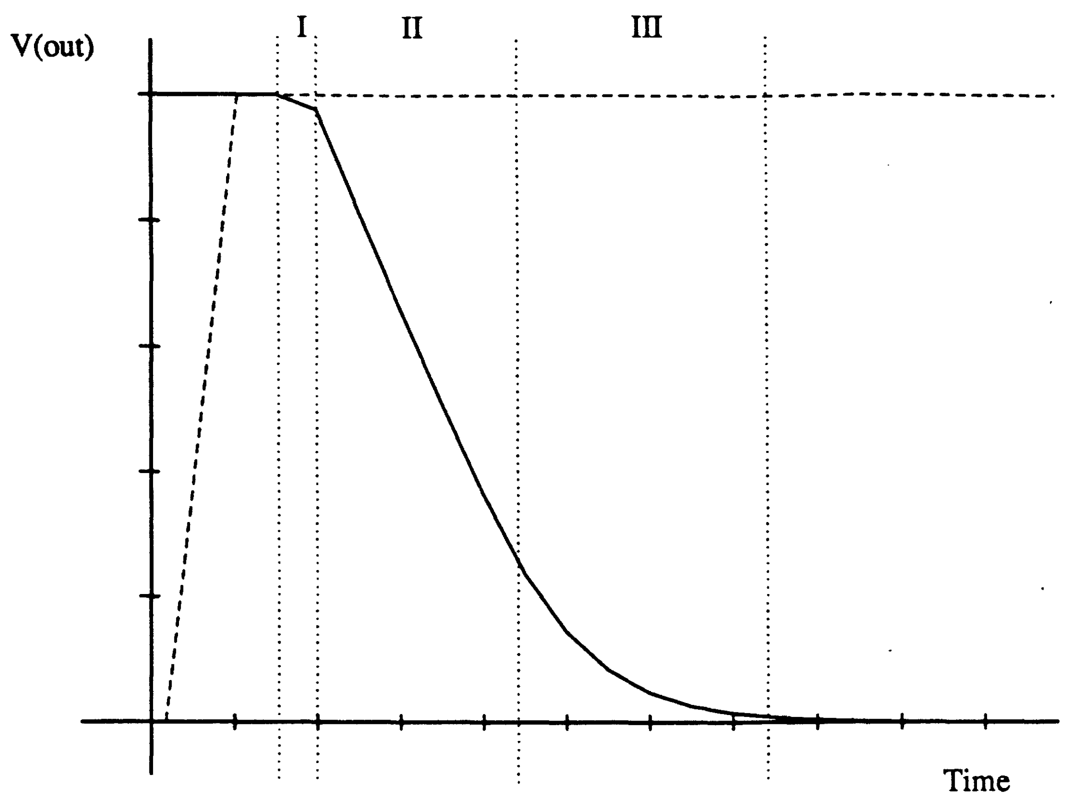


Figure 2-3: Output waveform of a CMOS inverter.

Region I is the initial response, region II is the linear region where the driving transistors are saturated, and region III is the decaying exponential tail. The dashed line represents the inverter input signal.

The delay time and transition time may define a ramp waveform model, as Figure 2-5 illustrates. The points on the waveform could also easily be interpreted as an exponential waveform with a delay time and a time constant, as in Figure 2-6. However, for the most part, we will view the waveform as a ramp with a delay time and transition time.

The ramp model works very well when a logic gate is driving a purely capacitive load, since the output waveform behaves as described in previous paragraphs. When the load is not purely capacitive, as in an RC tree or transmission gate, the waveform is better approximated with a multi-time-constant exponential or some other waveform. Waveform modeling for these situations is explained in Section 2.4.1.

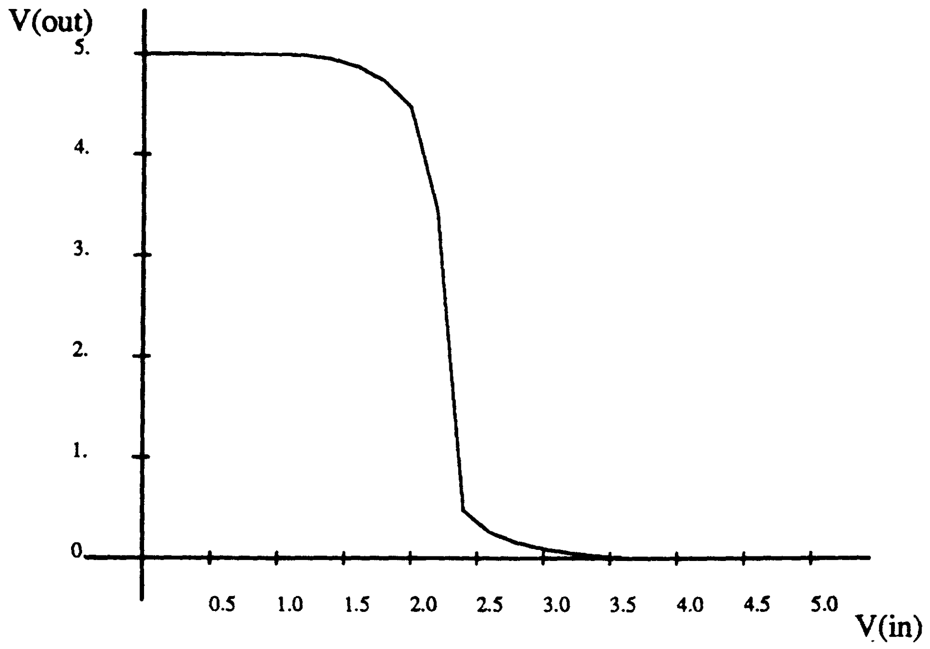


Figure 2-4: D.C. transfer function of a CMOS inverter.

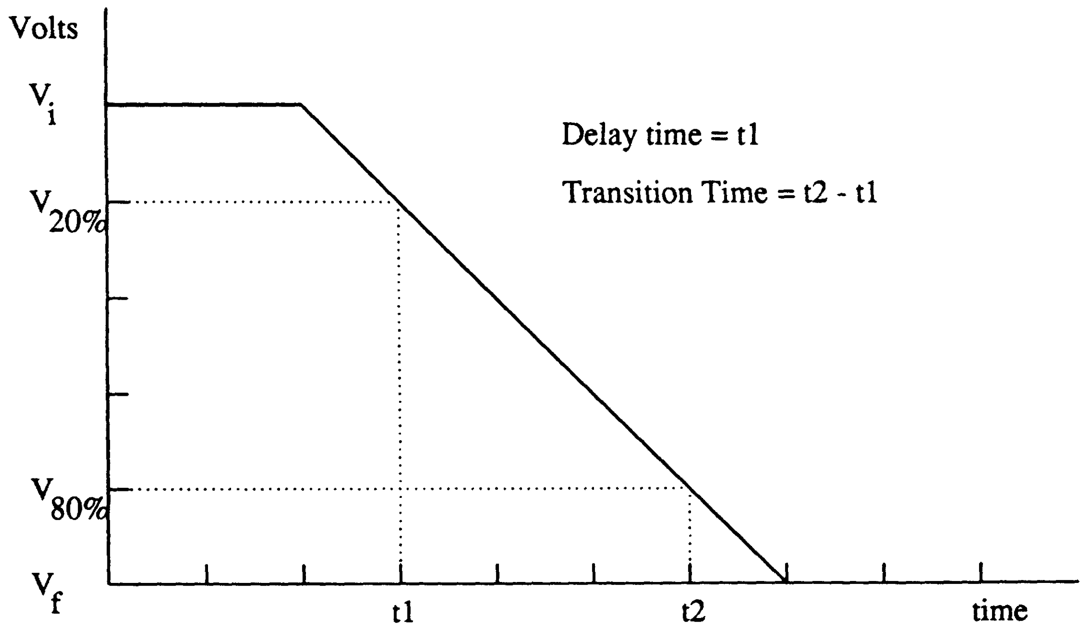


Figure 2-5: Ramp model of a waveform

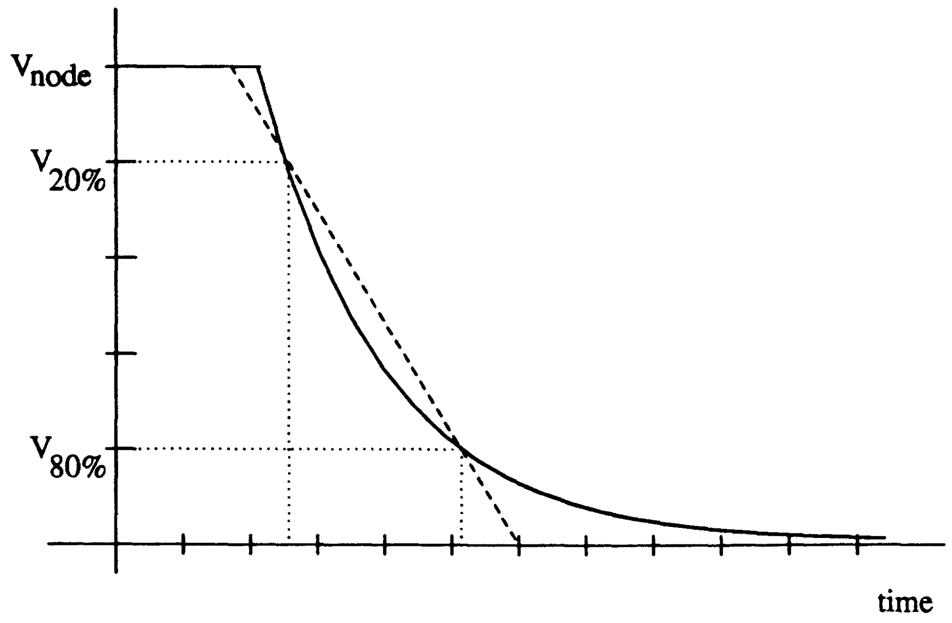


Figure 2-6: Fitting a ramp and an exponential to the same two points.

2.3 Logic Gate Models

The macromodel for a logic gate cell is shown in Figure 2-7. As previously mentioned, external variables are load impedance, input and output transitions, and node impedances (for use by other cells). The internal variables consist of the parameters used to calculate output waveforms.

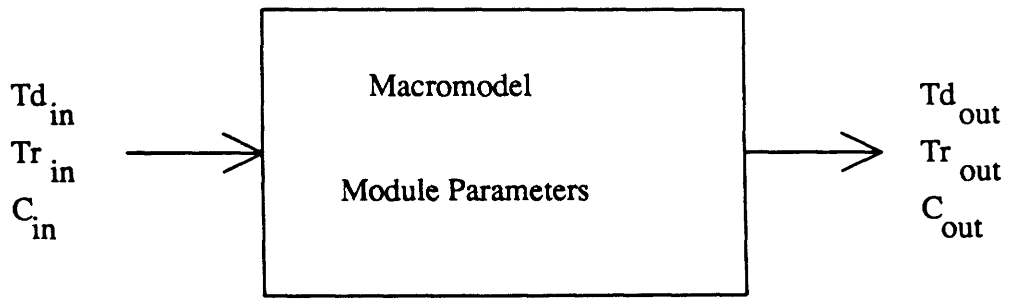


Figure 2-7: Macromodel for a logic gate cell

2.3.1 External Cell Variables

The input variables include load impedance and input waveform. If the load is composed of pure capacitance, then delay calculation is done by the method discussed in Section 2.3.2. If the load consists of resistance and capacitance, then the circuit must be characterized as an RC circuit with delay offset as described in the Section 2.4. The input waveform is defined in terms of delay time, transition time, and rising/falling transition as described in Section 2.2. Also specified is the node name upon which the input is acting, especially necessary when there is more than one input to the cell.

The output variables are output waveform and cell node capacitances. The output waveform is calculated as functions of the input variables, which are load capacitance and input waveform. The names of the output nodes affected are also provided.

The impedance of each input and output node is available for use in calculations for other cells. Impedance may be a function of the node values of the cell, although in practice, this is usually only necessary when calculating the load of a transmission gate.

The output node capacitances consist of parasitic capacitances of drain nodes of the transistors, as well as other layout parasitics which may be provided by a circuit extraction program.

Input capacitance includes the effective gate capacitance of the logic gate plus any parasitics. The parasitics may again be calculated by a circuit extraction procedure. The gate capacitance is calculated by

$$C_{eff} = \sum_{gates} C_{OX} W_{gate} L_{gate},$$

where

$$C_{OX} = \frac{\varepsilon_{SiO_2}}{T_{OX}},$$

T_{OX} is gate oxide thickness, ε_{SiO_2} is permittivity of the gate oxide, and W_{gate} and L_{gate} are width and length of the transistor gate. The summation sign indicates that all gates connected to the input must be included. For example, the effective input capacitance of a CMOS inverter would include the gate capacitance of both the n- and p-transistors.

A constant input capacitance is assumed. In reality, the gate capacitance is not constant, but a function of V_{DS} and V_{GS} [37]. However, SPICE simulations on an inverter show that overall effective gate capacitance is very close to $C_{gate,N} + C_{gate,P}$. SPICE simulations also show that overall effective capacitance of a single n-type transistor during a 0V to 5V transition is very close to $C_{gate,N}$. One must keep in mind, however, that SPICE may not be the best proof of gate capacitance behavior, as the issues of charge conservation and correct capacitance modeling has not yet been resolved. Theoretical calculations show that the above assumptions are reasonable.

2.3.2 Output Waveform Calculation

In general, output delay times and transition times are linear functions of load capacitance and piecewise linear functions of input transition time. Logic functions of the cell determine whether the output is rising or falling and also which set of parameters to use in delay calculation.

The transition time, Tr, calculation may be viewed as a function of load capacitance in the following manner:

$$Tr_{out} = Tr_{noload}(Tr_{in}) + R_{tr}(Tr_{in}) \times C_{load} \tag{2.1}$$

Tr_{noload} and R_{tr} are both piecewise functions of input transition time. Tr_{noload} is the output transition time when there is no load capacitance. R_{tr} is the ''transition time resistance''. Transition time will therefore increase at a constant rate with C_{load}.

The output transition time in terms of input transition time is usually a two section piecewise linear function. The first section is a constant value with a slope of zero. This is the *non-tracking* section. The input transition is completed before the output transition is well into its response. The effective output resistance of the gate and load capacitance determine the transition time of the gate output. After a certain break point, the input transition time is long enough to start having some effect on the output. This is the second section, or the *tracking* region, of the function. It is called the tracking region because the input signal is slow enough that the output signal will, in effect, trace it, although the output waveform will be a function of the input waveform as defined by the D.C. transfer function of the gate. The slope of the output transition time versus input transition time in this section is non-zero and positive. Figure 2-8 shows a typical relationship between input and output transition time for an inverter.

In some cases, for a cell composed of many transistors, or having a large load capacitance, the function consists of only the first non-tracking region in a practical range of input transition times. In the first case, the defined input is electrically removed enough from the output that input transition time has little effect. In the second case, the effect of large load capacitance is dominant, and the step input response is long enough

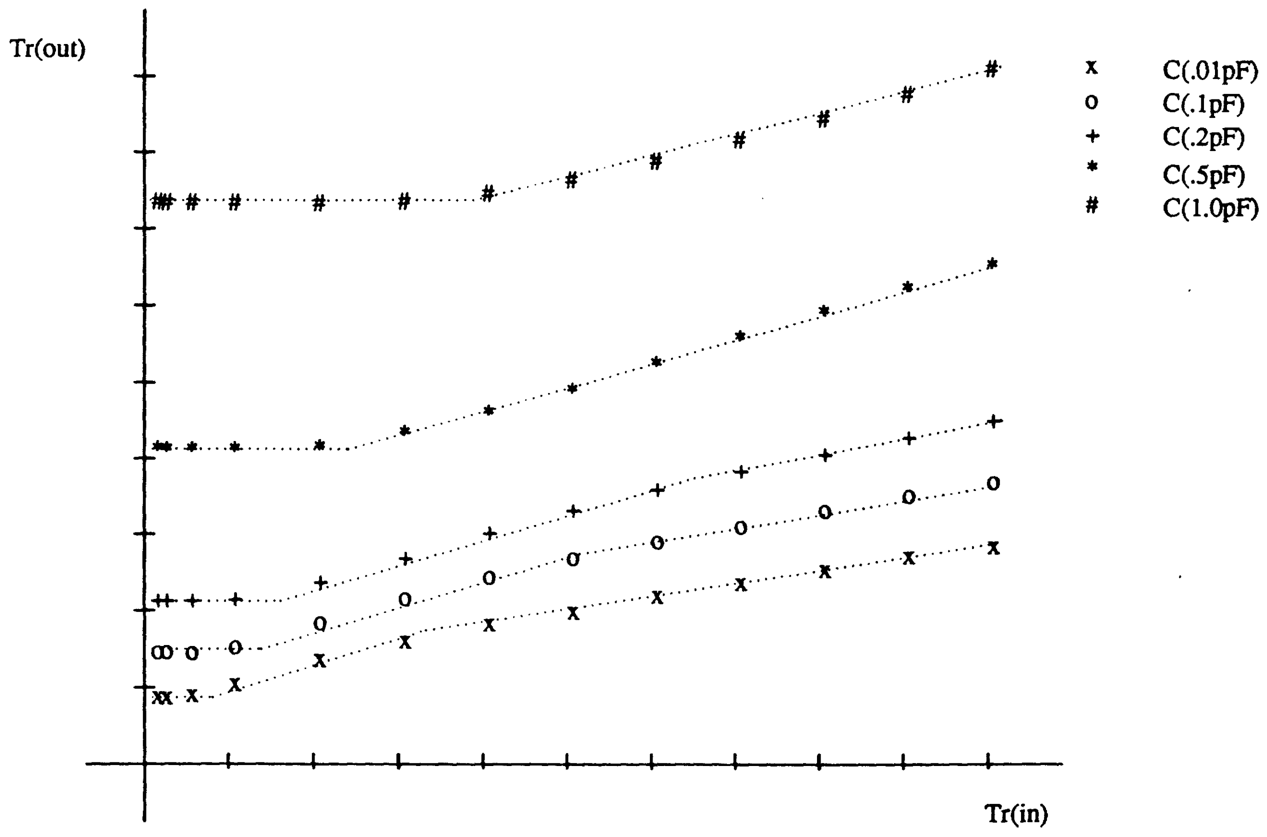


Figure 2-8: Output transition time function for an inverter.

Output transition time as a function of input transition time with load capacitance as a parameter.

that the input transition time would have to be unrealistically long in order to affect output transition time.

In other cases, the second region is actually better modeled by two sections, where the last section has a slightly smaller slope. This effect is usually seen for small cells, as in inverters, for smaller load capacitance. Under these condition will non-linearities in timing behavior show up. Larger circuits tend to have outputs which are more decoupled from inputs than smaller circuits. Capacitive effects will dominate timing results more when the load capacitance is large. If the load capacitance is small, then non-linear behavior will be more prevalent.

The delay time, *Td*, of a node is measured as the time the node completes 20% of its transition minus the time at which the circuit input completed 20% of its transition.

The delay time calculation may be viewed as a function of load capacitance in the following manner:

$$Td_{out} = Td_{noload}(Tr_{in}) + R_{td}(Tr_{in}) \times C_{load} + Td_{in} \tag{2.2}$$

Td_{in} is the delay time of the input signal. Td_{noload} and R_{td} are both piecewise functions of input transition time. Td_{noload} is the delay when there is no load capacitance. R_{td} is the "delay resistance". The corresponding model for this equation is shown in Figure 2-9.

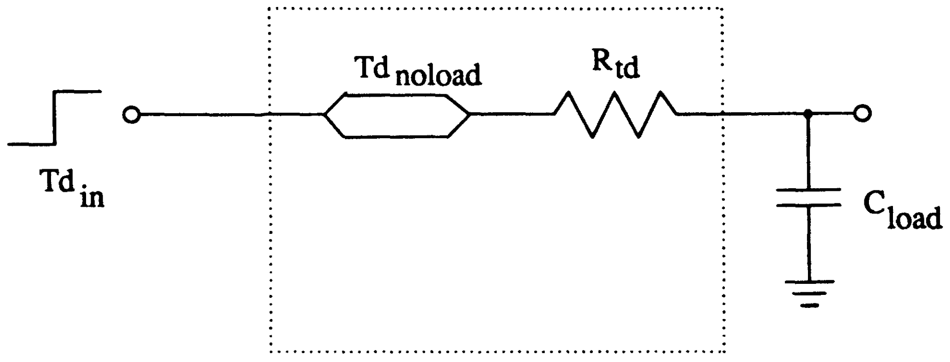


Figure 2-9: Model for delay calculation of a cell

Figure 2-10 illustrates an example of output delay time versus input transition time for an inverter. Output delay time in terms of input transition time is fairly linear, as one might expect. However, some curves exhibit a slightly smaller slope for longer input transition times and are therefore fit with two piecewise linear sections. The reason for this is related to the tracking and non-tracking regimes explained in previous paragraphs. Consider the critical switching time of a gate, T_{sw}, as the time when the input voltage is equal to the point on the D.C. transfer curve where $V_{in} = V_{out}$. Let this voltage be V_{sw}. In this explanation, let delay be measured as the time the output reaches V_{sw} minus the time the input reaches V_{sw}. In the non-tracking regime, input transition times are completed before the output of the circuit has reacted measurably. The output is dominated by the RC time constant. Thus, In the non-tracking region of smaller input transition times, the delay referenced to the time the input reaches V_{sw} is $R_{eq}C_{load}$, where C_{load} is the load capacitance plus any output capacitance and R_{eq} is the equivalent resistance of the logic gate. When the input transition time is slow enough such that the output follows the D.C. transfer curve, then, by definition, the output voltage reaches V_{sw}

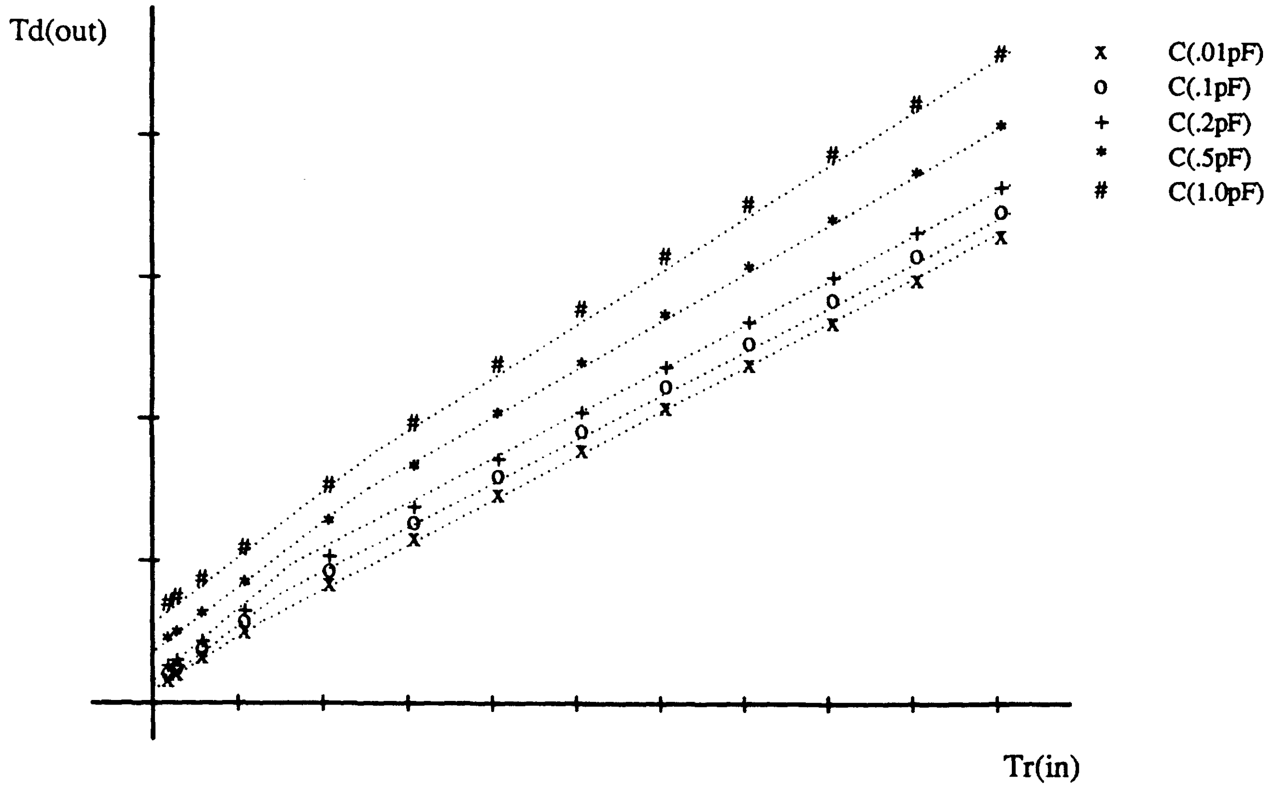
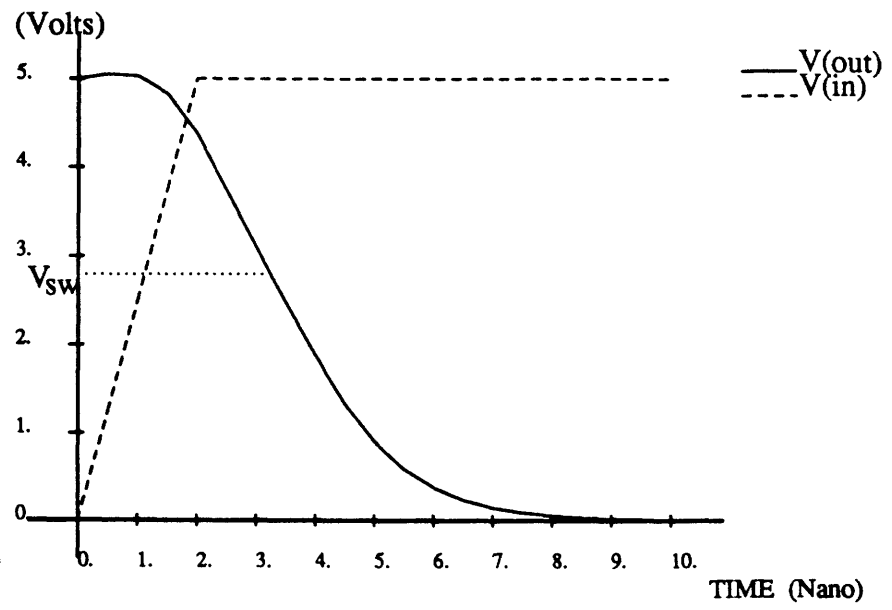


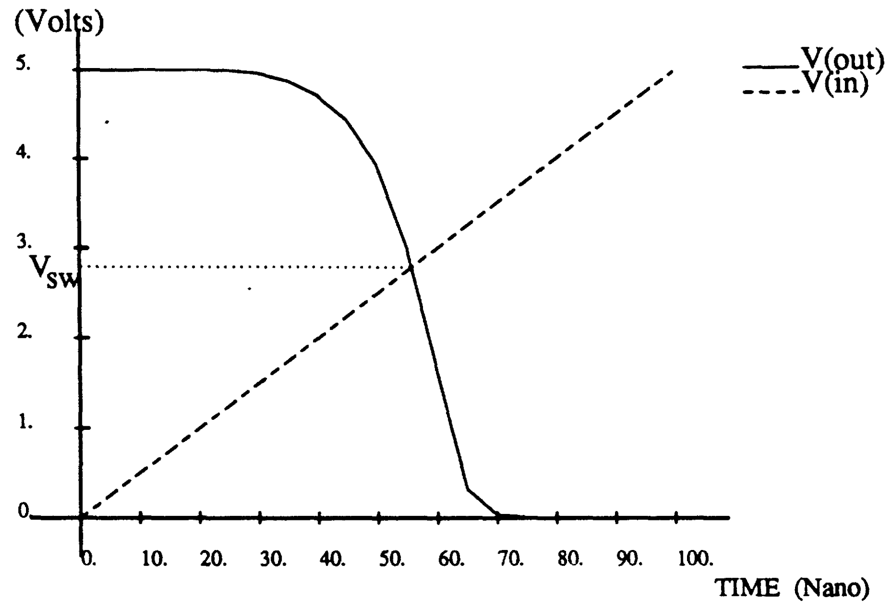
Figure 2-10: Output delay time function of an inverter.

Output delay time as a function of input transition time with capacitance as a parameter.

at the same time that the input voltage reaches that voltage. There is no delay with respect to those voltage points. These two situations are illustrated in Figure 2-11. Since, in the waveform model presented in Section 2.2, delay is measured with respect to the 20% transition values, there should never be zero delay. Delay will increase with input transition time, the transition will reach the 20% transition voltage before it will reach V_{sw}. However, this increase in delay will be less when the output waveform is defined by the D.C. transfer curve than when it is defined by the RC time constant. This effect appears in the delay functions as the two piece-wise linear functions, where the non-tracking region has a larger slope than the tracking region.



a)



b)

Figure 2-11: Effects of input transition on delay.

a) With a fast input transition, there is delay with respect to V_{sw} because of the additional delay caused by the RC time constant. b) With a slow input transition, there is no delay referenced to V_{sw} since the circuit's D.C. response determines the output waveform.

2.3.3 Derivation of Parameters

Parameters are derived from SPICE simulations of the cell. Simulations are done for each input/output pair, and for rising/falling transitions.

The cells modeled at the time of this writing have been simulated at five different loads, ranging from .01pF to 1.0pF, and at ten different transition times, ranging from .6ns to 48ns. This range is variable and may be adjusted depending on expected conditions under which the cell will operate. If it is known that certain outputs will not be sensitive to certain types of input transitions, simulations for those situations may also be omitted.

A waveform analysis is performed on the results, and plots are made of the analyzed data. The plots consist of points which are output delay time or transition time versus input transition time with load capacitance as a parameter. The slopes, intercept, and break points of the curves are then plotted with load capacitance as the independent variable. The slopes and intercepts of these lines are then the parameters that are entered into the cell's delay procedure file. The input and output load capacitances, as well as driving transistor widths, are also entered into the delay procedure.

2.3.4 Physical Interpretations

Equation (2.2) in Section 2.3 showed one way of viewing calculation of delay time. There is a delay offset, an effective resistance, and a load capacitance. Thus, the delay calculation may be viewed as a simple operation using the Elmore delay [38], defined as $\sum_k R_{ke} C_k$, with an added delay offset. Node e is the output node, C_k is the capacitance at node k, and R_{ke} is the common series resistance between the driving source and nodes k and e. Furthermore, this delay offset, which could also be called an *internal delay*, could be modeled as a capacitor internal to the cell but connected to this effective output resistance. Figure 2-12 shows the circuit for this model. The resulting equation is then

$$T_d = Td_{in} + R_{td} \times (C_{int} + C_{load}). \tag{2.3}$$

The Elmore delay calculation works for non-step inputs, but only if the elements are linear [38]. Since this is not the case, waveform effects are used in calculating the delay offset and the effective delay resistance and RC calculations are performed assuming a step input.

Calculation of transition time with respect to load capacitance may be approached in a similar manner. There is a no-load, or internal, transition time, which may be modeled by an internal capacitance. There is also an effective resistance. The result, this

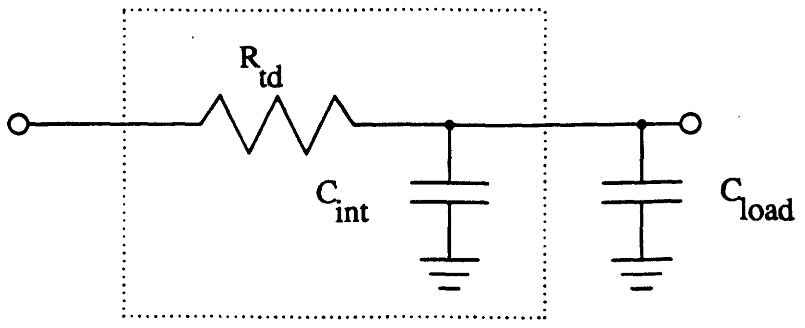


Figure 2-12: Model of a cell with internal delay modeled with a capacitor

time, is not the Elmore delay of the circuit, but the transition time, since resistance and internal transition time parameters are derived with transition time as a result. Again, waveform effects are accounted for in calculation of the effective transition time resistance and no-load transition time.

The question of modeling the cell with two circuits, or more precisely, one circuit with two sets of values, one set for delay time and one for transition time, instead of one circuit with one set of values may arise. The latter would be more physically appropriate. One reason is the definition of transition time in terms of the ramp model instead of an exponential model, since an exponential waveform would result from the physical model. However, a transform between the time constant of an exponential model and the transition time of a ramp model is a simple mathematical operation. More importantly, separating the calculation of the delay time and the transition time gives a greater degree of freedom in calculating accurate values. The best fit for the transition time is found independently of the best fit for delay time. Again, this is illustrative of the trade-offs of empirical modeling versus physical modeling.

2.3.5 Scaling Parameters as a Function of Transistor Width

Ideally, the modeling methods used should be versatile enough to allow for scaling the driving transistor widths of a logic gate. Transistor width specification may be a parameter of the cell. The delay and transition time parameters would be scaled as a function of the driving transistor width.

Tests were performed on inverters of various widths. Parameter values as a function of inverter width were plotted. Relationships between the parameter values and functions of inverter width were explored and tested. As would be expected, resistance parameters decrease as width increases. This inversely proportional relationship is:

$$R_W = R_{W_{min}} (\frac{W_{min}}{W})$$

The other parameters, such as intercepts and break points, seemed to scale exponentially with increases in inverter width. The scaling relationship for these parameters is:

$$P_W = P_{W_{min}} \times (sb)^{sf}$$

sb is the scale base, which varies with each parameter. sf is the scale factor, which indicates how many times the width has doubled. The equation for the scale factor is:

$$sf = \log_2 (\frac{W}{W_{min}})$$

Results for simulations of a parameterized inverter cell, with transistor width as the external parameter, proved favorable. Inverters with transistor widths of 4 (minimum width), 8, and 16 microns were tested. Although errors increase as the width grows larger, part of this is because the absolute delay of the inverter is becoming much smaller. Still, experimental delay values were within 10% of SPICE values.

Unfortunately, there did not seem to be a clean relationship in which all parameters could be scaled using the same scale base. Values for the scale base ranged from .1 to 1.1 (dimensionless) for different parameter types. This is due to the different nature of different parameters—some are slopes, some are break points, and others are intercepts. Perhaps different scaling methods may lead to cleaner and more consistent relationships for all types of parameters.

2.4 Transmission Gate and RC Circuit Modeling

As mentioned in the introduction to this chapter, there are two basic methods of delay calculation. Waveform models and delay models were discussed for one method. A more complex waveform model and delay calculation method will now be developed for circuits containing RC trees and transmission gates.

2.4.1 Waveform Model

As mentioned in Section 2.2, there are some circumstances when the ramp waveform model is not sufficient to model waveform behavior, such that large errors may result in delay times and transition times. A ramp model is used when a logic gate is driving a purely capacitive load. If the driving transistor is approximated by a resistor, then the resulting circuit is a single RC circuit driven by a voltage source. The output voltage is then modeled by a single time constant exponential.

When the load is not purely capacitive, as in an RC tree or conducting transmission gate, or when the driving circuit cannot be approximated well by a single resistor, then the output may not be as well behaved. Two or more dominant time constants may affect the output waveform. This is likely to happen when the waveform that is being modeled is at a node closer to the driving end of the tree than the output of the tree, or when there is significant capacitance in the tree not on the path between the source and the output. A more complex waveform model is then needed. Figure 2-13 shows an example RC circuit and resulting waveforms. Although V(e) has a slow, single time constant decay, V(k) shows effects of a fast initial transient with a slowly decaying tail.

The two time constant waveform model was developed by Mark Horowitz [15]. The model is calculated from RC tree values. This exponential waveform model has two time constants. For circuits modeled in this application, one time constant models the fast initial transient caused by the node capacitance on the path from the source to the node in question, and the other time constant models the slow decay of the latter part of the transition. This slow decay is caused by large loads at remote nodes off the path from the input to the node in question.

The transfer function for the two time constant model is

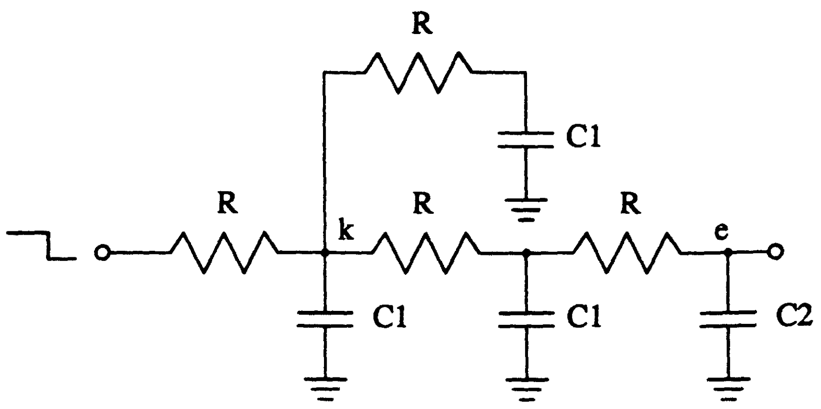
$$H(s) = \frac{1 + s\tau_z}{\left(1 + s\tau_1\right)\left(1 + s\tau_2\right)}.$$

The output voltage response of this transfer function to a step input is

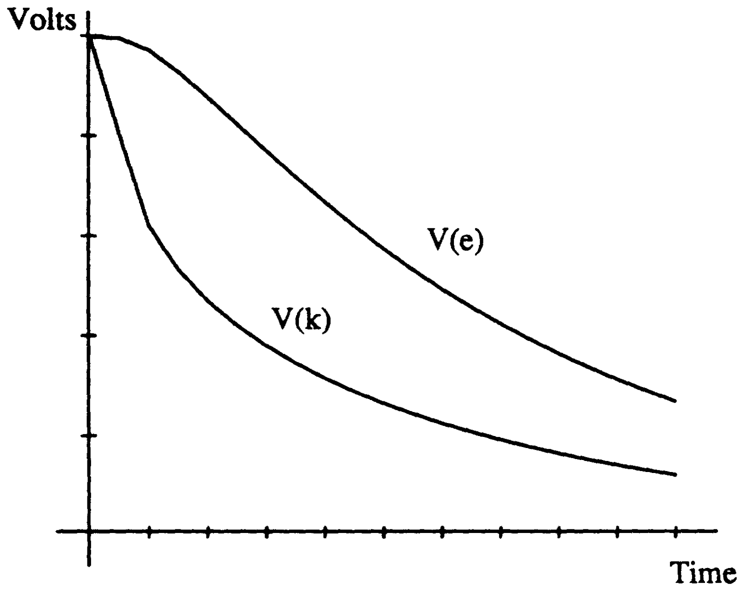
$$V_e = \frac{\left(\tau_z - \tau_1\right)e^{-t/\tau_1} + \left(\tau_2 - \tau_2\right)e^{-t/\tau_2}}{\tau_2 - \tau_1}. \qquad (2.4)$$

In general, τ_1 is the time constant modeling the fast initial transient, while τ_2 is the time constant modeling the slow decay. The time constants may be calculated from the RC tree values. The method for doing this will be discussed in Sections 2.4.4.2 and 2.4.5.2.

The two time constant model may not model all situations adequately. If the output has many time constants, as in the beginning of a long, distributed RC line, the waveform model will not be able to model all the decay rates present [15]. It would,



a)



b)

Figure 2-13: RC circuit and waveforms.

a) R = 15KΩ, C1 = .1pF, C2 = .2pF; b) Waveforms at nodes *k* and *e* with step input.

however, do better than a first order model. If the circuit has one decay rate, then the two time constant model will reduce to the single time constant case.

Once the two time constant model is calculated for a circuit, it must be transformed to an adequate ramp model for logic gate inputs. Basically, the ramp should be specified in a way to best model the effects that the two time constant waveform would have on the logic gate. Thus, we want to choose two points on the two time constant waveform that correspond to critical regions on the loading gate's D.C. response. One choice is the -1 slope points on the D.C. response, since those points are the boundaries of the high gain region of the gate where the output is most sensitive to the input. For a typical high gain inverter, these points may be 2.2V and 2.3V. The ramp will then pass through the 2.2V and 2.3V crossings of the two time constant response. Figure 2-14 illustrates a two time constant waveform and a ramp that might be derived from it.

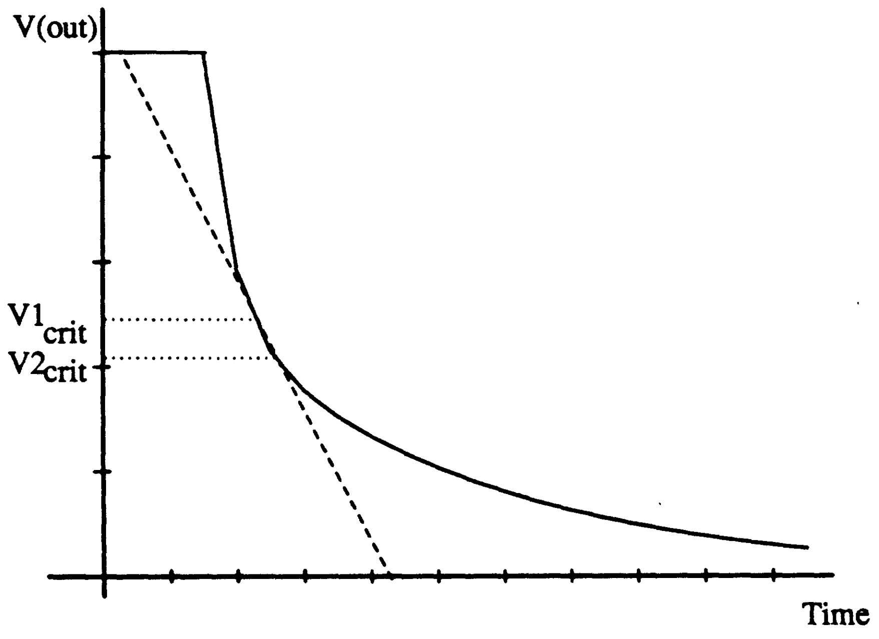


Figure 2-14: Two time constant waveform and example ramp

2.4.2 Modeling Circuits Containing Transmission Gates

The same basic method is used to analyze all transitions affecting transmission gates. First, the transistor group including the transmission gate(s) is converted to an RC circuit. A transistor group is defined similarly to a stage in Crystal, which is a chain of conducting transistors starting from a driving voltage source with each path ending in either a transistor gate, a source-drain connection of a non-conducting transmission gate, or a circuit output [6]. The driving logic gate, assuming a logic gate is the path to the driving voltage source, must be converted to its RC equivalent. Any conducting or switching transmission gates in the group are then also converted to RC circuits. The result is an RC tree. The time constants necessary for the two time constant waveform model of the nodes of the RC tree network are calculated. Only waveforms of nodes which have a loading logic gate connected to it or are otherwise specified as outputs must be evaluated. Lastly, equivalent ramps are derived from the two time constant waveforms based on the characteristics of the loading gates.

A step input is assumed for waveform calculations of an RC tree. Waveform effects are taken into account at an earlier step of the analysis, that is, calculation of effective resistances of transmission gates. Effective resistance is a function of, among other variables, input transition time. Along with effective resistances, delay offsets are calculated to further account for internal delays and input waveform effects. These delay offsets are added to the resulting output ramps derived from the RC networks.

When compared to the view of logic gate modeling in Section 2.3, delay calculation is not so totally different. Logic gate calculations use a delay time, or delay offset, plus a single time constant exponential curve to model waveforms. RC calculations use a delay offset plus a two time constant exponential curve to model waveforms. The two time constant waveform is then converted back to a single time constant exponential for logic gate delay calculations.

There are two basic types of circuits that will be analyzed using RC tree equivalent circuits. The first type is a driving gate with an RC tree as its load. Transitions in the RC tree are initiated by a transition taking place in the driving gate. A circuit like this may be an inverter with conducting transmission gate(s) as a load, as in Figure 2-15. The input to the inverter changes, causing the output of the inverter to propagate through the transmission gate(s).

The other type of circuit involves two RC trees at different logical node values which are suddenly connected. Thus, assuming the driving tree has a resistive path to a voltage source, node values of the loading tree will change to the node values of the driving tree. The driving tree is assumed to be connected to a voltage source through driving transistors of a logic gate. A circuit of this type may again be an inverter with a transmission gate as a load, but this time the transmission gate turns on to connect its

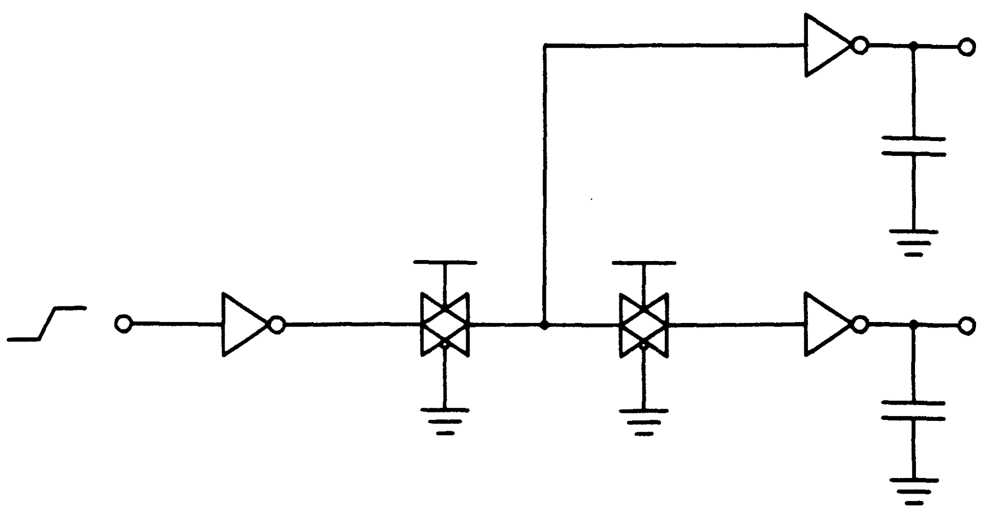


Figure 2-15: Conducting transmission gates driven by a logic gate

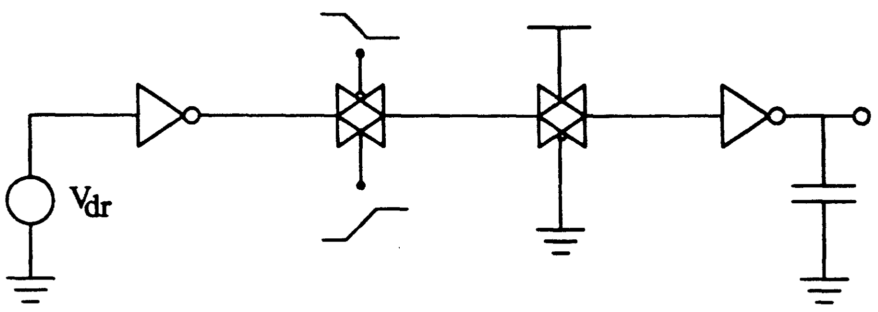


Figure 2-16: Circuit with a switching transmission gate

input to its output. Figure 2-16 is an example of this type of circuit.

In the event that the driving tree is not connected to a voltage source, then two isolated RC trees are connected. If one tree is clearly dominant and will force the other tree to the dominant node value, then the analysis of connecting RC trees may be used. If this is not the case, then both trees are forced into the unknown, or X, logical value, and delay analysis is meaningless. The nodes are set to the unknown value and no delay analysis is performed. The capacitances of each tree determine which action will be taken.

2.4.3 Effective Resistance of Transmission Gates

In order to convert circuits containing transmission gates to RC trees, the effective resistance of the transmission gates must be found. There are two modes of operation for a transmission gate for which an effective resistance must be found. The first is when a conducting transmission gate passes a signal from one source/drain node to the other. The second is when signals at the gate nodes of the transmission gate cause it to start conducting, connecting nodes that are at different node values. The node values must then settle to the same equilibrium value.

2.4.3.1 Conducting Transmission Gates

The model of a conducting transmission gate consists of an input capacitance to ground, an output capacitance to ground, and a series resistance between the two capacitors, as in Figure 2-17. This model is used whenever a signal passes from one source-drain node to the other. If a transmission gate is non-conducting, then the model would simply be an input capacitance and an output capacitance, with no connection between them. These capacitances would also terminate the branches of the RC trees connected to them.

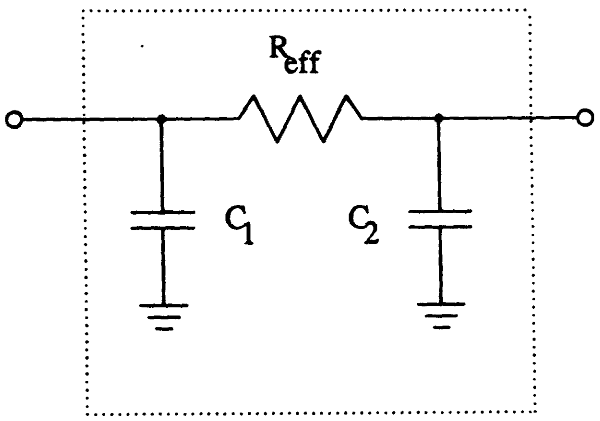


Figure 2-17: Model of conducting transmission gate

The input and output capacitances are considered to be constant and are usually derived from circuit extraction results. These capacitances include parasitic capacitances to the substrate of the sources and drains of transistors and interconnect. Also included is the channel capacitance of the transmission gate. Half of this value is added to the input capacitance and half is added to the output capacitance.

The resistance is modeled as a function of input transition time and the load capacitance of the transmission gate in the direction the signal is propagating. Thus, an effective resistance is calculated based on loading and signal conditions of that particular transition.

Transmission gate resistance parameters are derived in much the same manner as delay time and transition time parameters for logic gate cells. That is, SPICE simulations are done for rising and falling transitions while varying input transition time and load capacitance. A waveform analysis is performed on the results, and plots made of the analyzed data. The data in this case is resistance. Resistance is calculated by dividing the Elmore delay [38] of the output with respect to the input by the total capacitance seen at the output source/drain node of the transmission gate. This may be done since the Elmore delay of a single RC section is simply RC, where R is the resistance value and C is the capacitance value. Penfield and Rubinstein [13] showed that the Elmore delay for a rising waveform is equal to the area above the waveform and below 1 (assuming a unit step response).[1] Thus, the Elmore delay needed for resistance calculations is simply the area above (below) the output waveform minus the area above (below) the input waveform. After the resistance data is plotted versus input transition time and load capacitance, then the slope, intercept, and breakpoint parameters may be derived from those plots.

Transmission gate effective resistance is also assumed to be piecewise linear with respect to input transition time and a linear function of load capacitance. Figure 2-18 shows the relationship between conducting transmission gate resistance and input transition time and load capacitance. The effective resistance increases immediately with increasing transition time and then quickly levels off to a constant value. The effective resistance is roughly equal for all load capacitances with a step input. As transition time increases, the effective resistance rises more sharply and levels off at a slightly higher resistance at smaller loads. All in all, the variation in resistance is not great. One transmission gate example varies from 13 KOhm to 15 KOhm, versus both load capacitances and input transition times. Since this function describes overall effective resistance during a voltage transition, many effects dictate the behavior of effective resistance versus input transition time. One such effect is the relationship of resistance to V_{gs} when the transistors are in the non-saturation regime. The average value of V_{gs} is larger for fast input transitions than for slower input transitions. This assumes that the

[1]For a falling waveform, the Elmore delay is the area *below* the waveform.

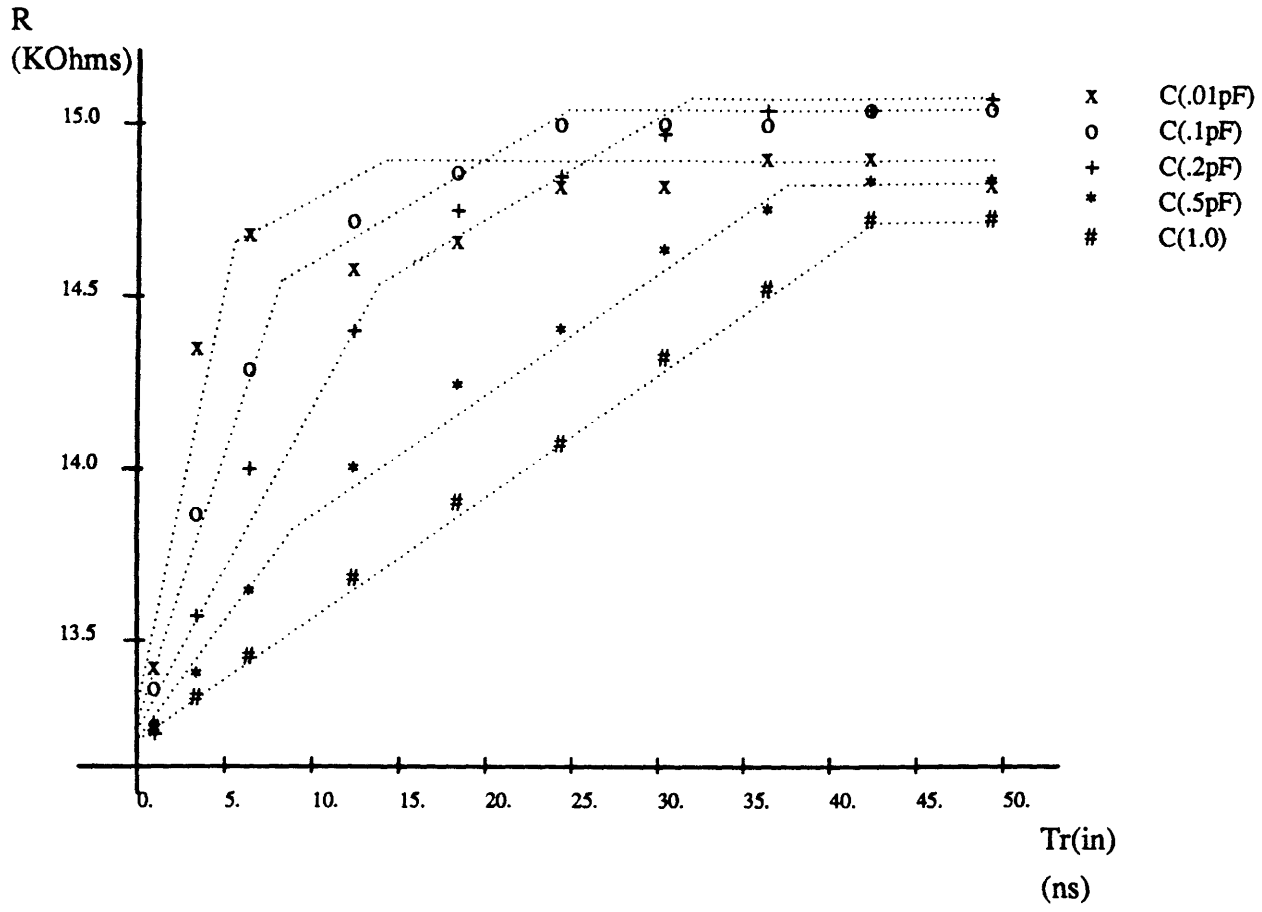


Figure 2-18: Conducting transmission gate effective resistance function.

Effective resistance of a conducting transmission gate as a function of input transition time with load capacitance as a parameter. The N-type transistor is 4 microns wide, 3 microns long, and has a threshold voltage of 1.2V. The P-type transistor is 8 microns wide, 3 microns long, and has a threshold voltage of -1.2V.

average value is taken over the time interval between the start and end of the output transition. Since transistor resistance is inversely proportional to V_{gs}, resistance will be smaller for faster inputs. At a certain breakpoint in input transition time, the average value of V_{gs} will be approximately constant with respect to input transition time and the resistance will level off. This breakpoint occurs earlier for smaller load capacitances.

2.4.3.2 Switching Transmission Gates

The model for a switching transmission gate consists of an input capacitance and an output capacitance to ground connected by a resistor in series with a switch, as shown in Figure 2-19. This model is used whenever the inputs to the gates of the transistors turn on, causing the transmission gate to start conducting. There are, of course, two inputs to a CMOS transmission gate, one for the n-transistor and one for the p-transistor. There will almost always be some delay between the time that these transitions will reach the transmission gate inputs. In order for the switch to be on, it is defined in this model that both the n- and p-transistors must be on. It is not necessary, however, for both to switch at the same time.

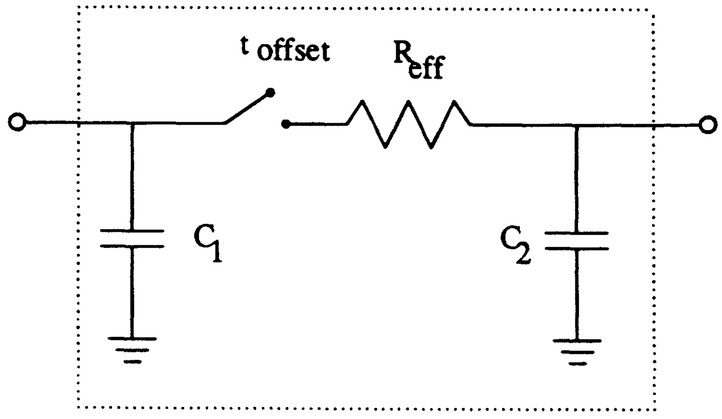


Figure 2-19: Model of switching transmission gate

The parameters that must be calculated are the time that the switch closes, t_{offset}, and the value of the resistor, R_{eff}. The switching resistance is calculated in much the same way as conducting resistance, except that the input signal is at the gate nodes of the transistors instead of a source/drain node. The resistance is again piecewise linear with input transition time and linear with load capacitance. Figure 2-20 shows the relationship between effective switching resistance and input transition time and load capacitance. The resistance starts at a constant value, and this value is approximately the same for all load capacitances. As the transition time increases, after a certain break point the resistance starts increasing. This behavior can be related to the tracking and non-tracking regions of output transition time versus input transition time that were described in Section 2.3.2. For fast inputs, the output will be in a non-tracking region where the output waveform will vary with load capacitance only. Thus, effective resistance will be constant with input transition time. As the input becomes slower, the output waveform

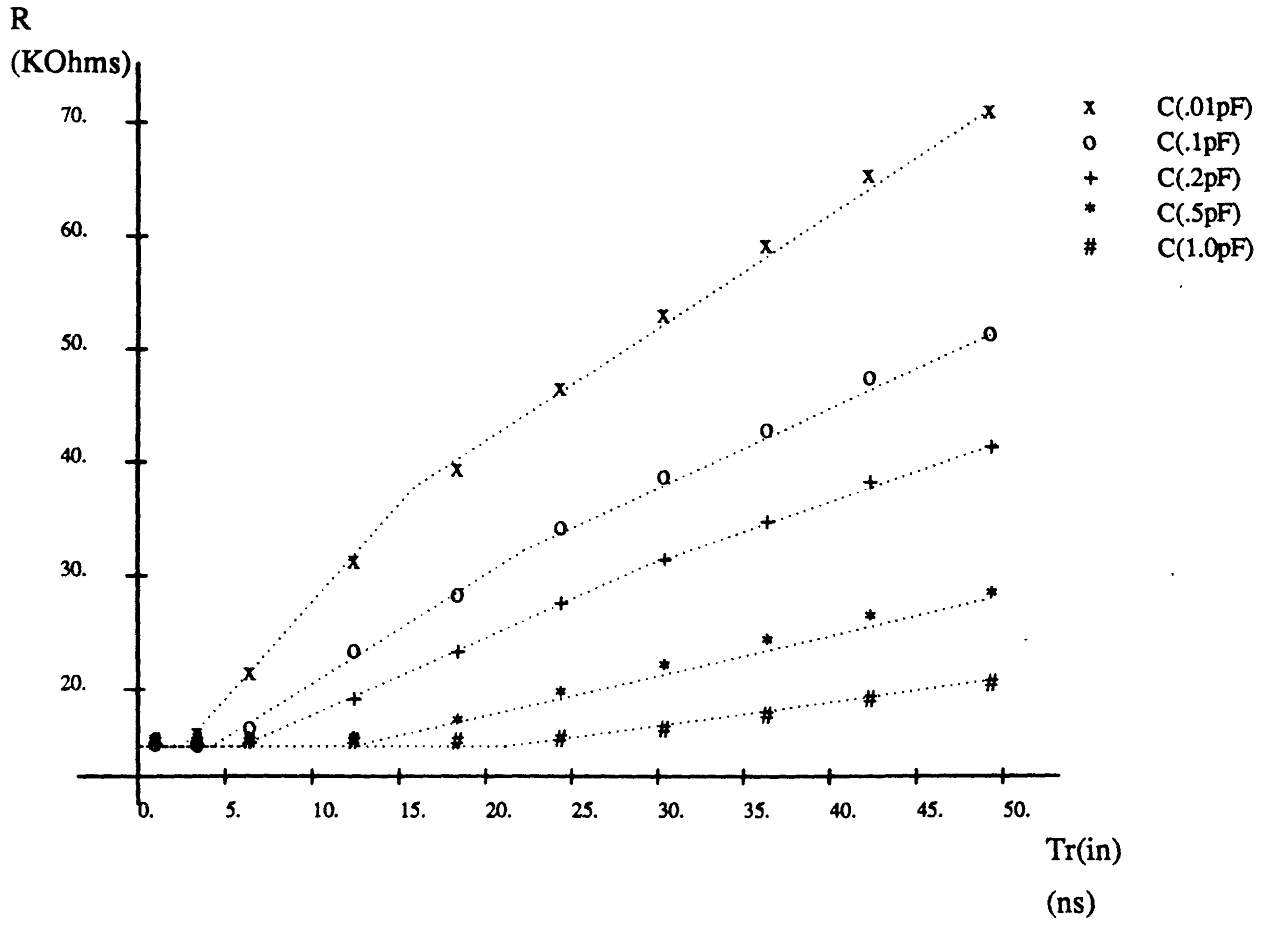


Figure 2-20: Switching transmission gate effective resistance function.

Effective resistance of a switching transmission gate as a function of input transition time with load capacitance as a parameter. The input signal is assumed to be v_g. The N-type transistor is 4 microns wide, 3 microns long, and has a threshold voltage of 1.2V. The P-type transistor is 8 microns wide, 3 microns long, and has a threshold voltage of -1.2V.

will start to track the input waveform. Thus, the effective resistance will become greater with increasing input transition time. This breakpoint occurs earlier for smaller load capacitances. Also, the effect of input transition time on resistance is greater for smaller load capacitances.

The time of closing of the switch, or *switching delay offset*, is also calculated as a function of input transition time and load capacitance. This is analogous to the output delay functions of logic gates.

The calculations mentioned above assume that input transitions of both the n- and p-type devices occur at the same time, and both with the same transition time. Most of the time this is not true. Differences in the input delay times and the input transition times must be accounted for.

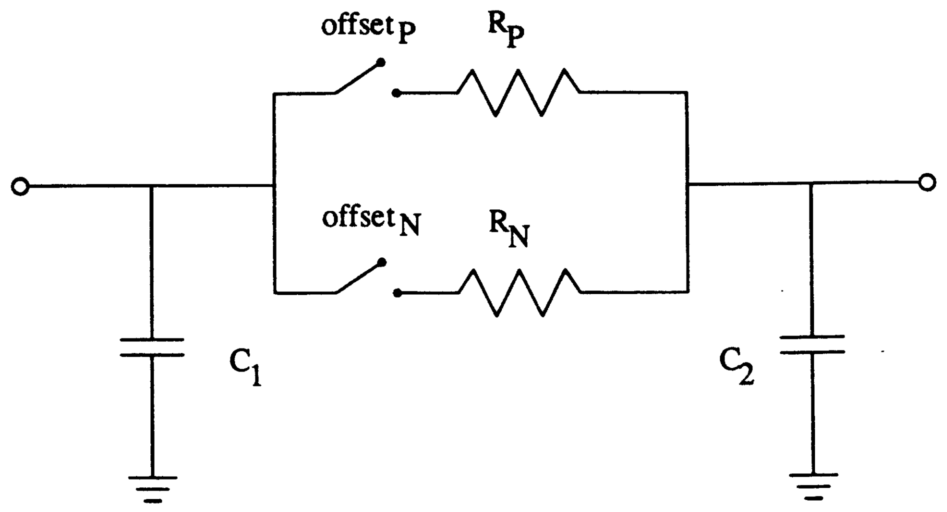


Figure 2-21: Switching transmission gate model.

Effects of separate transitions for the p- and n-transistors on the switching transmission gate model. This model is used to derive the effective resistance and delay offset of the transmission gate.

The switching delay offset and resistance are calculated separately for each input transition. Thus, there are two values for delay offset and two for switching resistance, one corresponding to the n-transistor input and one for the p-transistor input. The resulting circuit model is shown in Figure 2-21.

Assume that total effective resistance, R_T of a conducting transmission gate may be modeled as a parallel combination of an n-transistor effective resistance and a p-transistor effective resistance, as in Figure 2-22. Given a total resistance, R_T, these components may be found:

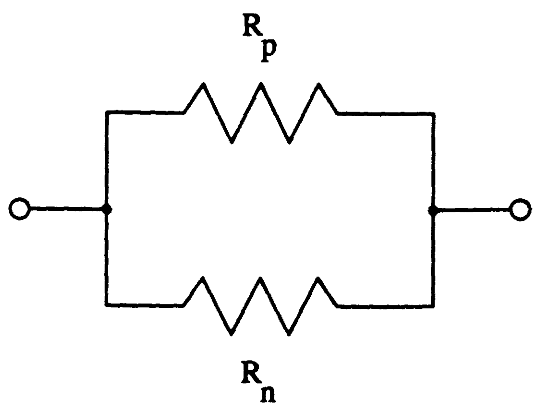


Figure 2-22: Resistance model of conducting transmission gate.

Effective resistance is modeled as a parallel combination of effective n- and p-resistances.

$$R_{NT} = \left(\frac{a+1}{a}\right) R_T$$

$$R_{PT} = (a+1) R_T,$$

where

$$a = \frac{\mu_n W_N L_P}{\mu_p W_P L_N}.$$

The desired resistance values are effective resistance of the n-transistor as a function of n-transistor gate input, and effective resistance of the p-transistor as a function of p-transistor gate input.

$$R_N = \left(\frac{a+1}{a}\right) R_{T,\,n\text{-}input}$$

$$R_P = (a+1) R_{T,\,p\text{-}input}$$

$R_{T,n\text{-}input}$ is the effective resistance of a switching transmission gate as a function of the n-transistor gate input. $R_{T,p\text{-}input}$ is the effective resistance of a switching transmission gate as a function of the p-transistor gate input. Obviously, only the resistance of the p-transistor is meaningful when the p-transistor gate changes, and only the resistance of the n-transistor is meaningful when the n-transistor gate changes. The resulting circuit is shown in Figure 2-21.

The circuit in Figure 2-21 will then give the following waveform, assuming, for this case, the p-transistor turns on (at t_p) before the n-transistor (at t_n) and a falling waveform:

$$V_{out} = \begin{cases} 1 & t < t_p \\ e^{-(t-t_p)/R_P C_L} & t_p \le t < t_n \\ V_{int} e^{-(t-t_n)/R_T C_L} & t \ge t_n, \end{cases} \quad (2.5)$$

where

$$V_{int} = e^{-(t_n - t_p)/R_P C_L}, \text{ and}$$

$$R_T = \frac{R_P \times R_N}{R_P + R_N}.$$

Assuming a unit initial value, the output voltage will be 1V until the first input arrives at t_p. Once the p-transistor is on, the voltage will fall exponentially with time constant $R_P C_L$, resulting in the second part of the equation. When the second input arrives at t_n, the waveform will continue falling, but with time constant $(R_P // R_N) C_L$, where $||$ denotes parallel combination. The voltage at the time when the second input arrives is V_{int}, which may be derived by solving the first exponential by substituting t_n for the time variable, t.

The final effective resistance and delay offset are then found from the waveform. Two critical voltage points are chosen. The points used are 1V and 4V, in order to find effective values for a wide range of combinations of switching times of the two transmission gate inputs as possible. The time constant and starting time of a single time constant exponential fitting the critical points are found. The resistance is calculated from the time constant and the load capacitance. The switching delay offset is the starting time of the exponential. Figure 2-23 shows the waveform defined by Equation (2.5) and two sample critical points. $offset_1$ is the time at which the first transistor turns on, and $offset_2$ is the time at which the second transistor turns on. A problem with this method is that if the second transistor switches after the piecewise linear waveform completes its transition, the effective resistance and delay only include effects of the first transistor. This may happen when an n-transistor passes a high signal or a p-transistor passing a low signal. A threshold drop would prevent a complete transition until the other transistor turned on. These threshold drops would not be present, however, when

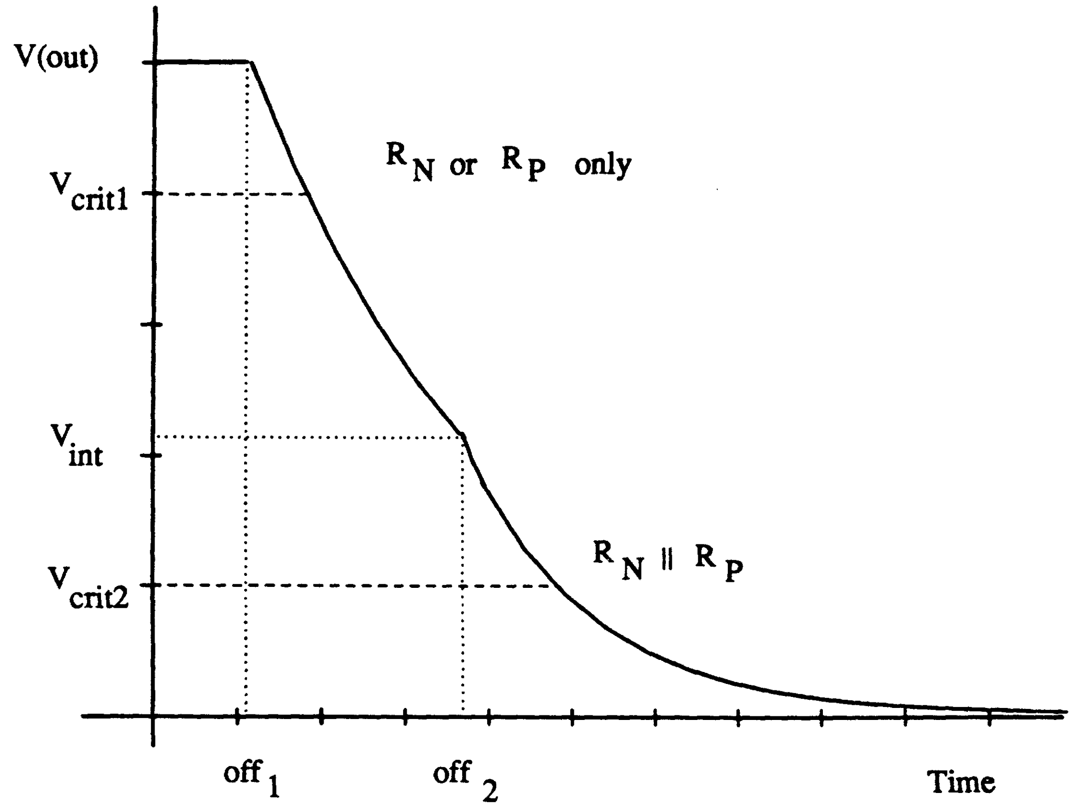


Figure 2-23: Waveform of circuit in Figure 2-21.

The piecewise exponential waveform is used for derivation of switching transmission gate effective resistance and delay offset.

an n-transistor passes a low signal or a p-transistor passes a high signal.

2.4.4 Analysis of Transmission Gate Circuits: Input from Driving Gate[2]

Although finding effective resistance of transmission gates is certainly a critical step in this delay calculation, the delay offset of the output waveforms and the driving resistance must also be found. Once the RC tree is constructed, then the actual waveform calculation on the RC tree is performed.

[2]This analysis assumes transmission gates are conducting.

2.4.4.1 Constructing the RC Tree

The first step in constructing the equivalent RC tree is to find the effective driving resistance and delay offset. Also needed is the output capacitance of the driving gate.

The output capacitance is mostly parasitic capacitance. Values of channel capacitance are usually very small compared to the magnitude of the parasitics.

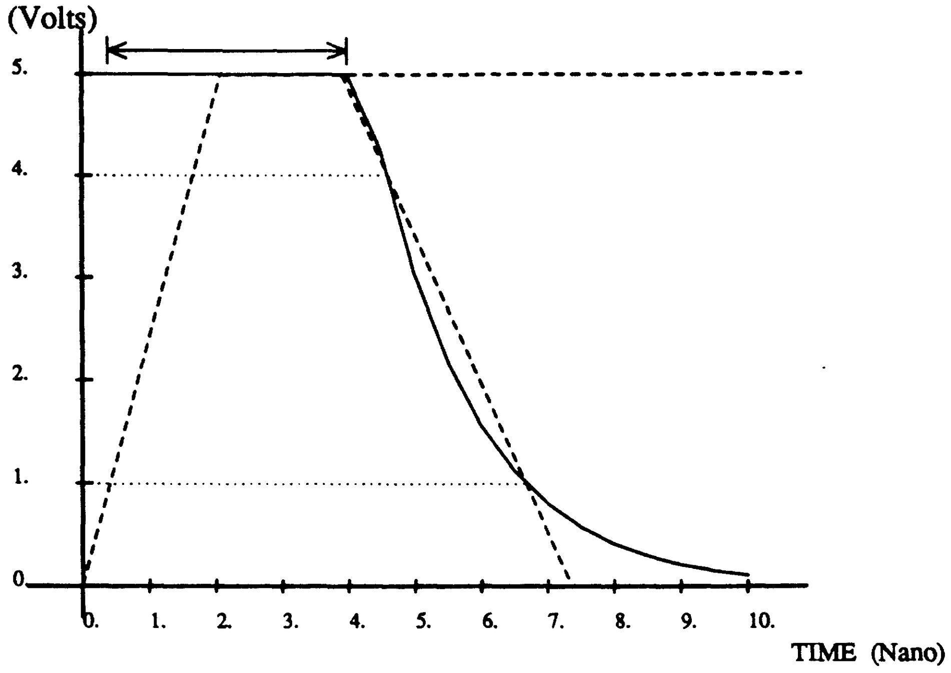


Figure 2-24: Finding driving resistance and delay offset for a logic gate.

The dashed lines represent the ramp waveform input and output to the driving logic gate. The solid line is an exponential curve fitted to the ramp waveform at the critical points shown. The driving resistance is found from the time constant of the exponential. The delay offset is indicated by the arrows at the top of the graph.

The driving resistance is a function of input transition time and load capacitance. A logic gate simulation is performed on the inverter with the specified input signal and the capacitance on the inverter output node as the load. This results in a delay time and a transition time. From the points defined by the delay time and the transition time, the output is viewed as a single time constant exponential. The time constant of the

exponential is used to calculate the driving resistance, and the starting time of the exponential is the delay offset. The input and output waveforms of the driving gate simulation, as well as the exponential fitted to the output waveform, is shown in Figure 2-24. The equations for calculating delay offset and driving resistance for a single time constant exponential waveform are

$$t_{offset} = t_1 - \tau \ln\left(\frac{V_f}{V_f - V_1}\right) \quad \text{and}$$

$$R_{dr} = \frac{t_2 - t_1}{C\left[\ln\left(\frac{V_f}{V_f - V_2}\right) - \ln\left(\frac{V_f}{V_f - V_1}\right)\right]}$$

for rising waveforms. V_f is the final voltage of the waveform. Rising waveforms will have an assumed initial voltage of zero. C is the capacitance of the node in question. V_1 and V_2 are voltages at times t_1 and t_2, respectively. The derivation of these relationships may be found in Appendix A. Similar relationships may be derived for falling waveforms. If V_1 and V_2 are defined as the 20% and 80% voltage points, which are the 1V and 4V points, the t_1 is the delay time and $t_2 - t_1$ is the transition time of the waveform, as defined in Section 2.2. The relations reduce to

$$T_{offset} = T_d - .161\,T_r \tag{2.6}$$

$$R_{dr} = \frac{T_r}{1.386\,(C_{ld} + C_{int})}.$$

T_d and T_r are the output delay time and transition time, respectively, of the driving gate simulation. The equations assume that the delay time and transition time are defined on the 1V and 4V points, but can easily be derived for other points. C_{int} is the parasitic capacitance of the output node of the driving gate. C_{ld} is the total load capacitance of the entire RC tree, including the driving gate output capacitance. For falling waveforms under the same assumptions, the relationships reduce to the same ones as Equation (2.6).

It was stated above that load capacitance has less effect on driving resistance than delay offset of a driving logic gate. This is also true for the calculations of the effective resistance of conducting and switching transmission gates. If the equations for T_{offset} and R_{dr} are combined with Equation (2.3), it is clear to see why.

$$T_{offset} = Td_{in} + R_{td} \times (C'_{int,Td} + C_{ld}) + .161 R_{tr} \times (C'_{int,Tr} + C_{ld})$$

$$R_{dr} = \frac{.161 R_{tr} \times (C'_{int,Tr} + C_{ld})}{1.386 (C_{ld} + C_{int})}.$$

$C'_{int,Tr}$ and $C'_{int,Tr}$ are the internal cell capacitances defined by the no-load output transition time and delay time, respectively, as in Equation (2.3). C_{int} is the output capacitance of the driving node, as specified by the cell representation. If $C'_{int} \approx C_{int}$, then the driving resistance is approximately $.116 R_{tr}$ and has little or no dependence on load capacitance. It should be noted, however, that R_{tr} is a piecewise linear function of input transition time. An analogous argument may be used to show the relatively small dependence of transmission gate resistance on load capacitance. The delay offset, however, is a linear function of load capacitance. The delay time of any output node will be substantially affected by the delay offset. Thus, it is important to specify the correct effective capacitance, C_{ld}, seen by the driving gate.

The calculations of delay offset and driving resistance account for effects of the input waveform on the driving gate. The input waveform is used for calculations of T_d and T_r, which are then used in equations for driving gate resistance and offset.

Once the RC equivalent for the driving gate is found, the equivalent RC circuits for any conducting transmission gates are calculated. An approximation for the input waveform is needed in order to calculate the resistance. The approximation used is the result from the driving gate simulation that was also used to find driving resistance. The approximation includes the effects of all the capacitances, but cannot include resistive effects since the values had not been determined yet. As a result of this, the approximation may be an overestimate on nodes toward the driving end of the tree, but an underestimate at the loading end.

The branches of the RC tree are terminated at transistor gate-type inputs, transmission gate source-drain connections when the gate is off, or at otherwise specified circuit outputs.

2.4.4.2 Waveform Calculation for a Driven RC Tree

Once the RC circuit is derived, then the two time constant waveforms for the nodes in the tree must be calculated.

From Equation (2.4), the quantities that must be found are τ_1, τ_2, and τ_z. The equations needed to find these values are derived in Chapter 3 of [15].

$$\tau_{De} = \sum_k R_{ke} C_k$$
$$\tau_P = \sum_k R_{kk} C_k$$
$$\tau_{Me} = \sum_k R_{ke} C_k \left(1 - \frac{\tau_{Dk}}{\tau_P}\right)$$
$$\tau_z = \tau_P - \tau_{De}$$
$$\tau_2, \tau_1 = \frac{\tau_P}{2}\left(1 \pm \sqrt{1 - 4\tau_{Me}/\tau_P}\right)$$

These equations all use resistance and capacitance values from the RC tree alone. C_k is the capacitance at node k. R_{ke} is the series resistance shared by the paths from the driving node to node k and from the driving node to node e. τ_{De} is the Elmore delay, or the first moment of the impulse response, of node e in the network[3]. τ_P is the upper bound of the responses of the network, and is also the sum of the open circuit time constants of the network. τ_{Me} may be found from the second moment of the impulse response of node e. The second moment is $2\tau_P(\tau_{De} - \tau_{Me})$.

An equivalent ramp is then derived for all nodes that are outputs or connected to the input of a logic gate. The procedure for this was described in Section 2.4.1.

2.4.5 Analysis of Transmission Gate Circuits: Switching Transmission Gate

As in the case when a transmission gate is driven by a switching logic gate, a circuit involving a switching transmission gate is analyzed by first finding equivalent RC trees. In this case, however, two equivalent RC trees must be calculated. One tree is connected to the driving gate. The other is the loading RC tree. The two trees are connected by the switching transmission gate. The input capacitance is included with the driving tree and the output capacitance is included with the loading tree. The effective resistance is included with the loading tree. An example of a switched RC circuit is shown in Figure 2-25. Also designated are the driving tree and the loading tree of the switched RC circuit.

---

[3]The N^th^ moment of waveform $v(t)$ is defined as $\int_0^\infty t^n v'(t)\,dt$, where $v(t)$ is the step response and $v'(t)$ is the impulse response of a network. The first order moment can be found more effectively by $\int_0^\infty (1 - v(t))\,dt$. Similarly, the second order moment can be found by $\int_0^\infty t(1 - v(t))\,dt$.

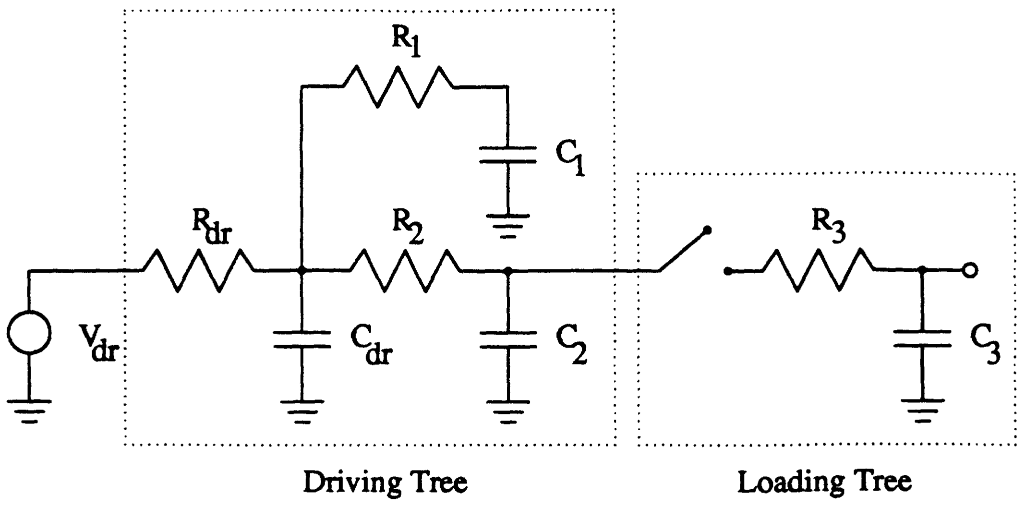


Figure 2-25: Switched RC circuit example

2.4.5.1 Constructing the RC Tree

The driving tree is found by calculating the *fixed-voltage resistances* of the devices connected to it. This includes the driving logic gate and any conducting transmission gates. Fixed-voltage resistance is the resistance of a driving gate or conducting transmission gate, where the gate is at a given voltage, assuming the voltage is not going to change. Thus, for an inverter with a high input, the fixed-voltage resistance of the n-type transistor will be used, since that is the conducting transistor in this case. The fixed-voltage resistance is used because it is assumed that there will be no waveform transitions on any of the driving tree nodes. Thus, the dynamic effective resistances used before have no meaning as there is no waveform upon which to base the calculations.

The fixed-voltage resistance is the resistance of a fully turned on transistor. It may be found by incrementally changing the voltage across the transistor and dividing the voltage change by the resulting current change. The fixed-voltage resistance per area should be the same for all transistors of that type. The fixed-voltage resistance of a transistor is simply

$$R_S = R_{/\mathrm{sq}} \times L / W$$

$R_{/\mathrm{sq}}$ is the resistance per square of the conducting transistor type of the gate. W and L are the width and length of the transistor, respectively. If there are series and parallel paths, as in a NOR or NAND gate, then fixed-voltage resistances of those transistors are combined accordingly. For a conducting transmission gate, the fixed-voltage resistance will be the n-transistor fixed-voltage resistance if the source/drain nodes are at a high voltage and the p-transistor fixed-voltage resistance if they are at a low voltage.

The capacitances of the driving tree are simply the output parasitics of the driving gate, the source-drain input and output capacitances of the conducting transmission gates, input capacitances of loading gates, and any other explicitly stated capacitances.

Since waveform transitions are occurring in the loading gate, the dynamic method of calculating effective resistances is used. The capacitances are found in the same way as described in Section 2.4.3.

The effective switching resistance and delay offset for the transmission gate that is turning on is found as described in Section 2.4.3.2.

If there are any conducting transmission gates in the loading tree, the method for calculating conducting resistance of transmission gates is used. However, an assumption for the waveform propagating through the transmission gate is needed since the resistance is a function of waveform. Given the effective resistance and load capacitance of the switching transmission gate, a time constant $R_{sw}C_L$ is found. The defined transition time of an exponential having this time constant is $1.386 R_{sw}C_L$. This transition time is used to find the effective resistance.

2.4.5.2 Waveform Calculation for a Switched RC Tree

The calculations to find τ_1, τ_2, and τ_z for connecting two RC trees is similar to those used for driving a single RC tree. The steps that differ are the calculations of τ_{De}, τ_P, and τ_{Me}. These results are also derived in Chapter 3 of [15].

$$\tau_{De}' = \sum_{k \in 2} (R_{nn_1} + R_{ke_2}) C_{k_2}$$

$$\tau_P' = \sum_{k \in 1} R_{kk_1} C_{k_1} + \sum_{k \in 2} (R_{nn_1} + R_{kk_2}) C_{k_2}$$

$$\tau_{Me}' = \sum_{k \in 2} (R_{ke_2} + R_{nn_1}) C_k \left[1 - \frac{\tau_{Dk}'}{\tau_P'} \right] - C_{T_2} \sum_{k \in 1} R_{kn_1} \frac{R_{kn_1} C_{k_1}}{\tau_P'}$$

The notations $k \in 1$ and $k \in 2$ mean "for all nodes k in tree 1" (the driving tree) and "for all nodes k in tree 2" (the loading tree), respectively. Node n is designated as the node in tree 1 that will be connected directly to tree 2. C_{T_2} is the sum of all the capacitances in tree 2.

CHAPTER THREE

Timing Simulation using Macromodels

Once the models for timing simulation have been developed, issues of implementation must be confronted. Decisions must be made as to what simulation method will be used, how cells will be represented, how circuits will be represented, and exactly how the delay calculations will be performed. The following sections deal with these topics.

3.1 Event Driven Simulation

There are two main approaches to circuit simulation, event driven simulation and time step simulation. They are also sometimes referred to as the waveform approach and the incremental approach, respectively, since event driven simulators apply an entire waveform at once to the sub-circuit to be analyzed while incremental approach updates the state of the entire network for every time step [35].

Time step simulation is usually used for circuit analyzers like SPICE. It operates by analyzing the entire circuit in small increments of time. There is a uniform step size at any given time for the entire circuit. This step size is chosen in order to produce the required accuracy for the most sensitive parts of the circuit [39]. Thus, the parts of the circuit that are relatively inactive are still analyzed using the minimum step size, which is very inefficient. Since the amount of computation needed to analyze a circuit grows more than linearly with the number of nodes, resources are strained if the circuit grows to much more than 100 nodes. Thus, incremental techniques are better used to carefully analyze the behavior of a smaller circuit. Larger circuits are more efficiently simulated using waveform techniques.

Event driven simulation uses a more modular approach for timing simulation. Cells may be analyzed for the entire time, or waveform, of interest, without analysis of other cells. Cells are then independent of each other in terms of timing simulation, with the exception of load impedance information needed from other cells connected to the cell outputs. Simulation of a cell starts when the inputs to the cell starts changing. If no inputs to the cell are changing, no simulation is performed. The amount of computation needed to simulate a circuit is in part dependent upon the inputs to the circuit, which determine the number of events that will take place. The computation may be expected to grow in a linear fashion with the number of elements in the circuit.

Cell macromodels are assumed to be uni-directional. This assumption makes event driven simulation practical. Uni-directional models assume that signal propagation is from node inputs to node outputs only. Thus, a cell is simulated only when its inputs are changing, and using only load impedance information from other cells. Since local feedback outside cell boundaries is not allowed, small time step analysis using both input and output waveforms is not needed.

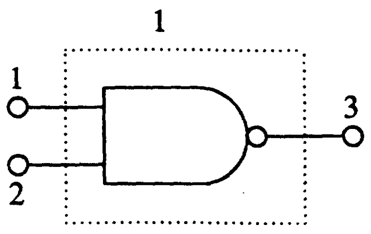
Large MOS circuits generally have a high degree of latency, meaning that large portions of the circuit are inactive at any given time during circuit operation. Exploiting latency saves both computation time and storage by taking advantage of this inactivity [35]. Macromodeling using event driven simulation exploits latency by simulating only those cells whose inputs are changing. Since most cells in a large circuit are inactive at any one time, much less computation is needed.

Event driven simulation takes advantage of many properties of large circuits. If these circuits are represented by smaller cells in a macromodeling approach, event driven simulation is made practical and desirable. Savings in both computation and storage may be gained.

3.2 Module Representation

A *module* is defined as a reduced data representation of a sub-circuit. Each type of lowest level sub-circuit, or leaf cell, of a system must be modeled as a module. The technology and layout of a module is assumed to be fixed, although scaling parameters with variables such as transistor width has been discussed in Section 2.3.5. A module is described by a module name, input node names, output node names, internal node names, input and output impedances, and a procedural delay representation.

An experimental simulator program uses a library of procedures to create and initialize all modules. Each module has a separate short procedure to perform the creation, assigning the correct representation for that module to each field. The cell name, input node names, and internal node names are constants. The cell name is represented internally as an integer while the input and internal nodes are arrays of integers. If no internal nodes are to be represented, then an empty integer array is used. The other fields use separate procedures as part of the module representation. An example representation of a two-input NAND gate module is shown in Figure 3-1. The



```
Record{
    Module_name : 1,
    Inputs : array[int][1,2],
    Outputs : Nand2_outputs,
    Internal_nodes : array[int][],
    Load_rep : Nand2_load_rep,
    Delay_rep : Nand2_delay_rep
    }
```

Figure 3-1: Sample module representation of a NAND gate.

integer representation for module and node names is used. *Nand2_outputs, Nand2_load_rep*, and *Nand2_delay_rep* are the names of the procedures used to yield output nodes, input and output impedances, and delay values respectively.

Most cells are inherently uni-directional, so definition of module inputs and outputs is simple and constant under all conditions. Transmission gates, however, are bi-directional circuits. Given a conducting transmission gate, there is nothing about the transmission gate itself that dictates which way a signal will propagate through its source-drain nodes. The circuit configuration containing the transmission gate, plus, in some circumstances, the state of the circuit, will dictate which source-drain node is the output and which is the input at any given time. Thus, a procedure *Out_node* is used to return output nodes of a cell when the program needs them. *Out_node* returns output nodes as a function of a cell input node. If the input node is either of the transistor gates, then both source-drain nodes will be returned. If the input is a source-drain node, then the other source-drain node will be returned. The code calling *Out_node* determines which node is the input, or driving, node from the circuit configuration.

Each input and output node also has a node impedance representation, *Calc_load*. The impedance of each node is either a capacitance or an RC circuit. The impedance may be dependent on the state of the nodes, as in a transmission gate. If the transmission gate is on, then the output impedance will be an RC circuit. If the transmission gate is off, then it will be the capacitance of one of the source-drain nodes. A run_time procedure is needed to yield these impedances because of possible dependence on node states. *Calc_load* is called when the input impedance is needed for loading information for another cell or the output impedance is needed for creating an RC tree.

Delay representation is an important field in the module representation, since all the data and code necessary to evaluate the delay of an instance is contained within it. The delay representation contains parameter values for delay and transition time calculation. It also contains parameters for static output resistance calculation. The static resistance values are carried along with the instance for RC tree calculations. Also, code exists for the above calculations based on the node states and the input transition, as well as any other instance parameter values, such as transistor width. This code updates the states of the nodes as well.

This procedure, *Calc_delay*, is called every time delay calculation is performed upon an instance of this module type. For transmission gate cells, *Calc_delay* is called to evaluate the effective resistance for construction of RC trees. Separate procedures exist to operate on all RC trees. These RC calculation procedures are independent of the module delay representations. This is because the calculation for an RC tree may include several instances. One group of calculations is done for all the instances contained in the RC tree.

Mathematical functions called by *Calc_delay* are piecewise linear functions with an arbitrary number of pieces (usually a function of input transition time) and linear functions (usually a function of load capacitance). There is also a procedure, used for transmission gate resistance calculations, that uses an inverse linear function of capacitance for the slope of one of the piecewise linear pieces, i.e.

$$f(Tr_{in}) = f_0 + m1 \times Tr_{in} \qquad t_{brk1} \leq t < t_{brk2}$$

where

$$m1 = (m1_{int} + m1_{sl} \times C_{load})^{-1}.$$

Variables f_0, t_{brk1}, $m1_0$, and $m1_{sl}$ are all module parameters. f_0 is the intercept, $m1$ is a slope, and t_{brk1} and t_{brk2} are breakpoints of a function with input transition time Tr_{in} as the independent variable. The slope, $m1$, is also shown as function of load capacitance, C_{load}. $m1_{int}$ is the intercept and $m1_{sl}$ is the slope of $m1$ with load capacitance, C_{load} as the independent variable.

The piecewise linear parameters are specified in the following manner, shown here in a generalization of the CLU language:

```
record{ $f_{0_{int}}$ $f_{0_{sl}}$,
    array[
        section{$m1_{int}$ $m1_{sl}$ $brk1_{int}$ $brk1_{sl}$},
        section{$m2_{int}$ $m2_{sl}$ $brk2_{int}$ $brk2_{sl}$},
        section{$m3_{int}$ $m3_{sl}$ $brk3_{int}$ $brk3_{sl}$},
        .
        .
        .    ]
```

Some of the above parameters are illustrated in Figure 3-2. The specification of a parameter set starts with a parameter set, $f_{0_{int}}$ and $f_{0_{sl}}$, which is used to find the initial value or intercept of the function with respect to input rise time. An array of specifications for piecewise linear sections follows. Each **section** is a record consisting of a set of parameters for the slope and a set for the breakpoint indicating the start of that section on the time axis. The size of the array of sections is variable, so the number of piecewise linear sections is variable as well. Each slope, breakpoint, and intercept is a linear function of capacitance and consists of two parameters, the slope and intercept with respect to capacitance. For instance,

$$m1 = m1_{int} + m1_{sl} \times C_{load}.$$

3.3 Circuit Structure

An *instance* is an occurrence of a sub-circuit within a circuit. It is described by a *module* within a circuit. A circuit is modeled as connections of instances of modules. This is illustrated in Figure 3-3. There are three basic module types in this example, *M1*, *M2*, and *M3*. The circuit contains three instances of *M1*, named *I1*, *I2*, and *I3* respectively. There is one instance of *M2*, named *I4*. Two instances exist, *I5* and *I6*, of type *M3*. Instance node names are the same as module node names.

An instance table contains instance names and instance representations of the circuit. A connection table contains information about how the instances are connected. Hierarchy is not useful in delay calculations with the given macromodels, as delay calculations are always done on instances of one module. However, hierarchy may be useful in input specification of the circuit. In this case, a preprocessor could be used to flatten the circuit to the lowest level of defined instances. Hierarchy could be useful if more general delay calculations may be derived from other delay calculations. In other

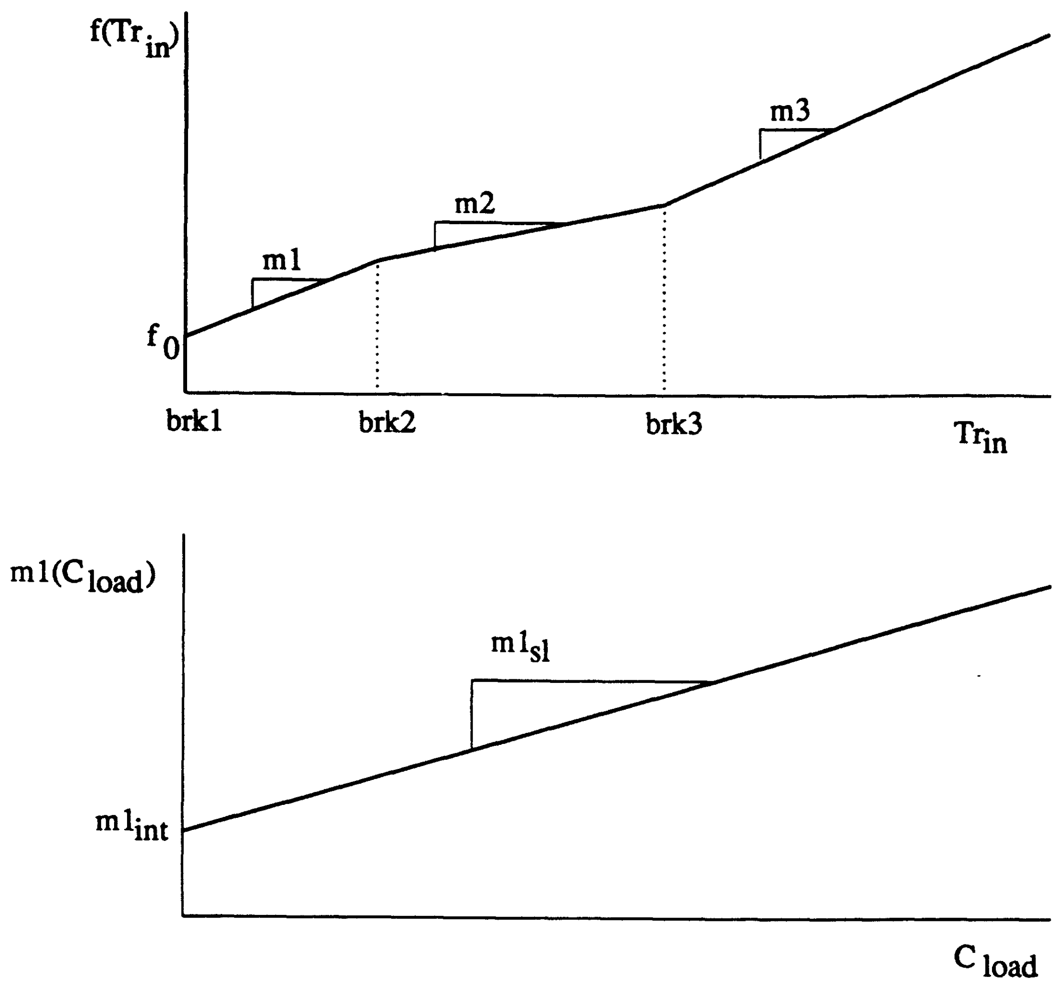


Figure 3-2: Functions described by a parameter set.

The top graph shows a piecewise linear function described by parameters with input transition time as the independent variable. Each of these parameters is then a linear function of capacitance as shown with slope *m1* in the bottom graph.

words, it would be useful to combine two or more macromodels to make a larger macromodel.

An instance is represented by the instance name, the module definition, the states of the nodes, the static output resistance, and a miscellaneous parameter field. An instance of the NAND gate module of Figure 3-1 is shown in Figure 3-4. The instance name is represented internally as an integer, as in the module name. The module definition states the module type of the instance and points to the module definition

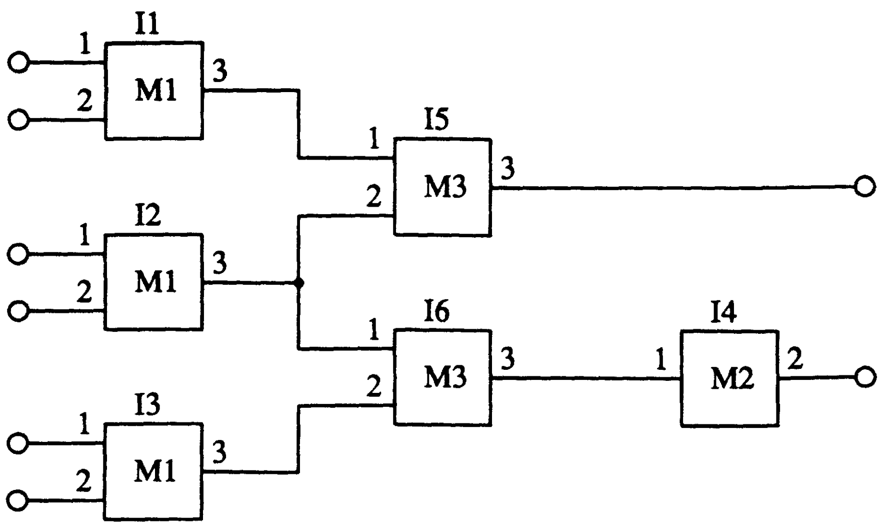
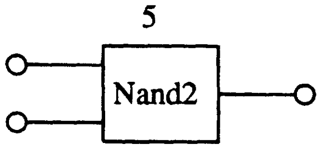


Figure 3-3: A circuit specified as connections of instances of modules.



```
Record{
    Instance_name : 5,
    Module_type : Nand2,
    Node_states :array[int][HI,LO,HI],
    Static_res : array[real][10000.],
    Parameters : array[real][],
    }
```

Figure 3-4: Instance representation of NAND gate module.

described in Section 3.2. Every input, output, and represented internal node is assigned a node state. The state of a node may be high (HI), low (LO), or unknown (X). Node states are assigned during the initialization of the circuit and during delay calculation procedures. The circuit initialization step is done in the same way as a switch-level simulation, that is, performing the simulation without incorporating any delay times.

The static output resistance of each output node, which is the effective output resistance of the driving conducting transistors, is calculated during the delay calculation procedure. The values are stored as an array of real numbers as a part of the instance representation for use in analysis of switching transmission gate circuits.

The remaining field of representation is an array of real numbers that may be used for miscellaneous parameters. The meaning of the parameters may vary with different module types. For a capacitor module, it is used to specify the value of capacitance. For a variable width inverter, it may be used to store the transistor widths. Each procedure of the module representation may use these parameters, and they may be used differently for different modules. If no other parameters are needed to specify the instance, then the array may be empty and it will be ignored.

3.4 Event Scheduling

Event driven simulation implies the need for an event queue. Events are placed in the queue in chronological order, that is, the order in which the events will take place in time. Thus, the next event to take place at any time is simply the event at the top of the queue.

The event representation consists partly of the waveform model, represented by delay time, transition time, and a rising/falling flag. Also included are the instance name and node number upon which the waveform will act. The delay time of the transition is repeated as a separate field, which is equal to the time the event will occur. The ordering of the queue is based on this value.

The basic simulation loop of the program is simply a loop that runs until the queue is empty. Once the queue is empty, no more events will occur in the circuit. Inside the loop are procedures that fetch the next event from the queue, perform delay calculations caused by the event, and then schedule the events resulting from the delay calculation.

```
Schedule Forced User Defined Inputs
While not{queue$empty(event_queue)} do
    Get next event from event_queue
    Calculate delay of instance affected as a function of the event
    For each output node of the instance
        Find instances affected by output transition, if any
        Schedule affected instances and transitions in event queue
        End
    End
```

3.4.1 Coincident Events

The event scheduling described above does not adequately handle coincident events, i.e. two transitions occurring on the same instance at approximately the same time. This may give rise to a series of transitions on the output node closely spaced in time. A pair of such transitions would effectively cancel each other out. Figure 3-5 illustrates an example of this type of action. Two issues arise from this situation: queue priority and event cancellation.

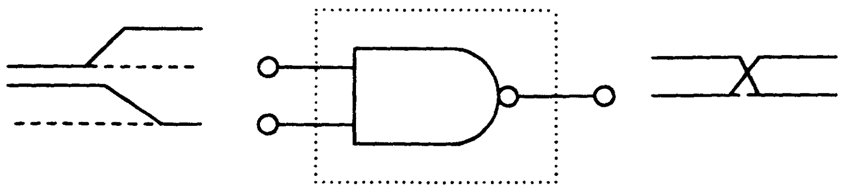


Figure 3-5: Two almost coincident transitions on a NAND instance.

Scheduling queue events was described briefly in previous paragraphs. It was stated that event scheduling was done according to the delay time of the signal. This implies that events scheduled to start at the same delay time are fetched in the order they were put into the queue, without regard to any other variables, such as transition time. However, situations may arise where this may not be adequate. Consider the set of transitions shown in Figure 3-6. Although transition *T1* has a delay time earlier than transition *T2*, it is not clear that it should be scheduled earlier in the event queue. If both transitions were to act upon the same instance, the instance may react first to *T2*. This is because although *T1* occurs first in terms of delay time, *T2* reaches the switching voltage of the instance, V_{sw}, before *T1* because of the faster transition time. Thus, events may be more correctly scheduled as a function of delay time, transition time, and the switching voltage of the instance. This may be difficult because the switching voltage varies with different circuits. Thus, queue priority would be a function of the instance affected. Of course, one could estimate a switching voltage that would be close to most encountered. It would be more accurate then simply using the 20% delay time.

Figure 3-5 shows the output transitions of the instance, consisting of two closely spaced transitions whose effects would ultimately cancel. Instead of calculating the effects of both these transitions throughout the circuit, which again would cancel, both events could be removed from the queue, saving unnecessary computation. Obviously,

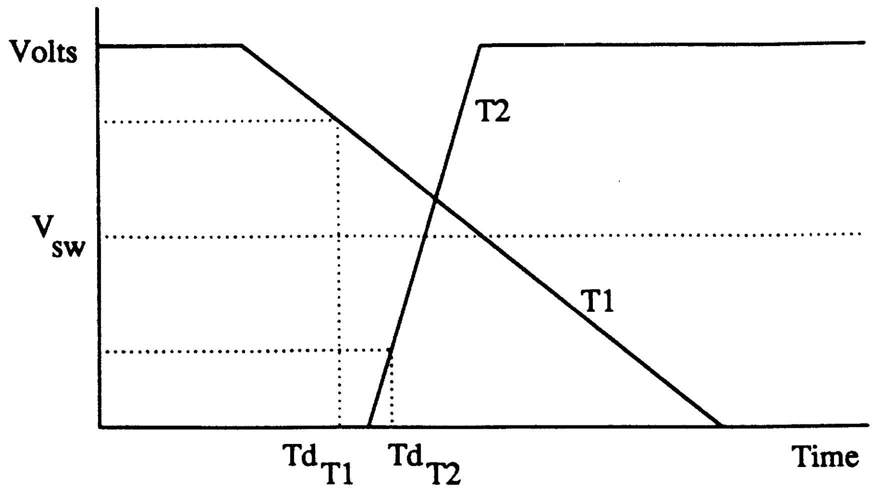
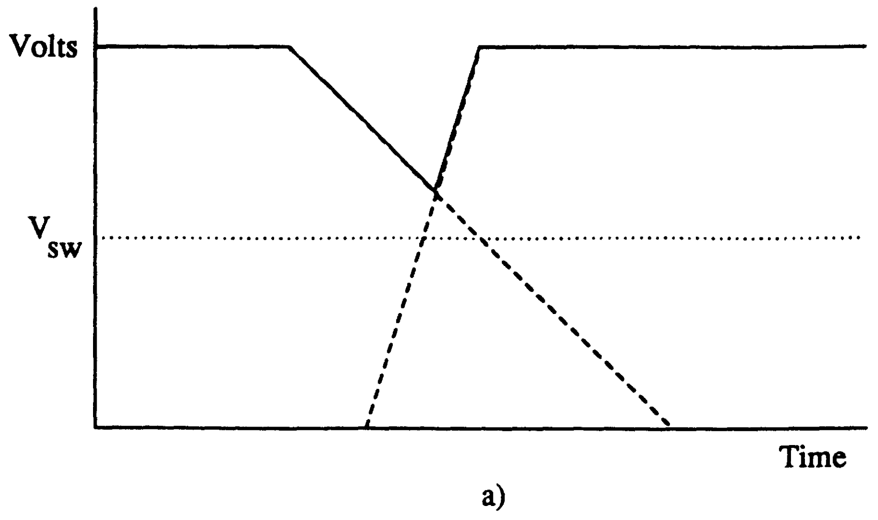


Figure 3-6: Closely spaced transitions with different transition times.

only pairs of conflicting events may be canceled, such that the final state would be the same as the initial state. Also, the event pair must consist of a rising transition and a falling transition. The question arises of determining exactly when to represent the transitions as unique transitions and when to cancel them. One way is to cancel them whenever the delay times are closer together than a certain tolerance. However, again the issue of transition time and the delay of the transitions relative to the switching voltage of the instance arises. Another method might be to take the maximum (for a high-low-high transition) voltage at any time point or the minimum (for a low-high-low transition) voltage at any time point of the two transitions. If this maximum (minimum) of the transitions does not reach the switching voltage of the instance, then the two events are canceled. If the switching voltage is reached, the events are scheduled. These two situations are shown in Figure 3-7. This way, there is no scheduling of events having transitions that would not reach the point of having an effect on the instance as a result of another transition occurring.

Since nodes driven by two different sources are not allowed, then two successive transitions in the same direction should never occur at the same node. This assumes that correct event canceling algorithms are implemented for coincident events, as described above.

Certain types of circuits depend on two or more actions taking place at the same time. Consider the circuit in Figure 3-8. When *Phi* is low and *Phi-bar* high, the feedback loop is closed. Input *D* is free to change and will not affect the node states in



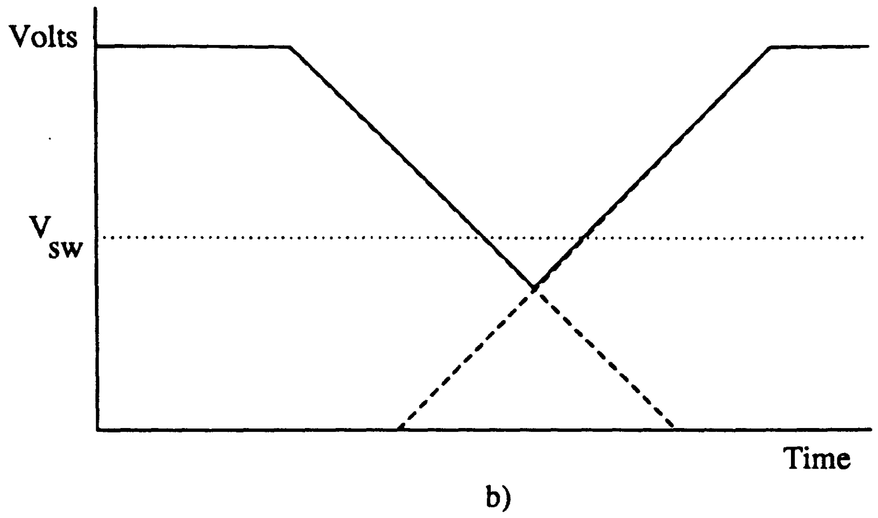


Figure 3-7: Canceling events.

a) Two transitions that will be canceled. The maximum of the transitions does not reach the designated switching voltage. b) These two transition will be scheduled as the maximum of the transitions reaches the switching voltage.

the cross-coupled inverter circuit since the input transmission gate is off. Due to the event scheduling method used, serious problems may arise when *Phi* changes, even ignoring the race problems that the circuit may actually experience. When *Phi* and *Phi-bar* change, events are scheduled for simultaneous times on both transmission gates. For correct functioning, the transmission gate that is turning off should be evaluated

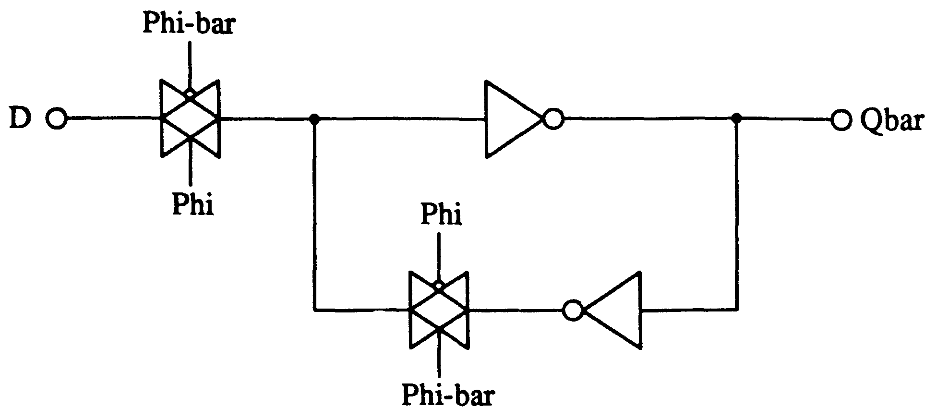


Figure 3-8: CMOS two phase flip-flop.

before the transmission gate that is turning on, since with the type of event driven simulation used, the transmission gates cannot be evaluated at the same time. However, there is no guarantee that this would happen. Thus, the states may be updated in such an order that both transmission gates could be on. Thus, the input node to the forward inverter is driven by two sources. If input *D* has changed, then these sources may be at different voltages. One way to handle this would be to make the entire circuit a module. The problem with this, as stated in Section 2, is that modeling a combination logic/transmission gate cell with transmission gates on the cell input is difficult. This problem has not been addressed thoroughly in this thesis, so future work is needed.

3.5 Input Specifications

The input to the program presently consists of two files, a connection file specification, and an input transition specification. The connection file contains information about the content and configuration of the circuit. This includes instance names and their module types, along with any instance parameters. The module type is designated by the module name and must exist in the module library. Modules that have been implemented include an inverter, a two input NAND gate, a two input NOR gate, a domino logic block, an adder cell, and a transmission gate, all in CMOS technology. Explicit circuit inputs and outputs are also specified. Finally, connections between instances are designated.

Information about the node states and input transitions are specified in the input transition file. First, any user specified initialized node states are input. Then, any scheduled input changes are included, with instance name and node, delay time, transition time, and rising/falling data.

3.6 Program Organization

The overall order of operations in the experimental simulator program is:

Read input files
Create modules
Create instances
Build instance table and connection table
Initialize nodes
Schedule inputs and start simulation loop
After all changes processed, output results

Most of the above steps are initialization. Node initialization is done with a queue much like delay calculation. Instead of delay calculation, the instance output states are determined from the input states, i.e., a logic simulation is done. Logic simulation may be thought of as the simplest simulation mode of a circuit, in terms of circuit models, variables, and information provided. For instance, transistors are often modeled simply as a switch, compared to the complex device models of circuit analyzers or delay functions of macromodels. Since devices have zero delay in logic simulation, there is no time scale for a given set of input values as in timing or circuit simulators. The output information provided is simply the node values of the circuit, without any delay or waveform information that would be provided by higher level simulators. The simulation steps for node initialization used in the experimental simulator are the same as for delay calculation, except that modules now have zero delay. If an output state is updated, for example, from an unknown state to a low state, then any instances connected to that output node are then scheduled to be updated in the initialization queue. At the end of the node initialization step, all non-isolated nodes should be either at a high or a low state.

The real work occurs in the simulation loop. The basic structure of the simulation loop was described in Section 3.4. Inside the simulation loop, one of the steps described performs the vast majority of the computations. This is, of course, the step performing the delay calculations on the instances. This step is described in a little more detail.

The procedure used to initiate a delay calculation operation in the simulation loop is **instance$calc_delay**. This procedure, for a given instance and input transition, gets the node states of the instance, calls **instance$calculate_loads** to get the load impedance on the output nodes of the instance, which may be a capacitance or an RC tree, and calls **module$calc_delay**, which in turn will call the correct delay procedure for that module as a function of the load, input transition, and node states.

If the module is a logic gate, output transition(s) will be returned by **module$calc_delay**. Along with the input waveform and node states, the output transition calculated at this point is a function of load capacitance only. If the load was of RC tree form, additional analysis will be performed to complete the derivation of the RC tree and calculate the correct outputs. The output transition returned at this point will be used to calculate the driving resistance and the delay offset as described in Section 2.4.4.1. If the module is a switching transmission gate, a resistance and delay offset is returned. A summarized form of an example delay calculation procedure for an inverter is shown below.

```
inv4_calc_delay
  %%%% Delay and transition time parameters %%%%
  own dr := pm$create(
          .9091e-9,11360.,
          ars$[pm$create_section(0.,0.,.85,0.),
             pm$create_section(11.43e-9,28570.,.6259,9.259e10)])
  own df := pm$create(
          .6923e-9,5769.,
          ars$[pm$create_section(0.,0.,.7421,1.316e11),
             pm$create_section(9.33e-9,16667.,.4154,1.923e11)])
  own tr := pm$create(
          4.22e-9,40740.,
          ars$[pm$create_section(0.,0.,0.,0.),
             pm$create_section(3.6e-9,40000.,.23,-6.522e10),
             pm$create_section(38.e-9,100000.,.1033,1.667e11)])
  own tf := pm$create(
          2.4e-9,20000.,
          ars$[pm$create_section(0.,0.,0.,0.),
             pm$create_section(2.22e-9,22222.,.221,0.),
             pm$create_section(26.35e-9,76470.,.1222,1.389e11)])
  %%%% Static resistance parameters %%%%
  own nr:real := nres_per_sq*Ln/Wn
  own pr:real := pres_per_sq*Lp/Wp
  %%%% get direction of input waveform
  rising := transition$get_transition_type(in_wave)
  %%%% Update states and select parameter set %%%%
```

```
if rising
  then
     in_node_state := hi
     out_node_state := lo
     static_resistance := pr
     delay_parm := df
     transition_parm := tf
  else
     in_node_state := lo
     out_node_state := hi
     static_resistance := nr
     delay_parm := dr
     transition_parm := tr
  end
%%%% Do delay calculation %%%
if load is capacitive
  then  %% If a capacitive load
     Get load capacitance C_ld
     Calculate delay time Td_out from C_ld, in_wave,
       and delay_parm
     Calculate transition time Tr_out from C_ld, in_wave,
       and transition_parm
     Create out_wave from Td_out and Tr_out
  else   %% If an RC tree load
     %% Calculate delay offset and driving resistance
     Get loading RC tree
     Get total capacitance C_tot of RC tree
     Calculate delay time Td_out using C_tot, in_wave,
       and delay_parm
     Calculate transition time Tr_out using C_tot,
       in_wave, and transition_parm
     Calculate driving resistance dr_res using C_tot
       and Tr_out
     Calculate delay offset T_off using Td_out and Tr_out
     Assign dr_res to tree driving resistance
     Create out_wave from T_off and Tr_out
  end
return(out_node,out_wave,dr_res)
end inv4_calc_delay
```

For each output node of the instance, if the instance is a logic gate with a capacitive load, the results are yielded to the simulation loop. If the instance is a switching transmission gate, the driving and loading trees are constructed and RC tree calculations are performed. Results are yielded for any logic gate inputs connected to the RC tree. If the instance is a logic gate with an RC tree load, construction of the loading RC tree and RC tree analysis is done. Again, results are yielded for any logic gate inputs connected to the RC tree.

CHAPTER FOUR

Experimental Results

Now that macromodels and simulation methods have been developed, tests are needed for evaluation. The following sections describe testing methods and test results for various circuits.

4.1 Testing Method and Goals

A test program was written to test the accuracy of the models and approximate execution times of the delay calculation. The experimental simulator program is about 3000 lines of code implementing the basic functions needed to test the models. The procedures were written in CLU and run on a DECSYSTEM-20 and a VAX running UNIX.

Macromodels were developed for an inverter, a two input NOR gate, a two input NAND gate, a domino logic block, a transmission gate, and a three bit adder. All of these circuits are CMOS. Various types of circuits using these macromodels were tested. The following steps were followed to develop a typical macromodel.

1. The cell to be modeled is simulated over a range of load capacitances and input rise times. The cell is fixed at the layout level. The number of points needed per input/output terminal pair depends on the cell to be modeled. For example, an inverter would require more data points than an adder cell since the output is more dependent on the input transition than in the adder case. Non-linearities in the output transition time versus input transition time also tend to be more predominant for an inverter, requiring more data points and more piecewise linear sections to model the effects.
2. Output files resulting from the SPICE simulations are analyzed with a program written to calculate delay times and transition times of waveforms. This program consists of about 800 program lines. In the case of transmission gates, a similar program calculates effective resistance and delay offset.
3. The delay times and transition times (or effective resistance and delay offset) are then plotted versus input transition time and load capacitance. These linear and piecewise linear curves are the basis for the parameters used in the macromodel. Software for this step of the macromodeling is incomplete and is currently done manually.

4. Finally, procedures are written to represent the macromodel as a module. These procedures include output node representation, input and output impedance representation, and delay and logic representation. The same procedure format suffices for most simple gate forms, with changes needed for specific parameters, input and output node names, and logical behavior.

The evaluation of the models and the simulation method is done by comparing delay times (model evaluation) and program execution time (simulation evaluation). The basis for comparison is SPICE simulation of the circuits. These circuits were all originally extracted from layout. The SPICE model files used are for a 2 micron CMOS process and may be found in *The Design and Analysis of VLSI circuits* by Glasser and Dobberpuhl [37].

The circuit delay time of a particular node is defined as the time of change of the node minus the delay time of the input signal. Absolute error is simply the experimental delay time minus the delay time derived from SPICE simulation waveforms. Percentage error is the absolute error divided by the SPICE result values.

Execution time improvement is measured by the execution time needed for the SPICE simulation divided by the execution time needed for the experimental simulation.

A variety of circuits were tested to illustrate the effectiveness of the models in several areas. Accuracy over a range of inputs and loads for a sample macromodel is observed. In order to test accuracy with a varying number of instances, simulations of circuits with a string of gates and circuits with combinations of gates are performed. Circuits with switching and conducting transmission gates are simulated to test those models. Accuracy in circuits with internal feedback is observed. A larger circuit, in this case an array multiplier, is tested. Finally, comparisons using an RC line model for polysilicon interconnect are made.

4.2 Test Circuits and Results

This section shows the test circuits, a table of results and some explanation for a variety of circuits. The execution times of all the circuits are included in one table, Table 4-14 on Page 87.

Inverter

The inverter circuit, shown in Figure 4-1, was used to test simulations over a range of input waveforms and load capacitances. The inverter layout is shown in Figure 4-2. Transistors are 4 microns wide and 3 microns long. Results included in Table 4-1. are for falling inputs.

The largest errors encountered were for a small load capacitance and a fast input transition. Of course, this is the condition for minimum delay and transition time, so a larger percentage error occurs for a given absolute difference. In this case, one must look to the absolute difference between experimental times and SPICE times to see that there is a small difference between the two.

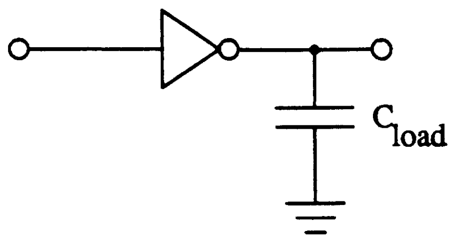


Figure 4-1: Inverter circuit.

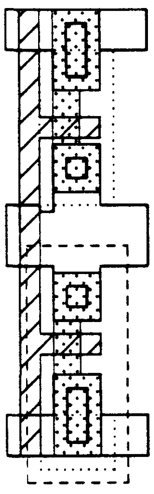


Figure 4-2: Layout of the inverter cell.

Inverter Circuit Results				
Case	Model Delay (ns)	SPICE Delay (ns)	ΔTd (ns)	% Td
1: .6ns,.05pF	1.987	2.360	-.373	-15.81
2: .6ns,1.0pF	12.78	12.52	.26	2.08
3: 24ns,.05pF	19.43	19.40	.03	.155
4: 24ns,1.0pF	32.67	32.99	-.32	-.97
5: 6ns,.2pF	8.28	8.59	-.31	-3.61

Table 4-1: Inverter circuit test results

Tests for varying input transition times and load capacitances.

The improvement in execution time for the experimental simulator, shown in Table 4-14, ranges from about 26 to 45 times faster than SPICE for this circuit.. Since only two instances and one event are needed in the test simulation, and delay calculations are very simple, most of the execution time is due to overhead. The gain in execution is less than other tests since SPICE is performing a relatively short and simple analysis.

Inverter String

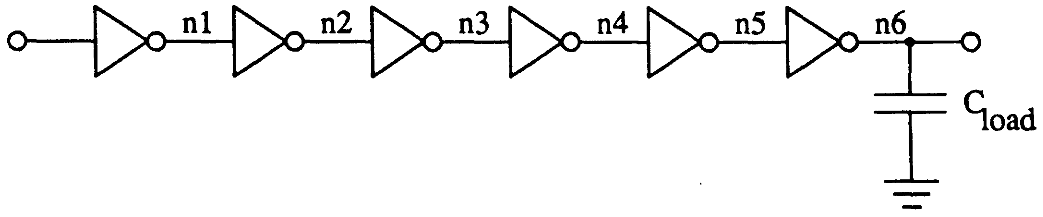


Figure 4-3: Inverter string test circuit.

A string of inverters, shown in Figure 4-3, was tested to observe how error behaves as a function of node position in the string. Table 4-2 shows that neither error percentage nor absolute difference in delay times consistently increased with increasing distance from the input node.

A larger improvement in execution time is seen as SPICE analyzes a larger circuit. The execution time of the experimental simulator is only very slightly higher than the single inverter case. Additional nodes and signal propagations seem to cause a relatively smaller rate of increase in execution times in the experimental case.

Inverter String Results				
Node	Model Delay (ns)	SPICE Delay (ns)	ΔTd (ns)	% Td
n1	14.05	14.27	-.22	-1.54
n2	21.63	21.93	-.30	-1.37
n3	26.79	26.60	.19	.714
n4	30.96	31.22	-.26	-.832
n5	35.61	35.75	-.14	-.392
n6	41.66	42.42	-.76	-1.79

Table 4-2: Inverter string test results

Inverter string test with rising input, input transition time of 24ns, and load capacitance on the output node of .2pF.

Domino logic block with p-transistor feedback

The test domino circuit is logically a trivial circuit, since statically, the output will be at the same state as the input. The point of this test is to illustrate how feedback internal to the macromodel affects accuracy and execution time.

All transistors are 4 microns wide and 3 microns long.

Both percent and absolute accuracy are very small. Although feedback is present in the circuit via the p-transistor across the output inverter, the feedback is decoupled from the input nodes. Feedback effects are then included in the circuit macromodel.

The improvement in execution time is higher than for the single inverter case because SPICE must incrementally analyze the feedback effects. For the experimental case, the effects of feedback are simply incorporated in the delay parameters of the macromodel and require no special analysis. Notice that the execution time of the experimental simulator for this circuit is approximately equal to its time for the single inverter case. This is because, again, there are two instances and only one event as in the inverter case. Thus, a much higher gain in execution time than for the single inverter case is seen.

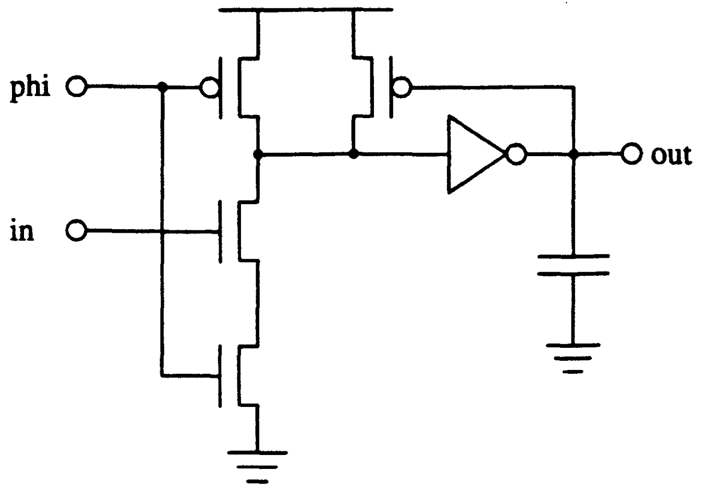


Figure 4-4: Domino logic test circuit.

Domino Circuit Results				
Input	Model Delay (ns)	SPICE Delay (ns)	ΔTd (ns)	% Td
1: phi falling	14.63	14.65	-.02	-.14
2: phi rising	39.20	39.65	-.45	-1.14
3: in rising	39.14	39.84	-.7	-1.76

Table 4-3: Domino circuit test results

Load capacitance on the output node is .2pF. All input transition times are 6ns.

String of Domino Logic Blocks

Figure 4-5 shows the connections for a string of domino logic gates. Again, comparison of delay propagation through this circuit shows no consistent increase in absolute difference or percent error.

Comparison of a string of these domino blocks shows a big improvement in execution time, 538 times faster, since SPICE must now perform analysis of the feedback for all five instances of the circuit. The test simulator CPU time increase is only a function of the increase of the number of cells and transitions and is not very different from the inverter string case.

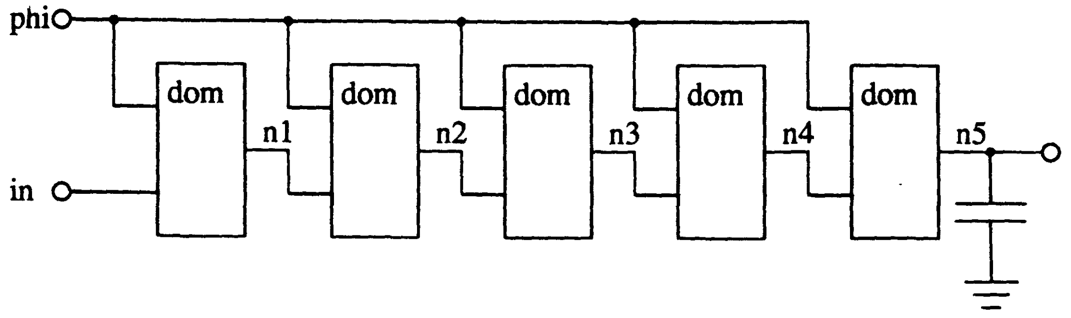


Figure 4-5: String of domino logic blocks.

The circuit represented by module *dom* is shown in Figure 4-4.

Domino String Results				
Node	Model Delay (ns)	SPICE Delay (ns)	ΔTd (ns)	% Td
n1	32.49	31.57	.92	2.91
n2	68.31	67.20	1.11	1.65
n3	104.1	102.8	1.3	1.26
n4	139.9	137.6	2.3	1.67
n5	181.5	180.9	.60	.33

Table 4-4: Domino string test results

The load capacitance is .2pF. *phi* is high and *in* is rising with a transition time of 6ns.

Logic gate realization of an adder cell

The circuit in Figure 4-6 is a 12 gate implementation of a three bit adder cell. It uses three types of gates. All gates are made up of minimum sized transistors, that is, 4 micron wide and 3 micron long transistors. Thus, interaction of three different macromodels in a more complex configuration is tested.

The input signals applied to the adder are shown in Figure 4-7, This series of input transitions suggests, assuming small gate delays with respect to the delay between inputs changing, that the *sum* output would rise and fall twice. The *carry* output would rise once and then fall. However, SPICE simulations show that this is not the case for the *sum* output. The gate delays were such that the sum only made two transitions, as succeeding inputs affected potential intermediate transitions of the sum output. The experimental simulator, however, because of the event driven nature of the simulation algorithm, produces intermediate transitions. These transitions overlap, however, to produce a waveform similar to the actual SPICE waveforms. Also, the final delay times of the *sum* and *carry* output are still very close to the SPICE delay times, even though the experimental simulator exaggerated the intermediate transitions. The SPICE and test waveforms are shown in Figure 4-8.

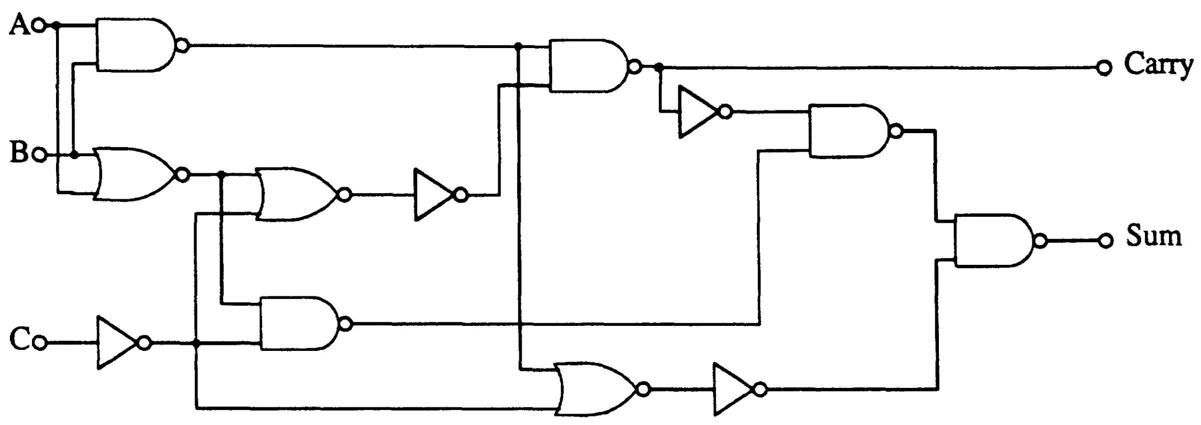


Figure 4-6: Logic gate implementation of a three-bit adder.

Results for Adder Circuit				
Output	Model Delay (ns)	SPICE Delay (ns)	ΔTd (ns)	% Td
Sum rising	27.86	28.87	-1.01	-3.5
Carry rising	36.93	37.53	-.6	-1.6
Sum falling	87.54	86.49	1.05	1.21
Carry falling	116.4	115.6	.8	.69

Table 4-5: Adder circuit test results

The load capacitance on the output nodes are both .2pF. The inputs to the cell are shown in Table 4-7.

Results are shown in Table 4-5. Errors in delay time behave consistently with those found in previous tests. The execution time of the experimental simulator is larger than for the previous tests since there are more instances and several transitions occur. However, SPICE execution time also increases because of the larger circuit and longer simulation time. The gain, however, is very large as the experimental simulator runs 900 times faster than SPICE. Increase in execution time as a function of circuit size and circuit activity is much smaller in the experimental case than for SPICE.

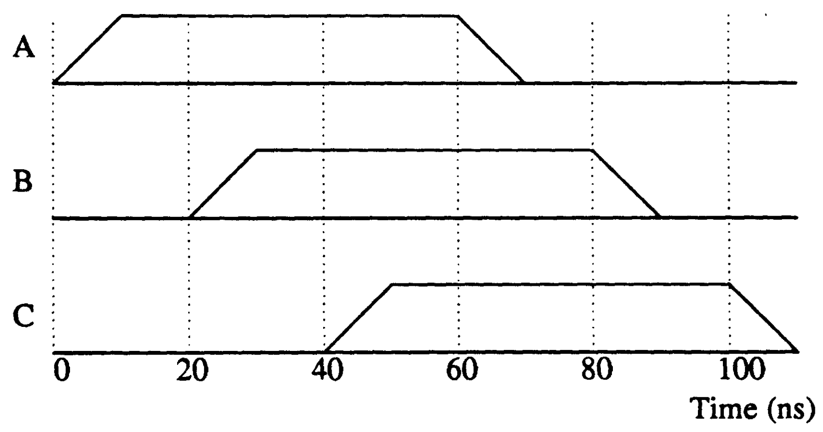
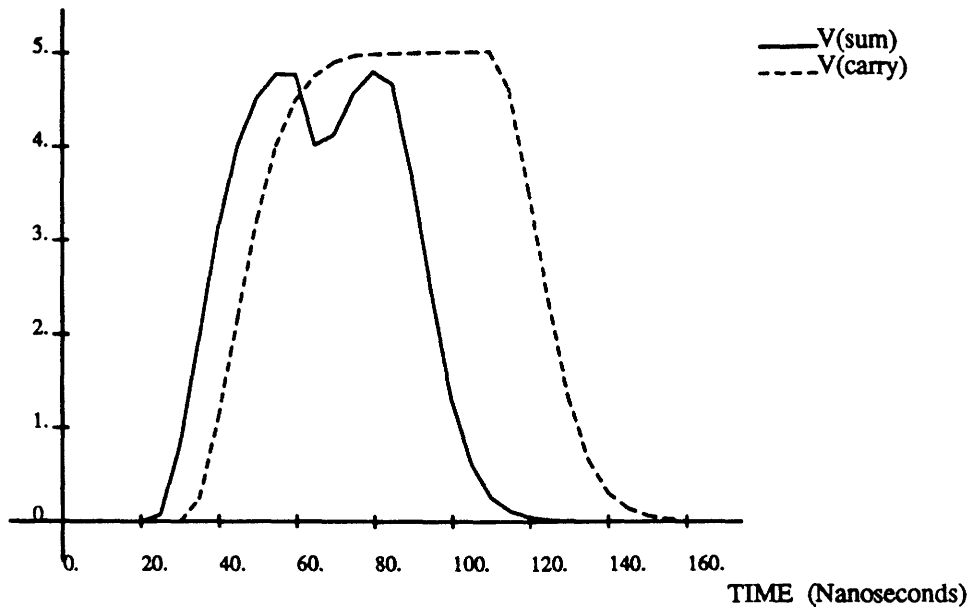
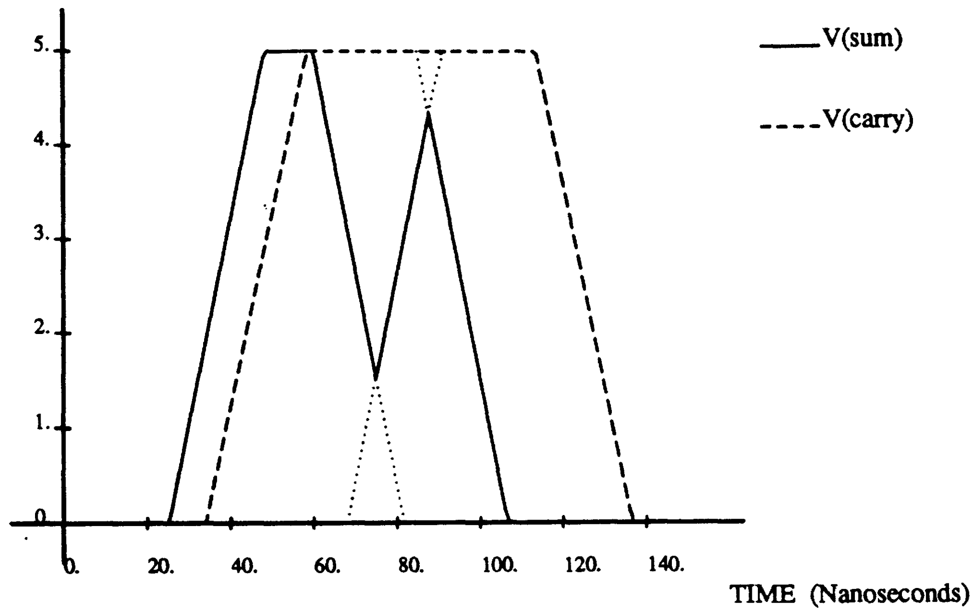


Figure 4-7: Input signals to the adder cell.



a)



b)

Figure 4-8: Output waveforms for the circuit of Figure 4-6.

a) Waveforms from a SPICE simulation of the circuit.
b) Waveforms from the experimental simulation of the circuit model.

Conducting transmission gates

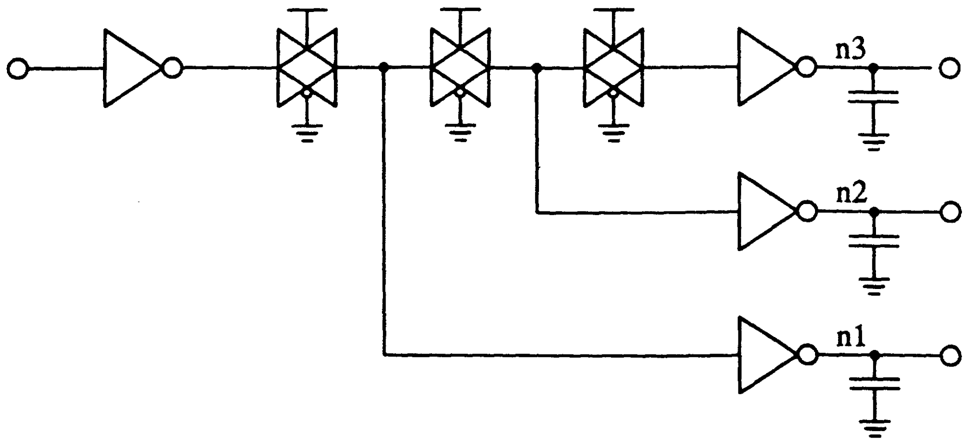


Figure 4-9: Circuit testing conducting transmission gate model.

Conducting Transmission Gate Ckt				
Node	Model Delay (ns)	SPICE Delay (ns)	ΔTd (ns)	% Td
n1	37.15	38.45	-1.3	-3.38
n2	43.90	46.65	-2.75	-5.90
n3	46.02	49.81	-3.79	-7.61

Table 4-6: Conducting transmission gate test

Load capacitance on output nodes is .2pF. Input waveform is falling with a transition time of 6ns.

The conducting transmission gate model is tested using the circuit in Figure 4-9. All transmission gates have 3 micron long, 4 micron wide n-type transistors and 3 micron long, 8 micron wide p-type transistors. The waveforms into the three loading inverters will exhibit different degrees of multi-time-constant behavior. The closer to the driving inverter the node is, the more multi-time-constant behavior it will experience.

The results in Table 4-6 show that percent errors are a little higher than for the logic gate cases. This is due to the non-linear properties of transmission gate resistance as well as the more complex waveform behavior.

The execution time for the experimental simulator is a little higher for the given number of instances than for a circuit for the same number of instances of logic gate macromodels. This is because additional calculation is needed to transform the transmission gates and driving circuits to RC circuits. Also, the delay calculation is more complicated. Nevertheless, an improvement of 146 is still seen when compared to the SPICE simulation.

Switching transmission gates

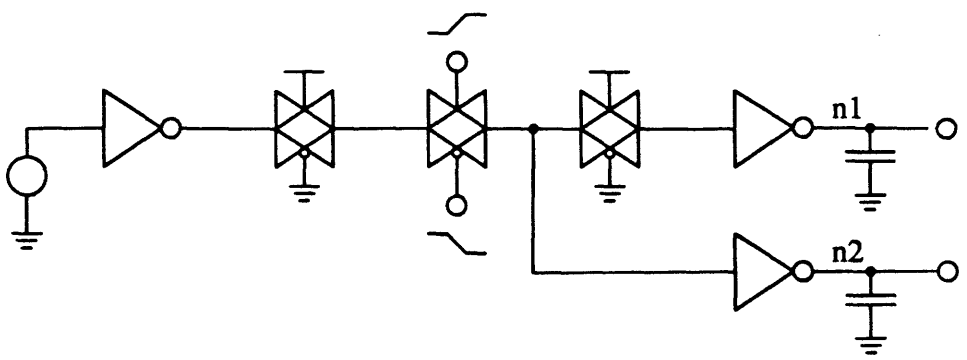


Figure 4-10: Circuit testing switching transmission gate model.

Switching Transmission Gate Circuit				
Node	Model Delay (ns)	SPICE Delay (ns)	ΔTd (ns)	% Td
n1	21.38	23.38	-2.0	-8.55
n2	24.37	26.48	-2.1	-7.97

Table 4-7: Switching transmission gate test

The circuit testing the switching transmission gate mode consists of a switching transmission gate, a driving circuit of an inverter and conducting transmission gate, and a loading circuit of a conducting transmission gate. Load capacitance on the output inverters are 0.2pF. The n-transistor gate voltage of the switching transmission gate rises at delay time 2ns with a transition time of 6ns. The p-transistor gate voltage falls at delay time 3.2ns with a transition time of .6ns. The driving circuit is initially at ground while the loading circuit is initially at VDD. Thus, voltages on nodes *n1* and *n2* will fall from VDD to ground when the transmission gate starts conducting. The methods described in Section 2.4.3.2 are used to model the effects of the transmission gate transistors turning on at different times. Again, errors are higher than those of logic gate circuits due to waveform modeling and nonlinear transistors.

The execution times are very similar to those for the conducting transmission gate circuits. The same reasons apply as for that case.

Shift register

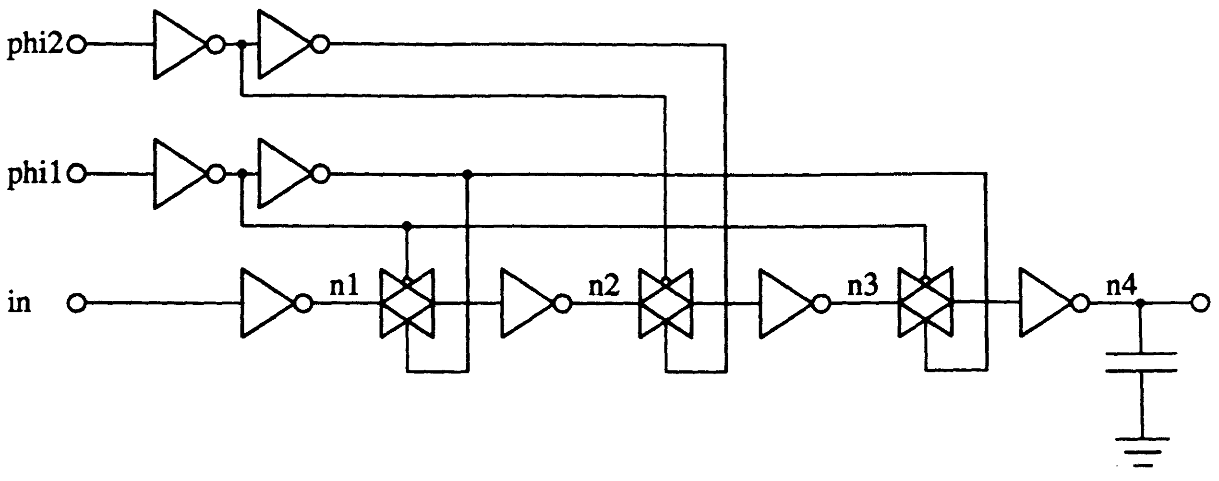


Figure 4-11: Shift register circuit.

Shift Register Results				
Node	Model Delay (ns)	SPICE Delay (ns)	ΔTd (ns)	% Td
n1	2.925	3.176	-.25	-7.9
n2	21.85	23.34	-1.49	-6.38
n3	66.42	64.58	1.84	2.85
n4	102.4	103.1	-.7	-.68

Table 4-8: Shift register test results

A dynamic shift register is shown in Figure 4-11 and the results in Table 4-8. The input to the shift register falls at delay time 0.2ns with a transition time of 0.6ns. The initial and final values of the nodes are shown in Table 4-9. The input clock waveforms are shown in Figure 4-12.

In this circuit we have switching transmission gates with skewed clock inputs plus logic gate signal propagation. All errors are below 7.5%, however the absolute error is only -.25ns for the highest percentage error case. The execution time gain is 308.

Shift Register Node Values		
Node	Initial State	Final State
n1	0	1
n2	1	0
n3	0	1
n4	1	0

Table 4-9: Shift register node values

The final state exists after the second cycle of *phi1*.

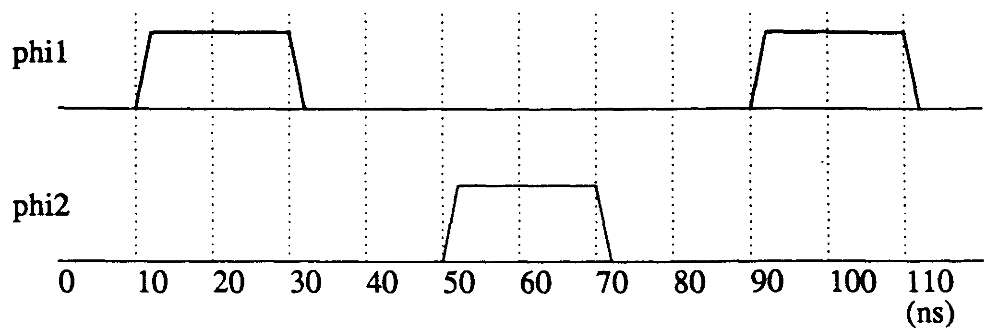


Figure 4-12: Clock waveforms for shift register.

Array Multiplier

Array multipliers are regular circuits that lend themselves well to the type of modeling and simulation developed in this work. An $N \times N$ array multiplier has N^2 instances of the same type of cell. Figure 4-13 shows the connections of these cells to form a 4×4 array multiplier. The circuit represented in an array multiplier cell is shown in Figure 4-14. One bit from each operand is input to the AND gate (or, equivalently, a NAND gate in series with an inverter). The output of the AND gate is connected to one of the adder inputs. The other adder inputs are *carry-in* and *sum-in*. In an array multiplier the carry-in is from the previous column, same row and the sum-in is from the previous row and subsequent column. Edge conditions can be seen in the array multiplier circuit in Figure 4-13.

The specific adder circuit used for array multiplier tests is shown in Figure 4-15. The adder was macromodeled as one cell. The macromodel contains many different sets of parameters for different input conditions. However, some of the input conditions were approximated as equivalent in the timing sense, eliminating the derivation of some parameters. Also, the output delay times and transition times were all linear with respect to input transition time (for the range of input transition times considered). This drastically reduced the number of simulations needed for each set of parameters.

The number of variables required for a SPICE simulation was so large that SPICE could not analyze even a 2×2 array multiplier. So, results from only a single array multiplier cell will be compared with SPICE results. The array multiplier cell consists of three modules, an adder model, a NAND gate model, and an inverter model. The results for the input conditions of Table 4-11 are shown in Table 4-10. All output loads are 0.2pF. All input delay times are 0.2ns and transition times are 6ns.

The delay errors are larger than previously encountered. This may be a result of assuming linear functions for output delay time and transition time with respect to input transition time. Also, fewer input transition time and load capacitance points were used to develop this macromodel than in previous tests. More SPICE simulations may be needed to provide enough points for more accurate delay time and transition time functions.

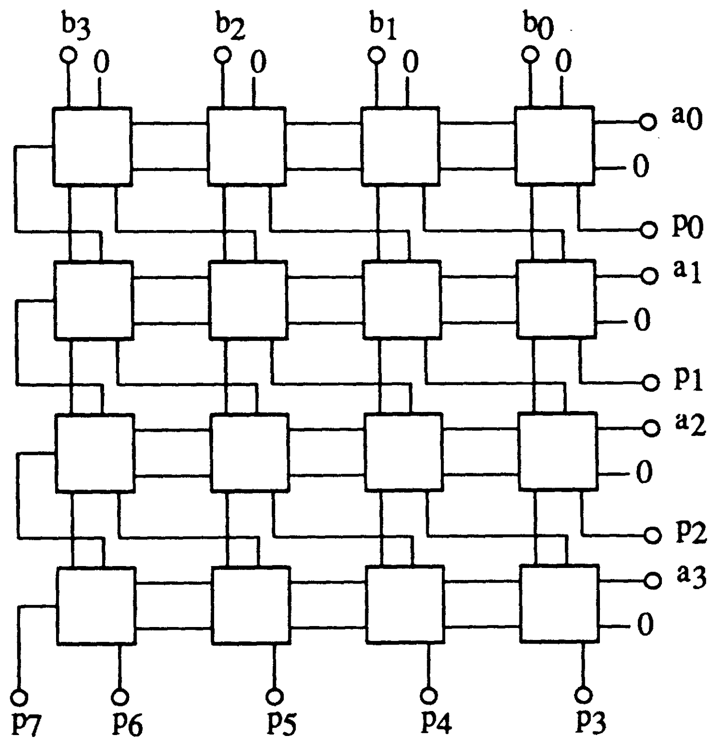


Figure 4-13: 4x4 Array Multiplier

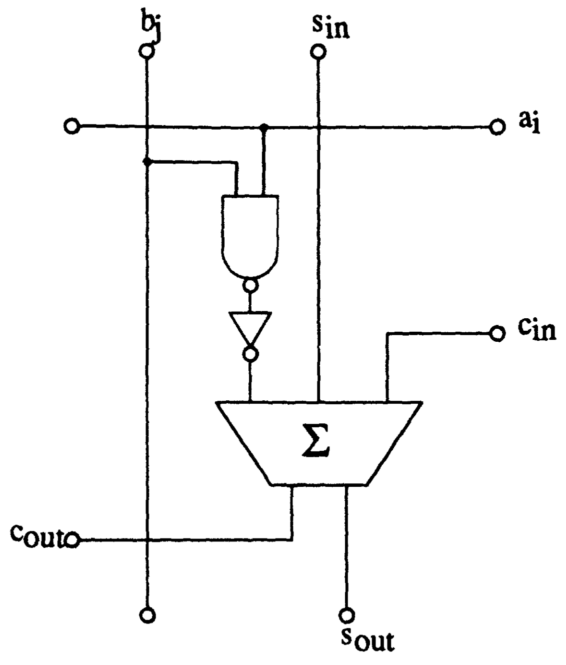


Figure 4-14: Array multiplier cell

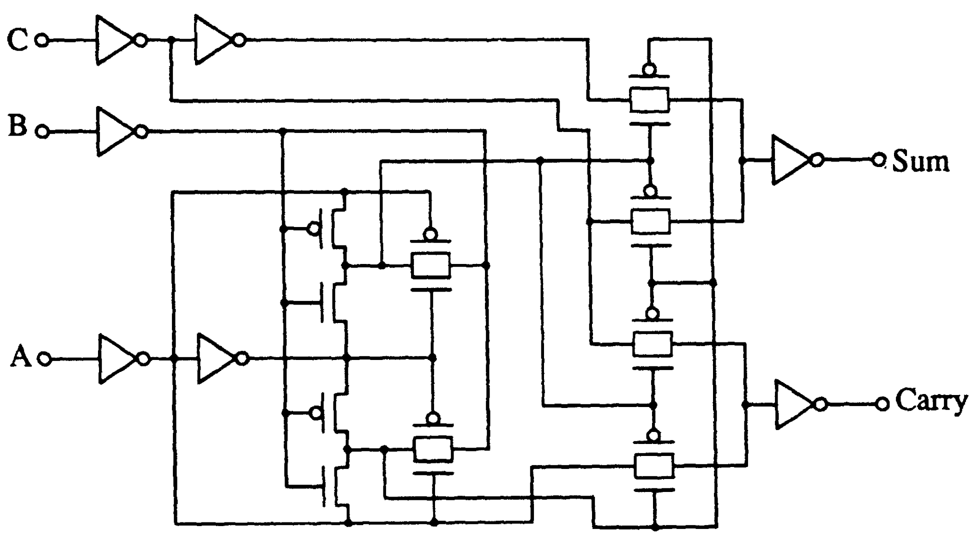


Figure 4-15: Three-bit adder circuit

Multiplier Cell Results				
Case	Model Delay (ns)	SPICE Delay (ns)	ΔTd (ns)	% Td
Case 1				
sum	69.60	76.37	-6.77	-8.86
Case 2				
sum	41.64	48.62	-6.98	-14.3
carry	49.90	53.52	-3.62	-6.76
Case 3				
sum	26.19	32.06	-5.87	-18.3

Table 4-10: Results for one array multiplier cell.

Multiplier Cell Inputs				
Case	A	B	Sum in	Carry in
Case 1	Rising	High	Low	Low
Case 2	High	High	Rising	Low
Case 3	High	High	High	Rising

Table 4-11: Multiplier cell inputs

Even though comparison with SPICE was not possible, results for a 2 × 2, 4 × 4, and an 8 × 8 array multiplier are shown in Table 4-12, along with execution times. This table illustrates how the execution time increases with the number of cells. Operands were chosen in order to increase the number of transitions during the simulation as the size of the array multiplier grows. The CPU time needed to simulate the multipliers grows with order N^2. Another way of stating this is that the CPU time grows roughly linearly with the number of instances in the circuit, since the instances increase with order N^2.

Array Multiplier Results				
Size	A	B	Td (ns)	CPU_{TEST} (sec)
2 by 2	3	2	120.1	.533
4 by 4	14	7	197.6	1.626
8 by 8	148	185	598.9	5.788

Table 4-12: Results for array multipliers

Polysilicon Interconnect Line

Figure 4-16 shows an RC line model for polysilicon interconnect. The line is 5 microns wide and 5mm long. Average capacitance and resistance values for polysilicon are assumed to be $C_p = 0.5 \times ()10^{-4}$ pF/μm^2 and $R_p = 35$ Ω/sq. Using ten RC sections, one arrives at the model in the figure. Delay was measured and compared at the fifth section and the tenth section. Results are shown in Table 4-13. The errors are higher than the previous results since the two time constant waveform model less discussed in Section 2.4.1 accurately models circuits with many time constants, as in a long RC line.

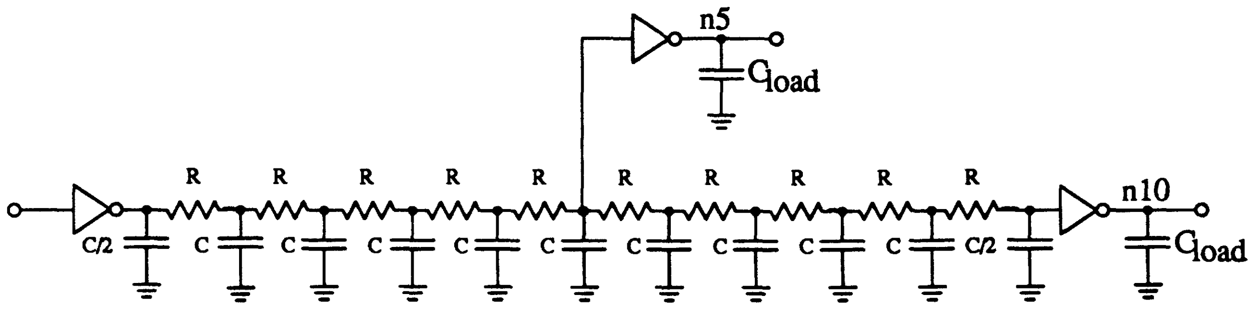


Figure 4-16: RC model for poly interconnect line

Ten RC sections are used to model a 5 micron wide, 5mm long polysilicon line. R = 350Ω and C = .125pF.

Poly Interconnect Line Results				
Node	Model Delay (ns)	SPICE Delay (ns)	ΔTd (ns)	% Td
n5	37.99	42.53	-4.54	-10.67
n10	38.66	43.26	-4.60	-10.63

Table 4-13: Results for RC model of poly interconnect.

Execution Times of Tests			
Circuit	CPU_{Test} (sec)	CPU_{SPICE} (sec)	CPU_{SPICE} / CPU_{Test}
Inverter			
— Case 1	.288	7.58	26.32
— Case 2	.292	10.53	36.06
— Case 3	.279	12.46	44.66
— Case 4	.277	12.22	44.16
— Case 5	.284	7.94	27.96
Inverter String	.361	63.99	177.3
Domino Circuit			
— Case 1	.349	44.61	128.0
— Case 2	.292	37.55	128.0
— Case 3	.284	32.24	113.0
Domino String	.377	202.95	538.0
Adder Circuit	.813	731.9	900.0
Conducting transmission gate ckt.	.465	67.97	146.0
Switching transmission gate ckt.	.427	59.07	138.0
Shift Register	.761	234.2	307.7
Array Multiplier Cell			
— Case 1	.372	216.0	580.6
— Case 2	.634	247.8	390.9
— Case 3	.570	191.7	336.3

Table 4-14: Execution times of test simulations

CHAPTER FIVE

Conclusions

A method has been presented for quick and accurate delay calculation of regular structure or standard-cell based CMOS VLSI circuits. Current simulators do not meet both speed and accuracy requirements needed for large circuits while modeling both logic gate and transmission gate structures. The techniques presented in this work perform calculations over two orders of magnitude faster than SPICE while keeping typical logic gate accuracies under 5% and typical transmission gate or RC circuit accuracies under 10%.

The modeling approach is empirical, but insight into model behavior was gained by considering physical properties of the circuits. For logic gates, output transition time and delay time is calculated as a function of load capacitance and input transition time. For transmission gates, resistance for the conducting case and switching case is modeled as a function of input waveform and load capacitance. The effects of transmission gate inputs having different delay times and transition times is also considered.

Methods for delay calculation of logic gate circuits and RC circuits were developed. Waveform models were derived for both types of delay calculation. An interface was found to link the two methods of delay calculation and waveform models.

Representations of sub-circuit modules and instances were presented. These representations were implemented in an experimental event driven simulator. Considerations in event driven simulation were discussed. Several tests were done to illustrate accuracy and computation speed compared to SPICE.

5.1 Future Work

Although test results indicate that the efforts in this work have been largely successful, there are still many issues that could be addressed and questions that need to be answered.

To further the modeling work done thus far, a method could be derived to calculate the effective input resistance of a cell. This input resistance would be a function of input waveform, as is the output resistance of a cell. Thus, cell inputs could be modeled correctly when that input is a transmission gate. In this thesis, only cell input capacitance is used as input impedance information. Of course, if the entire cell is a transmission gate, or series of transmission gates, then this case is modeled as an RC circuit.

More work on the effects of scaling transistors on macromodel parameters could be done. Ideally, macromodel parameters would be a function of such variables as transistor length and width, parasitic capacitance, and variation in cell layout. Also, adapting macromodel parameters for use with different technologies would be valuable.

A more general implementation of the logic evaluation of cells could be done. Logic evaluation is needed for node initialization as well as delay calculation. At present, the logic behavior is implemented in program code individually for each module. A more general and efficient way of describing the transistor structure of the module could be provided. This may be possible by providing a switch-level description of the cell, along with cell delay calculation parameters, as user input for a new library addition. General switch-level simulation procedures could be used for logic evaluation on all modules. This could potentially reduce the amount of program code needed for the delay representation of each module.

Finally, a useful technique would be combining existing macromodels to create a new macromodel. This would add a new dimension to the work, creating a hierarchical simulator using macromodels. This would probably be initially feasible for smaller sub-circuits, such as inverters and transmission gates. As cells with larger fan-in and fan-out would be composed, the number of possible delay paths would grow larger. Some case analysis may then be needed.

5.2 Final Thoughts

As developments in VLSI technologies and design remain a dynamic field, so must the development of computer-aided VLSI design tools. Only through continuing research in VLSI CAD can full use be made of technological advances in fabrication techniques, device sizes, alternate technologies and design methods. It is hoped that the ideas presented in this work contribute to this cause.

APPENDIX A

Derivation of Driving Resistance and Delay Offset for a Driving Gate

The goal of this section is to find a resistance and delay offset for an exponential waveform, given the final value of the exponential, V_f, and two sets of points on the waveform, (t_1,V_1) and (t_2,V_2).

Assume a rising exponential waveform:

$$V(t) = V_f\left(1 - e^{-(t - t_0)/\tau}\right). \tag{A.1}$$

V_f is the final value and τ is the time constant of the exponential waveform. t_0 is the time at which the exponential starts rising. The independent variable, t, may be solved for:

$$t = \tau \ln\left[\frac{V_f}{V_f - V(t)}\right] + t_0. \tag{A.2}$$

The time-voltage pairs are then substituted in equation (A.2). The resulting equations are:

$$t_1 = \tau\left[\frac{V_f}{V_f - V_1}\right] + t_0 \tag{A.3}$$

and

$$t_2 = \tau\left[\frac{V_f}{V_f - V_2}\right] + t_0. \tag{A.4}$$

Subtracting Equation (A.3) from Equation (A.4) and solving for τ yields:

$$\tau = \frac{t_2 - t_1}{\left\{ \ln\left[\frac{V_f}{V_f - V_2}\right] - \ln\left[\frac{V_f}{V_f - V_1}\right] \right\}}. \tag{A.5}$$

The resistance, R, of an exponential with time constant τ, assuming a capacitance of C, is then:

$$R = \frac{\tau}{C} \tag{A.6}$$

where τ is given in Equation (A.5).

The delay offset, t_0, of the exponential may be solved by using the results of Equations (A.3) and (A.5) to yield:

$$t_0 = t_1 - \tau \ln \left[\frac{V_f}{V_f - V_1}\right]. \qquad (A.7)$$

The results for a falling waveform may be found by going through the same steps. A falling exponential is described by:

$$V(t) = V_i e^{-(t - t_0)/\tau}. \qquad (A.8)$$

Where V_i is the initial voltage, t_0 is the delay offset, and τ is the time constant of the waveform. Again, given two sets of time-voltage points on the waveform, (t_1,V_1) and (t_2,V_2), and the procedure described above, The time constant is:

$$\tau = \frac{t_2 - t_1}{\left\{ \ln \left[\frac{V_i}{V_2}\right] - \left[\frac{V_i}{V_1}\right] \right\}}. \qquad (A.9)$$

Resistance may be found using Equation (A.6). The delay offset is then:

$$t_0 = t_1 - \tau \ln \left[\frac{V_i}{V_1}\right]. \qquad (A.10)$$

References

1. L.W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits", Memo ERL-M520, University of California, Berkeley, May 1975.

2. W.T. Weeks, A.J. Jimenez, G.W. Mahoney, D. Mehta, H. Qassemzadeh, and T.R. Scott, "Algorithms for ASTAP: A Network Analysis Program", *IEEE Transactions on Circuit Theory,*Vol. 11, 1973, pp. 628-634.

3. Bernard Conrad Cole, "Stretching the Limits of ASIC Software", *Electronics,*23 June 1986, pp. 34-38.

4. Randal E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems", *IEEE Transactions on Computers,*Vol. C-33, February 1984, pp. 160-177.

5. C.J. Terman, *Simulation Tools for Digital LSI Design,* PhD dissertation, Massachusetts Institute of Technology, October 1983.

6. J.K. Ousterhout, "A Switch Level Timing Verifier for Digital MOS VLSI", *IEEE Transactions on Computer Aided Design,*Vol. CAD-4, July 1985, pp. 336-349.

7. J.K. Ousterhout, "Switch-Level Delay Models for Digital MOS VLSI", *21st Design Automation Conference,* IEEE, 1984, pp. 542-548.

8. V.B. Rao, T.N. Trick, and I.N. Hajj, "A Table-Driven Delay-Operator Approach to Timing Simulation of MOS VLSI Circuits", *International Conference on Computer Design: VLSI in Computers,* IEEE, Port Chester, NY, October 1983, pp. 445-448.

9. Rolf Sundblad and Christer Svensson, "Fully Dynamic Switch-Level Simulation of MOS Circuits", *IEEE Transactions on Computer-Aided Design,*Vol. CAD-6, March 1987, pp. 282-289.

10. Genhong Ruan and J. Vlach, "Current Limited MOS Model in Logic and Timing Simulation", *International Symposium on Circuits and Systems,* IEEE, San Jose, CA, May 1986, pp. 747-750.

11. Young H. Kim, J.E. Kleckner, R.A. Saleh, and A.R. Newton, "Electrical-Logic Simulation", *International Conference on Computer-Aided Design,* IEEE, Santa Clara, CA, November 1984, pp. 7-9.

12. J. Rubinstein, P. Penfield, Jr., and M.A. Horowitz, "Signal Delay in RC Networks", *IEEE Transactions on Computer-Aided Design,*Vol. CAD-2, July 1983, pp. 202-211.

13. Paul Penfield, Jr. and Jorge Rubinstein, "Signal Delay in RC Tree Networks", *Caltech Conference on VLSI,* Pasadena, CA, January 1981, pp. 269-283.

14. M. Horowitz, "Timing Models for MOS Pass Networks", *Proc. of the Int. Symp. on Circuits and Systems,* 1983, pp. 198-201.

15. M. Horowitz, *Timing Models for MOS Circuits,* PhD dissertation, Stanford University, December 1983.

16. T. Lin and C.A. Mead, "Signal Delay in General RC Networks", *IEEE Transactions on Conputer-Aided Design,*Vol. CAD-3, October 1984, pp. 331-349.

17. T. Lin, *A Hierarchical Timing Simulation Model for Digital Integrated Circuits and Systems,* PhD dissertation, California Institute of Technology, August 1984.

18. T. Lin and C.A. Mead, "Signal Delay in General RC Networks with Application to Timing Simulation of Digital Integrated Circuits", Computer Science Department 5089:TR:83, California Institute of Technology, 1983.

19. Basant R. Chawla, Hermann K. Gummel, and Paul Kozak, "MOTIS—An MOS Timing Simulator", *IEEE Transactions on Circuits and Systems,*Vol. CAS-22, December 1975, pp. 901-910.

20. S.P. Fan, M.Y. Hsueh, A.R. Newton, and D.O. Pederson, "MOTIS-C: A New Circuit Simulator for MOS LSI Circuits", *International Symposium on Circuits and Systems,* IEEE, Phoenix, AZ, April 1977, pp. 700-703.

21. Chin Fu Chen and Prasad Subramaniam, "The Second Generation MOTIS Timing Simulator— An Efficient and Accurate Approach for General MOS Circuits", *International Symposium on Circuits and*

Systems, IEEE, Montreal, Canada, May 1984, pp. 538-542.

22. David Tsao and Chin-Fu Chen, "A Fast-Timing Simulator for Digital MOS Circuits", *IEEE Transactions on Computer-Aided Design*,Vol. CAD-5, October 1986, pp. 536-540.

23. Amrish Patel, Wally Bridgewater, and Rao Pokala, "Newton: Logic Simulation with Circuit Simulation Accuracy for ASIC Design", *IEEE 1986 Custom Integrated Circuits Conference*, IEEE, Portland, Oregon, June 1986, pp. 456-459.

24. Sani R. Nassif and Stephen W. Director, "WASIM: A Waveform Based Simulator for VLSIC's", *IEEE International Conference on Computer-Aided Design*, IEEE, Santa Clara, CA, November 1985, pp. 29-31.

25. M.D. Matson, *Macromodeling and Optimization of Digital MOS VLSI Circuits*, PhD dissertation, Massachusetts Institute of Technology, February 1985.

26. Mark D. Matson and Lance A. Glasser, "Macromodelling and Optimization of Digital MOS Circuits", *IEEE Transactions on Computer-Aided Design*,Vol. CAD-5, October 1986, pp. 659-678.

27. Rathin Putatunda, "AUTODELAY: A Second Generation Automatic Delay Calculation Program for LSI/VLSI Chips", *International Conference on Computer-Aided Design*, IEEE, Santa Clara, CA, November 1984, pp. 188-189.

28. Rathin Putatunda, "Automatic Calculation of Delay in Custom Generated LSI/VLSI Chips", *International Conference on Circuits and Computers*, IEEE, NY, NY, September 1982, pp. 193-196.

29. Rathin Putatunda, "AUTODELAY: A Program for Automatic Calculation of Delay in LSI/VLSI Chips", *19th Design Automation Conference*, IEEE, Las Vegas, Nevada, June 1982, pp. 616-621.

30. C.J.R. Fyson and K.G. Nichols, "Simple Characterisation of Logic Gates for Use in Macrosimulation", *International Conference on Circuits and Computers*, IEEE, NY, NY, September 1982, pp. 189-192.

31. J. White and A.L. Sangiovanni-Vincentelli, "RELAX2: A Modified Waveform Relaxation Approach to the Simulation of MOS Digital Circuits", *Proc. of the Int. Symp. on Circuits and Systems*, IEEE, 1982, pp. 756-759.

32. E. Lelarasmee, A.E. Ruehli, and A.L. Sangiovanni-Vincentelli, "The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*,Vol. CAD-1, July 1982, pp. 131-144.

33. Karem A. Sakallah and Stephen W. Director, "SAMSON: An Event Driven VLSI Circuit Simulator", *Custom Integrated Circuits Conference*, IEEE, Rochester, NY, May 1984, pp. 226-231.

34. John L. Wyatt. Private communication, November 1986

35. A.E. Ruehli, A.L. Sangiovanni-Vincentelli, and G. Rabbat, "Time Analysis of Large-Scale Circuits Containing One-Way Macromodels", *IEEE Transactions on Circuits and Systems*,Vol. CAS-29, March 1982, pp. 185-189.

36. Steven P. Mccormick, "Cross-Talk Noise Modelling for VLSI Interconnections", Ph.D. Thesis Proposal, Massachusetts Institute of Technology, 1986

37. Lance A. Glasser and Daniel W. Dobberpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley Publishing Company, Reading, MA, 1985.

38. W. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers", *Journal of Applied Physics*,Vol. 19, January"" 1948, pp. 55-63.

39. C.A. Zukowski, *Bounding Enhancements for VLSI Circuit Simulation*, PhD dissertation, Massachusetts Institute of Technology, June 1985.

40. T. Tokuda, K. Okazaki, K. Sakashita, I. Ohkura and T. Enomoto, "Delay-Time Modeling for ED MOS Logic LSI", *IEEE Transactions on Computer-Aided Design*,Vol. CAD-2, July 1983, pp. 129-134.

41. A.E. Ruehli, R.B. Rabbat, and H.Y. Hsieh, "Macromodelling - An Approach for Analysing Large Scale Circuits", *Computer-Aided Design*,Vol. 10, March 1978, pp. 121-130.

42. A.E. Ruehli and G.S. Ditlow, "Circuit Analysis, Logic Simulation, and Design Verification for VLSI" *Proceedings of the IEEE*,Vol. 71, January 1983, pp. 34-48.

43. J.L. Wyatt, "Signal Propagation Delay in *RC* Models for Interconnect", to be published

44. Bryan Ackland and Neil Weste, "Functional Verification in an Interactive Symbolic IC Design Environment", *Caltech Conference on VLSI*, Pasadena, CA, January 1981, pp. 285-298.

45. Bryan D. Ackland, Sudhir R. Ahuja, Teri L. Lindstrom, and Deborah J. Romero, "CEMU - A Concurrent Timing Simulator", *IEEE International Conference on Computer-Aided Design*, IEEE, Santa Clara, CA, November 1985, pp. 122-124.

46. Jonathan Allen, *Introduction to VLSI Design*, Massachusetts Institute of Technology, Cambridge, MA, 1985.