September 1989

UILU-ENG-89-2229
CSG-106

COORDINATED SCIENCE LABORATORY
College of Engineering

AN ACCURATE TIMING MODEL FOR FAULT SIMULATION IN MOS CIRCUITS

Sungho Kim

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-89-2229 (CSG-106)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (*If applicable*) N/A	7a. NAME OF MONITORING ORGANIZATION Semiconductor Research Corporation Joint Services Electronics Program
6c. ADDRESS (*City, State, and ZIP Code*) 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS (*City, State, and ZIP Code*) Semiconductor Research Corp. P.O. BOx 12053 (OVER) Research Triangle Park, NC 27709
8a. NAME OF FUNDING / SPONSORING ORGANIZATION same as 7a 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER 1-5-80381 N00014-84-C-0149
8c. ADDRESS (*City, State, and ZIP Code*) same as 7b	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)

An Accurate Timing Model For Fault Simulation In MOS Circuits

12. PERSONAL AUTHOR(S) Kim, Sungho

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Technical	FROM _____ TO _____	September 1989	42

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	fault simulation, MOS circuits, delay fault testing, timing model, SPICE, RSIM

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

With the increasing need for manufacturers to maintain quality requirements for high performance and high density MOS VLSI integrated circuits and the multitude of physical failures and defects that can occur in such circuits, delay fault testing is gaining importance. An extremely important component of such a delay test generation environment is an accurate yet fast delay fault simulator. The goal of fault simulation is to generate the responses of digital circuits under both fault and fault-free conditions, without prohibitive cost. Most simulators available today either sacrifice accuracy or are very time-consuming. Therefore, choosing the right simulation model is the key in fault simulation.

MOS circuit models that were previously developed for true value simulation, after being modified to accommodate fault models, have been used by fault simulators, such as FMOSSIM. These fault simulators are incapable of detecting timing errors and even some logic errors, both of which occur in actual failures in the MOS circuits. Others

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (*Include Area Code*) / 22c. OFFICE SYMBOL

7b. 800 N. Quincy st.
Arlington, VA 22217

19. have shown that some faults only affect the circuit timing and the delay may invalidate test sets. Therefore, fault simulators must consider the circuit delay in order to evaluate test sets accurately. FAUST is one such fault simulator which very accurately detects timing as well as logical errors. It was based on using a combination of table lookup and numerical integration techniques to solve differential equaltions. With today's large MOS circuits, it is desirable to propose faster timing models without sacrificing the accuracy too much, to complete the simulation.

In this thesis, we propose a timing model for fault simulation for large MOS circuits which effectively balances the need for speed and accuracy. Like RSIM, this simulation model represents a transistor with a linear resistor and uses a simple RC approximation for timing estimates, but we achieve greater accuracy through the use of five logic values and an improved timing model. Related timing simulators have been developed by other researches. In addition, our simulator can handle faults such as stuck faults, resistive shorts, and threshold voltage shifts. This is possible because our timing model can accurately predict the behavior of the circuit under these faults.

Due to its rapid and accurate evaluation of node state and delays, our timing model can be used to simulate large MOS circuits. In addition, it can be easily implemented as a concurrent fault simulator which w ould reduce total simulation time even further.

The remainder of this thesis is organized as follows. In Chapter 2, the proposed timing model and the fault model will be described. The implementation details of FACT, Fault Simulation with ACcurate Timing, will be discussed in Chapter 3, and in Chapter 4, results on various circuits will be shown and compared with those of SPICE and RSIM.

AN ACCURATE TIMING MODEL FOR FAULT SIMULATION IN MOS CIRCUITS

BY

SUNGHO KIM

B.S., Worcester Polytechnic Institute, 1986

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1989

Urbana, Illinois

ACKNOWLEDGMENT

I wish to express my sincere appreciation and gratitide to Professor Prithviraj Banerjee, my advisor, for his guidance, encouragement and many insights which lead to the result of this research. It has been a most exciting and rewarding period to work under his guidance.

Finally, I wish to thank my parents for their everlasting love, understanding and encouragement. This thesis is dedicated to both of them.

This work was supported in part by the Semiconductor Research Corporation and in part by the Joint Services Electronics Program.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION	1
2. TIMING MODEL FOR FAULT SIMULATION	3
2.1. FACT Model of Transistor	3
2.2. FACT's Node Model	8
2.3. FACT's Final Value Computation	10
2.4. FACT's Transition Time Computation	13
2.5. Slope and Load Effect	21
2.6. Fault Model of FACT	22
3. IMPLEMENTATION	25
3.1. Scheduling	25
3.2. Processing Scheduled Events and Stage Evaluation	26
3.3. Event List Management	26
4. SIMULATION RESULTS	28
5. CONCLUSIONS	37
LIST OF REFERENCES	38

CHAPTER 1.

INTRODUCTION

With the increasing need for manufacturers to maintain quality requirements for high performance and high density MOS VLSI integrated circuits and the multitude of physical failures and defects that can occur in such circuits, delay fault testing is gaining importance [1,2,3,4,5]. An extremely important component of such a delay test generation environment is an accurate yet fast delay fault simulator. The goal of fault simulation is to generate the responses of digital circuits under both fault and fault-free conditions, without prohibitive cost. Most simulators available today either sacrifice accuracy or are very time-consuming. Therefore, choosing the right simulation model is the key in fault simulation.

MOS circuit models that were previously developed for true value simulation, after being modified to accommodate fault models, have been used by fault simulators, such as FMOSSIM [6]. These fault simulators are incapable of detecting timing errors and even some logic errors, both of which occur in actual failures in the MOS circuits [7]. Others have shown that some faults only affect the circuit timing and the delay may invalidate test sets [8,9]. Therefore, fault simulators must consider the circuit delay in order to evaluate test sets accurately. FAUST [10] is one such fault simulator which very accurately detects timing as well as logical errors. It was based on using a combination of table lookup and numerical integration techniques to solve differential equaltions. With today's large MOS circuits, it is desirable to propose faster timing models without sacrificing the accuracy too much, to complete the simulation.

In this thesis, we propose a timing model for fault simulation for large MOS circuits which effectively balances the need for speed and accuracy. Like RSIM [11], this simulation model represents a transistor with a linear resistor and uses a simple RC approximation for timing estimates, but we achieve greater accuracy through the use of five logic values and an improved timing model. Related timing simulators have been developed by other researches [12,13,14]. In addition, our simulator can handle faults such as stuck faults, resistive shorts, and

threshold voltage shifts. This is possible because our timing model can accurately predict the behavior of the circuit under these faults.

Due to its rapid and accurate evaluation of node state and delays, our timing model can be used to simulate large MOS circuits. In addition, it can be easily implemented as a concurrent fault simulator which would reduce total simulation time even further.

The remainder of this thesis is organized as follows. In Chapter 2, the proposed timing model and the fault model will be described. The implementation details of FACT, Fault Simulation with ACcurate Timing, will be discussed in Chapter 3, and in Chapter 4, results on various circuits will be shown and compared with those of SPICE and RSIM.

CHAPTER 2.

TIMING MODEL FOR FAULT SIMULATION

A new model for timing simulation is proposed in this chapter. The first section describes the transistor model. Using this model, an MOS network is simulated as a resistor network where each node's value is determined by the resistance of its connections to various inputs. The second section describes the node model. This is followed by an explanation of the method for calculating the value of each node. The fourth section discusses the use of the model to predict the propagation of new input values through a network. The fifth section shows the use of an input slope and an output load capacitance for more accurate timing predictions, and the last section describes the fault model.

2.1. FACT Model of Transistor

In our delay fault simulator, Fault simulation with ACcurate Timing (FACT), a transistor is modeled by a linear resistor with a switch in series. To see how the model works, let's consider a simple inverter in Figure 2.1. We can think of the effective resistance of its transistor as

$$R_{eff} = \frac{v_{ds}}{i_{ds}}$$

The figure shows the actual effective resistance of both pullup and pulldown transistors as a function of the inverter's output voltage. We can see that the effective resistances of the transistors change as their terminal voltages vary. However, "average channel resistances" might be enough to characterize the transistor's behavior.

Another important characteristic of the transistor is that it behaves like a switch. With certain voltages applied at a transistor's terminal nodes, it makes no connection between its source and drain terminals - the transistor is "off." As the terminal voltages change, the transistor turns "on," conducting current between its source and drain terminals.

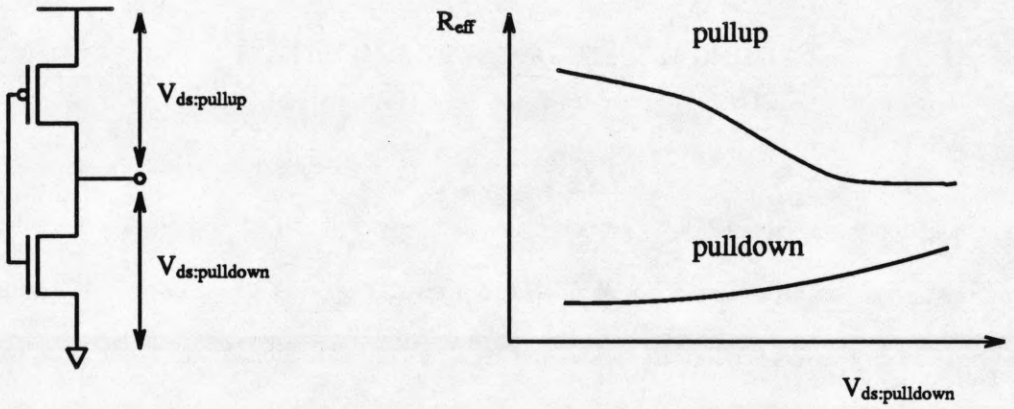
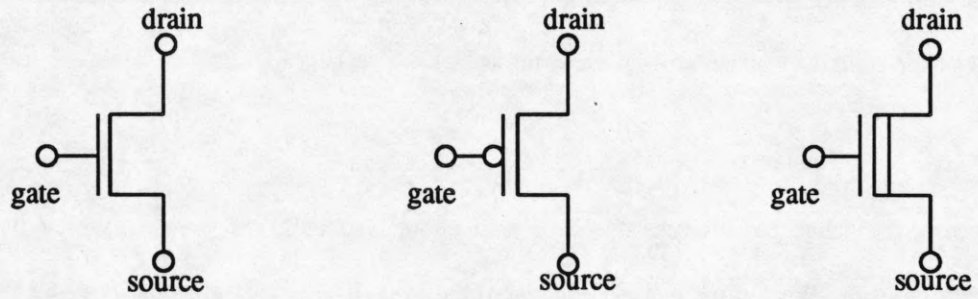


Figure 2.1. Effective resistances in an inverter

In Figure 2.2 are the three basic types of the transistor found in MOS circuits. The only difference between the n-channel and p-channel transistors is the gate voltage level which turns on the transistor. The depletion transistor is usually connected to VDD and used with its switch always on.



ON when gate ≥ 1V
OFF when gate < 1V

(a) n-channel switch

ON when gate ≤ 4V
OFF when gate > 4V

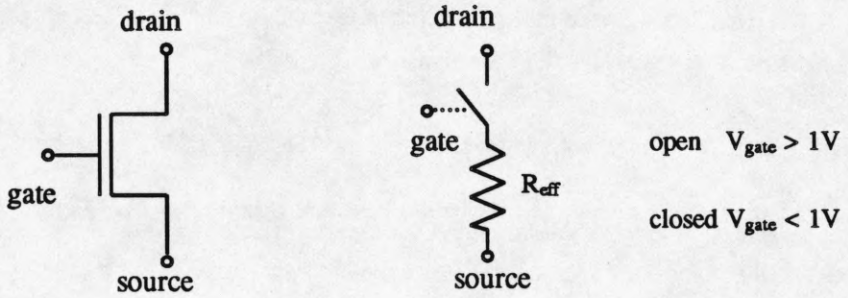
(b) p-channel switch

always ON

(c) depletion switch

Figure 2.2. Three types of MOS transistors

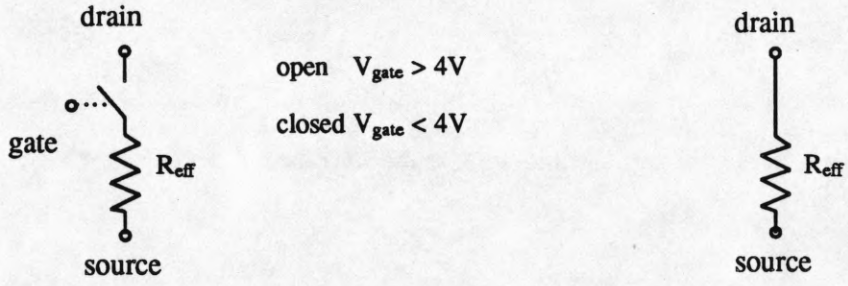
From these observations on the MOS transistor, one can construct the following transistor model that FACT uses:



(a) n-channel transistor

(b) FACT model

Figure 2.3. FACT model for n-channel transistor



(a) p-channel transistor model

(b) depletion transistor model

Figure 2.4. FACT models for p-channel and depletion transistor

Since R_{eff} is the only device information FACT uses about a transistor as seen in the above figures, choosing the value for R_{eff} is very important in our simplistic transistor model. In FACT, the effective resistance R_{eff} is determined separately for each transistor and depends on

width, length	Dimensions of the active transistor area.
type	Most MOS circuits contain more than one type of transistor.
context	Most MOS transistors are used in various contexts.
input slope	Slope with which the input changes its value.
output load	The load capacitance at the output that needs to be either charged or discharged.
gate voltage level	This determines how "on" the transistor is.

Among the above parameters, it is the gate voltage level that affects the value of R_{eff} the most. Therefore, R_{eff} is first determined by the transistor's gate voltage level. To see how this is done, consider Figure 2.5 which shows V-I characteristics of an MOS transistor with various voltages applied at the gate.

In the figure, as the gate voltage changes from one to five volts, the V-I curve of the transistor also varies. Because we use five logic levels to represent a node state (this will be explained in the following section), we have a V-I curve corresponding to each of five logic levels applied at the gate. Modeling a transistor by a linear resistor is equivalent to replacing each of the above V-I curves with a straight line, with its slope equal to the conductance of R_{eff} .

The values for R_{eff}, the inverse of the slope of each linearized V-I curve, must be chosen such that it leads to accurate predictions for both the final value and transition time. In FACT, this is done in a pre-simulation phase, in which a series of experiments are performed to measure the resistance of the transistor in various circuit con-

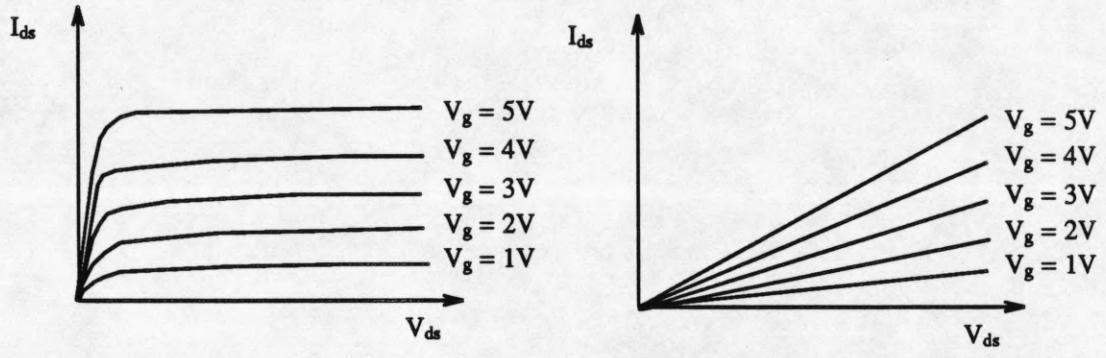


Figure 2.5. V-I characteristics of an MOS transistor

texts. Ideally, the experiments should be performed using actual circuits, but an accurate circuit simulator such as SPICE can be used to gather the needed measurements. At this point, we would like to point out that R_{eff}, actually, consists of three resistances:

R_{static}	when calculating the final voltage.
R_{dynlow}	when calculating the transition time for high-to-low transitions.
$R_{dynhigh}$	when calculating the transition time for low-to-high transitions.

Three resistances are used because one resistance value cannot give accurate predictions for both the final value and transition time.

Static resistances, that are used to estimate node voltages, are comparatively easy to choose. When all the nodes in a stage are connected to only one polarity of input, the values chosen for static resistances do not affect the voltage computation. When a circuit makes a connection to inputs of different polarities, the transistor resistances must be chosen to predict the output voltage. Since only the ratio of the pullup and pulldown devices is constrained, there is considerable freedom in choosing the actual resistance values.

Choosing the appropriate dynamic resistance is, however, not so simple. Dynamic resistances are used to predict the transition time. Therefore, each of the experiments consists of measuring the length of time required for the output to rise or fall from its starting voltage to the switching threshold. If the load capacitance is known, R_{dynlow} and $R_{dynhigh}$ can be calculated, essentially inverting the computation performed by FACT. As an example, consider the experiment with an n-channel pulldown in Figure 2.6. Initially, the gate is at zero volt and the capacitance at the output node is fully charged to 5 volts. By applying various voltages to the gate, we turn the transistor on and discharge the output capacitance. This is followed by observing the output waveform and approximating it with an exponential curve. We, then, measure the delay, τ , the time taken for the output voltage to drop to 36.7% of its initial value, the transition threshold, and obtain R_{dynlow} which is equal to $\frac{\tau}{C}$. The values currently used in FACT are given in Figure 2.7.

Once obtained in the pre-simulation phase, R_{eff} are used throughout the actual simulation phase. However, as seen in the previous example, the method for obtaining R_{eff} does not take the input slope, the load size, and the transistor size into account. Therefore, the effective resistance must go through a series of calibrations taking

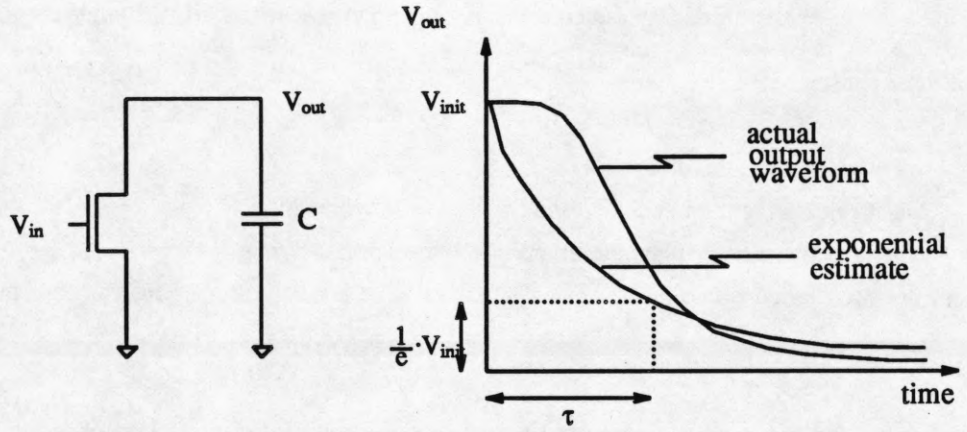


Figure 2.6. An experiment with an n-channel pulldown to measure R_{eff}

Transistor Type	Input change	Kohms/sq. (passing 1)	Kohms/sq. (passing 0)
n-mos	$0 \rightarrow 2V$	510	315
	$0 \rightarrow 3V$	130	80
	$0 \rightarrow 4V$	65	35
	$0 \rightarrow 5V$	35	21
p-mos	$5 \rightarrow 3V$	315	510
	$5 \rightarrow 2V$	80	130
	$5 \rightarrow 1V$	35	65
	$5 \rightarrow 0V$	21	35

Figure 2.7. $R_{dynamic}$ used by FACT

these parameters into account before it is ready to be used. This calibration phase will be discussed in detail in Section 2.4.

2.2. FACT's Node Model

To simulate a digital circuit, at least two logic states, logic-one and logic-zero, are necessary to represent the node voltage. Also, a third logic state, referred to as the "X" state, is often used to represent the unknown or the intermediate voltage levels. Representing the node by this binary or ternary logic values may be sufficient if the

circuit is designed so conservatively that intermediate voltage levels rarely occur. Unfortunately, in today's digital MOS circuits, intermediate voltage levels frequently occur. For example, let us consider the circuit in Figure 2.8. In the figure, the gate voltage at node A turns the transistor T1 on to charge up the capacitance at node B. However, the voltage at node B rises only up to 4 volts, instead of the desired voltage, 5 volts, due to the threshold voltage drop between the gate and the drain node of transistor T1. This weak logic-one value is, next, used to turn the next transistor T2 on. Due to the threshold voltage drop again, the voltage at node C is pulled only up to 3 volts, an intermediate voltage.

Intermediate voltages can also arise by a charge sharing which happens when two source free transistor networks become one by a source and a drain connection of a transistor. As soon as the connection is made, nodes in the network are allowed to share charge and all will reach the same intermediate voltage level.

Intermediate voltage levels also occur when a fault is injected into the circuit. Bryant of FMOSSIM [6] showed that many fault models spread intermediate voltages throughout the circuit. For an accurate fault simulation, therefore, intermediate voltages should be modeled accurately.

Modeling intermediate node voltages is also important because it directly affects the operation of the transistor; it is responsible for the different "on" states of a transistor. For example, if the gate of an NMOS transistor is driven by a full 5 volts, the transistor is fully "on" and behaves like a closed switch. As the gate voltage decreases, however, the transistor starts acting like a resistor, with its resistance increasing as the gate voltage

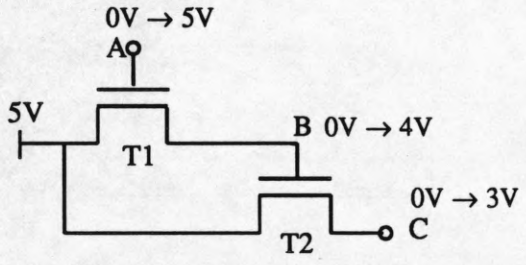


Figure 2.8. A circuit showing an occurrence of intermediate voltages

decreases. Finally, it becomes an open switch when the gate voltage drops to and below one volt.

In order to build an accurate fault simulator, we chose to quantize the node into five logic levels (See Figure 2.9). Now, we can not only describe the transistor's behavior more accurately but also model the intermediate voltages frequently occurring in fault simulation. A detailed study on the effect of physical failure of MOS circuits[15] has also shown that five logic levels are necessary to represent the intermediate voltages caused by a wide variety of fault models.

2.3. FACT's Final Value Computation

The final value of a node is obtained as follows. Using the transistor model described in the previous section, the original network is transformed into a network of resistors and capacitors as seen in Figure 2.10 and Figure 2.11. When a node evaluation takes place, the evaluation is performed not on the node in isolation, but on an entire set of nodes that are currently connected via source-to-drain connections of non-off transistors. We call such a group of nodes a stage (See Figure 2.12.). Stages do not extend through input, power or ground nodes.

In our node value computation, we evaluate the stages as trees. If the stage has any cyclical connections, it is nevertheless analyzed as a tree by arbitrarily ignoring any transistors which complete a cycle (See Figure 2.13.). This simplification can result in serious errors in estimation; fortunately, stages containing cycles are rare.

voltage range	logic value
0 — 1V	0
1 — 2V	0*
2 — 3V	I
3 — 4V	1*
4 — 5V	1

Figure 2.9. Multiple logic values used by FACT

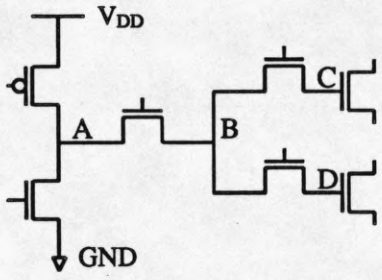


Figure 2.10. Typical MOS Signal Distribution Network

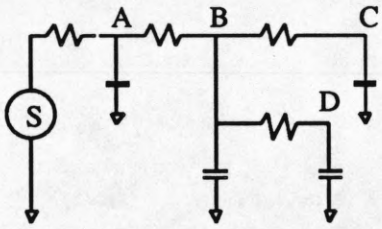


Figure 2.11. Linear circuit model for the network in Figure 2.10

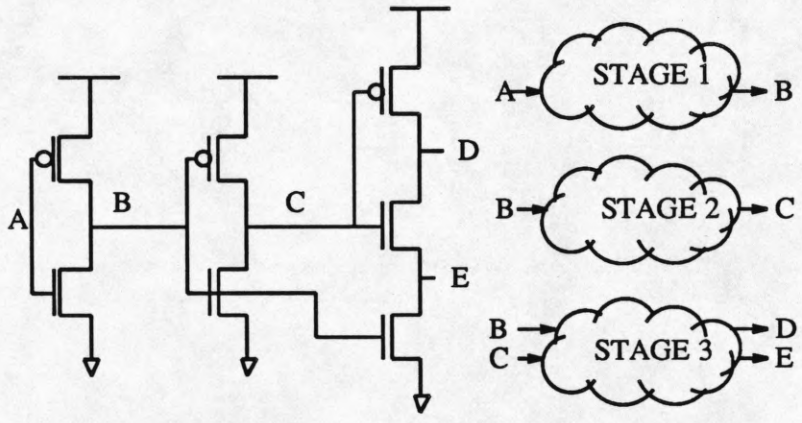


Figure 2.12. Simple circuit that has three stages

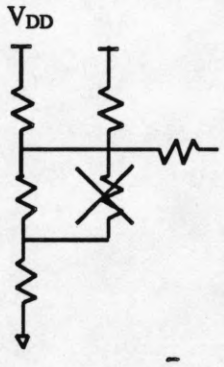


Figure 2.13. Tree approximation

Now, the node value computation is equivalent to solving a system of linear equations. Quite often, the stage has only one driving source, in which case the solution is trivial. But, multiple driving sources may also occur as seen in Figure 2.14. In such a case, FACT solves the linear system exactly, using a tree-based algorithm.

In the tree-based algorithm, given a particular node, the effect of inputs on the node is found by finding the exact Thevenin equivalent for the stage, with the given node as the output (See Figure 2.15.).

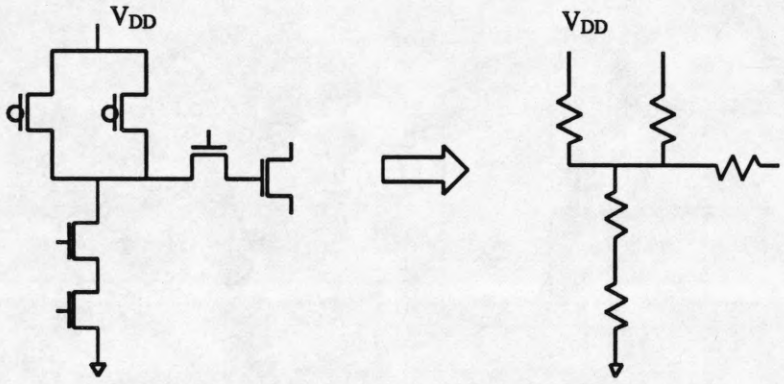


Figure 2.14. A multiple-source tree

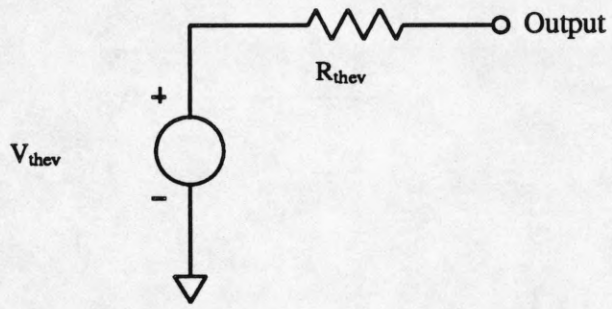


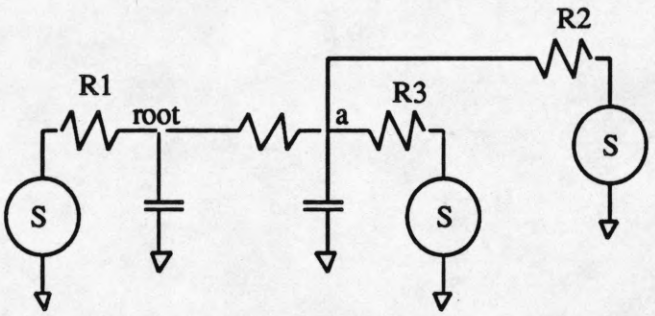
Figure 2.15. Equivalent circuit for a network node

Values for all the nodes in a stage are obtained in O(n) by passing the tree twice. For example, consider the RC tree with multiple sources in Figure 2.16(a). Since we are only interested in steady-state node values, we can ignore capacitances from the network. The algorithm, first, identifies a 'root' which is a node directly connected to a source through a resistor. In the first pass of the tree, V_{thev} and R_{thev} at the root, looking from the source side, are obtained by using two rules shown in Figure 2.17. By applying these rules, we can reduce the original network into the equivalent network shown in Figure 2.16(b). The exact voltage at the root node is, then, found by applying Rule 2. The rest of the node values are obtained in the second pass by walking back to the tree network.

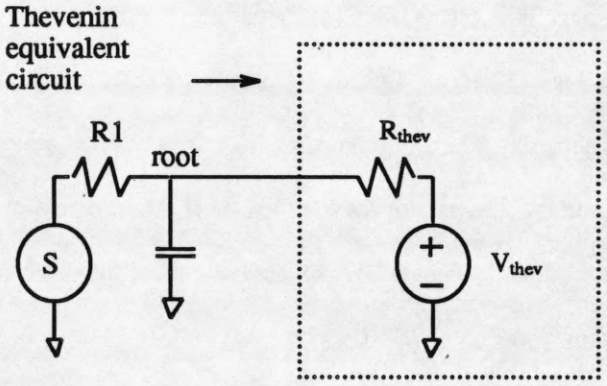
In the second pass, V_{thev} and R_{thev}, looking toward the root node, are found for each node in the network as seen in Figure 2.16(c). At each node, these Thevenin equivalent values are combined with those obtained in the first pass to obtain the node value by using Rule 2. The tree walk in the second pass continues to proceed until a source node is reached. A similar approach was taken by VTIsim [16].

2.4. FACT's Transition Time Computation

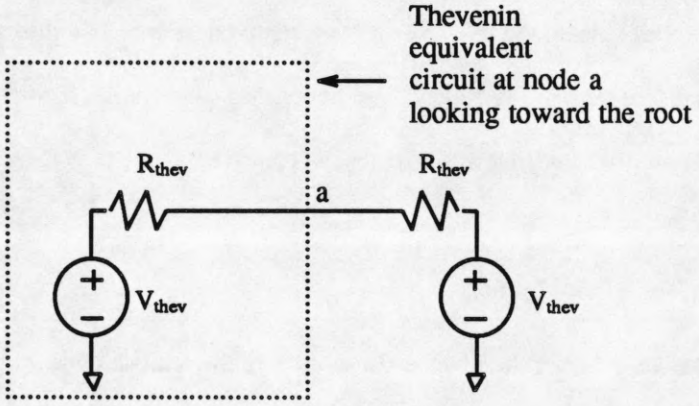
The aim of FACT is to generate the time response of digital circuit designs with less detail and at higher speed than circuit level simulation, but without losing accuracy as measured in terms of voltage levels and timing. This is achieved by modeling the nonlinear transistor network as linear resistor networks as previously described, and apply an RC network technique to find the delay.



(a) An RC-tree network

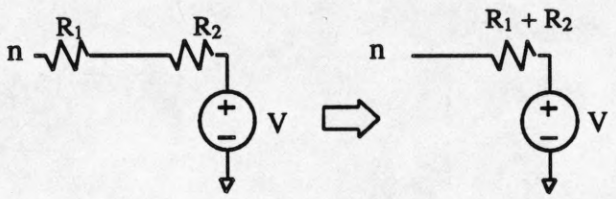


(b) Thevenin equivalent in the first pass

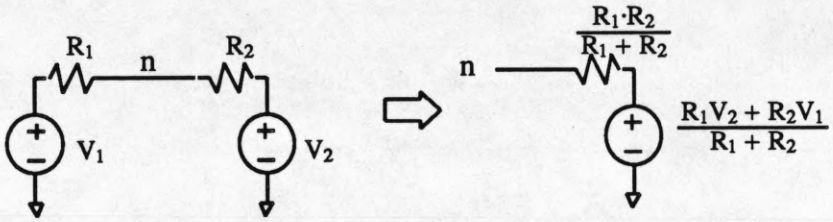


(c) Thevenin equivalent in the second pass

Figure 2.16. Tree-based algorithm for final value computation



(a) Rule 1.



(b) Rule 2.

Figure 2.17. Rules to obtain the Thevenin equivalent circuit

FACT's method for delay computation is based on the Penfield-Rubinstein's models for delay estimate [17]. Penfield and Rubinstein derived a tight upper and lower bound for a delay waveform in an RC tree and proposed a delay estimate that always lies within the bounds. To see how the model works, consider the RC tree in Figure 2.18.

An RC tree is a resistor tree where the root of the tree is the input and each node in the tree has a capacitor to ground. Since the only dc path to a node is from the input, given enough time all the nodes in the tree will settle to the same voltage: the voltage at the input node. When a step voltage is applied to the input, all the nodes in the tree lag the input, eventually settling to the new value. Because the output waveforms change gradually, the most complete method of specifying the delay for an input voltage step is to find the output waveforms versus time. The delay estimate proposed by Penfield and Rubinstein trys to approximates this output waveform with an exponential.

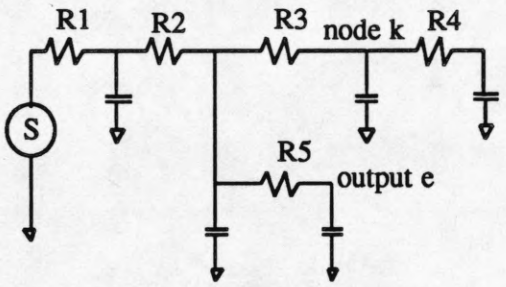


Figure 2.18. An RC tree

Consider any output node e, and any capacitor at node k with capacitance C_k. The resistance R_{ke} is defined as the resistance of the portion of the unique path between the input and e, that is common with the path between the input and node k. The approximate output waveform at node e is defined as

$$V_e(t) = e^{-\frac{t}{\tau_{De}}}; \qquad \tau_{De} = \sum_k R_{ek}C_k$$

This simple delay estimate takes the parasitic capacitances at all other nodes in the stage into account. This method is, therefore, likely to produce more accurate results than those of table-driven methods [18, 19, 20]. Its efficiency in computation also makes it suitable for a fast timing simulator such as FACT. One disadvantage of this delay model is that it is only applicable to an RC tree: it cannot be used for circuits with multiple driving voltages. Because many sub-circuits have multiple same-polarity source connections, crude approximations are unavoidable if this model is used for such circuits. For an accurate simulation, therefore, it is necessary to extend this model to handle non-RC trees.

In FACT, this is achieved by a computationally efficient algorithm we developed based on Wyatts's work on delay estimates in RC meshes [21]. Wyatt has shown, with redefinition of R_{ke}, that the exponential estimate for an RC tree can also be applied to an RC mesh. In the RC mesh in Figure 2.19, the voltage at each node can be represented by the following matrix, with a new definition for r_{ke} .

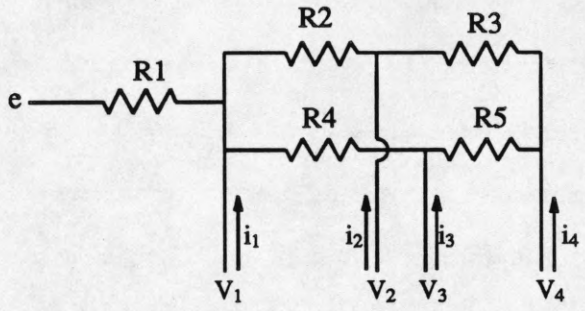


Figure 2.19. An RC mesh

$$\begin{bmatrix} v_1-e \\ v_2-e \\ v_3-e \\ v_4-e \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix}$$

$$r_{ke} = \frac{v_k}{i_e} \text{ where } i_j = 0 \quad \text{for all } j>0,\ j\neq k$$

Unlike the R_{ke} of RC trees, which can be read off by observing the circuit, calculation of r_{ke} is quite complex. For example, r_{ke} can be obtained by inverting an $(n-1)\times(n-1)$ conductance matrix, where n is the number of non-ground nodes in circuit. This matrix will have positive diagonal elements and non-positive off-diagonal elements. Each node except for ground corresponds to one row and one column of the conductance matrix. Entries in the matrix are determined as follows. The diagonal entry of row j is equal to the sum of the conductances incident on node k. The off-diagonal entries are zero unless node i corresponding to that column is a neighbor of node k. In this case that entry is set equal to the negative of the conductance between i and k. It is easily shown that this is equivalent to writing KCL at node k. The resulting conductance matrix is augmented with an $(n-1)\times(n-1)$ identity matrix. Gaussian elimination or the Crout algorithm [22] is used to reduce the conductance matrix to an identity matrix. At this point, the augmented matrix holds the inverted conductance matrix. Thus, $[G:I] \rightarrow [I: G^{-1}]$. The G^{-1} matrix contains the resistance values defined by Wyatt. Since the inversion of a matrix requires an $O(n^3)$ algorithm, this method is computationally too expensive.

For an RC tree with a multiple same-polarity source connection, it is possible to obtain r_{ke} efficiently without having to invert a matrix. This is done by using a tree-based algorithm we developed. To see how the algorithm works, consider the RC tree driven by two sources in Figure 2.20. As seen in the figure, we partition the tree with multiple sources into sub-trees such that the root of each sub-tree is directly connected to each source through a resistor. By representing an RC tree as an RC forest in this way, we can easily find the necessary resistances, r_{ke}. First, consider the case in which both nodes e and k belong to the same sub-tree. For the purpose of illustration, let k and e be any arbitrary nodes in Tree 1 of Figure 2.18; k = G and e = H. By definition, $r_{GH} = \frac{v_G}{i_H}$ with all the currents across the capacitor at each node set to zero with the exception of i_H. Therefore, i_H originating from the capacitor at node H can flow to ground only through the resistive paths, R9→R5→R3→R2 and R9→R5→R1. All other resistors can be ignored from the picture because no current flows through them. The resulting circuit is shown in Figure 2.21.

Since the current through R8 is also zero, $V_D = V_G$ by the KVL and $I_D = I_H$ by the KCL. R_{GH} is, therefore, equal to $\frac{V_D}{I_D}$, which is the equivalent resistance between node D and ground. The problem of finding R_{ke} reduces to finding this equivalent resistance at node D. Rather than trying to find this resistance directly, we divide this

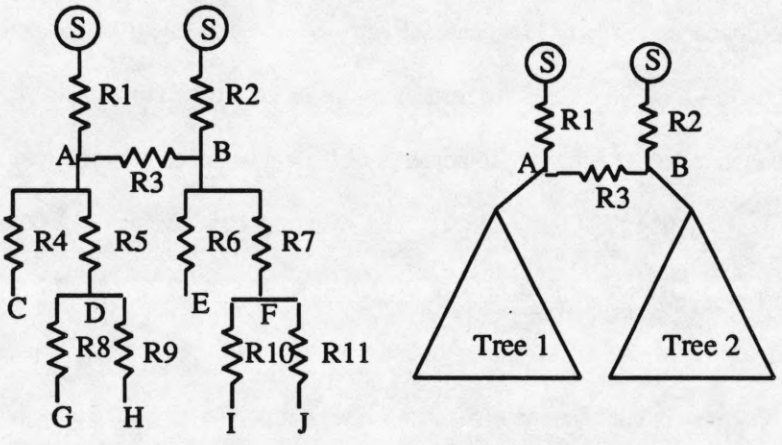


Figure 2.20. Multiple source tree

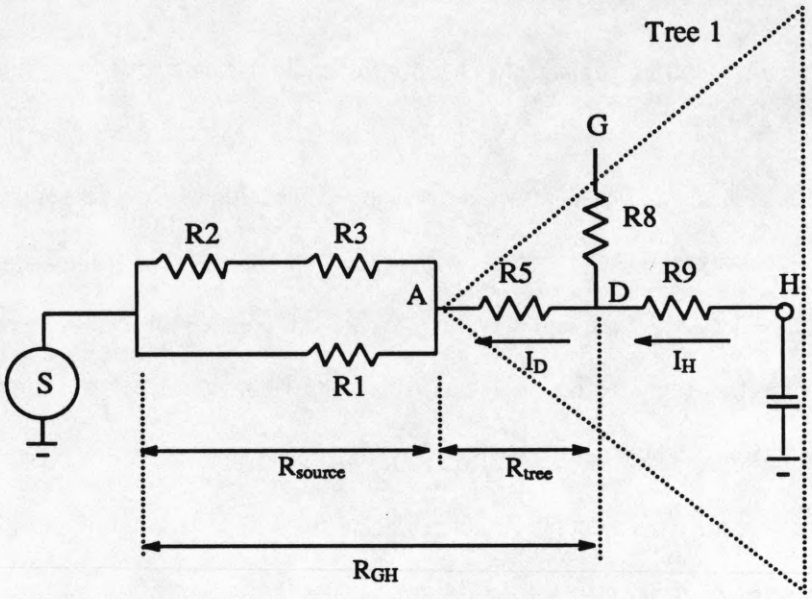


Figure 2.21. Reduced network to find R_{GH}

equivalent resistance into two parts; R_{source}, the resistance between the root node and the source node, and R_{tree}, the resistance of the unique path between the root and e, that is common with the path between the root and node k. R_{ke} is simply the sum of R sub source and R sub tree.

The computation method for R_{tree} is identical to the one used in calculating R_{ke} of a one-source RC tree. Therefore, we can treat the n-source RC tree as n separate one-source RC trees and use the same algorithm. If nodes e and k are not in the same sub-tree, the path from node e to the root of its sub-tree is exclusive from the path from node k to the root. Therefore, R_{ke} is, simply, equal to R_{source} in this case.

After obtaining R_{tree}, we compute R_{source} and add it to R_{tree}. Computing R_{tree} requires an O(n) algorithm, where n is the number of nodes in the stage, and R_{source} is obtained by using an $O(s^2)$ algorithm, where s is the number of sources. Since s is usually limited to five or less per stage, the computation is dominated by the first one. Therefore, the complexity of the above algorithm is O(n).

The extension of the delay model to handle an RC tree with multiple source enables a simulator to cover a wide variety of circuits for an accurate simulation. For instance, delay at the output of a NOR gate can be

accurately computed even when the two inputs change simultaneously. The effect of having multiple paths to V_{DD} for a faster pull-up can also be seen clearly in our timing model.

With a slight modification, our timing model can also handle dynamic CMOS logic structures accurately. For example, consider the basic dynamic CMOS gate in Figure 2.22(a). It consists of an n-transistor logic structure whose output node is precharged to V_{DD} by a p-transistor and conditionally discharged by an n-transistor connected to V_{ss}. ϕ is a clock signal. The precharge phase occurs when $\phi = 0$. The path to V_{ss} supply is closed via the n-transistor when $\phi = 1$. And, if the inputs to the n logic block are set such that there exists a path from the output to this evaluation transistor, the output is pulled down to zero. If the n logic block does not include a cycle, the delay at the output can be accurately modeled.

In its evaluation phase, a dynamic logic circuit can be modeled by the RC circuit in Figure 2.22(b). To find the delay at the output, we need to find $R_{e,output}$ for e = all nodes in the stage. Let e be an arbitrary node in the figure. To find $R_{e,output}$, we inject current I_{output} at the output and measure the voltage at node e, V_e. The desired ratio, $\frac{V_e}{I_{output}}$, is obtained as follows:

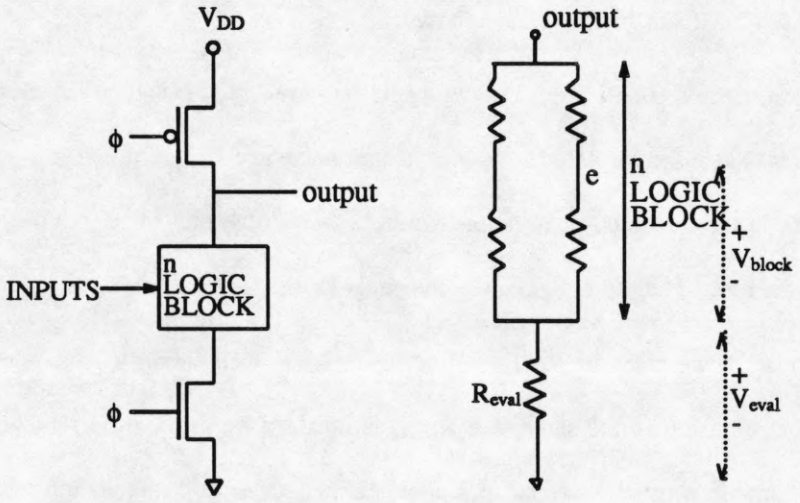


Figure 2.22(a) A dynamic CMOS gate (b) An equivalent resistive network

$$V_e = V_{block} + V_{eval}$$

Since $V_{eval} = I_{output} \cdot R_{eval}$,

$$V_e = V_{block} + I_{output} \cdot R_{eval}$$

Dividing each side by I_{output},

$$\frac{V_e}{I_{output}} = \frac{V_{block}}{I_{output}} + R_{eval}$$

If $R_{eval} = 0$, the circuit becomes an RC tree with two same-polarity sources and $R_{e,output}'$ for this RC tree equals $\frac{V_{block}}{I_{output}}$. Therefore, $R_{e,output}$ for the dynamic circuit is obtained by adding R_{eval} to $R_{e,output}'$ of this RC tree. $R_{e,output}'$ is obtained by using the same algorithm we developed for multiple source RC trees, and is added to R_{eval}:

$$R_{e,output} = R_{e,output}' + R_{eval}$$

Therefore, the algorithm for multiple-source RC trees can also be used for such dynamic circuits.

2.5. Slope and Load Effect

A significant source of error in the RC model comes from its inability to deal with waveform shape. In practice, the effective resistance of a transistor depends on the waveform on its gate. If the trigger transistor turns on instantaneously, then its full driving power is used to drain the output capacitance and the transistor has a relatively low effective resistance. If the trigger transistor turns on slowly, then it may do much or all of its work while only partially turned-on. In this case its effective resistance will be higher.

If all waveforms in a circuit have the same shape, then the effective resistances of transistors can be characterized using that waveform and the RC model will produce accurate results. Unfortunately, this is not the case in actual VLSI circuits. Although almost all waveforms have an exponential shape, they vary by more than three orders of magnitude in their slopes. As a result, the effective resistance of the transistors varies by more than a factor of ten. Therefore, it is highly desired to include the effect of the input slope in the calculation of the effective resistance.

Unfortunately, the effective resistance of a transistor depends not only on the slope of its gate voltage, but also on the load being driven by the stage and on the sizes of the transistors in the stage. If a stage is driving a large load, or has very small transistors, then only very slowly rising input slopes will affect the stage's delay. If a stage is driving a small load or has very large transistors, its delay will be more sensitive to the slope of its input.

As shown previously, effective resistances are obtained with the assumption that the input is driven by a step function. Therefore, if the input is actually a step voltage, this value will be used as its effective resistance. For inputs with slopes, R_{eff} is modified as follows:

$$R'_{eff} = R_{eff}(1+k\frac{load}{input\ slope})(\frac{L}{W})$$

Here, k is a fudge factor that is determined in the pre-simulation phase by using SPICE. W and L are the size of the transistor in λ. The load is approximated by the sum of capacitances driven by the transistor.

The accuracy of this model depends largely on the accuracy with which slope can be calculated. The current FACT implementation approximates the slope by using the delay of the input. For any given voltage, the slope of an exponential waveform at that voltage is proportional to the delay time to reach that voltage. This means that the slope at a node is proportional to its delay, which is the delay computed by the RC timing model. A similar approach was also taken by the Crystal timing analyzer [23].

2.6. Fault Model of FACT

The faults that can be modeled in FACT are: (1) Node stuck-at-zero (2) Node stuck-at-one (3) Short between nodes (4) Line open (5) Transistor stuck-open (6) Transistor stuck-closed (7) Gate-to-drain, gate-to-source, source-to-drain shorts of transistors. (8) Threshold voltage shifts of transistors giving rise to changes in characteristics. In FACT, each of the shorts and opens described can be "resistive" with various ranges of resistance values. This is possible because these faults are represented by adding extra fault transistors such as FMOSSIM [6], and any desired resistance of the fault can be obtained by adjusting the gate voltage and the width of the fault transistor.

Figure 2.23 shows various faults and the corresponding model. Those gate nodes labeled f are normally 0, but are set to 0*, I, 1*, and 1 to activate the fault with various resistances; gate nodes labeled $\overline{f}$ are normally 1, but are set to 0, 0*, I, and 1* to activate the fault. Those gate nodes labeled s are normally 0, but are set to 1 to short the source and the drain of the transistor; gate nodes labeled $\overline{s}$ are normally 1, but are set to 0 to open the source and the drain nodes. The effective resistances of these transistors are set to 0 ohm to have the effect of a short between the source and the drain nodes.

A stuck-at-zero or stuck-at-one node fault can be modeled by inserting a transistor with an infinite width to short the node to Gnd or Vdd, respectively. A stuck-closed transistor fault is injected by shorting the transistor's source and drain nodes. The short is achieved by a fault transistor in parallel with the fault-free transistor. To eliminate the effect of the fault-free transistor when simulating the fault, we insert an S-transistor, which acts like a switch, at the source node of the fault-free transistor. Similarly, a stuck-open transistor fault is modeled by a fault transistor in series and an S-transistor in parallel as seen in Figure 2.23. A gate-to-source short of a transistor is injected by connecting a fault transistor to the gate and the source terminals of the transistor. Gate-to-drain and source-to-drain shorts are modeled similarly. Threshold voltage shifts of transistors are injected by shifting the values for R_{eff} in discrete steps.

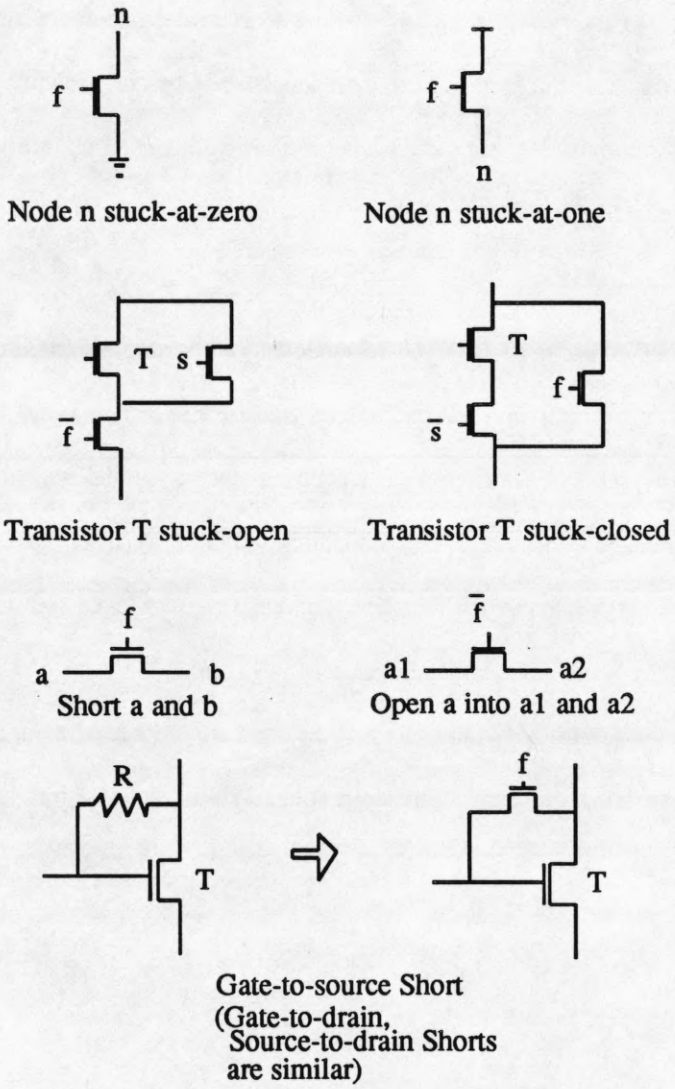


Figure 2.23. Fault models used by FACT

CHAPTER 3.

IMPLEMENTATION

FACT uses the event-driven method, which essentially propagates the effects of signal changes through a circuit. The main operations to be performed in an event-driven simulation algorithm are scheduling of events, processing of events, evaluation, and event list management.

3.1. Scheduling

Events are generated as a result of primary input changes and the evaluation of stages. All events for a particular time are kept in a list, called the event list, for that time. Insertion of events in the event list for the appropriate time is termed scheduling.

Only signal changes are inserted into the event list. This requires a comparison of the computed value of each node with its present value if a change has not already been scheduled. If a change has been scheduled, the newly computed value is compared with the last scheduled value. Suppression of spikes of short duration is performed at this time.

For detecting a spike, the newly computed value of a signal is compared with its last scheduled value. A spike is present if the values are different and the time between the last scheduled change and the new change to be scheduled is smaller than a prescribed value. The spike is suppressed by deleting the last scheduled event and ignoring the new change.

In FACT, it is sometimes necessary to cancel certain scheduled events in order to obtain more accurate transition time. For example, consider a CMOS NOR gate in Figure 3.1, where input IN2 rises immediately after input IN1 does. When input IN1 changes its state, the output gradually goes to zero, and a new event is scheduled at the transition time, τ. But before τ is reached, input IN2 changes its state and turns on an n-channel transistor, adding another pulldown to the output. A new transition time is computed using the two pull-downs and is

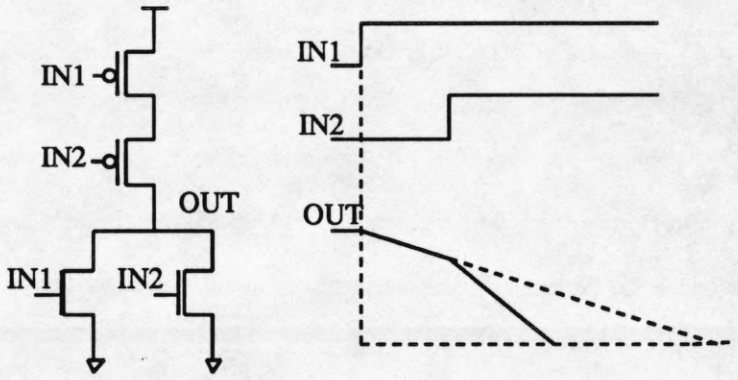


Figure 3.1. Effect of multiple changing inputs

scheduled before τ. The previously scheduled event at τ should, therefore, be canceled and replaced by the newly computed one.

3.2. Processing Scheduled Events and Stage Evaluation

The processing of scheduled events consists of updating node values and determining the stages to be evaluated as a consequence of the node value change. For every event in the event list at the current simulated time, the node value is updated to its new value. If the updated node is connected to gates of other transistors, a new stage is formed for such transistors and is put on a list, called the evaluation list, if it is not already there.

After all the updates and fan-out processing for a particular simulated time have been performed, the stages on the evaluation list are evaluated. The stage evaluation consists of computing new node values and their transition times. Events generated as a result of the stage evaluation are inserted into the event list.

3.3. Event List Management

FACT uses the time wheel to manage the event list (See Figure 3.2.). The event lists on the time wheel are separated by fixed increments of 1 nanosecond of simulated time, allowing faster access to events for a desired time. The event wheel is managed as a circular buffer in which the N array elements hold events for the next N

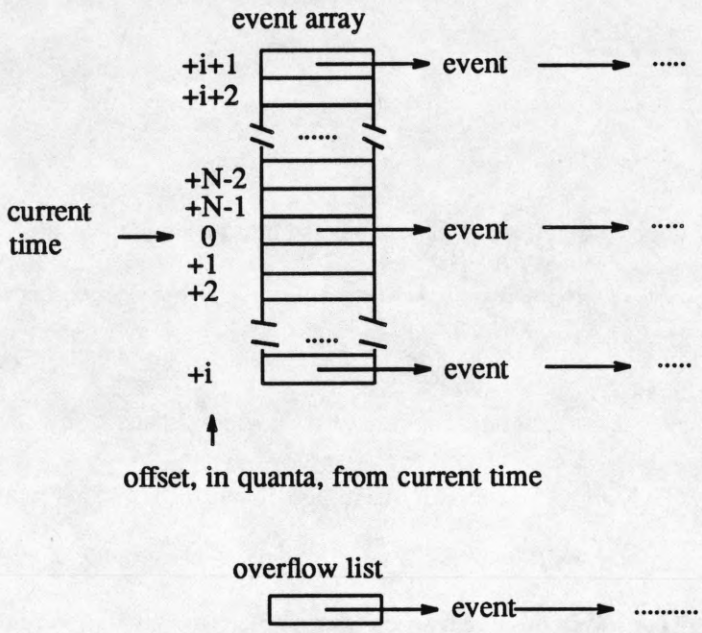


Figure 3.2. The time wheel is implemented with an event array

time quanta. An array index indicates the array element which corresponds to the current simulated time. If a new event is scheduled for a time M quanta in the future, where M < N, the event is added to the end of the event list sorted in the array element (index + M) mod N. If M > N, the event is inserted into the overflow list according to its scheduled time.

To find the next event to process, the event array is searched starting at the current index, until an event is found. Each increment of the index corresponds to advancing the simulated time by one nanosecond. If the array is empty, the simulated time is advanced to equal the scheduled time of the first event on the overflow list; this event then becomes the next one to be processed. When an event is located for processing, the overflow list is examined to find events whose scheduled times are less than N time quanta away from the new simulated time. Such events are moved from the overflow list to the appropriate list in the time wheel.

CHAPTER 4.

SIMULATION RESULTS

The timing simulation model described in Chapters 2 and 3 has been implemented. The fault simulator with an accurate timing model, FACT, is written in C and runs on the SUN workstation under the Unix operating system. We evaluate the performance of FACT based on its computational speed and the accuracy of its timing estimates, first, for some circuits without fault injection, and then for some circuits with faults.

In Figure 4.1, a CMOS NOR gate is shown. Here, the effect of the second input change, IN2, is clearly seen by the change in the output slope (Figure 4.2.). This is possible because the voltage transitions are modeled by ramps in FACT. RSIM[11] solves this event scheduling problem by ignoring the first input change, resulting in a serious timing error.

The next example is a Manchester carry chain (Figure 4.3.). Here, we can see how our improved delay algorithm adds accuracy to the timing estimate. In this circuit, the length of delay at the output of the carry chain depends upon the number and the positions of pull-down paths. This variation in delay is more accurately predicted by the multiple-source delay algorithm of FACT than that of RSIM (Figure 4.4).

One large benefit of using five logic values over 0, 1 and X values can be seen when simulating circuits with a feedback such as EXOR gates in Figure 4.5. Each EXOR gate in the figure contains a pass-gate and a feedback line which controls the pass-gate. For certain input transitions, the logic transition at the output has an initial slow rise time due to a slow feedback response. RSIM and other switch level simulators fail to predict the correct value at the output because the small voltage change at the output is not recognized as an event. In FACT, however, the correct value is predicted because the voltage change as little as 0.2V is recognized as an event (Figure 4.6.). This sensitivity added to the event scheduling enables the simulator to predict the correct output. Table 4.1 shows a comparison of the execution times of RSIM, FACT, and SPICE for various circuits.

Figure 4.7 shows a chain of CMOS inverters with a gate-to-source resistive short fault. Figure 4.8(a) shows the output waveform at node D when R = 200 ohms. This fault causes a timing error only at node D. When the resistance of the shorted path is 5 ohms, the output is inverted (Figure 4.8(b)). These faults are all detected by both SPICE and FACT. RSIM, however, predicted an X value at node D when the fault was present. Because X is interpreted as an indeterminate state rather than an intermediate voltage, there is no guarantee that the X value will produce an effect different than the state of the node in the good circuit.

Figure 4.9 shows a dynamic AND decoder with a resistive open. The effect of the fault is seen in Figure 4.10 as a delayed output value. Figure 4.11 shows the CMOS latch with various line-open faults. The waveforms produced by SPICE in Figure 4.12 show that faults f2, f5, f6, and f7 are detected by the applied input patterns. Although not able to produce the detailed waveform such as SPICE, FACT is accurate enough to detect all the faults detected by SPICE as seen in Figure 4.13. This shows that FACT predicts the behavior of the circuit under faults accurately in terms of voltage levels and timing.

These simulation results clearly indicate that the proposed timing model gives a significant accuracy to both fault-free and fault simulations at a reasonable cost.

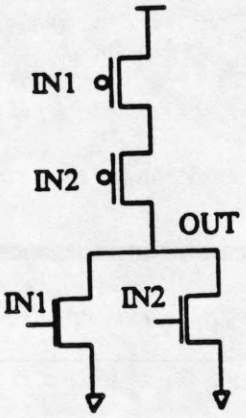


Figure 4.1. A CMOS NOR gate

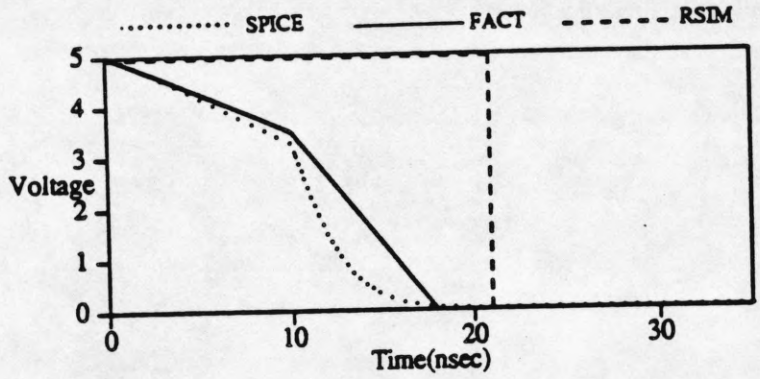


Figure 4.2. Waveforms for a CMOS NOR gate

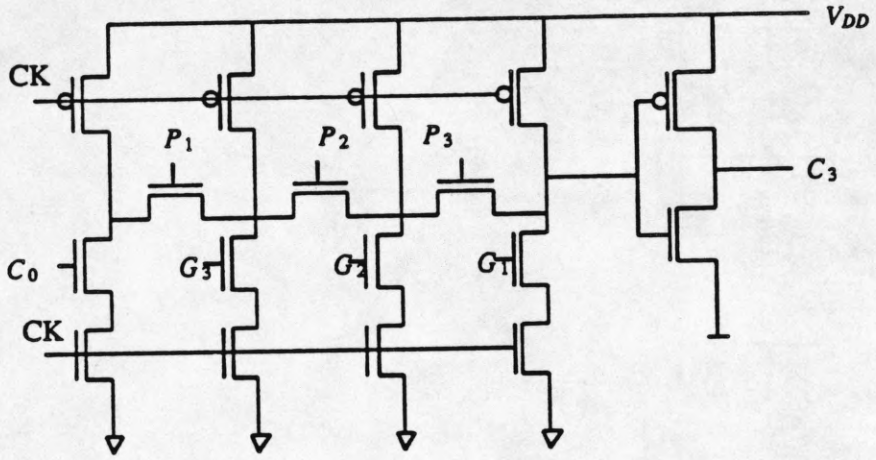


Figure 4.3. A Manchester carry chain

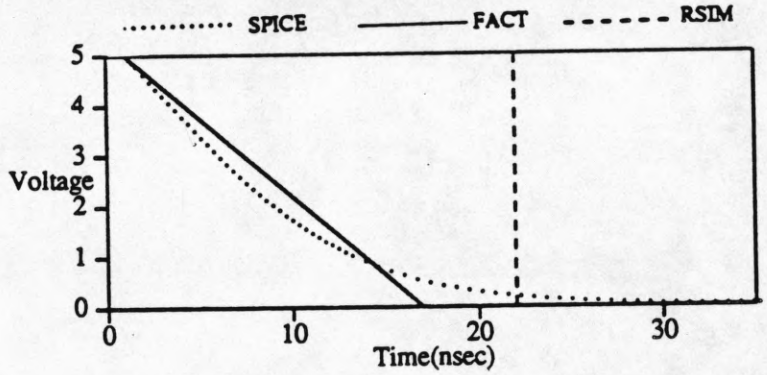


Figure 4.4. Waveforms for a Manchester carry chain

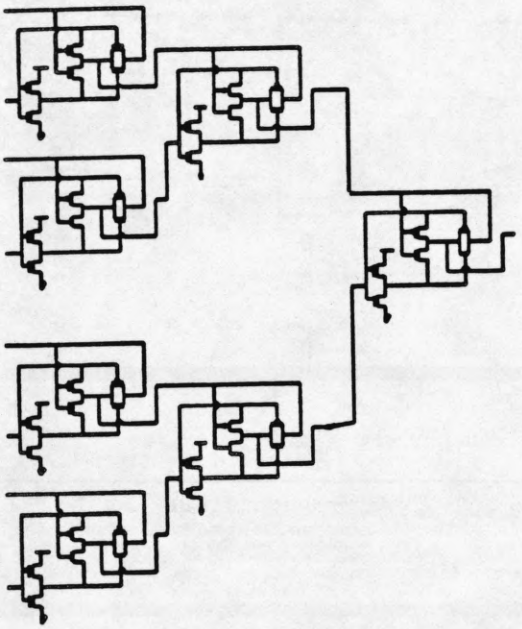


Figure 4.5. A parity generator

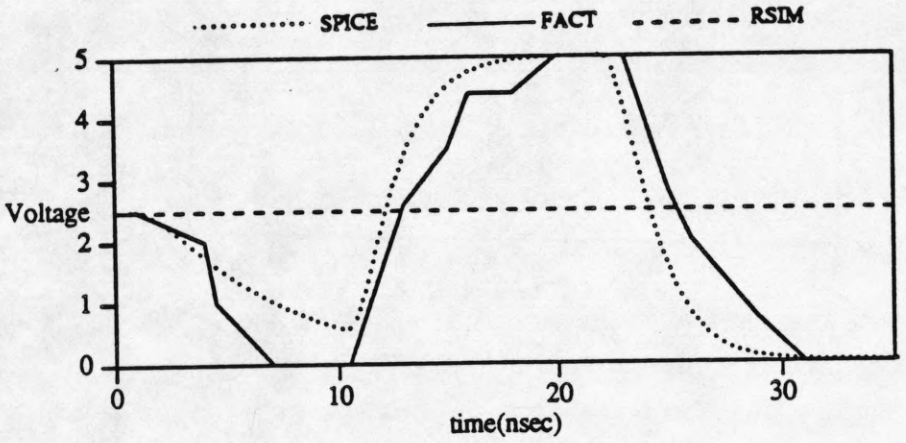


Figure 4.6. Waveforms for a parity generator

Table 4.1. Time Taken by Various Circuits

Circuits	RSIM (sec.)	ATS (sec.)	SPICE (sec.)
A latch	0.7	1.2	60
8 input Parity Generator	0.8	1.5	82
A chain of 20 Inverters	1.0	1.8	110
8-bit Asynchronous Counter	2.2	3.4	224

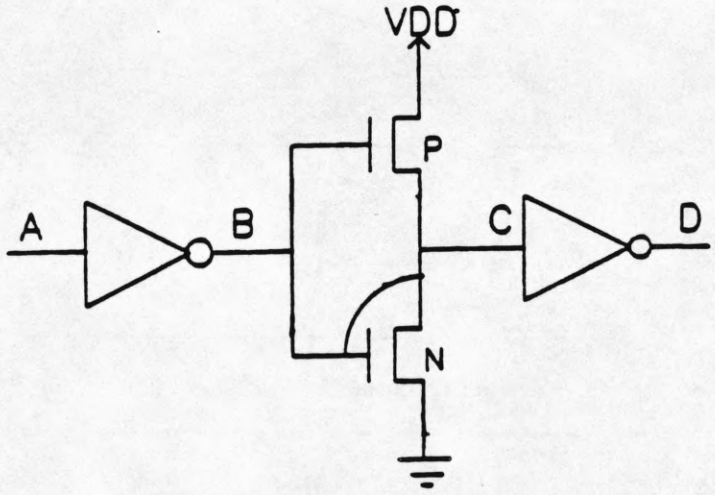


Figure 4.7. A chain of CMOS inverters with a fault

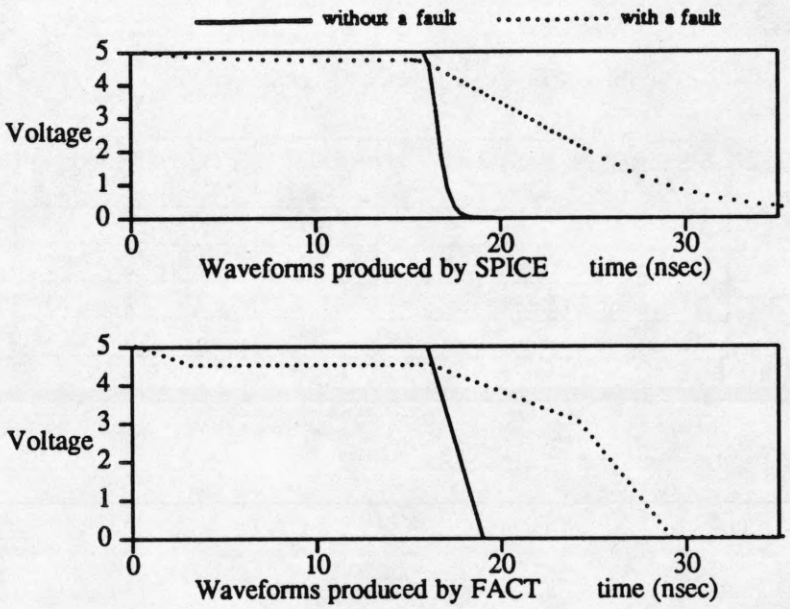


Figure 4.8. (a) Waveforms at node D when R = 200 ohms

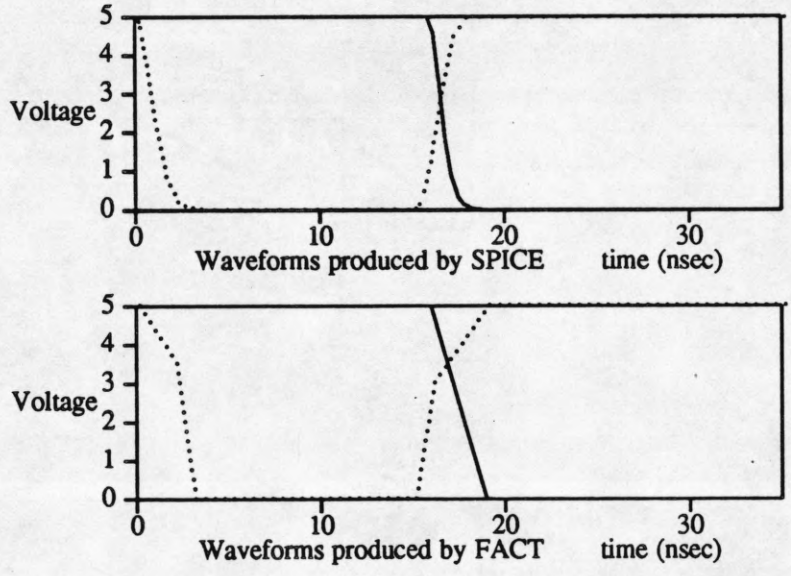


Figure 4.8. (b) Waveforms at node D when R = 5 ohms

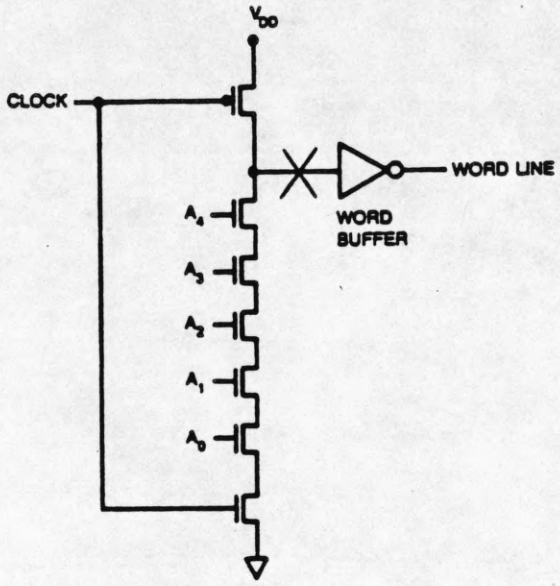


Figure 4.9. A dynamic AND decoder

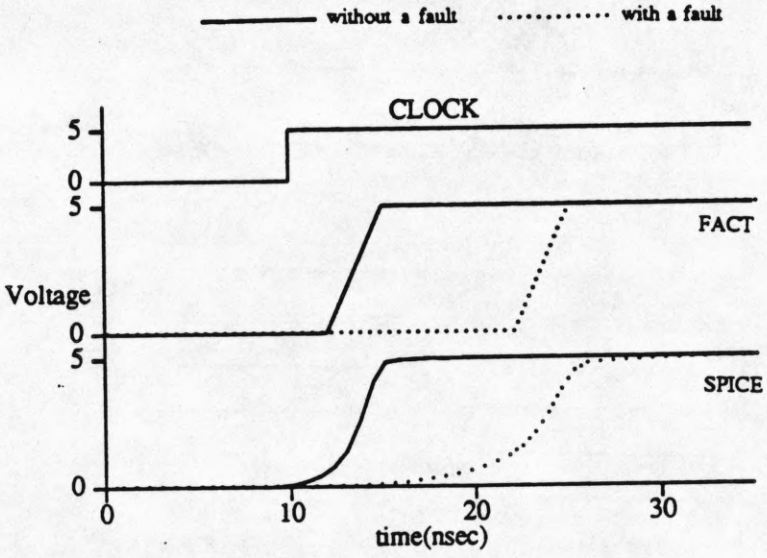


Figure 4.10. Waveforms at WORD LINE

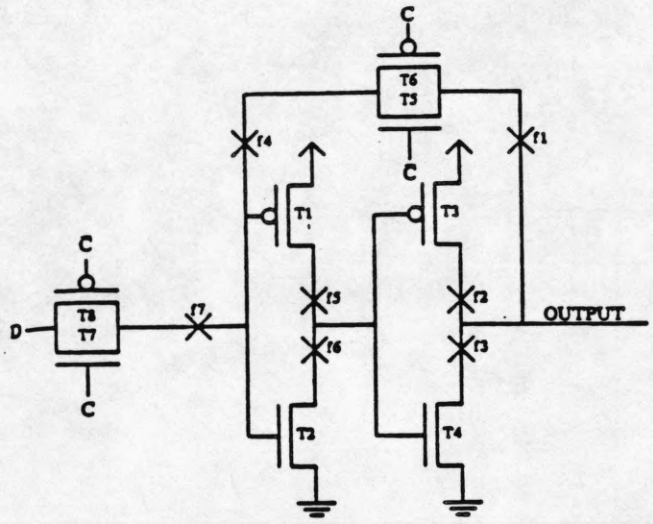


Figure 4.11. A CMOS latch

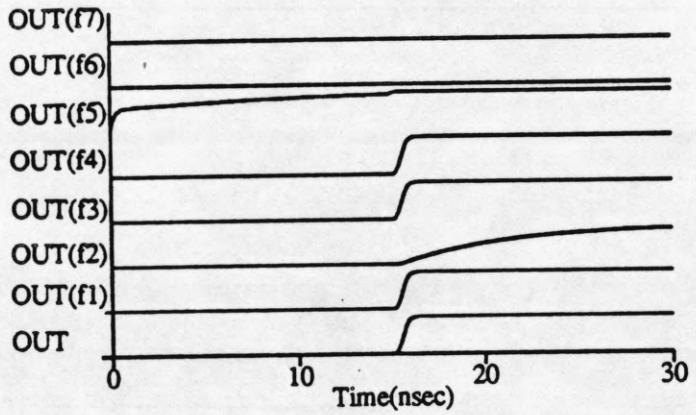


Figure 4.12. Waveforms for a CMOS latch under various faults (SPICE)

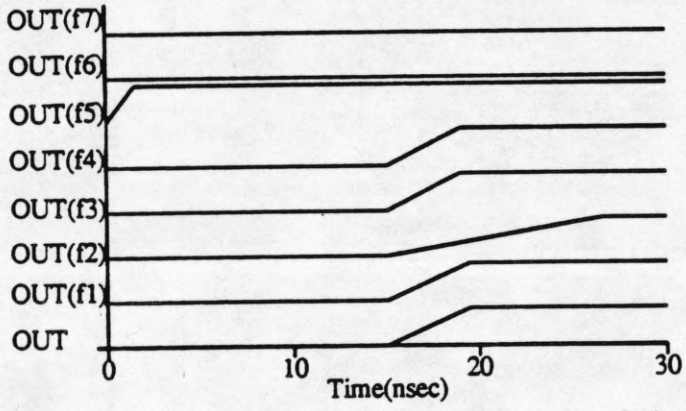


Figure 4.13. Waveforms for a CMOS latch under various faults (FACT)

CHAPTER 5.

CONCLUSIONS

In this thesis, we presented a timing model for fault simulation. The significant features of the model are as follows.

1) It uses five logic values for more accurate fault-free and fault simulations.

2) The delay calculation can handle multiple source networks and is therefore a significant improvement over the conventional single source models of Penfield-Rubinstein [17].

3) The fault model is very realistic and incorporates resistive shorts and opens.

4) It can handle multiple faults and will be used in the future in a concurrent fault simulator.

The simulation results obtained by FACT indicate that the model gives more accurate prediction than that of RSIM, and has a clear advantage over fault simulators implemented at the switch level such as FMOSSIM in terms of accuracy, all without much added cost.

LIST OF REFERENCES

[1] S. Koeppe, "Modeling and Simulation of Delay Faults in CMOS Logic Circuits," *Proc. IEEE Internat. Test Conf.*, pp. 530-536, September 1986.

[2] T Hayashi, K. Hatayama, K. Sato, and T. Natabe, "A Delay Test Generator for Logic LSI," *Proc. Internat. Conf. on Fault-Tolerant Computing*, pp. 146-149, June 1984.

[3] E. P. Hsieh, R. A. Rasmussen, L. J. Vidunas, and W. T. Davis, "Delay Test Generation," *Proc. 14th ACM-IEEE Design Automation Conf.*, pp. 486-491, June 1977.

[4] V. S. Iyengar, B. K. Rosen, and I. Spillinger, "Delay Test Generation 1 -- Concepts and Coverage Metrics," *Proc. IEEE Internat. Test Conf.*, September 1988.

[5] V. S. Iyenger, B. K. Rosen, and I. Spillinger, "Delay Test Generation 2 -- Algebra and Algorithms," *Proc. IEEE Internat. Test Conf.*, September 1988.

[6] R. E. Bryant and M. D. Schustor, "Fault simulation of MOS digital circuits," *VLSI DESIGN*, pp. 24-30, Oct. 1983.

[7] P. Banerjee and J. A. Abraham, "Characterization and testing of physical failures in MOS logic circuits," *IEEE Design and Test*, pp. 76-86, August 1984.

[8] M. A. Breuer, "The effects of races, delays, and delay faults on test generation," *IEEE Trans. Comput.*, vol. C-23, pp. 1078-1092, Oct. 1974.

[9] S. M. Reddy, M. K. Reddy, and V. D. Agrawal, "Robust tests for stuck-open faults in CMOS combinational logic circuits," *Proc. 14th Int. Fault-Tolerant Computing Symp.*, pp. 44-49, June 1984.

[10] H. C. Shih, J. T. Rahmeh, and J. A. Abraham, "FAUST: An MOS fault simulator with timing information," *IEEE Trans. CAD*, vol. CAD-5, pp. 557-563, Oct. 1986.

[11] C. J. Terman, "Simulation Tools for Digital LSI Design," *M.I.T. Laboratory for Computer Science, TR-304*, 1983.

[12] B. R. Chawla, H K. Gummel, and P. Kozak, "MOTIS - An MOS timing simulator," *IEEE Trans. on CAS*, vol. CAS-22, No. 22, pp. 901-910, Dec. 1975.

[13] A. J. de Geus, "SPECS: Simulation Program for Electronic Circuits and Systems," *International Symposium on Circuits and Systems*, pp. 534-537, 1984.

[14] S. H. Hwang, Y. H. Kim, and A. R. Newton, "An accurate Delay Modeling Technique for Switch-Level Timing Verification," *23rd Design Automation Conference*, pp. 227-233, 1986.

[15] P. Banerjee and J. A. Abraham , "Fault Characterization of VLSI MOS Circuits," *Proceedings, International Conference on Circuits and Computers*, pp. 564-568, September 1982.

[16] T. J. Schaefer, "A transistor-level logic-with-timing simulator for MOS circuits," *22nd Design Automation Conference*, pp. 762-765, 1985.

[17] P. Penfield, Jr., and J. Rubinstein, "Signal delay in RC tree networks," *18th Design Automation Conference*, pp. 613-617, 1981.

[18] H. N. Nham and A K. Bose, "A Multiple Delay Simulator for MOS LSI Circuits," *Proc. ACM IEEE 17th DAC*, pp. 610-617, June 1980.

[19] V. B. Rao, T. N. Trick, and I. N. Hajj, "A Table-Driven Delay-Operator Approach to Timing Simulation of MOS VLSI Circuits," *Proc. 1983 IEEE ICCD*, pp. 445-448, Nov. 1983.

[20] F. Lai, V. B. Rao, and T. N. Trick, "JADE: A Hierarchical Switch Level Timing Simulator," *Proc. 1987 IEEE ISCAS*, pp. 592-595, May 1987.

[21] A. Wyatt, "Signal delay in RC meshes," in *VLSI Memo, No. 84-196, MIT*, Cambridge, Mass., pp. 1-7, August 1984.

[22] J. Vlach and K. Singhal, in *Computer Methods for Circuit Analysis and Design* New York, NY: Van Nostrand Reinhold Company, 1983.

[23] J. K. Ousterhout, "Switch-level delay models for digital MOS VLSI," *21st Design Automation Conference*, pp. 542-548, 1984.