COMPUTER SYSTEMS LABORATORY

STANFORD UNIVERSITY · STANFORD, CA 94305-4055

Improved Models for Switch-Level Simulation

Chorng-Yeong Chu

Technical Report No. CSL-TR-88-368

This dissertation was supported in part by National Science Foundation contract MIP 8451822.

Improved Models for Switch-Level Simulation

Chorng-Yeong Chu

Technical Report No. CSL-TR-88-368

November 1988

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305-4055

Abstract

Simulation plays an important role in design verification. With increasingly large VLSI designs, the switch-level representation has become the only approach that is both reasonably accurate and computationally feasible.

At present, switch-level simulators use relatively unsophisticated techniques to extract information from the switch-level representation, and even these small amounts of information are not always fully utilized. As a result, these simulators often lack accuracy. Most notably, the way some switch-level simulators compute the final value can potentially generate undesirably pessimistic results, and charge-sharing problems are widely ignored.

This thesis shows how to extract more information based on the same set of widely adopted switch-level assumptions. Using more sophisticated analyses, this thesis presents better final-value and charge-sharing models. The new final-value model uses a systematic way to look at the relationship between the voltage and the resistance. This approach can also objectively compare the accuracy of different DC-computation schemes. Charge-sharing problems are modeled with two time constants. The two-time-constant approach is based on the observation that most waveforms due to charge sharing are dominated by a pair of time constants. Charge-sharing models are first constructed on resistor networks, then they are extended to transistor networks.

These models have been incorporated into nRSIM --- a RSIM-based switch-level simulator. The new simulator has the same running time as the original RSIM, but it can handle a larger class of circuits.

__Key Words and Phrases:__ Charge-sharing models, delay estimation, final-value computation, MOS VLSI, RSIM, switch-level simulation.

Contents

List of Figures

Chapter 1

Introduction

Designers of integrated circuits (IC's) rely on simulators to do design verification, which includes the checking of the functionality as well as the performance of their designs. At present, metal-oxide-semiconductor (MOS) IC's with several hundred thousand transistors are quite common, whereas, for these designs, a detailed analysis in a reasonable time frame is well beyond today's computing power. In the early eighties, a new concept called switch-level simulation was developed to cut down drastically the simulation time but still give reasonably accurate results.

The basic idea behind the switch-level representation is to substitute each transistor with a much simplified resistive-switch model. This abstraction filters out all the nonessential details of a transistor but still catches its basic functionality. This technique is particularly suitable for simulating digital designs because digital designs are less sensitive to the exact characteristics of transistors. Switch-level models are gaining popularity because of their flexibility — trade-offs between the accuracy of the simulation result and the simulator's computational requirement can be made by adjusting the details of the resistive-switch model.

1.1 Organization

The next chapter describes previous work in switch-level modeling. Two well-known simulators, MOSSIM and RSIM, are reviewed. Even though these simulators have been widely used, there is still room for improvement. Chapter 2 then identifies problems in the logic and timing aspects of RSIM. Only delay estimation in the timing aspect has been investigated by other researchers, and their single-time-constant and two-time-constant models are summarized. The rest of the thesis will try to improve other shortcomings of switch-level simulation.

Chapter 3 discusses problems encountered in evaluating the final value of a network. In switch-level simulations, there are times when the state of the gate input of a transistor cannot be determined. When this happens, it is not clear whether the source and drain terminals of the transistor are electrically connected. Since the final value of a network can change with the electrical configuration of the network, determining the connectivity is important. This chapter first surveys and compares the techniques suggested by other researchers in a systematic way, which involves the construction of the solution space formed by possible resistance-voltage combinations at each node. It then proposes a new scheme and verifies that the new scheme can do better than other schemes under many situations.

Chapter 4 looks at charge-sharing problems in resistor-capacitor networks. Charge sharing is a timing-related problem caused by the redistribution of charge among capacitors of a network. This problem can be classified into two categories depending on whether a network is driven. The existing timing models are inadequate for charge-sharing problems because they do not take the charge-redistribution process into considerations. This chapter suggests methods to approximate both kinds of charge-sharing problems. Mathematical and intuitive arguments are provided to substantiate the basis of the proposed methods.

Chapter 5 applies insight gained from Chapter 4 to model charge-sharing problems in transistor-capacitor networks. The major problem with modeling a transistor network as a resistor network is that transistors are nonlinear devices, and it is not clear how the nonlinearity of transistors can affect the accuracy of the result. By looking at MOS transistors as pseudo-linear devices, it is actually possible to directly apply the linear formulations to transistor networks. This chapter explores the differences such as the shapes of waveforms and the determination of time constants in charge-sharing models of resistor and transistor networks. A part of this chapter and most of Chapter 4 have been published in [6,7].

Chapter 6 examines the implementation issues of the models presented in Chapters 3, 4, and 5. It is shown that RSIM's existing algorithm can effectively implement all these models in complexity that is linear with respect to the number of transistors in an electrically connected cluster. It also shows that the previous timing models can be generalized straightforwardly from the single-driver assumption to multiple drivers without complicating the evaluation algorithm.

Finally, Chapter 7 summarizes the contributions of this thesis and describes areas for further investigations.

Chapter 2

Previous Work in Switch-Level Simulation

2.1 Overview

The switch-level representation of a transistor circuit is an abstract model that lies between the logic-level and circuit-level representations. This representation is particularly suitable for simulating digital VLSI designs because it makes a good compromise between the speed of simulation and the accuracy of its result. Generally speaking, switch-level simulators run about two to three orders of magnitude faster than circuit-level simulators such as ASTAP[27] and SPICE[17], while providing most of the information needed to analyze digital designs. Although, by solving complicated differential equations, circuit-level simulators are capable of generating detailed analog waveforms, this information is often irrelevant for analyzing pure digital circuits. In contrast, a fast simulator is invaluable for large designs.

Problems with most logic-level simulators[9,25] are not their speed; their shortcomings are in the evaluation of logic and the estimation of delay. For MOS technologies, logic-level simulators' ability to handle pass transistors (transmission gates) is limited by their Boolean-gate model, which does not manipulate bidirectional signals well. In addition, logic-level simulators which use designer-supplied minimum, nominal, and maximum gate delays to estimate circuit speed are often awkward at modeling MOS circuits where the loading depends on the logic value. These shortcomings negate the speed advantage of logic-level simulators.

This chapter summarizes prior works in switch-level simulation. In Section 2.2, the fundamentals of two widely used switch-level simulators are presented. These simulators substitute transistors and capacitors with abstract elements. In doing so, they try to maintain the essential features of the original circuit while simplify those which are irrelevant to the analysis of digital VLSI designs. However, it is not completely effortless to analyze circuits that are in the switch-level representations. These difficulties are explained in Section 2.3. Only one of the difficulties, namely delay estimation, has been extensively investigated by other researchers. Their results are reviewed in Section 2.4. The following chapters will then describe methods of improving the remaining problems.

2.2 Switch-Level Models

In switch-level models, a node is characterized by its capacitance, which is determined by the node's physical layout. This capacitor has to be charged or discharged in order to change the node's state. Since a capacitor does not charge or discharge instantaneously, there is a delay associated with each state change. This delay is directly proportional to the capacitance of the capacitor, and is inversely proportional to the current driving the node.

To model the sources of current, each transistor is represented by a resistive switch which links the transistor's source and drain terminals. The switch is further modeled by a perfect switch in series with a resistive device. The state of the switch is determined by the transistor's gate input. For nMOS devices, the switch conducts if the gate is high; for pMOS devices, the switch conducts if the gate is low. If the value of a transistor's gate input cannot be determined by its simulator, then the state of the corresponding switch cannot be specified either.

The resistive component reflects the current-conducting ability of the transistor. There are many ways to calibrate the conductivity. The two most well-known switch-level schemes were proposed by Bryant[4,5] and Terman[26]. Bryant's scheme is more abstract in that only a small set of discrete values are used to do the calibration. In contrast, Terman uses a continuous spectrum of effective resistances, which is more realistic in measuring the conductivity of a transistor.

2.2.1 Bryant's Scheme (MOSSIM)

Bryant's scheme, which is adopted by switch-level simulators such as MOSSIM[5], MOSSIM II[3], and COSMOS[2], does not calculate the exact conductivities of transistors: it only keeps track of the relative conductivities. The notion of relative conductivity is essential in modeling ratioed logic, which requires one conducting transistor to overpower another conducting transistor. The scheme labels relative conductivity with discrete strengths $\gamma_1, \gamma_2, \ldots, \gamma_{\max}$. The required difference in conductivity to put transistors into different strength classes is set by the user, and is normally set such that ratioed logic can operate properly. Consequently, this scheme is also known as the "order-of-magnitude" scheme. The number of different strengths required for simulation is usually small. For example, most CMOS designs do not involve ratioing, hence, they can be modeled with one strength; while most Mead-and-Conway[16] style nMOS designs only need two strengths: one for depletion load transistors and one for all other transistors. Since conductivities of devices with different strengths differ significantly[1], a weaker device can be ignored

[1]Typically a factor of four for Mead-and-Conway style nMOS designs.

when connecting in parallel with a stronger device, but forms a bottleneck if the connection is in series. In other words, no matter whether two devices are connected in parallel or in series, they can always be replaced by one device. For parallel devices, the strength of the new device is set by that of the stronger one of the original devices, while for series devices, it is set by that of the weaker one of the original devices.

Not only are transistors' conductivities measured in a relative sense, but sizes of capacitors are also relative. A small number of strengths $\kappa_1, \kappa_2, \ldots, \kappa_{\max}$ are assigned to capacitors. The criterion in assigning these strengths is such that when two capacitors with different strengths are connected together, the capacitor with the weaker strength can be charged to the voltage level of the other capacitor. For example, a precharged node is often assigned a stronger strength than an ordinary node because a precharged circuit works through capacitance ratioing.

A third kind of strength is assigned to input nodes. These nodes are labeled with strength ω regardless of their logical states. Strengths of transistors, capacitors, and inputs can be compared to one another. An input has the strongest strength because it can directly change the state of a node. A node which has an electrically connected path to an input node is said to be *driven*, and the path is called a *driving path*. When passing through a transistor, the strength of an input is attenuated by the transistor. Hence, the strength on the other side of the transistor is set by the strength of the transistor. When two or more driving paths merge at a node, the path with the strongest strength suppresses all other paths weaker than it. Driven values always overpower capacitors because capacitors can always be driven to the state of the driving source. This reasoning implies that capacitors are "weaker" than any driving path, and thus have weaker strengths than transistors. In a nondriven network, a signal originating from a capacitor retains its strength when passing through a transistor. Such a path is called a *charging path*. A weaker charging path can be ignored when a stronger one is present at the same node. The ordering of the three kinds of strengths is

$$\kappa_1 < \kappa_2 < \ldots < \kappa_{\max} < \gamma_1 < \gamma_2 < \ldots < \gamma_{\max} < \omega.$$

This ordering is handy in filtering out weak paths with negligible effects. By blocking out weak paths. the problem of determining the state of a node is simplified: the state depends on unblocked paths only. If all the unblocked paths have the same logical state. then the node has that state as well. In contrast. if the unblocked paths have different logical states. then the state of the node is undefined because it is unclear which of the paths is the dominant one.

2.2.2 Terman's Scheme (RSIM)

The other widely used switch-level model was developed by Terman for RSIM[26]. This scheme models a transistor as a linear resistor. Terman assigns two sets of equivalent resistances to each transistor in its conducting state: one set for dynamic purpose and the other for static purpose. Thus, this scheme is also known as the "effective-resistance" scheme. Both sets of equivalent resistances are. among other factors. functions of a device's dimensions and material.

The dynamic equivalent resistance is used for delay estimation. Its value depends on the circuit context of the transistor. For example, it is much easier to discharge than to charge a capacitor through an nMOS transistor due to the device's nonlinear current-to-voltage characteristic. RSIM, with help from a circuit simulator, assigns dynamic equivalent resistance to take nonlinearity into account. In order to find the equivalent resistance of an nMOS transistor charging a capacitor, Terman assumes a step function for the transistor's gate input. and finds the duration required for the capacitor to reach certain switching voltage. This duration is defined as the rising time constant. The ratio between the rising time constant and the capacitance is defined to be the dynamic equivalent resistance of the transistor. A complete set of dynamic equivalent resistances can be gathered by doing experiments on all possible combinations of simple circuit configurations.

In addition to a set of dynamic equivalent resistances, Terman uses, probably incorrectly,[2] a different set of equivalent resistances, namely the static equivalent resistances, to find the DC voltage of each node in a transistor network. For a CMOS circuit which does not depend on device ratioing for its correct operation, static equivalent resistances can have any value. For an nMOS gate in which a voltage divider is formed between a depletion load and one or several enhancement pull-down paths, there is still considerable freedom in assigning static equivalent resistances to ensure correct simulation.

Since a transistor network in Terman's scheme is modeled as a resistor-capacitor network (or an RC network), linear circuit theories can be used to simplify the network. For example, two transistors in series are modeled as one linear resistor having the sum of their equivalent resistances; similarly, the resistance of two parallel transistors is the parallel combination of their equivalent resistances. The DC voltage of a node in an RC circuit can be determined by analyzing its resistors alone. RSIM also provides a delay estimation, which is approximated by the product of the lumped sum of resistances and the lumped sum of capacitances. This approach is illustrated by an example in Figure 2.1, where the delay at node n is estimated to be

$$\tau = (\sum_{k=1}^{n} R_k)(\sum_{k=1}^{n} C_k). \tag{2.1}$$

[2]Since the same factors that affect a transistor's ability to drive a capacitor also affect the DC value, one can use a single effective resistance for both the dynamic and static analyses. This not only simplifies the analyses, but also give better results.

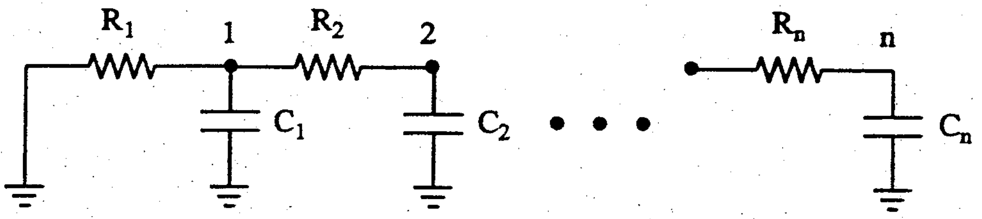


Figure 2.1: A distributed RC line.

2.2.3 Comparisons between MOSSIM and RSIM

MOSSIM and RSIM put emphases on different issues. By assigning discrete strengths to transistors and capacitors, MOSSIM does not have to deal with detailed current-voltage relationships. In contrast, RSIM uses resistors and capacitors to form a more accurate, but also more complex, model of a circuit. Although MOSSIM makes major approximations up front, its higher level abstraction allows it to achieve generality through simplicity. MOSSIM's path blocking and state evaluating algorithms are cheap to implement and can accurately model the *abstracted* circuit. On the other hand, RSIM has to deal with a more complex model and must often make approximations at the evaluation stage.

In order to limit its complexity, RSIM only handles tree-like networks. A network is tree-like if it does not contain any closed path (loop) formed by transistors (or resistors). Loops are rare in digital designs, and they are expensive to analyze because a simple parallel-and-series collapsing of resistors can no longer be used. To solve a general resistor network with loops requires finding the inverse of a matrix, which is prohibitively expensive for an effective-resistance based simulator. On the other hand, MOSSIM solves feedbacks by an iterative algorithm which terminates when nodes in a loop stabilized. This process is relatively inexpensive for MOSSIM because only a small set of discrete strengths are involved.

One major drawback of MOSSIM's elegant high-level representation is its lack of timing information. The notion of relative strength is insufficient to specify the exact delay, hence, unit delay is assumed. Since speed is the principal advantage of a fully customized IC, timing information is crucial for most designers. RSIM, on the other hand, carries enough information to be as accurate as the resistor-capacitor modeling can be. As a matter of fact, the RC-modeling approach has even been used by timing verifiers[14,18,21] whose main purposes are to estimate delays and find critical paths.

Unfortunately, in their original implementations, neither MOSSIM nor RSIM is satisfactory in simulating nontrivial circuit structures such as complicated gates, pass-transistor networks, and charge-sharing designs. Although MOSSIM is intrinsically limited by the available information in its abstracted model, RSIM is only limited by its inability in extracting information from RC networks. Since this thesis aims at improving the accuracy of simulation, and the effective-resistance representation has a greater potential for accomplishing this objective, RSIM is chosen to be the groundwork.

2.3 Analysis of RSIM

The major challenge for an effective-resistance based simulator is how to inexpensively, but accurately, extract logic and delay information from RC networks. In order to determine the logical state of a node, the node's DC voltage has to be computed. In order to find the exact delay of a node, the node's waveform has to be derived from a set of differential equations.

The computational complexity of a simulator is further increased by the presence of the unknown logical state, X. A transistor with an X on its gate is called an X transistor, which can be either conducting or nonconducting. Thus, the number of possible electrical configurations that can be derived from a network increases exponentially with the number of X transistors in the network. Different circuit configurations can drive the same node to different final voltages with different delays, and only an exhaustive evaluation can uncover all possible outcomes.

An exhaustive algorithm which requires solving differential equations at each step is extremely expensive. At the expense of sacrificing some accuracy, RSIM uses two schemes to drastically reduce the computational complexity. One scheme is based on an intelligent way of handling X transistors such that not all circuit configurations are evaluated. The other is to use the easy-to-compute empirical model described in Equation 2.1 to approximate the delay time constant.

RSIM manages to avoid exhaustively evaluating X transistors by making conservative approximations on possible outcomes. An approximation is conservative if it is no more optimistic than the worst case scenario created by setting X transistors to all possible conducting and nonconducting combinations. The definition of a "worst case scenario" depends on the subject being examined. Since DC voltage is used to determine the logical state of a node, the worst case scenario is that the node reaches voltage levels which represent different logical states by setting X transistors to different combinations. If a node can have more than one possible state, then the node is considered as logically invalid.

Determining the achievable voltages is a difficult problem, as the example in Figure 2.2 illustrates. The circled part of the figure is a resistor divider in series with a switch in an unknown state. The switch conceptually represents an X transistor. In order to find the maximum achievable voltage at node a, the switch should be considered as OFF. However, if node a is later on merged with a strong pull-down such as node b, then in order to find the maximum achievable voltage at the combined node, the switch should have been considered as ON! In other words, the decision of whether a bridging X transistor should be considered as conducting or nonconducting depends on what it is going to combine with and the information

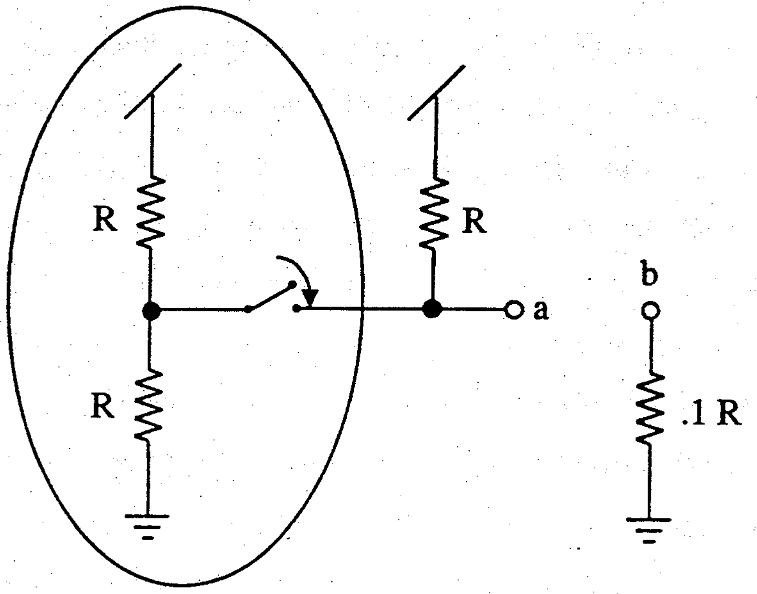


Figure 2.2: Potential complications that can be caused by a switch in an unknown state.

that is being looked for.

RSIM's X-transistor scheme can intelligently handle some X-transistor configurations, but it also generates pessimistic results on some other configurations. A detailed description and analysis of RSIM's scheme can be found in Section 3.3.

In terms of delay modeling, RSIM has been quite accurate for simple transistor clusters. A transistor cluster is a group of nodes that are electrically connected, and it is the smallest unit that RSIM operates on. Generally speaking, the complexities of transistor clusters do not vary with the complexities of designs, and in order to minimize delays, complicated transistor clusters such as pass-transistor networks are often avoided if possible. Thus, most transistor clusters are of the type in which a logic gate drives an output capacitor, and the lumped model described in Equation 2.1 is adequate for these clusters.

However, there are two classes of circuits which the lumped model does not handle well. The first one is distributed RC structures, which include the models of long

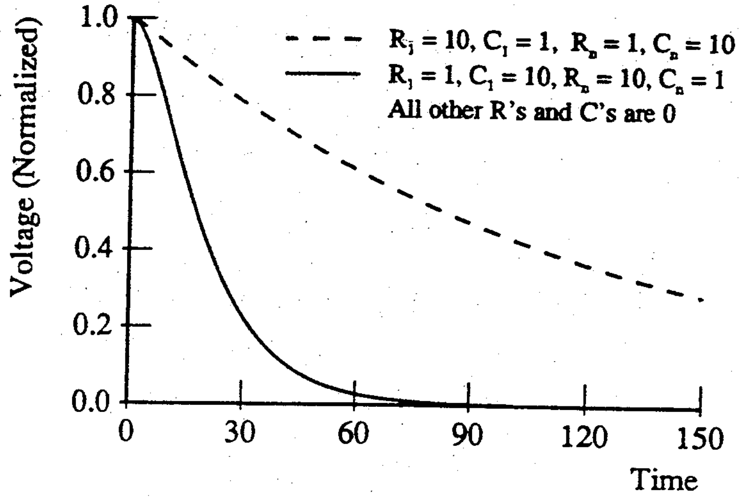


Figure 2.3: Waveforms at node n in Figure 2.1 with different sets of R's and C's.

polysilicon or diffusion lines and pass-transistor networks (which are unavoidable in some designs). With reference to the example shown in Figure 2.1. even without a rigorous definition of delay. the lumped model seems doubtful. The fallacy of the model is that not all capacitors discharge or charge through all resistors. Therefore. a voltage waveform can change significantly just by rearranging the R's and C's in a network while keeping $\sum_k R_k$ and $\sum_k C_k$ in constants. An example is shown in Figure 2.3.

In addition. the lumped model assumes that transistor clusters are always driven by one and only one voltage source, which, unfortunately. is not true. For example, some circuit structures. such as NAND and NOR gates. can have multiple voltage supplies. and the lumped model cannot effectively handle these structures.

The other class of circuits which the lumped model does not handle are those with timing problems caused by the redistribution of charge among capacitors (i.e. charge-sharing problems). Charge sharing is the focus of Chapters 4 and 5. and it will be discussed more extensively there.

Among all the shortcomings that can be potentially improved, only delay modeling has been extensively investigated by other researchers. In the early eighties, a number of researchers came up with more sophisticated timing models for RC networks. Their results are reviewed in the following section.

2.4 Enhancements in Timing Models

Research in timing has been concentrated on RC trees driven by a single voltage source with all capacitors starting at the same voltage. An early analytical result by Elmore[8] is adopted as the definition of delay. Elmore defines the delay of a node to be the first moment of its impulse response. This value is also equal to the centroid of the impulse response in the time domain, which matches the fuzzy notion of what a "delay" should be.

The definition of delay was then extended to approximate waveforms. This led to the single-time-constant model[12,15,19,24]. This model provides an estimate of a waveform, hence, the approximate time required for a node to reach any voltage level can be determined. Bounds of an estimate were also developed to check the accuracy of the approximation. Although waveform bounding is theoretically interesting[28], it is difficult to use in simulators.

When both the single-time-constant model and its bounds match poorly with the real waveform at a node, there is a good chance that the node does not have a dominant time constant. This observation prompted some researchers to experiment on other more complicated timing models. Among them, there is the two-time-constant model[12], which is most helpful in improving the estimates of a small class of unusual waveforms. However, its technique has profound influences in works presented in a later chapter. Both the single-time-constant and the two-time-constant models are summarized here.

2.4.1 Single-Time-Constant Model

The idea behind the single-time-constant model is quite simple. An RC tree is a linear system, and its solution is the sum of exponential functions. For circuits appearing in digital VLSI designs, an output waveform is often dominated by the slowest exponential, which is caused by the lowest-frequency pole. Thus, a single exponential function is a good model for an output waveform. The area under an exponential is equal to its time constant, and it also turns out that for a node in an RC tree, the area under its voltage waveform in the time domain is quite easy to find from the circuit's network topology. Hence, this area, which has the same value as the Elmore's delay, is used as the approximate time constant.

Assume that the root of an RC tree is the ground, and that all the capacitors in the tree are charged high initially. The voltage[3] drop V_e between any node e and the ground can be formulated by collecting the current from each capacitor and adding them together as follows:

$$V_e = \sum_k R_{ke} i_k = -\sum_k R_{ke} C_k \frac{dV_k}{dt}$$

where i_k is the current from the capacitor C_k at node k, and R_{ke} is the resistance of the path to the ground shared by both node k and node e. The area of V_e in the time domain is given by

$$\int_0^\infty V_e \, dt = \sum_k R_{ke} C_k \overset{\text{def}}{=} \tau_{D_e}.$$

Figure 2.4 shows a simple RC tree where nodes a, b, and c are charged high initially. The exact waveform at node b is plotted against the single-time-constant model in Figure 2.5.

Unfortunately, not all waveforms can be modeled successfully with a single time constant. The example in Figure 2.6 is derived from the one in Figure 2.4 by

[3]All voltages are normalized to range between 0 and 1 unless specified otherwise.

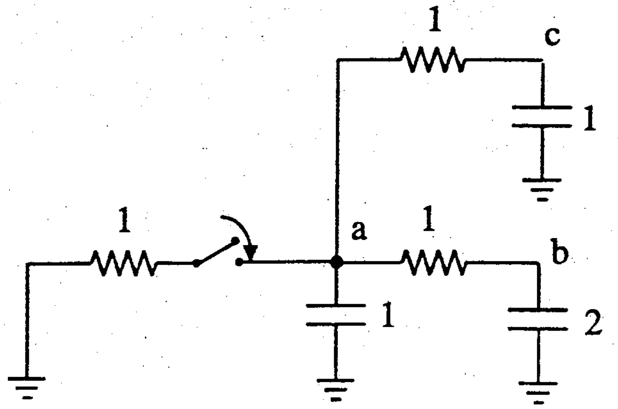


Figure 2.4: A simple RC tree.

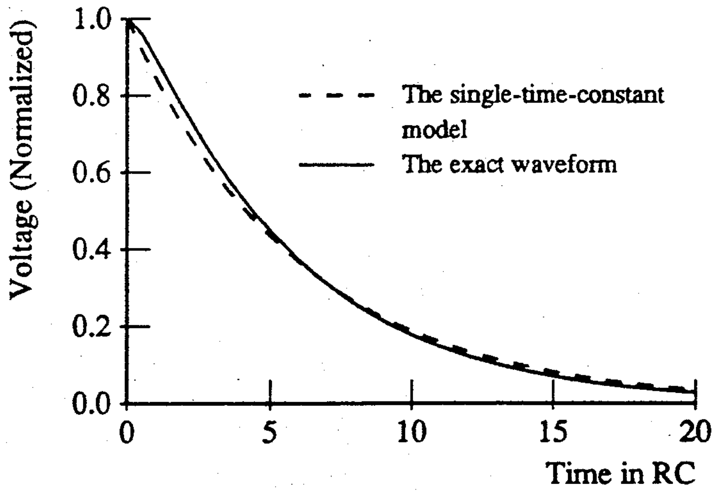


Figure 2.5: Voltage waveform of node b in Figure 2.4.

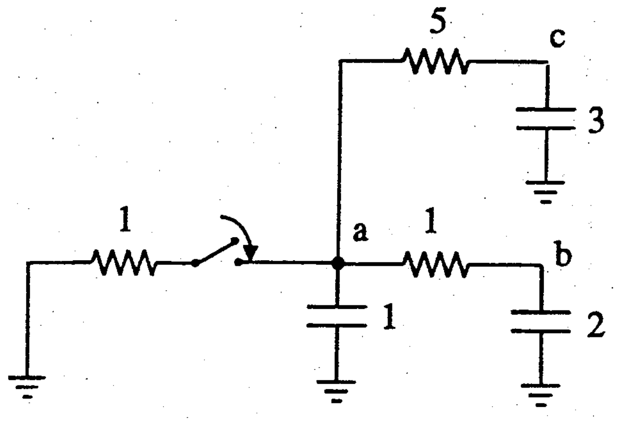


Figure 2.6: Node b in this network has a low-frequency pole-zero pair.

increasing the capacitance at node c and the resistance between node a and node c. The voltage at node b initially falls at its own rate; however, the rate of decay is eventually controlled by the dominant time constant set by the capacitor at node c. In frequency domain, this circuit has a low-frequency pole-zero pair, and the low-frequency zero partially cancels the dominant pole. This causes the output to have a two-time-constant behavior. As shown in Figure 2.7, although the areas below the single-time-constant model and exact waveform are still the same, the approximation is not good for most regions.

2.4.2 Two-Time-Constant Model

Horowitz[12] proposed a model with a slow and a fast component for this class of networks. His model approximates a node's *network transfer function*,[4] $H(s)$, which is defined as Laplace transform of the derivative of the node's voltage waveform (its step response) by

$$H(s) \approx \frac{k(1+s\tau_z)}{(1+s\tau_1)(1+s\tau_2)} = k(1+(\tau_z-\tau_P)s+\tau_P(\tau_P-\tau_z-\tau_{M\epsilon})s^2+\cdots) \quad (2.2)$$

[4]The only difference between a network transfer function and a network function is that the former can have arbitrary initial state while the latter is a zero-state response.

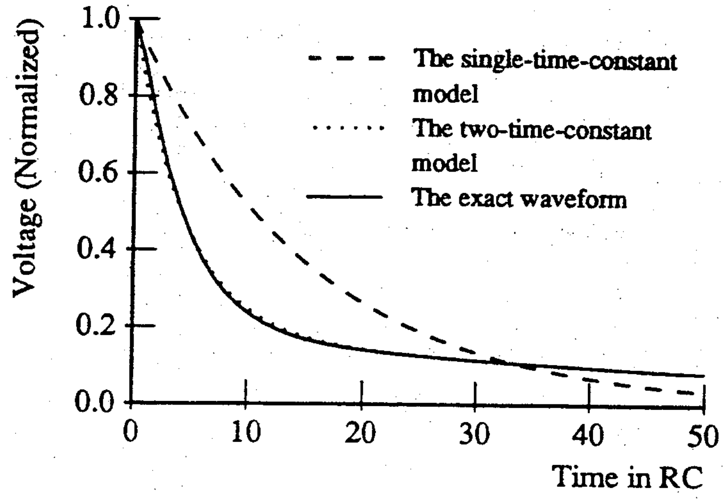


Figure 2.7: Voltage waveform of node b in Figure 2.6.

where $\tau_P = \tau_1 + \tau_2$ and $\tau_{M\epsilon} = \tau_1\tau_2/\tau_P$. The magnitude of k is equal to the product of all zeros in $H(s)$ divided by the product of all poles in $H(s)$. The model also approximates τ_P by the sum of all τ's in the original system, which is equal to $\sum_k R_{kk}C_k$. The network transfer function which the model tries to approximate can be derived from its circuit description by finding the moments of the real waveform as follows:

$$
\begin{aligned}
H(s) &= \int_0^\infty h_\epsilon \epsilon^{-st}\, dt \\
&= \int_0^\infty h_\epsilon\, dt - s\int_0^\infty t h_\epsilon\, dt + \frac{s^2}{2}\int_0^\infty t^2 h_\epsilon\, dt + \cdots \\
&= \int dV_\epsilon + s\int_0^\infty V_\epsilon\, dt - s^2\int_0^\infty tV_\epsilon\, dt + \cdots \\
&= \int dV_\epsilon - s\sum_k \left(R_{k\epsilon}C_k \int_0^\infty \frac{dV_k}{dt}\, dt\right) + s^2 \sum_k \left(R_{k\epsilon}C_k \int_0^\infty t\frac{dV_k}{dt}\, dt\right) + \cdots \\
&= -1 + s\tau_{D_\epsilon} - s^2 \sum_k R_{k\epsilon}C_k\tau_{D_k} + \cdots
\end{aligned} \qquad (2.3)
$$

where h_ϵ is equal to the derivative of V_ϵ.

The idea behind modeling a network transfer function by a two-pole-one-zero system is to match the boundary conditions at time zero and infinity as well as the area and the first moment of the estimate with that of the real voltage waveform. Similar to that of a Taylor's series expansion, the accuracy of an estimated waveform improves with the orders of matching moments, but the amount of computation also increases accordingly. By matching the coefficients of the first three terms in Equations 2.2 and 2.3, the time constants τ_1 and τ_2 can be uniquely determined. In Figure 2.7, the output of this two-time-constant model is compared with other models.

Since fan-outs in VLSI designs usually have roughly the same timing, circuits with low-frequency pole-zero pairs are rather rare. As a result, this model has not been widely used. Nevertheless, the technique of modeling by matching the terms of an expanded network transfer function can be applied to other unexplored timing and glitch-detection areas.

2.5 Summary

Switch-level simulators balance the accuracy of circuit-level simulators with the speed of logic-level simulators. They are accurate enough to simulate pure digital designs but fast enough for large networks. The order-of-magnitude scheme and the effective-resistance scheme are the two most widely used switch-level modeling techniques. The former scheme transforms transistors and capacitors to devices with discrete strengths. In contrast, the latter scheme transforms a transistor network to an RC network. Between these two schemes, the effective-resistance scheme is more suitable for timing and other detailed analysis.

The switch-level simulator RSIM follows the effective-resistance approach, but has many shortcomings. These shortcomings are caused by difficulties in extracting information from an RC network. Although delay modeling has been investigated by many researchers, there are still timing issues which need attentions. Most notably, the existing timing models assume that circuits are driven by a single voltage source; hence, they can neither model circuits without a driver nor model circuits with multiple drivers.

X transistors also cause problems to switch-level simulators because of the uncertainties of their states. RSIM tends to handle X transistors too conservatively, and produces pessimistic results. These issues are explored in the following chapters.

Chapter 3

Final-Value Computation

3.1 Overview

RSIM models a digital circuit as having three logical states — 1 (high), 0 (low), and X (invalid). It determines a node's logical state by first estimating its voltage and then comparing the voltage with two thresholds: if the voltage is higher than the *high threshold*, then it is considered as logical high; if the voltage is lower than the *low threshold*, then it is considered as logical low; if the voltage is in between the two thresholds, then it is considered as invalid. Theoretically, there can be a fourth state for simulators, which is usually referred to as "valid-but-unknown" state. For example, although the two outputs of an *R-S* latch must have opposite polarities, their states cannot be determined during initialization. In this chapter, a valid-but-unknown state is treated the same as an invalid state.

The two principal considerations in computing DC voltages at the switch level are accuracy and complexity. In terms of accuracy, design flaws such as drive fight and ratio error which result in invalidated outputs must be caught. Although simulators should err on the conservative side such that real design errors do not slip through, indiscriminately pessimistic results tend to discourage a designer because invalid states spread quickly in simulators. The spreading of invalid states has been examined by Breuer[1]. The fundamental problem is that ternary logic does not obey the Law of Excluded Middle ($Q + \overline{Q} = 1$) as digital circuits do. consequently, cross-coupled structures which rely on this property to settle valid-but-unknown states cannot be handled correctly. The dilemma is how to catch real design flaws without invalidating nodes which are otherwise valid. In terms of complexity, the method chosen should be computationally inexpensive such that simulation at the VLSI level can be done interactively if so desired.

While Kirchhoff's current law and voltage law provide a systematic way in analyzing any lumped electric circuit. their application to circuits in the switch-level representation is complicated by X transistors. Although all achievable voltages at a node can be found by evaluating X transistors exhaustively (by setting them to combinations of conducting and nonconducting states), it is both expensive and unnecessary to do so. For digital circuits. the logical state of a node is only determined by its minimum and maximum achievable voltages: hence, instead of solving for all achievable voltages, a simulator only has to find the achievable voltage range.

Yet, determining the achievable voltage range is still a nontrivial task, as the example in Figure 2.2 has already shown. In addition, although voltage range alone dictates the logical state of a node, combining two or more nodes requires the nodes' resistance information. Since the effective resistance associated with a node depends on how X transistors are set, it can also have multiple values. These values can be bounded by a resistance range. Voltage range and resistance range are closely related, and their relationship is best visualized in a two-dimensional V_{eq}-R_{eq} (voltage-resistance) plane. This technique is first documented by Terman[26], and it is reviewed in the next section.

3.2 Solution Space Representation

Thévenin's theorem states that any two-terminal network of resistors and voltage sources can be represented by an equivalent voltage source (V_{eq}) in series with an equivalent resistor (R_{eq}). As mentioned in the previous section, if some of the resistors in the network do not have fixed values, then the output of the network can be represented by a collection of Thévenin equivalents. The collection can be represented in a plot with its X axis being V_{eq} and its Y axis being R_{eq}.

A simple resistor divider shown in Figure 3.1 (a) illustrates the construction of such a plot. To start with, assume that $R_{U_l} = R_{U_h} = R_U$ and that $R_{D_l} = R_{D_h} = R_D$. In this case, the effective output resistance and the effective output voltage are equal to $R_U \parallel R_D$ and $R_D/(R_U + R_D)$ respectively. This Thévenin equivalent can be specified in the V_{eq}-R_{eq} plane by a point. The relationship between the effective output resistance and the effective output voltage, as a function of R_U and R_D, is:

$$R_{eq}(R_U, R_D) = R_U V_{eq}(R_U, R_D) = R_D(1 - V_{eq}(R_U, R_D)).$$

In the resistor divider, if the pull-down resistor varies between 0 and ∞ (that is $R_{D_l} = 0$ and $R_{D_h} = \infty$) while holding the pull-up resistor in constant at R_U, then all possible V_{eq}-R_{eq} pairs (Thévenin equivalents) form a segment of the line $R_{eq}(R_U, r) = R_U V_{eq}(R_U, r)$ where $0 \leq r \leq \infty$. This segment resides in the first quadrant of the V_{eq}-R_{eq} plane, has slope R_U, and terminates at $(0, 0)$ and $(1, R_U)$. In contrast, if the pull-up resistor has the range $[0, \infty]$ while the pull-down resistor is held at constant R_D, then all V_{eq}-R_{eq} pairs form the first-quadrant segment of the line $R_{eq}(r, R_D) = R_D(1 - V_{eq}(r, R_D))$ terminating at $(0, R_D)$ and $(1, 0)$.

If neither the pull-up nor the pull-down resistor is at constant, then the plot of all V_{eq}-R_{eq} pairs forms a diamond-shaped quadrilateral; see Figure 3.1 (a) and (b). A systematic way to construct this solution space is to vary the pull-up resistance between 0 and ∞ with the pull-down resistance equal to R_{D_l}, and then to repeat the

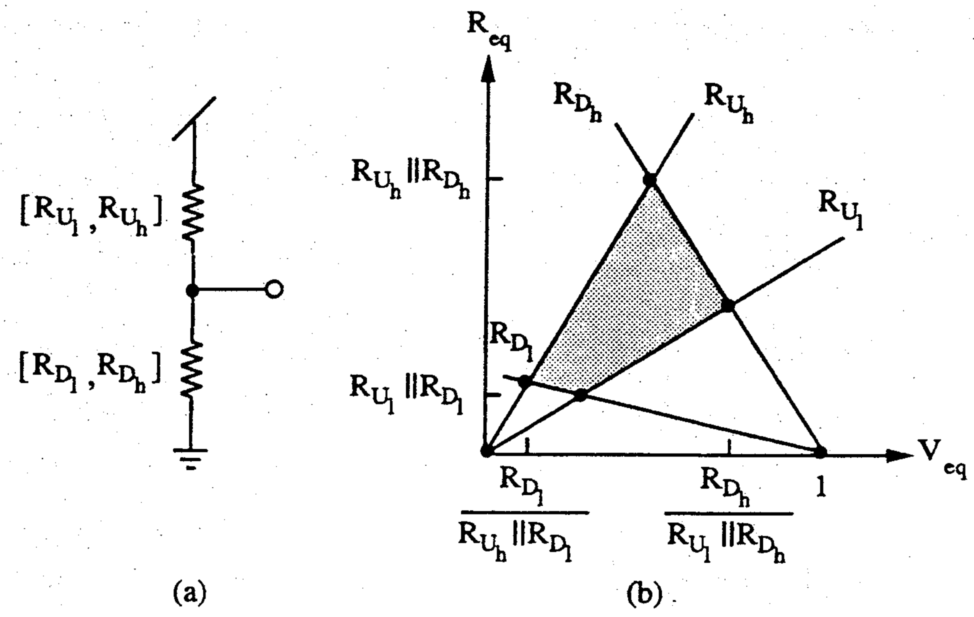
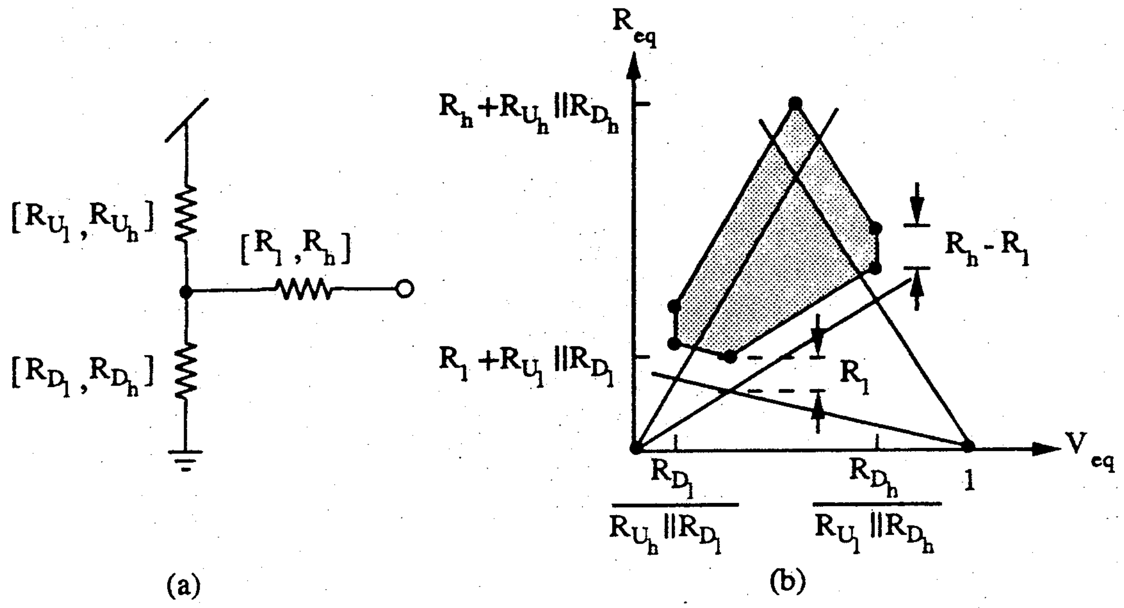


Figure 3.1: A resistor divider and its output solution space.

operation with the pull-down resistance equal to R_{D_h}. Thévenin equivalents formed by the operations can be described by two straight lines in the V_{eq}-R_{eq} plane. The region in the first quadrant between these two lines represents all possible output V_{eq}-R_{eq} combinations for the resistor divider with its pull-down resistance between $[R_{D_l}, R_{D_h}]$ and any positive pull-up resistance. By reversing the roles of R_U and R_D in the above process, one can calculate all V_{eq}-R_{eq} combinations for the resistor divider with its pull-up resistance between $[R_{U_l}, R_{U_h}]$ and any positive pull-down resistance. These combinations form another triangular region in the first quadrant of the V_{eq}-R_{eq} plane. The solution space is the intersection of these two regions.

Although a diamond-shaped quadrilateral can be expressed quite easily with four parameters, the solution space of a more complicated circuit can be highly irregular. With reference to the circuit shown in Figure 3.2 (a), since the series resistor only affects the output impedance, the circuit's output voltage range is the same as that of Figure 3.1 (a). Its solution space can be visualized by shifting



$[R_{U_l}, R_{U_h}]$
$[R_l, R_h]$
$[R_{D_l}, R_{D_h}]$
(a)
R_{eq}
$R_h + R_{U_h} \| R_{D_h}$
$R_l + R_{U_l} \| R_{D_l}$
$R_h - R_l$
R_l
V_{eq}
$\frac{R_{D_l}}{R_{U_h} \| R_{D_l}}$
$\frac{R_{D_h}}{R_{U_l} \| R_{D_h}}$
1
(b)

Figure 3.2: A resistor divider in series with a resistor and its output solution space.

the shaded area in Figure 3.1 (b) R_l units along the R_{eq} axis and stretching the upper boundary by another $R_h - R_l$ units. The result is graphically illustrated in Figure 3.2 (b).

Several seemingly reasonable final-value computation schemes are actually heuristics to handle complicated solution spaces. However, only a systematic analysis can explore the limitations and shortcomings of these heuristics. The following sections examine two methods of approximating complicated solution spaces.

3.3 Box Method (RSIM)

The implementation which comes with the original distribution of RSIM approximates the solution space of a circuit by a rectangular-shaped area in the V_{eq}-R_{eq} plane: see Figure 3.3 (a). Due to its highly regular shape, a solution space can be specified by two sets of *mutually independent* parameters $[R_l, R_h]$ and $[V_l, V_h]$. The

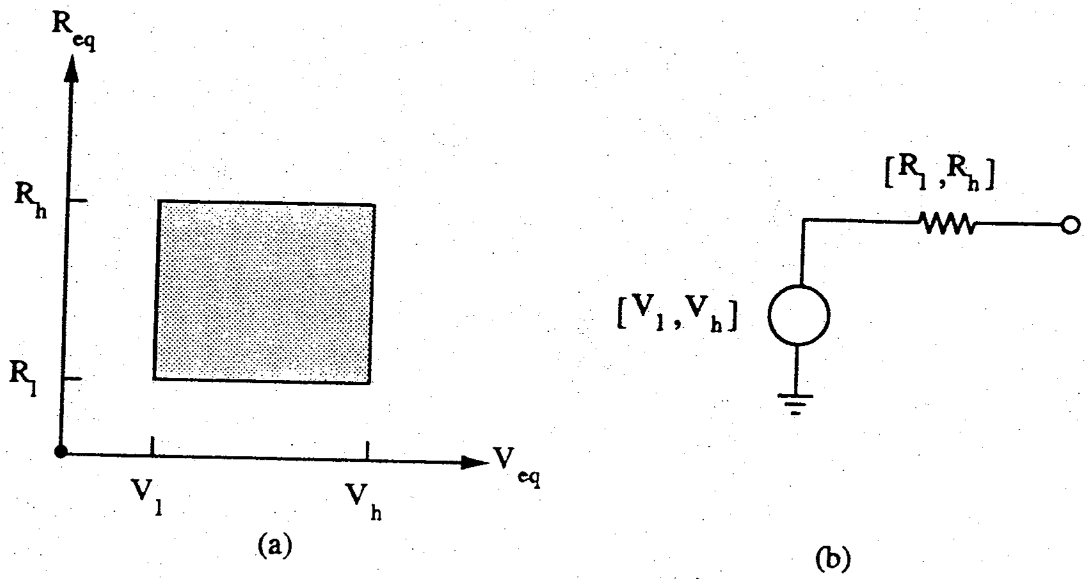


Figure 3.3: A rectangular solution space and its equivalent circuit.

notation $box(R_l, R_h, V_l, V_h)$ will be used to denote the above solution space. A box (a rectangular solution space) can be physically realized by a voltage range in series with a resistor range: see Figure 3.3 (b).

Boxes are basic building blocks in RSIM's DC-computation scheme. In order to find the solution space on one end of a resistor R when the other end is connected to the equivalent circuit of a box, one only has to transform the box R units up along the R_{eq} axis. Connecting the equivalent circuits of $box(R_{1_l}, R_{1_h}, V_{1_l}, V_{1_h})$ and $box(R_{2_l}, R_{2_h}, V_{2_l}, V_{2_h})$ in parallel yields a hexagonal solution space as illustrated by an example in Figure 3.4. Since only rectangular solution spaces are allowed in the box method, a hexagonal solution space is approximated by its bounding rectangle $box(R_l, R_h, V_l, V_h)$ where R_l, R_h, V_l, and V_h are determined as follows.

Since the effective output resistance is equal to the resistance of its component resistors in parallel, the minimum and the maximum output resistances are

$$R_l = R_{1_l} \parallel R_{2_l} \quad \text{and} \quad R_h = R_{1_h} \parallel R_{2_h}.$$

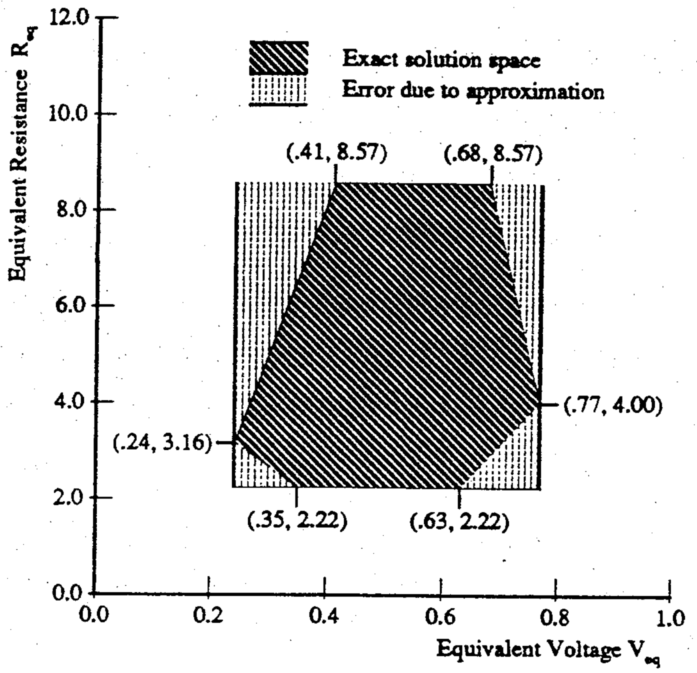


Figure 3.4: The exact solution space on top of the approximate solution space formed by concatenating the equivalent circuits of $box(4.0, 20.0, 0.15, 0.45)$ and $box(5.0, 15.0, 0.6, 0.85)$.

The effective output voltage is determined by the resistor divider formed between the voltage sources. To calculate its minimum value, one should set both voltage sources to their minimals and adjust the variable resistors such that the one which is associated with the voltage source with the lower voltage is in its minimal while the other resistor is in its maximal. The opposite applies to finding the maximum output voltage. In the above example, V_l and V_h are set by

$$\text{if } (V_{1_l} < V_{2_l})\ V_l = \frac{V_{1_l}R_{2_h} + V_{2_l}R_{1_l}}{R_{1_l} + R_{2_h}} \text{ else } V_l = \frac{V_{2_l}R_{1_h} + V_{1_l}R_{2_l}}{R_{1_h} + R_{2_l}};$$

$$\text{if } (V_{1_h} > V_{2_h})\ V_h = \frac{V_{1_h}R_{2_h} + V_{2_h}R_{1_l}}{R_{1_l} + R_{2_h}} \text{ else } V_h = \frac{V_{2_h}R_{1_h} + V_{1_h}R_{2_l}}{R_{1_h} + R_{2_l}}.$$

The box method is seemingly reasonable because it is conceptually familiar — the equivalent circuit of a box resembles a Thévenin equivalent. One advantage of the method is that the result is guaranteed to be conservative. Any operation on a box, be it series connection with a resistor or parallel concatenation with another box, generates a solution space which is either equal to or forms a superset of the exact solution space; therefore, any V_{eq}-R_{eq} combination at a node must belong to the solution space computed by the box method.

Unfortunately, the conservative merit also brings undesirable pessimism to the box method's results. For example, as shown previously, a hexagonal solution space has to be approximated by its bounding rectangle. When the equivalent circuit of such an approximate solution space is concatenated with that of another box, the box method uses the boxes' boundary information to compute the result. In other words, once an error or an approximation is introduced, it propagates and amplifies through operations between boxes. By building error upon error at places where they can cause the most damage, it is possible for the solution space approximated by the box method to have a much wider voltage range than that of the real solution space. Consider the example of connecting the equivalent circuits of $box(R_1, R_1, 0, 0)$, $box(0, \infty, V, V)$, and $box(R_3, R_3, 1, 1)$, which is shown graphically in Figure 3.5. Let the value of V be $R_1/(R_1 + R_3)$ such that the output voltage is equal to V regardless of the effective resistance of the second box. In this example, the box method does not always provide a good approximation for the output. For instance, if the first box is first combined with the second box, then with the third box, the approximate voltage range is shown as a function of V in Figure 3.6. Although the exact output voltage is equal to V, the approximate voltage range is equal to $[0, 2V - V^2]$, which grows wider with V. As a result, the box method signals the output to be invalid whenever $2V - V^2$ is greater than the low threshold; in reality, the output is invalid only if V lies between the low threshold and the high threshold.

The same example also illustrates another problem with the box method — its result is inconsistent with respect to the order of evaluation. For example, if the output voltage is computed by combining the third box with the second box and

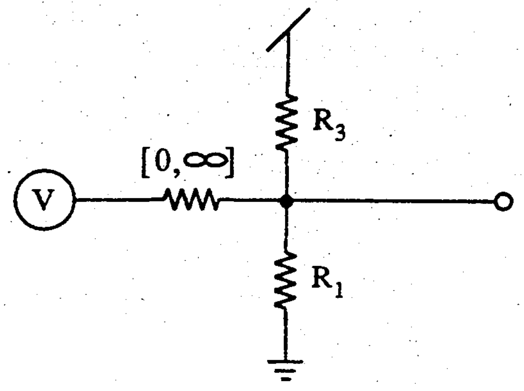


Figure 3.5: Concatenating the equivalent circuits of $box(R_1, R_1, 0, 0)$, $box(0, \infty, V, V)$, and $box(R_3, R_3, 1, 1)$.

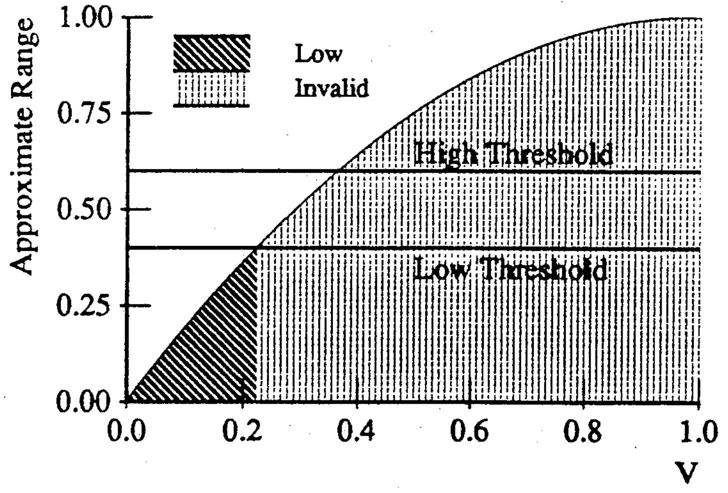


Figure 3.6: Approximate voltage range as a function of V if $box(R_1, R_1, 0, 0)$ is first combined with $box(0, \infty, V, V)$ then with $box(R_3, R_3, 1, 1)$.

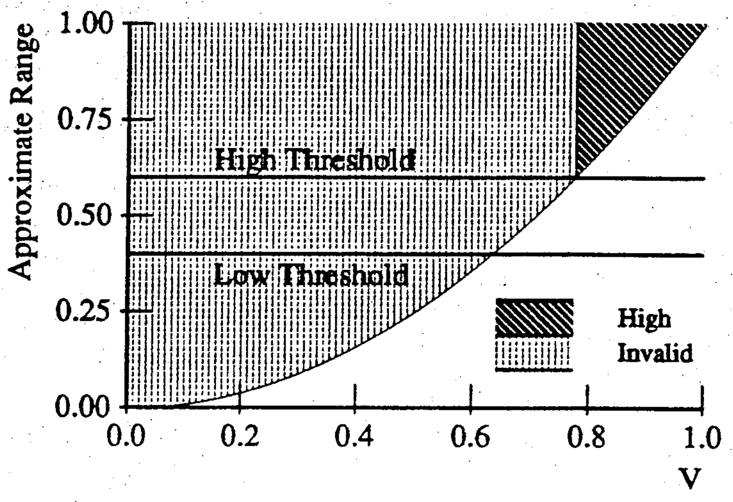


Figure 3.7: Approximate voltage range as a function of V if $box(R_3, R_3, 1, 1)$ is first combined with $box(0, \infty, V, V)$ then with $box(R_1, R_1, 0, 0)$.

then with the first box, then the approximate voltage range becomes $[V^2, 1]$, which is plotted as a function of V in Figure 3.7. Although either ordering guarantees a conservative approximation, the two results are quite different, and neither is very good.

3.4 Diamond Method

A different heuristic to approximate the solution space has been proposed by Terman in [26]. Terman suggests using resistor dividers same as the one shown in Figure 3.1 (a) as basic building blocks. As explained in Section 3.2, a diamond-shaped quadrilateral can be specified by four parameters: R_{U_l}, R_{U_h} (the minimum and the maximum pull-up resistances), R_{D_l}, and R_{D_h} (the minimum and the maximum pull-down resistances). The notation $diamond(R_{U_l}, R_{U_h}, R_{D_l}, R_{D_h})$ will be used to denote such a region.

Since all resistor dividers in this method are defined between the same power supply and the ground. connecting the outputs of two resistor dividers is straightforward. When two resistor dividers with solution spaces *diamond*(P_{1_l}. P_{1_h}. Q_{1_l}. Q_{1_h}) and *diamond*(P_{2_l}. P_{2_h}. Q_{2_l}. Q_{2_h}) are connected. the overall pull-up resistance must be within $[P_{1_l} \parallel P_{2_l}, P_{1_h} \parallel P_{2_h}]$ (similarly for the overall pull-down resistance). These two ranges are sufficient to specify a resistor divider with the solution space *diamond*($P_{1_l} \parallel P_{2_l}$, $P_{1_h} \parallel P_{2_h}$, $Q_{1_l} \parallel Q_{2_l}$, $Q_{1_h} \parallel Q_{2_h}$). Unlike the box method. no approximation is made in this step.

On the other hand. the solution space formed by connecting a resistor R in series with a resistor divider *diamond*(R_{U_l}. R_{U_h}. R_{D_l}. R_{D_h}) is much more complicated. It can be constructed from that of Figure 3.1 (b) by shifting the shaded region R units up along the R_{eq} axis. Although the result is still a diamond-shaped quadrilateral. no resistor divider of the kind shown in Figure 3.1 (a) can produce such a solution space. This result is unacceptable to the diamond method because the new solution space can no longer be merged with other resistor dividers using the previous algorithm. Hence. the diamond method must construct a resistor divider which approximates this solution space. Terman reasons that the most important part of a solution space is its voltage range. thus it is not compromised. This constraint fixes the left and the right vertices of a diamond-shaped area in the V_{eq}-R_{eq} plane. He also decides to preserve the lowest effective resistance seen at the output because it seems to be more prudent to overestimate than to underestimate resistances. These constraints uniquely define a resistor divider with the solution space *diamond*(A_l, A_h, B_l, B_h) where

$$A_l = R_{U_l} + R + R\frac{R_{U_l}}{R_{D_l}} \qquad A_h = R_{U_h} + R\frac{R_{U_h}}{R_{U_l}} + R\frac{R_{U_h}}{R_{D_l}}$$

$$B_l = R_{D_l} + R + R\frac{R_{D_l}}{R_{U_l}} \qquad B_h = R_{D_h} + R\frac{R_{D_h}}{R_{D_l}} + R\frac{R_{D_h}}{R_{U_l}}.$$

The exact and the approximate solution spaces are illustrated in Figure 3.8. Since the exact solution space is not a subset of the approximate solution space, this

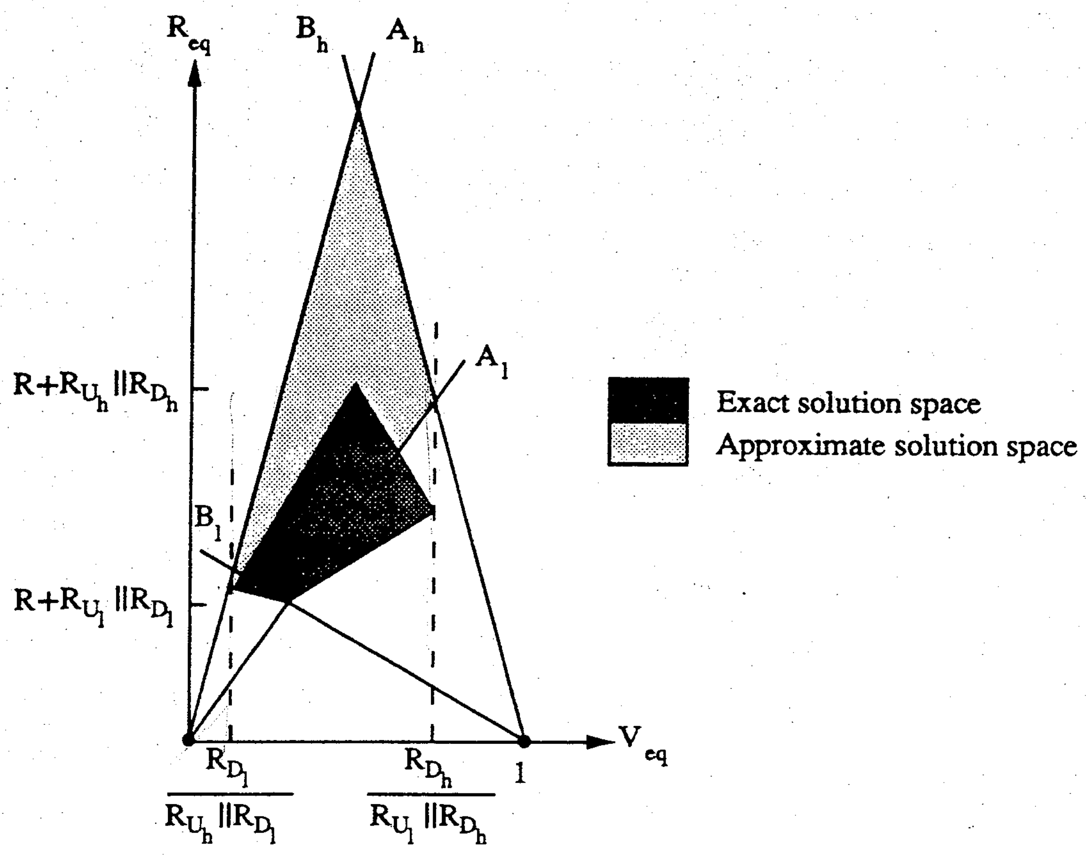


Figure 3.8: The exact solution space and the diamond method's approximation.

approximation is not conservative.

While the box method can overestimate as well as underestimate the effective resistances associated with the corners of a solution space, the diamond method never underestimates them. This is best illustrated by comparing the solution spaces in Figures 3.4 and 3.8. Since the corners of a solution space define the interface with other solution spaces, the box method is conceivably more vulnerable to error because it tends to exaggerate the strength of a stronger driver while understate the strength of a weaker one. In this context, a stronger driver is the one with the higher voltage when computing the maximum output voltage (and vice versa for a weaker driver).

The diamond method is also more consistent than the box method with respect to the order of evaluation. Approximations are introduced only at places where they are intrinsically order independent (i.e. series connections). Despite its conceptual complexity, the computation and memory requirements of this scheme are no worse than the box method.

Unfortunately, this scheme has one major drawback. As hinted in Figure 3.2 (b), a solution space can lose its diamond shape due to a series X transistor. However, it is beyond the diamond method's ability to take the upper bound of a series resistor range into consideration, hence, an X transistor is modeled indifferently from an 1 transistor! This inadequacy negates all the potential usefulness of this method; it is not even clear how any nontrivial diamond-shaped solution space can be formed without X transistors.

3.5 Improved Resistor-Divider Model

The major limitation of the diamond method can be removed by extending the set of basic elements used to model circuits. A new scheme is proposed here. This scheme uses four kinds of basic building blocks: resistor, resistor range, definite block, and indefinite block. A resistor is used to approximate the conductivity of a transistor when it is ON. Similarly, an X transistor is substituted by a resistor range with its bounds set by the equivalent resistance of the X transistor when it is ON and infinity.

A definite block is defined to be a resistor divider same as the one shown in Figure 3.1 (a) with the constraint that R_{U_h} and R_{D_h} cannot be infinite at the same time. Hence, the output of a definite block is guaranteed to be driven. The circuit shown in Figure 3.9 (a) is defined to be an indefinite block. Its solution space is shown in Figure 3.9 (b). There is no constraint on R_{U_h} and R_{D_h} of an indefinite

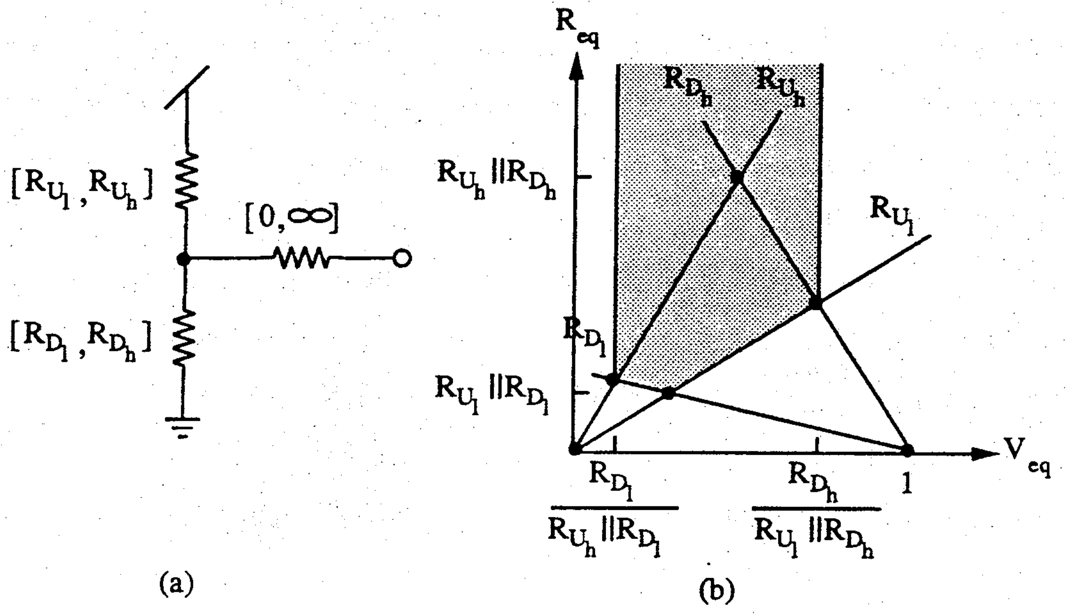


Figure 3.9: An indefinite block and its solution space.

block; thus, a high impedance node is also an indefinite block. Other than the high impedance case (whose output is nondriven), the output of an indefinite block may or may not be driven because the series resistor range is conceptually equivalent to a switch in an unknown state. Definite and indefinite blocks such as those introduced above are referred to as *definite*$(R_{U_l}, R_{U_h}, R_{D_l}, R_{D_h})$ and *indefinite*$(R_{U_l}, R_{U_h}, R_{D_l}, R_{D_h})$ respectively.

Two operators, *series operator*$(+)$ and *parallel operator*$(\|)$, will be defined on the four basic building blocks. Series operator operates between the output terminal of a definite or an indefinite block and one of the terminals of a resistor or a resistor range. The four possible combinations under this operation and their corresponding output types are shown in Figure 3.10. Parallel operator links the outputs of two definite and/or indefinite blocks. There are three possible combinations, and they are shown in Figure 3.11. Series and parallel operators are defined to generate definite and indefinite blocks only; thus the four basic building blocks are closed

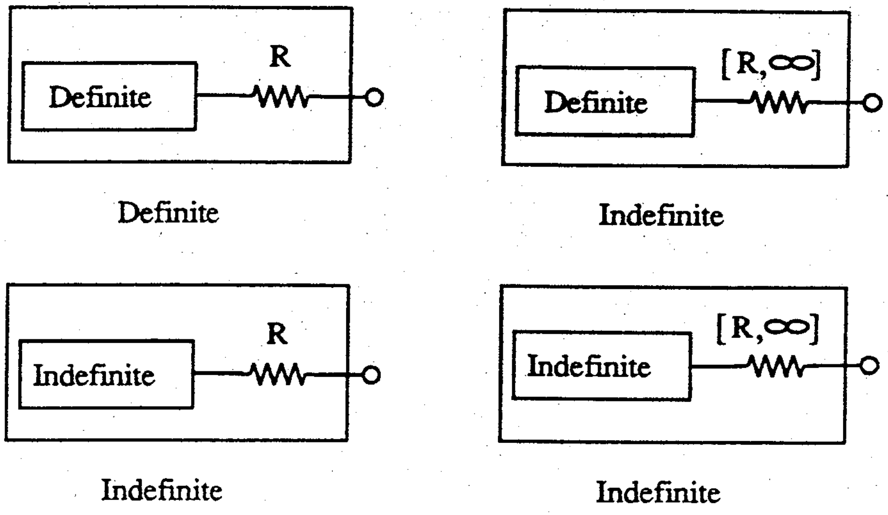


Figure 3.10: Domain and output type of the series operation.

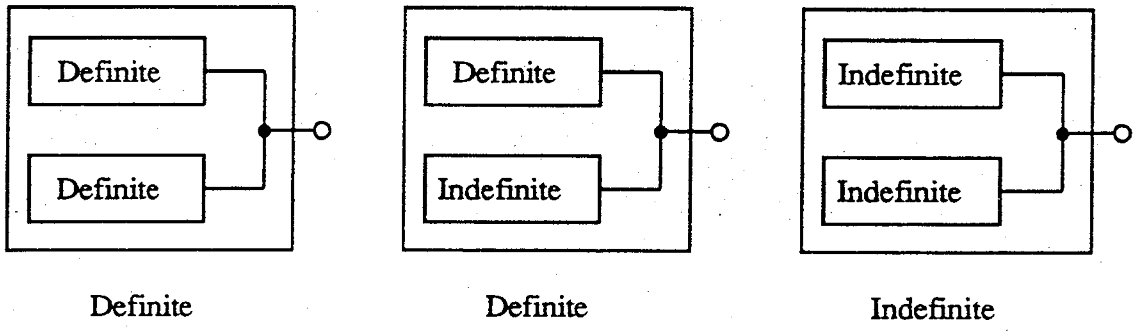


Figure 3.11: Domain and output type of the parallel operation.

under the two operators. This property is introduced such that a complicated circuit can be built or analyzed from the fundamentals.

3.5.1 Series Operator

Ideally, the series operator would find the exact equivalent, in the form of either a definite block or an indefinite block, for the unconnected terminal of the resistive element. However, it has been shown in Figure 3.8 that even for the simplest case of a definite block in series with a resistor, the result does not have a definite-block or an indefinite-block equivalent. In general, none of the combinations shown in Figure 3.10 has either a definite-block or an indefinite-block equivalent. In order to satisfy the property that basic building blocks are closed under the series operator. the operator is defined as follows: it produces a definite or an indefinite block. depending on whether the output is guaranteed to be driven, whose minimum and maximum output voltages and their corresponding minimum resistances match the exact solution space. This definition is different from the one used by the diamond method in that the diamond method matches the minimum and the maximum voltages and the *overall* minimum equivalent resistance seen at the output. The new definition is made to eliminate errors at the left and right corners of a solution space: after all. the maximum range of the voltage is the subject of this study. Of the four cases shown in Figure 3.10. three of them generate indefinite blocks because their outputs are not guaranteed to be driven: only the series connection between a definite block and a resistor produces a definite block. The latter combination is illustrated with an example in Figure 3.12. The series operator is summerized as follows:

$$\begin{aligned}
\mathit{definite}(R_{U_l}, R_{U_h}, R_{D_l}, R_{D_h}) + R &= \mathit{definite}(A_l, A_h, B_l, B_h), \\
\mathit{indefinite}(R_{U_l}, R_{U_h}, R_{D_l}, R_{D_h}) + R &= \mathit{indefinite}(A_l, A_h, B_l, B_h), \\
\mathit{definite}(R_{U_l}, R_{U_h}, R_{D_l}, R_{D_h}) + [R, \infty] &= \mathit{indefinite}(A_l, A_h, B_l, B_h), \\
\mathit{indefinite}(R_{U_l}, R_{U_h}, R_{D_l}, R_{D_h}) + [R, \infty] &= \mathit{indefinite}(A_l, A_h, B_l, B_h) \qquad (3.1)
\end{aligned}$$

where

$$A_l = R_{U_l} + R + R\frac{R_{U_l}}{R_{D_h}}, \quad A_h = R_{U_h} + R + R\frac{R_{U_h}}{R_{D_l}},$$

$$B_l = R_{D_l} + R + R\frac{R_{D_l}}{R_{U_h}}, \quad B_h = R_{D_h} + R + R\frac{R_{D_h}}{R_{U_l}}.$$

This operation is guaranteed to be conservative because the approximate solution space is always a superset of the real solution space.

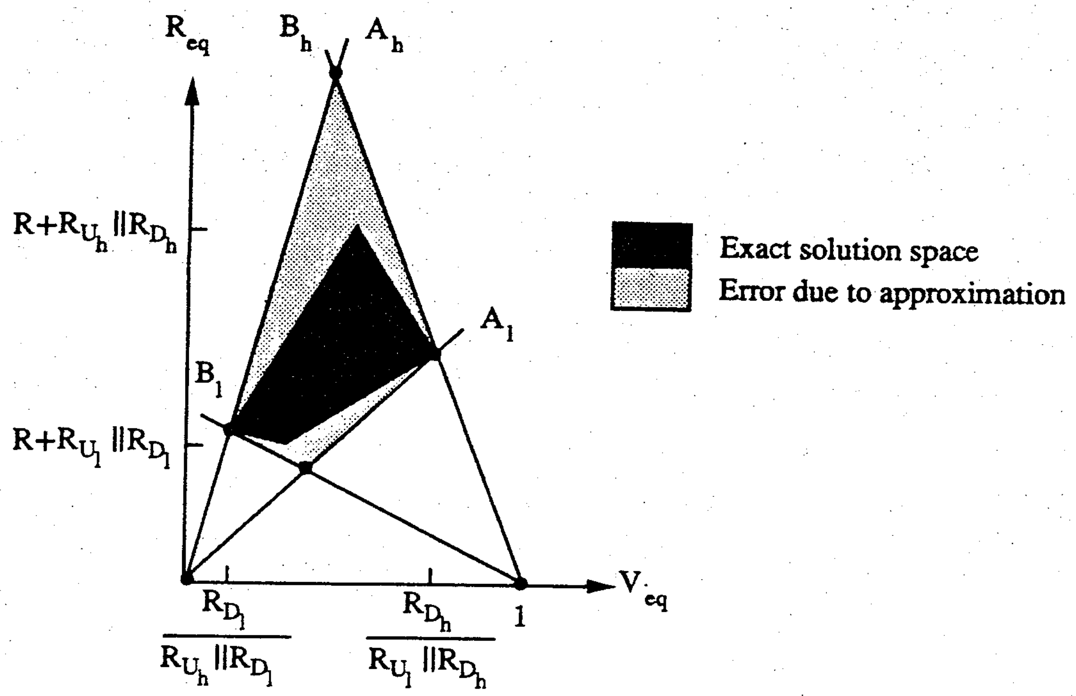


Figure 3.12: The exact solution space on top of the solution space defined by the series operator at the output of a resistor R when the other terminal of the resistor is connected to a definite block *definite*$(R_{U_l}, R_{U_h}, R_{D_l}, R_{D_h})$.

Error introduced in the series operation does little damage, if any, to the accuracy of the overall result due to a subtle but important property of the solution space: if all resistor dividers are defined between the same power supply and ground (in other words, only definite and indefinite blocks are allowed), then the solution space of a definite block can be bounded by its left and right corners. This is because the right corner of a definite block's solution space has a lower pull-up resistance but a higher pull-down resistance than any other point in the solution space. Hence, when the definite block is combined with any resistor divider between the same power supply and ground, this combination is guaranteed to minimize the combined pull-up resistance and maximize the combined pull-down resistance; as a result, it produces the highest combined voltage. Consequently, this combination is labeled as the ***strongest pull-up combination*** of the definite block. Similarly, the left corner has the highest pull-up resistance but the lowest pull-down resistance, and is labeled as the ***strongest pull-down combination.*** The analysis can also be applied to the solution space of an indefinite block because an indefinite block is basically a definite block in series with a switch.

With reference to Figure 3.12, the approximate solution space, according to the above argument, is bounded by the left and right corners of its solution space. At the same time, these corners also match the left and right corners of the exact solution space. Thus, the approximate solution space represents the smallest region which is still conservative and is realizable by either a definite block or an indefinite block.

3.5.2 Parallel Operator

The parallel operator would ideally provide an exact equivalent for those parallel connections depicted in Figure 3.11. Yet it suffers the same problem as the series operator — an exact equivalent is not always realizable by either a definite block or an indefinite block. Approximations targeted at minimizing errors at the left and right corners of a solution space will be introduced to define the parallel operator.

The parallel connection between two definite blocks, $\mathit{definite}(P_{1_l}, P_{1_h}, Q_{1_l}, Q_{1_h})$ and $\mathit{definite}(P_{2_l}, P_{2_h}, Q_{2_l}, Q_{2_h})$, has been worked out in Section 3.4. Its result is restated in the following expression:

$$\begin{aligned}\mathit{definite}(P_{1_l}, P_{1_h}, Q_{1_l}, Q_{1_h}) \parallel \mathit{definite}(P_{2_l}, P_{2_h}, Q_{2_l}, Q_{2_h}) = \\ \mathit{definite}(P_{1_l} \parallel P_{2_l}, P_{1_h} \parallel P_{2_h}, Q_{1_l} \parallel Q_{2_l}, Q_{1_h} \parallel Q_{2_h}).\end{aligned} \tag{3.2}$$

No approximation is required.

It is more complicated to define the parallel operator between a definite block $\mathit{definite}(P_{1_l}, P_{1_h}, Q_{1_l}, Q_{1_h})$ and an indefinite block $\mathit{indefinite}(P_{2_l}, P_{2_h}, Q_{2_l}, Q_{2_h})$.

The indefinite block may or may not contribute to the output depending on the resistance of its series resistor range. A conservative approach is to take both extremes into considerations. At one extreme, the resistance of the series resistor range is set to zero such that the indefinite block behaves like a definite block. In this case, the output solution space is equal to

$$\mathit{definite}(P_{1_l} \parallel P_{2_l},\ P_{1_h} \parallel P_{2_h},\ Q_{1_l} \parallel Q_{2_l},\ Q_{1_h} \parallel Q_{2_h}).$$

This solution space can be bounded by its left and right corners. The other scenario is to assume the series resistor range to have infinite resistance. In this case, the output of the indefinite block is in high impedance. Hence, the solution space of the combined output is the same as that of the definite block. This solution space can also be bounded by its left and right corners. In either extreme, the combined output is driven; hence, the output type is definite.

In order to satisfy the conservative criterion, the output solution space is approximated by the smallest region in the V_{eq}-R_{eq} plane which contains all four corners in the above two cases and which is realizable by a definite block. This concept is graphically illustrated in Figure 3.13 (b) where each pair of left-and-right corners are joined by a line for clarity. It is easy to tell from this figure that the top pair is set by the definite block alone because they have higher equivalent resistances than the lower pair.

The upper bounds of the approximate pull-up and pull-down resistor ranges, R_{U_h} and R_{D_h} in Figure 3.13, are set by the upper bounds of the definite block's pull-up and pull-down resistor ranges. By the same token, the lower bounds of the approximate pull-up and pull-down resistor ranges, R_{U_l} and R_{D_l} in Figure 3.13, are set by the lower bounds of the combined pull-up and pull-down resistor ranges when the resistance of the series resistor range of the indefinite block is set to zero. This

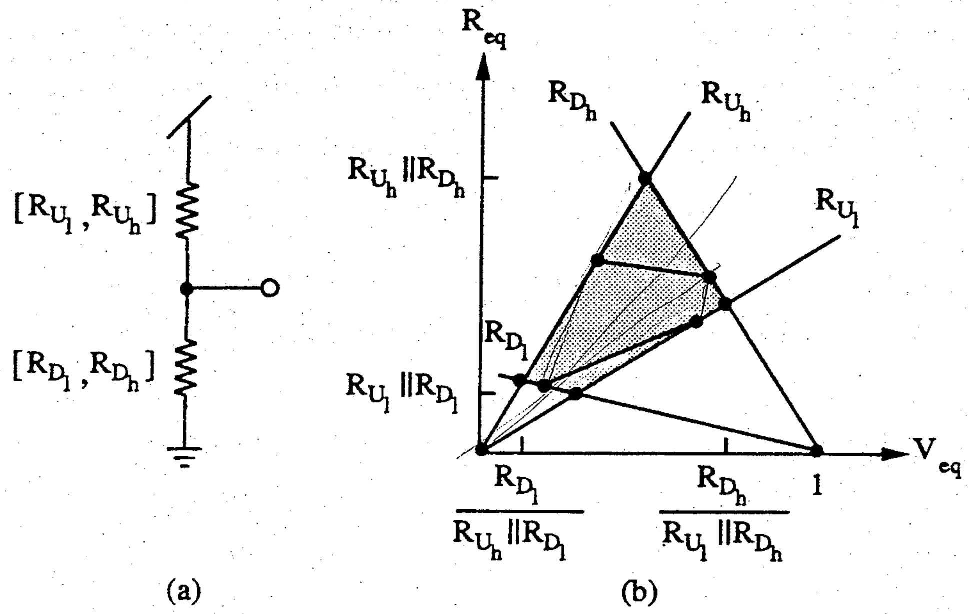


Figure 3.13: Approximate solution space of a definite block in parallel with an indefinite block.

result can be expressed as follows:

$$\begin{aligned}&\mathit{definite}(P_{1_l}, P_{1_h}, Q_{1_l}, Q_{1_h}) \parallel \mathit{indefinite}(P_{2_l}, P_{2_h}, Q_{2_l}, Q_{2_h}) = \\ &\qquad \mathit{definite}(P_{1_l} \parallel P_{2_l},\ P_{1_h},\ Q_{1_l} \parallel Q_{2_l},\ Q_{1_h}). \qquad (3.3)\end{aligned}$$

Error will be introduced in this step. As shown in Figure 3.13, the strongest pull-down combination (the left corner) of the approximate solution space is stronger than any pull-down combination of the exact solution space; similarly, the strongest pull-up combination (the right corner) of the approximate solution space is stronger than any pull-up combination of the exact solution space. Thus, the approximation is always conservative, but it can occasionally be pessimistic as well. The following example illustrates the most pessimistic scenario.

Assume that both the pull-up and the pull-down resistances of a given definite block are equal to R, and both the pull-up and the pull-down resistances of a given indefinite block are equal to r. When the outputs of these two blocks are connected together, one can easily see the combined output has a voltage of 0.5 regardless of the state of the X transistor associated with the indefinite block. However, according to Equation 3.3, the improved resistor-divider model approximates the result to be $definite(R \parallel r, R, R \parallel r, R)$. This approximation can be extremely pessimistic when $r \ll R$ — the approximate voltage range is equal to $[0, 1]$. In general, the approximation tends to be pessimistic when the pull-up and the pull-down resistances of the definite block are much larger than that of the indefinite block.

The parallel operator also has to be defined between two indefinite blocks; say $indefinite(P_{1_l}, P_{1_h}, Q_{1_l}, Q_{1_h})$ and $indefinite(P_{2_l}, P_{2_h}, Q_{2_l}, Q_{2_h})$. In essence, this case is the same as the case between a definite block and an indefinite block. However, the output here can either be in one of the three driven states or be in the high impedance state. The three driven scenarios are derived from different combinations of the two series resistor ranges. Each of their solution spaces is bounded by its left and right corners. The approximate solution space is defined to be the smallest region in the V_{eq}-R_{eq} plane which contains all three pairs of corners and which is realizable by an indefinite block. Mathematically, it can be stated as follows:

$$\begin{aligned} &indefinite(P_{1_l}, P_{1_h}, Q_{1_l}, Q_{1_h}) \parallel indefinite(P_{2_l}, P_{2_h}, Q_{2_l}, Q_{2_h}) = \\ &\qquad indefinite(P_{1_l} \parallel P_{2_l}, \max(P_{1_h}, P_{2_h}), Q_{1_l} \parallel Q_{2_l}, \max(Q_{1_h}, Q_{2_h})). \end{aligned} \quad (3.4)$$

The parallel operator introduces error in this operation just like it does to that between a definite block and an indefinite block.

3.5.3 Merits

The improved resistor-divider model has the following merits. First of all, the state of an X transistor is not predetermined; instead, it is properly handled as an unknown. As a result, the major drawback of the diamond method is eliminated. Secondly, this scheme computes the exact voltage range for a node when there is no bridging X transistors in the associated cluster. In other words, X transistors which lead to the power supply or ground do not result in any error. This is because without bridging X transistors, the only indefinite blocks are 1) a node driven to the Vdd through an X transistor, 2) a node driven to the ground through an X transistor, and 3) a node driven to the Vdd through one X transistor and to the ground through another X transistor. These circuits are shown in column (a) of Figure 3.14.

In order to proof that these three circuits do not introduce uncertainty to the final-value computation, one can look at their equivalents shown in Figure 3.14 (b). The equivalents have all the characteristics of a definite block except that they are not guaranteed to be driven. When any of them is combined with a definite block, Equation 3.2 gives the exact solution. Since Equation 3.3 gives the same result under the same situation, it is also free of error.

When any two of the circuits shown in Figure 3.14 (a) are combined, the output still belongs to the same group of three circuits. The exact solution can either be obtained by Equation 3.4 or by applying Equation 3.2 to the corresponding equivalent circuits. Thus, if a cluster has no indefinite blocks other than those shown in Figure 3.14 (a), then the improved resistor-divider model gives the exact solution.

The box method, on the other hand, does not have the same characteristic. As explained in Section 3.3, a nondegenerate box appears whenever there is an X transistor, and error propagates and amplifies in the box method through parallel box operations which involve at least one nondegenerate box.

This brings up another merit of the improved resistor-divider model: error propagates slower in this scheme than in the box method in general. This is because

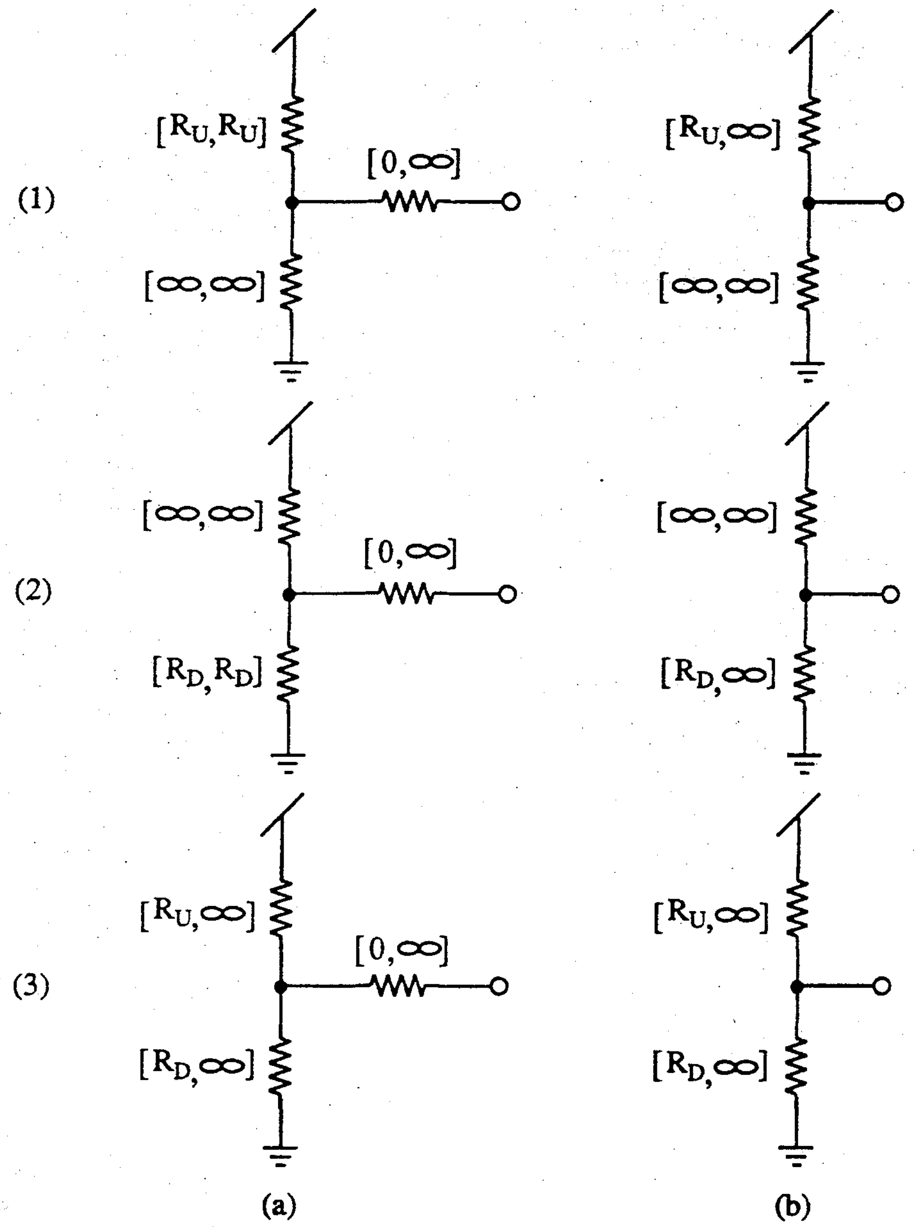


Figure 3.14: Three indefinite circuits (column (a)), which do not introduce uncertainty to the final-value computation, and their equivalents (column (b)).

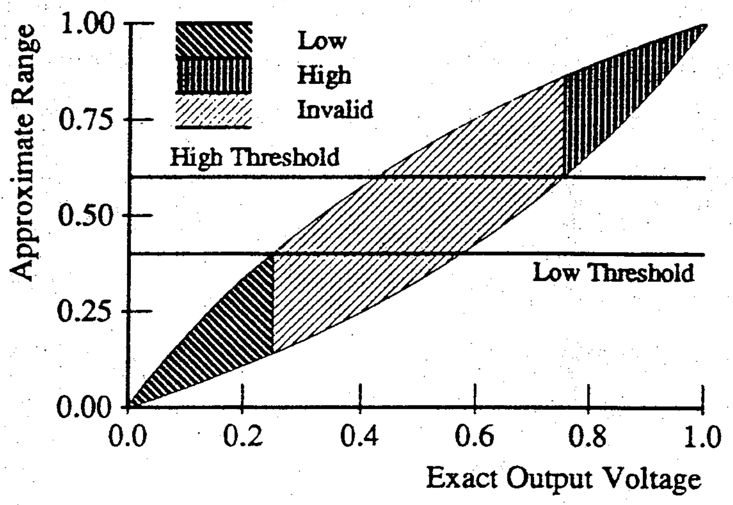


Figure 3.15: Approximate voltage range as a function of $V = R_1/(R_1 + R_3)$ if $definite(\infty, \infty, R_1, R_1)$, $indefinite(R_3, R_3, R_1, R_1)$, and $definite(R_3, R_3, \infty, \infty)$ are paralleled together.

parallel operations between definite blocks do not need any approximation, hence, error is confined to parallel operations involving indefinite blocks. When an indefinite block is combined with a definite block in parallel, error can be introduced; however, since the result is a definite block, the error does not amplify itself through future operations! For instance, if the improved resistor-divider model is applied to the example

$$definite(\infty, \infty, R_1, R_1) \parallel indefinite(R_3, R_3, R_1, R_1) \parallel definite(R_3, R_3, \infty, \infty)$$

as described in Section 3.3, the result is shown in Figure 3.15. This result is less pessimistic than the ones shown in Figures 3.6 and 3.7 because the bounds are much tighter; yet it is still conservative because the shaded region includes the exact solution space.

The result shown in Figure 3.15 is order independent because the improved resistor-divider model is order independent. This can be proved by looking at the definitions of the series and parallel operators. The series operator is intrinsically order independent because it is controlled by the circuit topology. The parallel operator has three variations. The two variations expressed in Equations 3.2 and 3.4 are order independent because finding the parallel or the maximum of a group of numbers is order independent. The third variation, which is expressed in Equation 3.3, involves a definite block and an indefinite block, and the upper bounds of the indefinite block's pull-up and pull-down components are ignored in computation. This variation does not cause order dependency because 1) if the indefinite block is first combined with an indefinite block, then the result is still an indefinite block; 2) if the indefinite block is first combined with another definite block, then its upper bounds are ignored in that operation.

Last but not the least, the computational complexity of this scheme is low, and its data structure is simple. As a matter of fact, this method uses no more resources than either the box method or the diamond method. The implementation details are discussed in Chapter 6.

3.6 Simulation Example

The example shown in Figure 3.16 is encountered during the simulation of the MIPS-X processor[11,10], which is a CMOS design with an on-chip instruction cache of 2K bytes. This example provides a realistic comparison between the box method and the improved resistor-divider scheme.

The figure shows the 6-T RAM designed for the instruction cache. When RSIM is first activated, nodes B, C, and D of the circuit are all in unknown states. In order to write a zero into the memory cell, signal A is first raised such that the bit-line capacitor can be discharged. Then the word-line control signal will be raised to pass the low signal into the memory cell. At this moment, the state of node D is still unknown, hence RSIM thinks that the lower inverter of the memory cell can

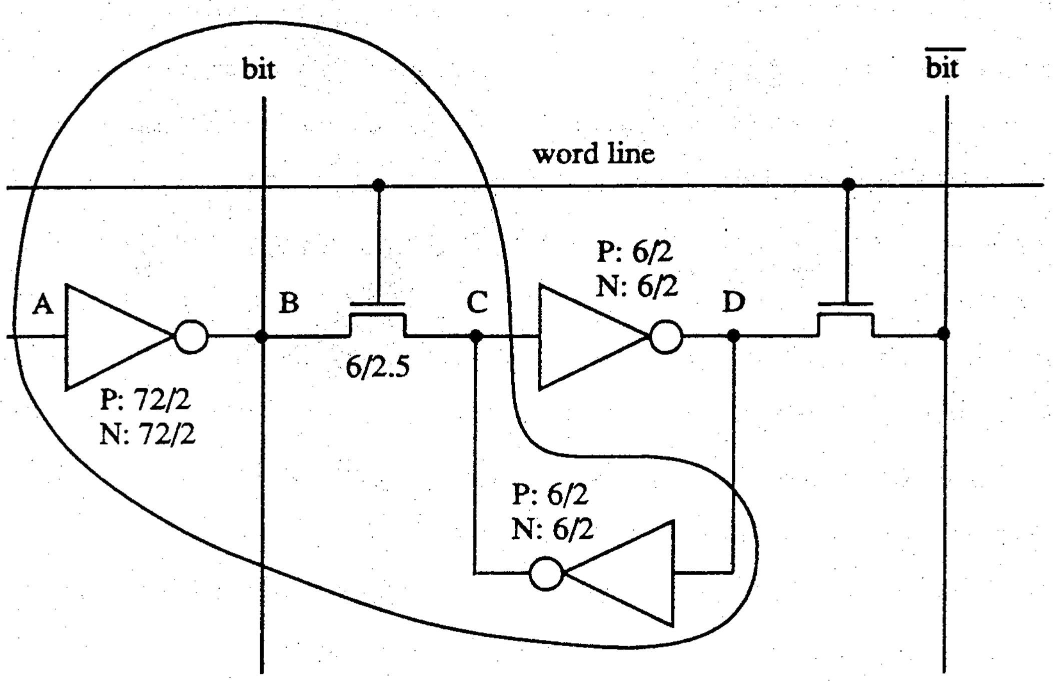


Figure 3.16: A 6-T RAM and its driver.

fight with the much stronger bit-line driver. The cluster of interest is circled in the drawing.

RSIM models the cluster as a resistor network. The version of RSIM that runs at Stanford sets the normalized low and high thresholds at 0.4 and 0.6 respectively. It also sets the effective resistances of nMOS and pMOS transistors as follows:

	nMOS	pMOS
Dynamic Low	6000 Ω/□	24000 Ω/□
Dynamic High	12000 Ω/□	12000 Ω/□

Thus, the circled cluster becomes a resistor network shown in Figure 3.17.

In order to compute the voltage at node C, Figure 3.17 can be partitioned into three subcircuits as illustrated. The box method views each subcircuit as a box, and computes the output according to rules described in Section 3.3. Assume that

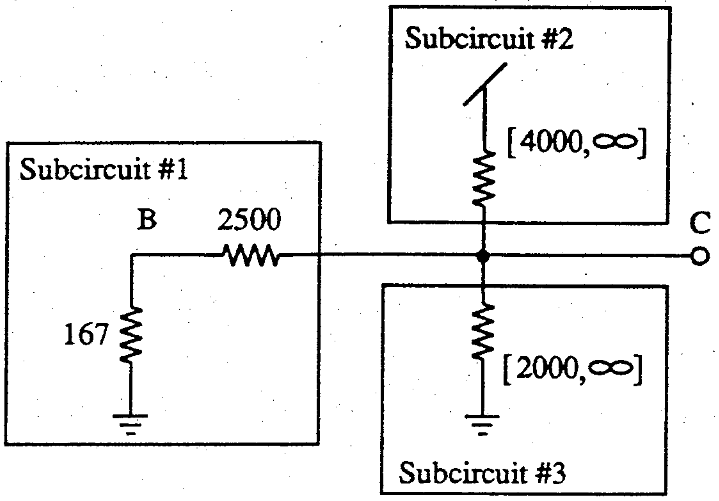


Figure 3.17: A resistor model of the circled cluster shown in Figure 3.16.

*box*1, *box*2, and *box*3 represent the solution spaces of subcircuits #1, #2, and #3 respectively. Since the box method is order dependent, its approximation can vary with how boxes are combined. If *box*1 is first combined with *box*2 or *box*3, then the approximate output voltage range is [0, 0.4]. This range accurately reflects the worst case scenario for the output, and node C can be properly driven to the logical low state. However, if *box*2 and *box*3 are combined first, then the output voltage range becomes [0, 0.67]! In this case, the box method signals the output to be invalid. Thus, if the order of evaluation is totally random, then in one third of the time, the box method cannot properly initialize this memory cell.

On the other hand, the improved resistor-divider model views subcircuit #1 as a definite block while it views subcircuits #2 and #3 as indefinite blocks. It computes the output voltage range to be [0, 0.4] regardless of the order of evaluation. Thus, the memory cell can always be properly initialized.

3.7 Summary

Transistors with unknown gate states (i.e. X transistors) post great difficulties to the determination of DC voltages. These transistors introduce uncertainties to simulators because they can be considered as either conducting or nonconducting; thus, they can vary the electrical connectivity of a circuit.

Since the logical state of a circuit depends only on the minimum and maximum voltages that the circuit can reach, simulators need not to exhaustively evaluate all achievable voltages of the circuit. There are many heuristics to approximate an achievable voltage range. A better known heuristic is the one implemented in RSIM. This heuristic is shown to be conservative, but it also has many problems. For example, it can occasionally give pessimistic approximations, and it suffers consistency problem — its results may depend on the order of evaluation.

A new DC-computation scheme is introduced in this chapter. This scheme is also conservative, but it is in general less pessimistic than RSIM's scheme. The new scheme is self-consistent in that its results are not order dependent. An example from an actual simulation shows that the new scheme compares favorably to RSIM's scheme.

Chapter 4

Linear Charge-Sharing Models

4.1 Overview

The timing models reviewed in Chapter 2 assume that charge flows unidirectionally between capacitors and a voltage supply, and that voltage waveforms are monotonic. In reality, neither the unidirectional nor the monotonic assumption is universally true. For example, if a network does not have a voltage source, then there is not an exclusive supplier or drainer of charge. Hence, if charge is to be redistributed among the capacitors of such a network, it is incorrect to assume that the charge will always flow in one direction. Another example is that if capacitors in a network start at different voltages, then their waveforms may not be monotonic. Complications caused by the redistribution of charge among capacitors are commonly referred to as charge-sharing problems. This chapter focuses on how to model charge sharing on RC networks.

Charge sharing often occurs in IC's and causes problems to switch-level simulators. Formal definition and detailed analysis of charge sharing are provided in Section 4.2. Sections 4.3 and 4.4 then introduce methods that can be used in a switch-level simulator to model charge sharing. These methods are presented with both mathematical and conceptual arguments. Mathematical argument is based on the frequency-domain analysis, which expands a network transfer function. On the other hand, a more intuitive physical picture provides insight for easy understanding and potential improvements. The parallel between mathematical rigor and physical basis also helps the construction of charge-sharing models for nonlinear networks in the next chapter.

4.2 Charge-Sharing Networks

Charge sharing refers to the voltage variations on nodes caused by the redistribution of charge when two RC networks at different steady-state voltages are connected together. There are two kinds of charge sharing. The first kind is *pure charge sharing* in which two nondriven RC subnets are connected through a resistive switch. Since sharing charge is the only means to achieve voltage variations, it is important to model the charge-redistribution process in order to calculate the elapsed time for a node to reach its switching voltage — a threshold between the old and the new logical states. A simulator can use this delay information to schedule the transition events at the correct time. The lack of an active drive in pure charge sharing causes a great problem to the single-time-constant model which uses the dominant-pole approximation. The dominant time constant in pure charge sharing is infinite (since the nodes are not driven to either 0 or 1), which is much slower than the time constants set by the redistribution of charge. Since a major voltage variation occurs during the redistribution of charge, a single time-constant model is inadequate.

The other kind of charge sharing is illustrated in Figure 4.1. *Charge sharing with a driven path* is the same as pure charge sharing except that one of its subnets has a resistive path to a voltage source. The subnet with the driven path will

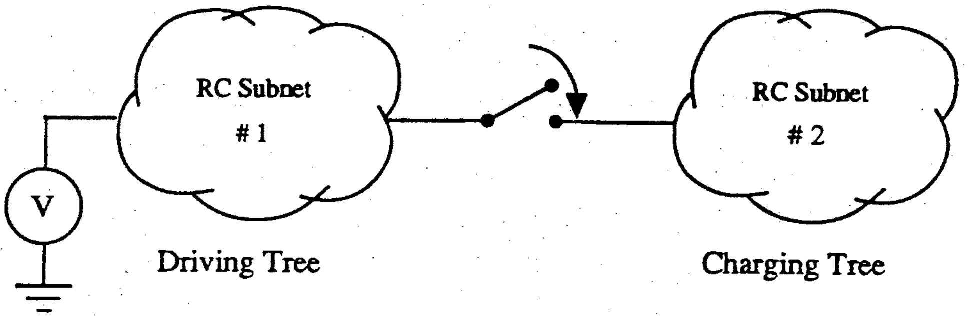


Figure 4.1: Charge sharing with a driven path.

be called the *driving tree*. In the steady states before and after the concatenation, capacitors in the driving tree have the same voltage as the voltage source. The other subnet which is not driven initially will be labeled as the *charging tree*. Capacitors in the charging tree start at a certain voltage but are eventually driven to the steady-state voltage equal to that of the driver in the driving tree. A model for this kind of charge sharing needs multiple time constants because there are two events happening simultaneously. One event is caused by the redistribution of charge, and the other is due to the driving source. A node in the charging tree always has a monotonic voltage waveform, and the two-time-constant model described in Section 2.4 works well. A node in the driving tree is more complicated since it starts and ends at the same voltage but can have a voltage spike whose amplitude is large enough to cause the node to temporarily change its state. The amplitude of a voltage spike is defined to be the maximum voltage fluctuation during the transition. In order to catch glitches, a simulator must calculate these fluctuations.

Charge sharing occurs in many places. In nMOS designs, precharged logic is used to gain speed while avoiding static power dissipation. This circuit technique works through pure charge sharing: a precharged node shares its charge with another node, which has a much smaller capacitance, without degrading its signal level.

Nondriven clusters can also occur due to flaws in design or simulation vectors.

Charge sharing with a driven path appears in all kinds of MOS technologies. It corresponds to situations where a switching pass transistor connects an actively driven network. such as a gate. with a high-impedance fan-out network whose capacitors are initially charged to a different polarity.

At present. most switch-level simulators, including RSIM. use ad hoc approaches to model these events. Without a good timing model. these simulators assume that charge-sharing events happen instantaneously and schedule them before driven events. To estimate the amplitudes of voltage spikes. these simulators use the ratio of the total charge to the total capacitance and in essence ignore all the resistors in the circuit. These schemes often introduce fictitious events that slow down the speed of simulation and sometimes cause flip-flops to latch incorrect values.

Two examples demonstrate these shortcomings. The circuit shown in Figure 4.2 is similar to the barrel shifter of the MIPS processor[20], which is an nMOS design. The node marked N reads from one of the several sources. Each source is a precharged capacitor of 2 pF. Assume that φ (phi) goes high ahead of *Load.1* such that by the time *Load.1* goes high. the voltages of the capacitors on both sides of the transistor gated by φ are the same, and their values are different from that of the source capacitor. The elapsed time for node N to change its logical state is crucial in determining the speed of this barrel shifter. An effective-resistance model of the circuit is shown in Figure 4.3. If node N is scheduled to change instantaneously as suggested by RSIM, then the performance of the circuit is overestimated.

The RAM cell from the MIPS-X processor. which is shown in Figure 4.4, illustrates another problem in simulating charge sharing. In order to read from the memory cell. both *bit* and $\overline{bit}$ are precharged. Assuming a zero is stored in the cell when the *word* line goes high, one can see that the large ratio in capacitances between the *bit* line and the internal node will cause a voltage spike on the latter. An

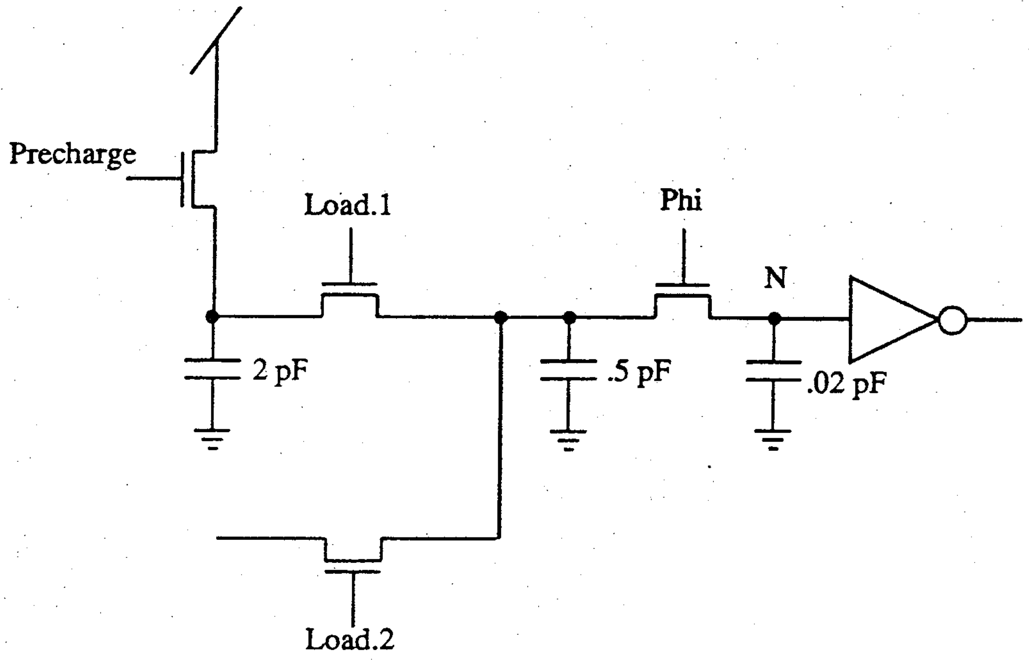


Figure 4.2: A circuit which is similar to the MIPS processor's barrel shifter.

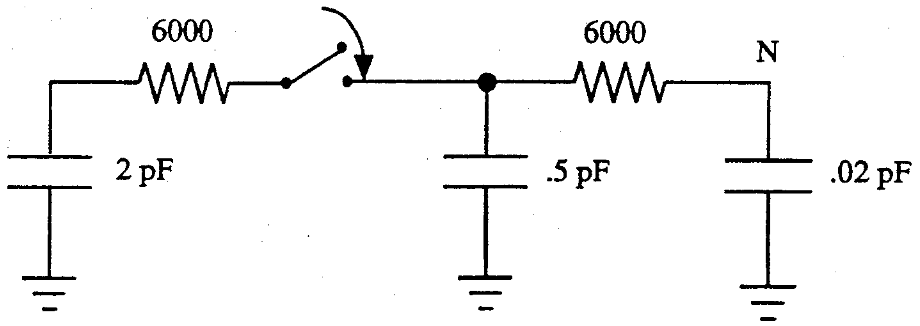


Figure 4.3: An effective-resistance model of the barrel shifter shown in Figure 4.2.

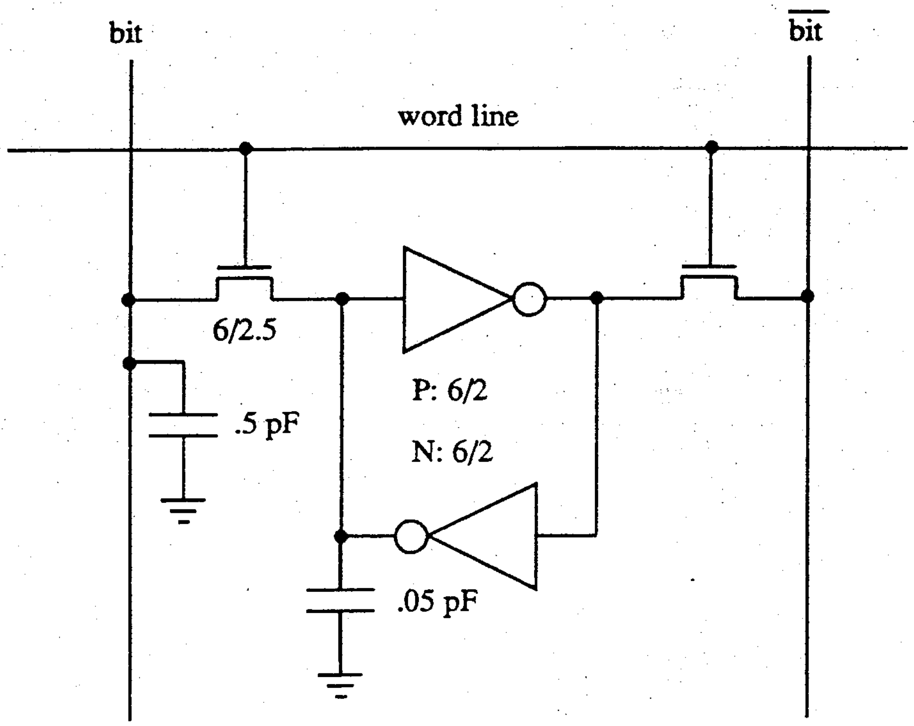


Figure 4.4: The RAM cell designed for the instruction cache of the MIPS-X processor.

effective-resistance model of the RAM's read/write network is shown in Figure 4.5. The access transistor's effective resistance is doubled because it is passing a high voltage. Despite the ten-to-one ratio in *bit*-line and internal-node capacitances, the amplitude of the voltage spike is merely 0.25. However, a simpleminded approach such as RSIM's, which uses the ratio of the total charge to the total capacitance, predicts a voltage spike with an amplitude of more than 0.9. This poor prediction propagates through the cross-coupled inverters and destroys the value stored in the cell!

As the next sections will show, by using only a slightly more complex method, one can form much better estimates of the charge-sharing effects.

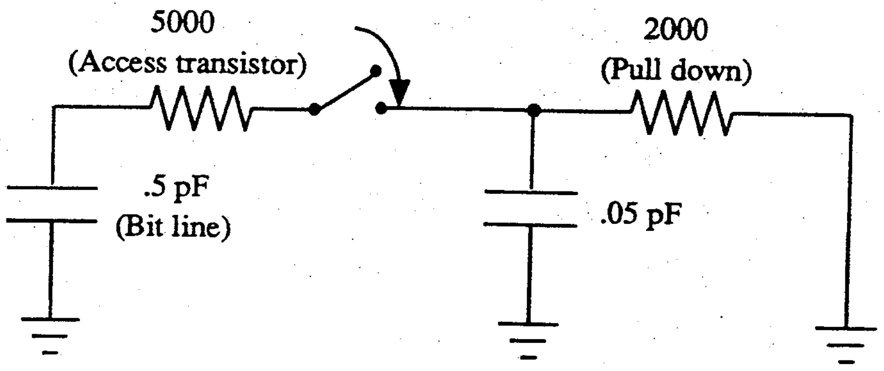


Figure 4.5: An effective-resistance model of the RAM's read/write network.

4.3 Pure Charge Sharing

In a pure charge-sharing network, the steady-state voltage V_f is determined by the ratio of the total charge to the total capacitance. However, during a transition, the voltage at any given node is closely coupled with its neighbors, which makes a closed-form solution very complicated. In some sense, even a single-driver RC network can be classified as a special case of pure charge sharing because the voltage source is functionally equivalent to a gigantic capacitor with the initial voltage equal to that of the voltage source. As a matter of fact, pure charge-sharing networks and single-driver networks share many common characteristics. For example, both kinds of networks are linear systems, and their transient waveforms are usually dominated by a single pole. The reason for the transient or the charge-redistribution portion of a pure charge-sharing waveform to have a single pole is that pure charge-sharing designs usually have a dominant capacitor that acts as a virtual voltage source. Thus, pure charge-sharing problems can be approached in a way similar to that of the single-time-constant model: by calculating the area between an output waveform and a time-independent line representing the final voltage and choosing an exponential function with the same area and boundary conditions. This approach was also independently suggested by Raghunathan and Thompson[23,22].

In the single-time-constant model. an output voltage is expressed in terms of capacitors' currents flowing towards a fixed voltage source. However, in pure charge-sharing networks. all nodes are floating: there is no distinguished node which acts as a supplier or drainer of charge. Nevertheless. a derivation similar to the one in Section 2.4 can be followed to express the voltage difference between any pair of nodes. After the voltage differences between every node and a randomly selected reference node r are established, conservation of charge can be used as an additional condition to decouple the relationship.

4.3.1 Waveform Estimate

If each capacitor C_k of a pure charge-sharing network is replaced by its equivalent current source $i_k = -C_k \frac{dV_k}{dt}$. the voltage difference between node ϵ and the reference node can be written as

$$V_\epsilon - V_r = \sum_k R^r_{k\epsilon} i_k = -\sum_k R^r_{k\epsilon} C_k \frac{dV_k}{dt} \tag{4.1}$$

where $R^r_{k\epsilon}$ is the resistance of the path to the reference node r shared by both node k and node ϵ. The area between V_ϵ and V_r in the time domain is equal to

$$\begin{aligned} \int_0^\infty V_\epsilon - V_r \, dt &= -\sum_k R^r_{k\epsilon} C^h_k \int_1^{V_f} dV_k - \sum_k R^r_{k\epsilon} C^l_k \int_0^{V_f} dV_k \\ &= \sum_k R^r_{k\epsilon} C^h_k - V_f \sum_k R^r_{k\epsilon} C_k \\ &\overset{\text{def}}{=} \tau_{A^r_\epsilon} \end{aligned} \tag{4.2}$$

where C^h_k is equal to C_k if C_k is at logical high initially and zero otherwise (vice versa for C^l_k). Equation 4.2 is not the desired result because it gives the area between two waveforms which are changing simultaneously. Conservation of charge provides the necessary information to decouple node r from node ϵ. Since $\sum_\epsilon C_\epsilon V_\epsilon = C_T V_f$ where C_T is the sum of all capacitances, one can multiply both sides of Equation 4.2

by the capacitance C_ϵ of node ϵ and sum over corresponding equations from every node to give an equation for $\int_0^\infty V_f - V_r \, dt$:

$$\sum_\epsilon \int_0^\infty C_\epsilon (V_\epsilon - V_r) dt = C_T \int_0^\infty V_f - V_r \, dt = \sum_\epsilon C_\epsilon \tau_{A_\epsilon^r}.$$

Combining this equality with Equation 4.2 gives the area between the waveform V_ϵ at any node ϵ and V_f:

$$\int_0^\infty V_\epsilon - V_f \, dt = \tau_{A_\epsilon^r} - C_T^{-1} \sum_k C_k \tau_{A_k^r} \,. \tag{4.3}$$

Assuming the transient waveform is dominated by a single time constant, the estimated voltage waveform V_ϵ^* is

$$V_\epsilon^* = \begin{cases} V_f + (1 - V_f)\epsilon^{-t/\tau_\epsilon} \\ V_f(1 - \epsilon^{-t/\tau_\epsilon}) \end{cases} \quad \text{where} \quad \tau_\epsilon = \begin{cases} \frac{C_T \tau_{A_\epsilon^r} - \sum_k C_k \tau_{A_k^r}}{(1-V_f) C_T} & \text{(for falling } V_\epsilon^*) \\ \frac{\sum_k C_k \tau_{A_k^r} - C_T \tau_{A_\epsilon^r}}{V_f C_T} & \text{(for rising } V_\epsilon^*) \end{cases} . \tag{4.4}$$

This model is used to estimate the voltage waveform at node X of the RC network shown in Figure 4.3. Comparison with the exact waveform is plotted in Figure 4.6. The model works extremely well for this example because the circuit in Figure 4.3 practically has only one nondegenerate pole. Although the approximation for a more complicated circuit is not guaranteed to be as good as this one, designs with pure charge sharing in mind are usually quite simple.

The time constant τ_ϵ in Equation 4.4 may seem counterintuitive at first glance because $\tau_{A_\epsilon^r}$ varies with the reference node. As a matter of fact, Raghunathan and Thompson[23,22] suggested always using the node of interest to be the reference. Their scheme results in excessive computation because a different set of parameters is required for each and every node. Computation can be drastically reduced by using one randomly selected reference node, since $C_T \tau_{A_\epsilon^r} - \sum_k C_k \tau_{A_k^r}$ is a constant regardless of the node chosen to be the reference! The consistency of this result is not a coincidence. The next section will prove that the above result is a degenerate version of the standard two-pole-one-zero model.

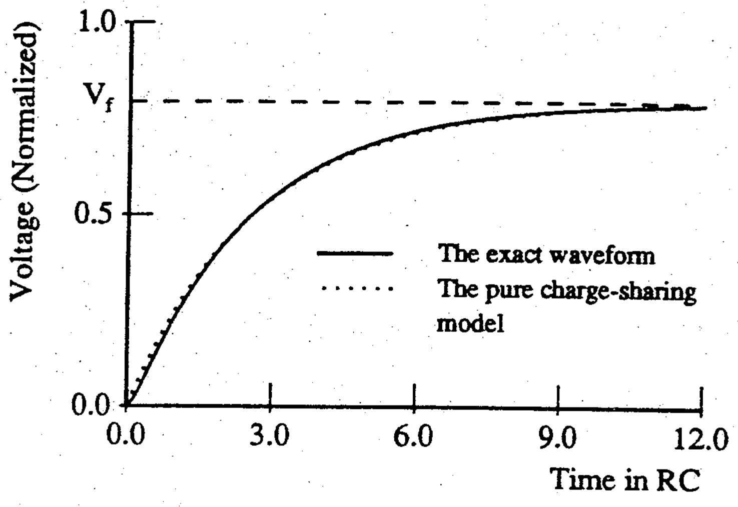


Figure 4.6: The pure charge-sharing model versus the exact waveform for node N in Figure 4.3

4.3.2 Frequency-Domain Interpretation

Since the time-domain derivation is based on an intuitive understanding of the single-time-constant model, it says little about the fundamentals of the model. In order to discover the limitations and pitfalls of this result, frequency-domain analysis is carried out. A new network is first constructed by connecting a weak driver to any node r of the original pure charge-sharing network. The weak driver consists of a fixed voltage source equal to V_f in series with a large resistance R_L. Although the total charge in the new network is the same at time zero and infinity, charge is continuously introduced or drained by the voltage source before the system reaches equilibrium. Yet in the limit of a very large R_L, voltage waveforms in the new and the old networks are practically the same for timing purposes. The reason to construct the new network is to provide a reference node with a fixed voltage, such that the standard frequency-domain formulation can be applied.

In this new system, voltage V_e^* at node e with respect to the voltage source becomes $V_e^* - V_f = -\sum_k (R_L + R_{ke}^r) C_k \frac{dV_k^*}{dt}$, and the network transfer function at node ϵ is equal to

$$\begin{aligned} H(s) &= \int d(V_\epsilon^{r*} - V_f) + s\int_0^\infty (V_\epsilon^{r*} - V_f)\, dt - s^2 \int_0^\infty t\,(V_\epsilon^{r*} - V_f)\, dt + \cdots \\ &= \begin{cases} (V_f - 1) \quad +s\tau_{A_\epsilon^r} - s^2 \sum_k (R_L + R_{k\epsilon}^r) C_k \tau_{A_k^r} + \cdots & (\text{if } V_\epsilon^{r*} \text{ starts at } 1) \\ V_f \quad +s\tau_{A_\epsilon^r} - s^2 \sum_k (R_L + R_{k\epsilon}^r) C_k \tau_{A_k^r} + \cdots & (\text{if } V_\epsilon^{r*} \text{ starts at } 0) \end{cases} \end{aligned} \tag{4.5}$$

It is worth noting that the area between V_ϵ^{r*} and V_f in the new system is exactly the same as the area between V_ϵ and V_r in Equation 4.2 but is quite different from the area between V_ϵ and V_f in Equation 4.3. Although V_ϵ^{r*} can be as close to V_ϵ as desired by increasing R_L. a large R_L results in a slow charge-restoration process. In other words, the weak driver significantly changes the area and higher moments of a waveform without a noticeable change to the waveform itself.

The network transfer function can be approximated by models with different degrees of accuracies. However, a single-time-constant model cannot catch both the charge-redistribution and the charge-restoration portions of a waveform. The two-pole-one-zero model given by Equation 2.2 is the next simplest model. By matching the coefficients of the first three terms in Equations 2.2 and 4.5, one can show that the ratio between the product and the sum of the two approximate time constants, $\tau_{M\epsilon} = \tau_1\tau_2/(\tau_1 + \tau_2)$. bears the following relationship:

$$\lim_{R_L \to \infty} \tau_{M\epsilon} = \begin{cases} \frac{C_T\tau_{A_\epsilon^r} - \sum_k C_k \tau_{A_k^r}}{(1-V_f)C_T} & (\text{if } V_\epsilon^{r*} \text{ starts at } 1) \\ \frac{\sum_k C_k \tau_{A_k^r} - C_T\tau_{A_\epsilon^r}}{V_f C_T} & (\text{if } V_e^{r*} \text{ starts at } 0) \end{cases}.$$

Since the sum of the two approximate time constants. $\tau_P = \sum_k (R_L + R_{kk}^r) C_k$, is unbounded when R_L approaches infinite, one of the poles is located at the origin, and the other is at $-1/\tau_{M\epsilon}$. A pole at the origin gives a degenerate time constant. which accounts for the restoration of charge by the weak driver. On the other hand, the faster time constant controls the transient. This derivation gives the same result as that of the time-domain approach.

In general. higher order of accuracy can be obtained, at the expense of computational complexity. by modeling the transient with more than one pole. In reality. such extra accuracy is seldom necessary for switch-level simulators.

4.3.3 Limitations of the Model

An intrinsic limitation of the model is due to the implicit assumption that waveforms in pure charge-sharing networks are monotonic — the area between a waveform and the time-independent line representing the final voltage increases monotonically with time. This condition is only guaranteed for charge sharing between two capacitors connected by a resistor. Therefore. it is not surprising to find out that the model gives the exact solution to such networks. Unfortunately. in its most general form. a waveform in a pure charge-sharing network can cross V_f several times before settling at the steady state.

In the time domain, the area computed by Equation 4.3 cannot differentiate a monotonic waveform from a waveform crosses V_f several times. In the latter case. the areas below and above the reference line can partially cancel each other. In the frequency domain. high-frequency poles are ignored when the transient is modeled by a single pole. With these limitations in mind. it is not hard to construct a hypothetical circuit which cannot be properly handled by the above model. An example is shown in Figure 4.7.

Assume that only the capacitor at node c is charged high initially. The capacitor at node b is negligible comparing to its neighbors. and the resistor to the right of the switch is much smaller than the one to the left. When the switch is closed, charge sharing is done quickly between nodes b and c. This drives the voltage of node b across the final voltage as shown in Figure 4.7. Due to the large resistor between nodes a and b, the capacitor at node a does not come to play until a later stage. Looking at the frequency domain. one can see that the network transfer function

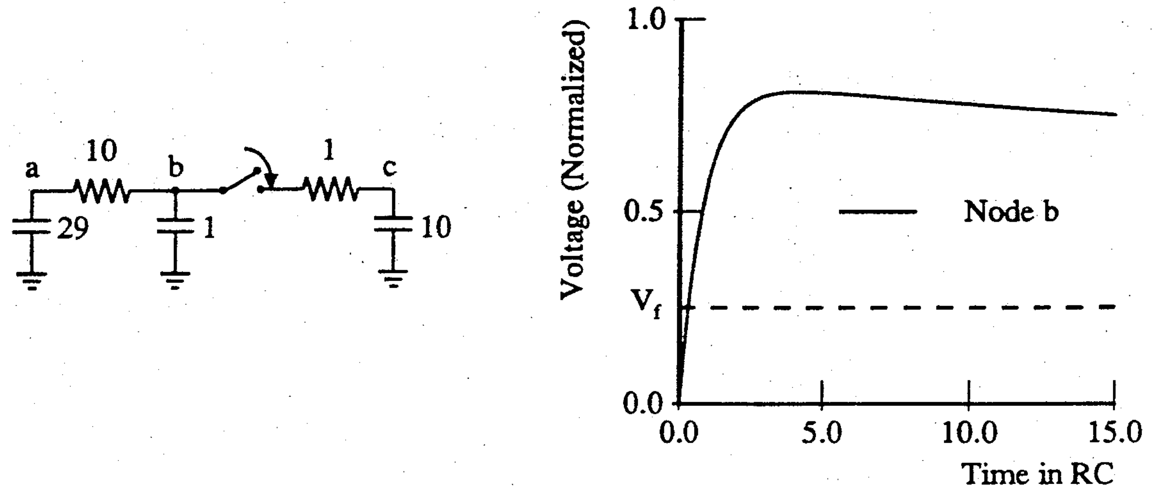


Figure 4.7: A hypothetical pure charge-sharing network which breaks the model.

of node b has a low-frequency zero which partially cancels the low-frequency pole contributed by node a. hence. the high-frequency pole becomes important. This is exactly the same reason why some circuits fail the single-time-constant model as described in Section 2.4.

The aforementioned pure charge-sharing model can only reasonably approximate the waveforms at nodes a and c. The area of the voltage waveform at node b is negative. and consequently its approximate time constant is also negative. which literally means that the waveform is unbounded when time approaches infinity. This unexpected result is easy to detect. and a safeguard can be built in. Since it is prohibitively expensive to come up with a more accurate model for this rare error. the time constant in Equation 4.4 can be redefined as

$$\tau_\epsilon = \begin{cases} \max\left(0. \ \frac{C_T \tau_{A_\epsilon^r} - \sum_k C_k \tau_{A_k^r}}{(1-V_f)C_T}\right) & \text{(if } V_\epsilon^* \text{ starts at 1)} \\ \max\left(0. \ \frac{\sum_k C_k \tau_{A_k^r} - C_T \tau_{A_\epsilon^r}}{V_f C_T}\right) & \text{(if } V_\epsilon^* \text{ starts at 0)} \end{cases} .$$

Even though the new definition can occasionally underestimate a delay. it is guaranteed to do no worse than scheduling a charge-sharing event instantaneously.

4.4 Charge Sharing with a Driven Path

The general two-pole-one-zero approximation can also be used to estimate the amplitudes of voltage spikes in driving trees. Assume. without loss of generality. that the driving source of a network is the ground. Thus. driving-tree nodes start and end at the ground voltage. The network transfer functions of these nodes must have a zero at the origin. A proof is provided in Appendix A. A zero at the origin contradicts the basis of the two-pole model described in Section 2.4. which assumes that one pole is closer to the origin than all other poles and zeros. The breakdown in assumption calls for a different kind of approximation.

In order to take the dominant zero into consideration, the network transfer function should be approximated by

$$H(s) \approx \frac{ks}{(1+s\tau_1)(1+s\tau_2)} = k(s - (\tau_1+\tau_2)s^2 + \cdots) .$$

Since the coefficients of s and s^2 shown here are not the same as the corresponding coefficients from the two-pole-one-zero model given by Equation 2.2, a separate derivation is necessary for this case.

4.4.1 Amplitude Derivation

For a two-pole-one-zero-at-the-origin system to match the area and the first moment of a real waveform, the sum of inverses of the two approximate poles. $\tau_{P_e} = \tau_1 + \tau_2$, is equal to

$$\tau_{P_e} = \frac{\sum_k R_{ke} C_k \tau_{A_k}}{\tau_{A_e}} \quad \text{where} \quad \tau_{A_e} = \int_0^\infty V_e \, dt = \sum_k R_{ke} C_k^h .$$

This is in contrast to other two-pole-one-zero models where the sum $\tau_1 + \tau_2 = \sum_k R_{kk} C_k$ depends strictly on the circuit configuration. τ_{A_e} can be interpretated as a modified form of the Elmore's delay — a form taking the initial charge distribution into formulation. However, delay is meaningless for nodes in driving trees because they start and end at the same voltage.

Since a two-pole waveform cannot be uniquely specified by its area and first moment, this model returns a family of waveforms as a function of the approximate lowest-frequency pole, $-1/\tau_1$. The approximate voltage waveform V_ϵ^* of node ϵ of the driving tree can be expressed in terms of τ_1:

$$V_\epsilon^* = \frac{\tau_{A_\epsilon}}{2\tau_1 - \tau_{P_\epsilon}} \left[\epsilon^{-t/\tau_1} - \epsilon^{-t/(\tau_{P_\epsilon} - \tau_1)}\right].$$

The peak amplitude of V_ϵ^*, as a function of τ_1, is equal to

$$\frac{\tau_{A_\epsilon}}{\tau_{P_\epsilon} - \tau_1} \left(\frac{\tau_1}{\tau_{P_\epsilon} - \tau_1}\right)^{-\tau_1/(2\tau_1 - \tau_{P_\epsilon})}.$$

By definition, the dominant pole is closer to the origin than any other poles, so for a two-pole system, $(\tau_{P_\epsilon}/2) \leq \tau_1 \leq \tau_{P_\epsilon}$. In this domain, the peak amplitude of waveforms in a family increases monotonically with τ_1 and is bounded by $(2/\epsilon)(\tau_{A_\epsilon}/\tau_{P_\epsilon})$ and $\tau_{A_\epsilon}/\tau_{P_\epsilon}$ (the value of the lower bound is approximately equal to 73.6 percent of that of the upper bound).

4.4.2 Physical Interpretation and Improvement

The above mathematical model can be improved by investigating its physical basis. The charge-sharing network shown in Figure 4.8 has exactly two poles. If C_2 is charged high initially, then the network transfer function of node E has a zero at the origin. In other words, the two-pole-one-zero-at-the-origin model actually tries to map a node in the driving tree to a node in a two-capacitor-two-resistor circuit. The mapping is done by matching geometric waveform characteristics such as the area and the first moment. The mapped two-capacitor-two-resistor circuit will be defined as the *reduced network* of the original node. Unfortunately, the two matches are insufficient to determine the four unknowns in a reduced network. As a result,

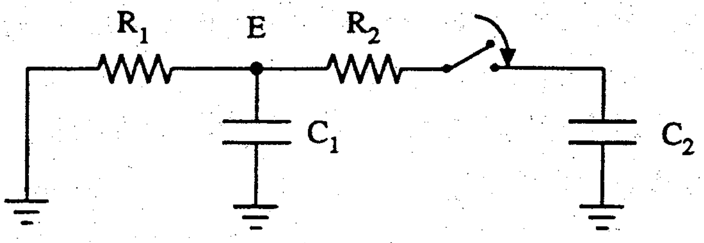


Figure 4.8: The simplest circuit with two poles and a zero at the origin.

the amplitude calculation leaves a 26.4 percent (i.e. $1 - \frac{2}{\epsilon}$) uncertainty between the lower and the upper bounds. Additional constraints can be introduced to narrow the search space. The obvious one is to compute the second moment of a node's step response, which, however, is computationally expensive. A subtler but much cheaper to compute constraint can be used.

Since both the original and the reduced network are linear systems, the locations of their poles are subject only to their network topologies. Hence, a reduced network should be reasonablely capable of modeling the original node regardless of the initial charge distribution. One way to ensure this capability is to introduce an additional constraint such that if all capacitors of the original and the reduced systems are charged high initially, the Elmore's delays of node E of the reduced network and node ϵ of the original network are equal. The Elmore's delay of node ϵ can be derived from the topology of the original network and is equal to $\tau_{D_\epsilon} = \sum_k R_{k\epsilon} C_k$. This constraint does not interfere with or contradict to the area and the first moment constraints. Solving these constraints gives

$$C_1 = \frac{\tau_{D_\epsilon} - \tau_{A_\epsilon}}{\tau_{A_\epsilon}} C_2$$

$$R_1 = \frac{\tau_{A_\epsilon}}{C_2}$$

$$R_2 = \frac{\tau_{P_\epsilon} - \tau_{D_\epsilon}}{C_2} . \qquad (4.6)$$

The analytic solution $V_E = a(\epsilon^{-t/\tau_1} - \epsilon^{-t/\tau_2})$ of a reduced network is easy to obtain. and the peak voltage V^*_{max} is given by

$$V^*_{max} = \frac{\tau_{A_e}}{\tau_{P_e}} f((\tau_{P_e} - \tau_{D_e})(\tau_{D_e} - \tau_{A_e})/\tau^2_{P_e})$$

where the normalized amplitude function f ranges between $2/\epsilon$ and 1, and its equation is presented in Appendix B. This model gives the exact solution if the original circuit itself has only two time constants. Two-time-constant circuits are quite common: the read/write network of the RAM cell shown in Figure 4.5 is an example. In fact, according to the simulation on the execution unit[1] of the MIPS-X processor, 77.5 percent of its dynamic clusters have two or fewer transistors. Hence, being able to solve a reduced network without error is an extra bonus of this model.

4.4.3 Limitations of the Model

The charge sharing with a driven path model is plagued by the same problem as the pure charge-sharing model: it is vulnerable to circuits with low-frequency zeros that cancel the effect of low-frequency poles. A circuit similar to the one shown in Figure 4.7 illustrates this deficiency. Assume that a resistor of value 10 connects node a to the ground. Since the capacitor at node b is insignificantly small compared to its neighbors and the capacitor at node a is large enough to be considered as a virtual ground, the resistive-divider between node a and node c dictates the initial voltage waveform at node b. The whole system is eventually discharged at approximately the same rate through the driving resistor. As a result, the amplitude at node b cannot be reasonably modeled with two poles. This failure can be caught by the model: R_2 in Equation 4.6 becomes negative, which is not physically realizable.

[1]The RAM cell is located in the instruction cache, which is not part of the execution unit.

The problem is expensive to fix. Matching higher-order moments is not only computationally demanding, it is still vulnerable to the same problem though at a different level. Fortunately, even though pedagogical circuits with multiple low-frequency zeros are easy to construct, they are very rare in digital designs. In order to recover gracefully after detecting that the model fails, the definition of R_2 needs to be modified:

$$R_2 = \max(0, \frac{\tau_{P_e} - \tau_{D_e}}{C_2}) .$$

4.5 Summary

Charge sharing can be classified into two kinds: pure charge sharing and charge sharing with a driven path. In the former kind, a waveform is hard to estimate because its dominant time constant is infinite. In the latter kind, delay is meaningless because a node starts and ends at the same level; yet its transient waveform may cause a glitch.

Pure charge sharing needs to be modeled by two poles in order to catch the charge-redistribution portion of a waveform. However, the previous two-pole model cannot be directly applied because there is no distinguished reference node. The similarity between pure charge sharing and a fully-charged driven case is used as the basis for the time-domain derivation. Frequency-domain interpretation of the result is also presented. It is shown that a waveform can be modeled with the sum of two exponential functions, though one of them degenerates to become a constant.

Charge sharing with a driven path is controlled by two events: sharing charge between the charging tree and the driving tree, and a driving voltage source. The amplitude of a voltage spike is determined by high-frequency components of a waveform. These components can usually be taken care of by introducing an additional pole in the model. However, some care must be taken in using the two-pole-one-zero model because there is a zero at the origin in this case. Although a similar approach to the one reviewed in Section 2.4.2 can match the area and the first moment of a waveform, these two constraints are not enough to uniquely determine a waveform. Fortunately, matching the Elmore's delay can provide an additional piece of information, which is then sufficient to specify a unique waveform estimate.

Chapter 5

Charge Sharing in Transistor Networks

5.1 Overview

Chapter 4 has introduced charge-sharing models for resistor-capacitor networks: however. the circuits being modeled are really transistor-capacitor networks (or *TC* networks). The major complication in modeling MOS networks is due to transistors' nonlinearity. For example. an nMOS transistor passes a low signal more effectively than a high signal. but the opposite is true for a pMOS transistor. To compensate for this asymmetry, switch-level simulators usually assign different effective resistances to transistors depending on the signal level being passed[26]. However, there is always an underlying uncertainty about how this ad hoc compensation can change the accuracy of a model. To address this doubt, this chapter attempts to construct nonlinear charge-sharing models which take transistors' nonlinearity into consideration.

Since a step function has been assumed for the gate input of a switching transistor, all transistors which belong to the same cluster enter the linear region either right away or shortly after the input changes. By assuming a quadratic transistor model. Horowitz[12] was able to formulate a pseudolinear current-voltage relationship for transistors operating in the linear region. This relationship has been used to develop timing models for TC networks[12]. The pseudolinear transformation is reviewed in Section 5.2.

Although the pseudolinear transformation does not remove the nonlinearity of a transistor, it allows the voltage of a node in a TC network to be formulated in a way similar to that in an RC network. Sections 5.3 and 5.4 rely on this property to construct charge-sharing models for nondriven and driven TC networks respectively.

5.2 Transistor as a Pseudolinear Device

The current through the drain and source terminals of a MOS transistor is not linearly related to the voltages associated with these terminals. Horowitz[12] noticed that when a transistor is in the linear region, its drain-source current, based on the quadratic transistor model, is linearly related to some function $g(V_D, V_S)$ of the terminal voltages. Moreover, the function is separable with respect to the two parameters V_D and V_S: $g(V_D, V_S) = f(V_D) - f(V_S)$. As a result, the drain-source current and the transformed voltage $f(V)$ of a terminal node are linearly related. This transformation, unfortunately, does not totally remove the nonlinearity of a TC circuit because linear capacitors become nonlinear devices in the transformed domain. Thus, such transformation is called pseudolinear.

Using an nMOS transistor as an example, the drain-source current i_{DS} is equal to

$$i_{DS} = \mu_n C_{ox}\frac{W}{L}(V_{GS} - V_{th} - \frac{V_{DS}}{2})V_{DS} = (1 - V_{th})\frac{f_n(V_D) - f_n(V_S)}{R_{eff}}$$

where all voltages are normalized by the power supply. The function $f_n(V_D)$ is equal to $1 - [1 - V_D/(1 - V_{th})]^2$, and likewise for the function $f_n(V_S)$. R_{eff} is equal to

$2L/[W\mu_n C_{ox}(1 - V_{th})]$ and, up to the first order, only varies with the dimensions of the transistor. Since R_{eff} is the proportional constant in the above expression, it has the semantics of an effective resistance. Due to a threshold drop between the gate and source terminals of a MOS transistor, noninput nodes of a transistor circuit with exclusively n-type devices have a range between 0 and $1-V_{th}$. Assuming a step gate input, one can see that V_{DS} of any transistor in such a circuit is always less than or equal to its corresponding $V_{GS} - V_{th}$: thus, all conducting transistors operate in the linear region.

The pseudolinear current-voltage relationship of a pMOS transistor is symmetric to that of an nMOS transistor, but the source and drain labelings are reversed. For a pMOS transistor, the drain-source current is equal to

$$i_{DS} = -\mu_p C_{ox}\frac{W}{L}(V_{GS} - V_{th} - \frac{V_{DS}}{2})V_{DS} = -(1 - |V_{th}|)\frac{f_p(V_D) - f_p(V_S)}{R_{eff}}$$

where $f_p(V_D) = 1 - [1-(1-V_D)/(1-|V_{th}|)]^2$, and likewise for $f_p(V_S)$. The threshold voltage V_{th} of a pMOS device is negative, hence, $|V_{th}|$ is actually equal to $-V_{th}$. R_{eff} in this expression is equal to $2L/[W\mu_p C_{ox}(1 - |V_{th}|)]$. Noninput nodes of a circuit with exclusively pMOS devices range between $|V_{th}|$ and 1 due to a gate-source threshold difference. When a step gate input is assumed, $V_{DS} \geq (V_{GS} - V_{th})$ for all transistors, thus they all operate in the linear region.

Since pMOS technology can easily be mapped[1] to nMOS technology, the following presentation will be in nMOS only. Furthermore, in order to simplify notations, all voltages will be normalized by $1 - V_{th}$, and symbol U will be used to represent the transformation of a normalized voltage; for instance, $U_D = 1 - (1 - V_D)^2$.

[1]This can be accomplished by replacing each pMOS device by an nMOS device with proper adjustments in R_{eff}'s (to compensate for the differences in the hole and electron mobilities and the magnitudes of threshold voltages) and reversing the polarity of each node (replacing V by $1 - V$).

5.3 Nonlinear Pure Charge Sharing

Modeling pure charge-sharing transistor networks consists of two sub-problems: the determination of waveform shapes and the approximation of time constants. These concerns are shared by the linear pure charge-sharing model, hence, insights can be gained by looking at the physical basis of that model.

In essence, the linear pure charge-sharing model maps a node in an RC network to a node in a two-capacitor-one-resistor network. The output of this model is an exponential, and the value of its time constant is set such that the areas under the true output and the model output match. An equivalent technique for a nonlinear model would be to map a node in a TC network to a node in a two-capacitor-one-transistor network. Adopting this equivalent technique is appropriate because, in spite of the differences between transistors and resistors, a waveform in a pure charge-sharing MOS network usually has a simple shape and is dominated by a single time constant.

For TC networks, the area under the true output is, in general, impossible to find because of transistor's nonlinear current-voltage relationship. Fortunately, the aforementioned nonlinear transformation from V to U allows a limited application of superposition. Equation 4.1 can be rewritten for a TC network as

$$U_e - U_r = -\sum_k R^r_{ke} C_k \frac{dV_k}{dt}$$

where R^r_{ke} is set by the R_{eff} of the path instead of the resistance of the path as in the linear model. The area between U_e and U_r in the time domain is equal to

$$\int_0^\infty U_\epsilon - U_r \, dt = \tau_{A^r_\epsilon}. \qquad (5.1)$$

The idea is to find $\int_0^\infty U_\epsilon - U_f \, dt$ and to use it to approximate the time constant for the U waveform at node ϵ. Since there is a one-to-one mapping between a U

and its corresponding V waveforms. the time constant for the V waveform can be estimated as well.

5.3.1 Shapes of Waveforms

The falling and rising waveforms for nodes in a two-capacitor-one-transistor network can be solved explicitly:

$$V = \begin{cases} 1 - \frac{(1-V_f)\tanh(t/\tau)}{(1-V_f)+V_f\tanh(t/\tau)} & \text{(for the falling node)} \\ \frac{V_f\tanh(t/\tau)}{(1-V_f)+V_f\tanh(t/\tau)} & \text{(for the rising node)} \end{cases} \tag{5.2}$$

where $\tau = RC_H$ (R is the R_{eff} of the connecting transistor, and C_H is the capacitance of the node which is originally charged high). Although falling and rising waveforms of nodes in an arbitrarily complex TC network can have much more complicated formulas. the functions in Equation 5.2 can roughly depict any falling or rising node because. to first order, all falling nodes fall at approximately the same rate and so do rising nodes.

These two waveforms are monotonic. and they approach V_f asymptotically from the opposite sides. In the U-domain, where $U = 1-(1-V)^2$, the transformations of these two waveforms are also monotonic, and they approach $U_f = 1-(1-V_f)^2$ asymptotically from the opposite sides. Equation 5.2 and its transformation will be used to approximate the voltage and U-domain waveforms in any pure charge-sharing MOS network.

5.3.2 Approximate Time Constant

Even though Equation 5.1 looks very similar to Equation 4.2, the charge-conservation technique (multiplying both sides of Equation 4.2 by the capacitance C_ϵ of node ϵ and summing over corresponding equations from every node) used to separate nodes ϵ from r in Equation 4.2 cannot be applied to Equation 5.1. That is because U is a function of V^2, hence $\sum_\epsilon C_\epsilon U_\epsilon$ is a function of the total energy in the system. Unlike the total charge, the total energy stored in all capacitors of a pure charge-sharing network does not conserve.

Fortunately, in the proposed scheme, the time constant of a waveform is approximated by its area in the time domain, and there are countless waveforms that have the same area. Hence, for the purpose of approximating a time constant, it is not necessary to find the exact shape of a waveform, but any waveform G which has the same area as U can be used. In addition, if it is possible to find a particular G function, P, which is linearly related to V, then conservation of charge can be applied to a set of $\int_0^\infty P_\epsilon - P_\tau \, dt = \tau_{A_\epsilon^\tau}$ equations to decouple nodes ϵ and τ.

In general, P is impossible to define without knowing the area under U, but finding the area under U is the reason P needs to be defined. However, for a two-capacitor-one-transistor network, its P function, U^*, can be defined for both the rising and falling nodes because the system can be solved analytically; see Equation 5.2. In this case,

$$U^* = \begin{cases} F(V_f)(V - V_f) + U_f & (\text{for } V \geq V_f) \\ R(V_f)(V - V_f) + U_f & (\text{for } V \leq V_f) \end{cases} \tag{5.3}$$

can be obtained by substituting Equation 5.2 to $\int_{V(0)}^{V_f} (U - U^*) \frac{dt}{dV} dV$ and setting the latter to zero. It is not surprising that, instead of linear in V, U^* is piecewise linear. This is because the shapes of the rising and falling waveforms of a two-capacitor-one-transistor network are quite different. The two portions of Equation 5.3 are monotonic and they approach U_f asymptotically from the opposite sides just like the transformations of Equation 5.2. These monotonic and asymptotic properties simplify the problem: the area between the transformation of the rising waveform and U_f is equal to $\int_0^\infty R(V_f)(V - V_f) \, dt$ (similarly for the falling waveform). The

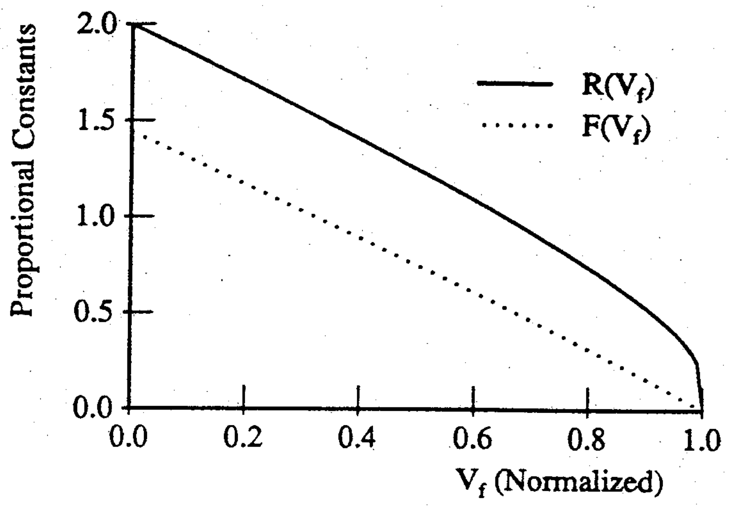


Figure 5.1: Proportional constants as functions of the normalized final voltage.

proportional constants of Equation 5.3

$$F(V_f) = \frac{2V_f(1-V_f)}{2V_f-1} + \frac{1-V_f}{\ln[2(1-V_f)]} \quad (\text{if } V_f \neq 0.5;\ F(0.5) = 0.75)$$

and

$$R(V_f) = \frac{2(1-V_f)^2}{1-2V_f} - \frac{V_f}{\ln[2(1-V_f)]} \quad (\text{if } V_f \neq 0.5;\ R(0.5) = 1.25)$$

are functions of the final voltage only, and they are plotted in Figure 5.1.

Although U^* is nothing more than the P function of the trivial case (a two-capacitor-one-transistor network), it has been decided previously that waveforms in any network usually resemble those in a trivial case. Hence, Equation 5.3 can, and will, be used to approximate the P function of any node in any network. Since $\int_0^\infty P_\epsilon - P_r\, dt = \int_0^\infty U_\epsilon - U_r\, dt = \tau_{A_\epsilon^r}$,

$$\int_0^\infty U_\epsilon^* - U_r^*\, dt \approx \tau_{A_\epsilon^r}. \tag{5.4}$$

After Equation 5.4 is collected for every node, conservation of charge can be applied to the sum of the weighted CU^* products. This technique is fundamentally the same as summing over the unweighted CV products in constructing the linear pure charge-sharing model. The trick in weighting CV^* products is to multiply the CV^* product of a rising node by $F(V_f)$ and the CV^* product of a falling node by $R(V_f)$ such that both products are proportional to $F(V_f)R(V_f)(V - V_f)$. The sum of the weighted products is

$$\begin{aligned}\int_0^\infty F(V_f)\sum_k C_k^l(V_k^* - V_r^*) + R(V_f)\sum_k C_k^h(V_k^* - V_r^*)\,dt \\ \approx F(V_f)\sum_k C_k^l\tau_{A_k^r} + R(V_f)\sum_k C_k^h\tau_{A_k^r}.\end{aligned}$$

When Equation 5.3 is substituted into the above equation, the left-hand side can be simplified to $(C_L F(V_f) + C_H R(V_f))\int_0^\infty V_f - V_r^*\,dt$ where $C_L = \sum_k C_k^l$ and $C_H = \sum_k C_k^h$. Thus,

$$\int_0^\infty V_f - V_r^*\,dt \approx \frac{F(V_f)\sum_k C_k^l\tau_{A_k^r} + R(V_f)\sum_k C_k^h\tau_{A_k^r}}{C_L F(V_f) + C_H R(V_f)},$$

and it provides the necessary information to do decoupling: $\int_0^\infty V_\epsilon^* - V_f\,dt$ for any node ϵ can be computed by combining $\int_0^\infty V_\epsilon^* - V_r^*\,dt$ with $\int_0^\infty V_f - V_r^*\,dt$.

The estimated waveform V_ϵ^* for node ϵ has the shape given by Equation 5.2, and its time constant, determined by $\int_0^\infty V_\epsilon^* - V_f\,dt$ through combining $\int_0^\infty V_\epsilon^* - V_f\,dt$ with Equation 5.3, is as follows:

$$\tau_\epsilon = \begin{cases} \dfrac{(C_L\tau_{A_\epsilon^r} - \sum_k C_k^l\tau_{A_k^r}) + \frac{R(V_f)}{F(V_f)}(C_H\tau_{A_\epsilon^r} - \sum_k C_k^h\tau_{A_k^r})}{(1-V_f)^2 C_T} & \text{(for falling nodes)} \\ \dfrac{\frac{F(V_f)}{R(V_f)}(\sum_k C_k^l\tau_{A_k^r} - C_L\tau_{A_\epsilon^r}) + (\sum_k C_k^h\tau_{A_k^r} - C_H\tau_{A_\epsilon^r})}{V_f(1-V_f)C_T} & \text{(for rising nodes)} \end{cases}.$$

Just like its counterpart in the linear model, this time constant is independent of the reference node. Since the time constants depend on the ratio between $F(V_f)$ and $R(V_f)$, this ratio is plotted in Figure 5.2.

This nonlinear model provides accurate estimates for networks with a single dominant time constant; for circuits with only two capacitors, the estimate is exact. On the other hand, the model fails the same class of circuits that causes problems to the linear model. The reason is that the nonlinear model also implicitly assumes

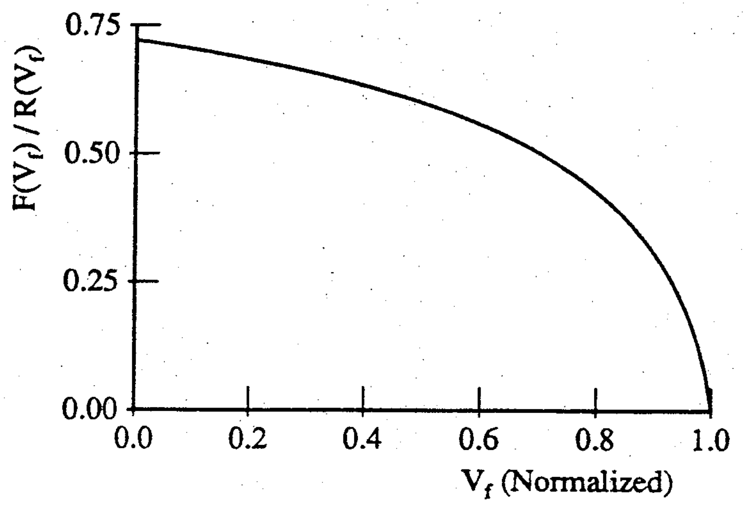


Figure 5.2: The ratio between the proportional constants as a function of the normalized final voltage.

that voltage waveforms are monotonic. Hence, it can occasionally give poor, but conservative, estimates to nonmonotonic waveforms. It is even more difficult to improve the nonlinear model than to improve the linear model because the concept of poles and zeros cannot be applied to nonlinear devices. In order to safeguard the consequence of a breakdown in the model, the aforementioned time constant can be modified as $\max(0, \tau_e)$.

The nonlinear model has been applied to the barrel shifter shown in Figure 4.2, and the result is compared with SPICE simulation in Figure 5.3. The excellent match is because the barrel shifter really has only one dominant time constant, even though the circuit has three capacitors.

5.4 Nonlinear Charge Sharing with a Driven Path

This section introduces a charge-sharing model for driven TC networks. The major concern in modeling TC networks is the nonlinearity of MOS devices, which usually

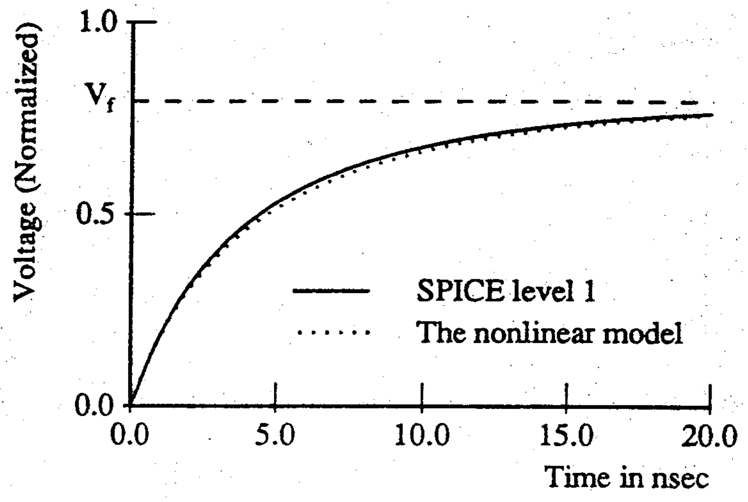


Figure 5.3: The nonlinear model versus SPICE simulation for node N of the barrel shifter shown in Figure 4.2.

has a decisive effect on the shape of a waveform. For example, neither the rising nor the falling waveform in a TC network is exponential in nature[12] as opposed to those in an RC network. Furthermore, the general shapes of waveforms in a pure charge-sharing TC network (described in Equation 5.2) are also quite different from those in an RC network.

The simplest driven TC network with a charge-sharing problem is a two-capacitor-two-transistor circuit shown in Figure 5.4, where C_2 is initially charged to a different polarity from the voltage source V (V can be either 0 or 1), and the transistor T_2 is being switched on. Even in such a simple case, the voltage spike at node 1 can have quite different shapes depending on the type of the MOS devices and the voltage level of the driver. This circuit configuration deserves special attention because it is statistically the most common charge-sharing-with-a-driven-path scenario (77.5 percent of the dynamic clusters in MIPS-X processor's execution unit have two or fewer transistors).

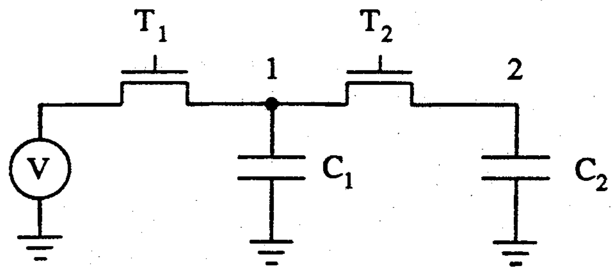


Figure 5.4: The simplest driven TC network that has a charge-sharing problem.

5.4.1 The Trivial Case

The trivial configuration (the two-capacitor-two-transistor configuration) is simple enough that it can be analyzed directly. Unfortunately, as far as the author knows, there is no closed-form solution for its output waveforms. Hence, they must be solved numerically.

Before attempting to solve the output waveforms, it is useful to put the circuits in a normal form. With reference to Figure 5.4, assume, without loss of generality, that the voltage source V is at the ground level. Let R_1 and R_2 be the R_{eff}'s of T_1 and T_2 respectively, and let $\alpha = R_1/(R_1 + R_2)$ and $\beta = C_1/(C_1 + C_2)$ such that both α and β lie between 0 and 1. A circuit parameterized by α and β is shown in Figure 5.5, and it is the normal form of the circuit shown in Figure 5.4. The proof in Appendix C shows that voltage V_{N_ϵ} at node N_ϵ in Figure 5.5 and voltage V_ϵ at node ϵ in Figure 5.4 are related as follows:

$$V_{N_\epsilon}(t') = V_\epsilon(RCt')$$

where $R = R_1 + R_2$, $C = C_1 + C_2$, and t' is dimensionless.

Since voltage waveforms at corresponding nodes of any two networks with the same normal form are different only by a stretching factor in time, voltage spikes in two such circuits will have the same amplitude. This property reduces the number of variables in a two-capacitor-two-transistor network from four to two, and it allows

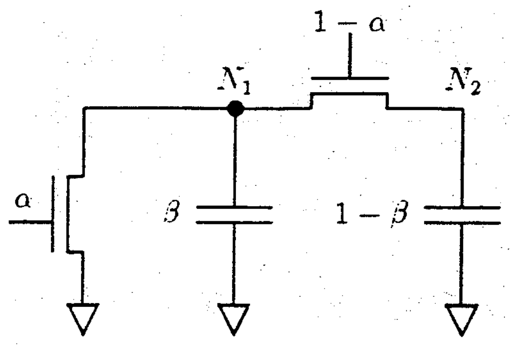


Figure 5.5: A normalized network.

a simulator to use a table, indexed by α and β, to look up the amplitude of a voltage spike.

The nonlinear differential equations associated with the circuit shown in Figure 5.5 can be solved numerically using schemes such as the fourth order Runge-Kutta method[13]. Figure 5.6 plots the maximum voltage fluctuation, as a function of α and β, of the voltage spike at node N_1. The color code used in the plot is explained in Figure 5.7. Jags on the curves in Figure 5.6 are due to the low resolution of the numerical data set.

When comparing the voltage spike in an nMOS two-capacitor-two-transistor network driven to the ground to that in a corresponding RC network, which is shown in Figure 5.8, one can see that the former has a lower amplitude. The differences in the two plots can be explained qualitatively by looking at nMOS transistor's current-voltage relationship. When a step gate input is assumed, the drain-source current (i_{DS}) and the drain-source voltage (V_{DS}) of an nMOS transistor are related as shown in Figure 5.9. The plot shows that i_{DS} decreases with a higher V_S (source voltage) or a lower V_{DS}, therefore, an nMOS transistor is more effective in discharging a capacitor than in charging one. For example, in Figure 5.5, capacitor C_1 (labeled with β) is charged through transistor T_2 (labeled with $1-\alpha$) and discharged through transistor T_1 (labeled with α). When the voltage of C_1

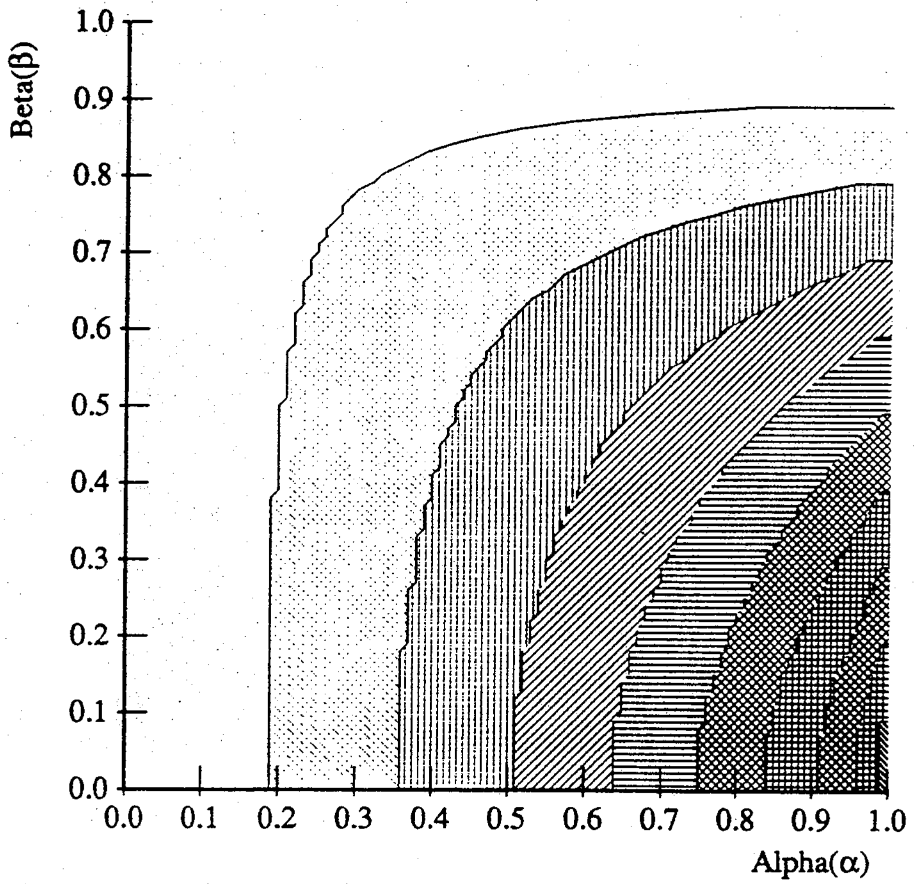


Figure 5.6: The maximum voltage fluctuation of a spike in a normalized nMOS network driven to the ground.

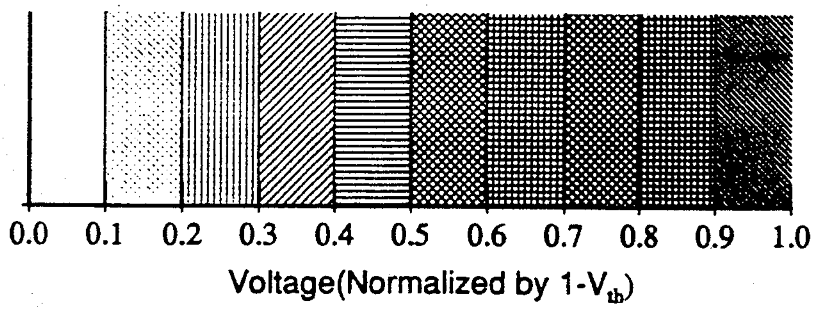


Figure 5.7: Color code for voltage-fluctuation plots.

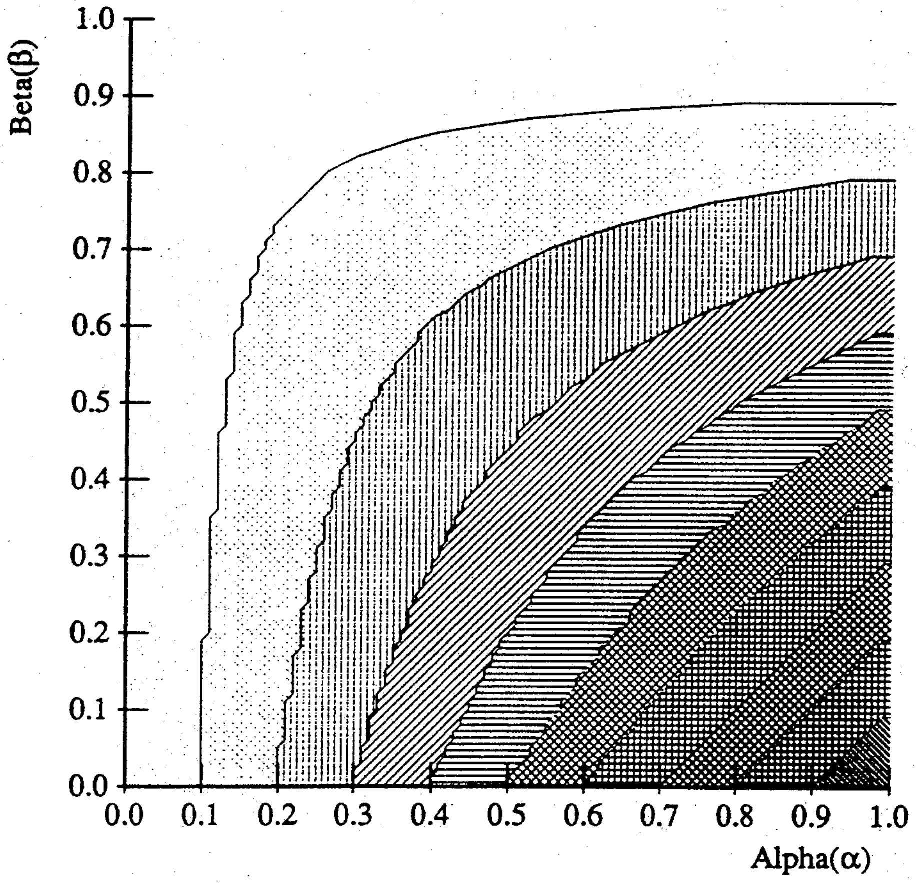


Figure 5.8: The maximum voltage fluctuation of a spike in a normalized RC network (driven to either Vdd or the ground).

increases, the current conductivity of T_1 increases while that of T_2 decreases. As a result, C_1 cannot be charged to as high a voltage as that in the corresponding RC network.

In contrast, if a two-capacitor-two-transistor nMOS network is driven by Vdd, then the amplitude of its voltage spike is expected to be higher than that of a corresponding RC network. This conjecture is verified by the numerical result shown in Figure 5.10.

Although Figures 5.6 and 5.10 are derived from the nMOS technology, they can also be applied to pMOS networks. Thus, only one set of plots is required for both

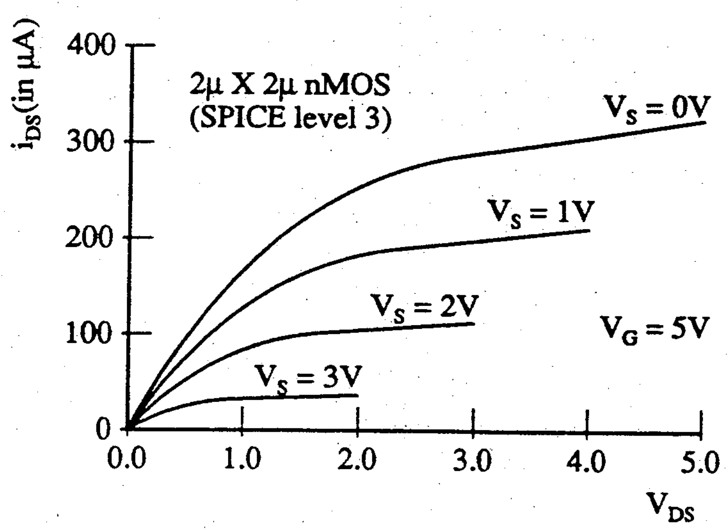


Figure 5.9: The drain-source current (i_{DS}) of an nMOS transistor as a function of its drain-source voltage (V_{DS}) for different source voltages (V_S).

the nMOS and pMOS technologies.

5.4.2 Nontrivial Cases

When a driven network consists of more than two capacitors and two transistors, its precise charge-sharing outcome is too complicated to find without a circuit-level simulator. However, Section 4.4 has shown that it is possible to model a nontrivial RC network by mapping it to a two-capacitor-two-resistor network. The output of the model can closely approximate the real output because most circuits' spikes are dominated by a pair of time constants. Since the two-time-constant characteristic holds for transistor networks as well, it is reasonable to apply the modeling-by-mapping strategy here.

For the linear model described in Section 4.4, the mapping is done through matching the first-order and the second-order geometric characteristics of a voltage waveform (namely the area and the first moment). For a nonlinear model, it is more convenient to do the mapping in the transformed domain because the nonlinear

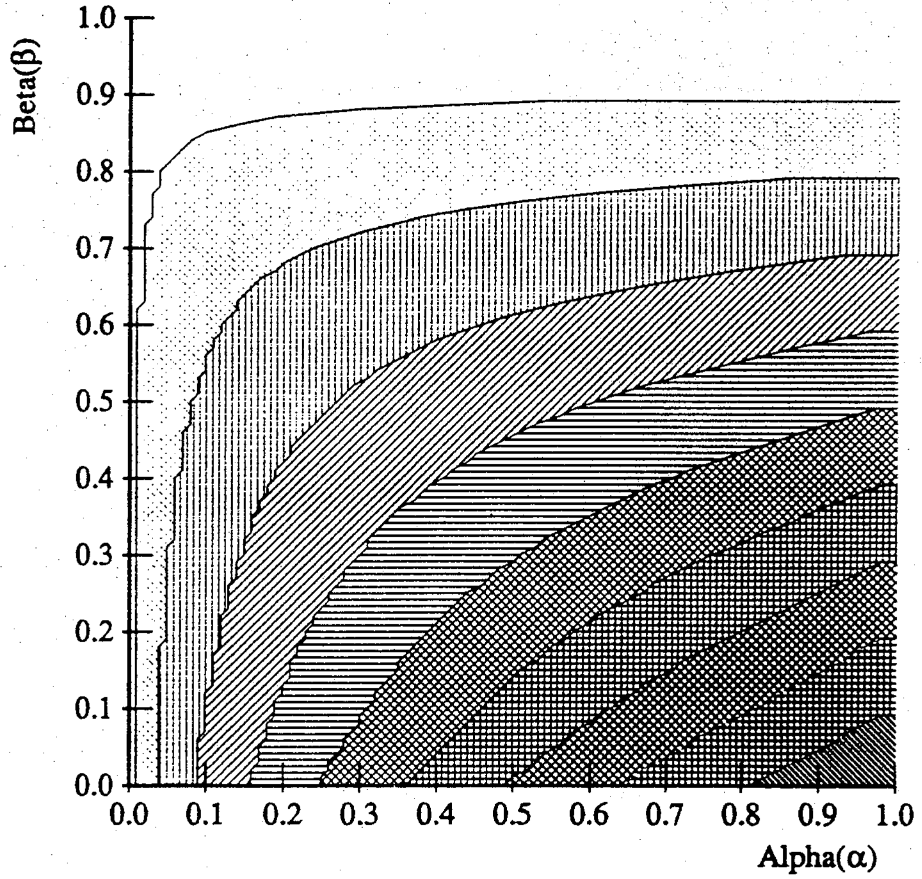


Figure 5.10: The maximum voltage fluctuation of a spike in a normalized nMOS network driven to Vdd.

current-voltage equations can be written in a pseudolinear form. For example, given a TC tree driven to the ground, the transformed waveform U_ϵ at any node ϵ can be written as

$$U_\epsilon = -\sum_k R_{k\epsilon} C_k \frac{dV_k}{dt}$$

where $R_{k\epsilon}$ is set by the R_{eff} of the path to the ground shared by both nodes k and ϵ. Integrating both sides of the above equality, one can see that the area under U_ϵ in the time domain is equal to $\sum_k R_{k\epsilon} C_k$.

Unfortunately, without having the exact shape of U_ϵ, it is impossible to derive $\int_0^\infty V_\epsilon \, dt$ from the transformed-domain area. As a result, the first moment of U_ϵ

$$\int_0^\infty t U_\epsilon \, dt = \sum_k R_{k\epsilon} C_k \int_0^\infty V_k \, dt$$

cannot be determined. Thus, the linear modeling technique cannot be directly applied to transistor networks. Yet its result still provides useful insights to the mapping process because the technique's dependency on the second-order information only supplements but not invalidates its dependency on the first-order information.

As a matter of fact, the first-order information alone makes some major modeling decisions for the linear model. This is best illustrated by constructing a simpler linear model using exclusively first-order intuitions and comparing its result with the more elaborate linear model described in Section 4.4. For node ϵ of an RC network shown in Figure 5.11, one can construct a charge-sharing model as follows. To the zeroth order, the charging tree is just a large capacitor whose capacitance is set by the sum of all the charging-tree capacitors. Since charging-tree capacitors discharge through R_{cc} (which is the resistance of the path between the ground and the node connecting the charging tree and the driving tree), R_{cc} needs to be included in the first-order model to connect the charged capacitor to the ground. In order to customize the model for node ϵ, the waveform characteristics at node ϵ have to be emphasized. There is only one point on R_{cc} of the first-order model whose voltage

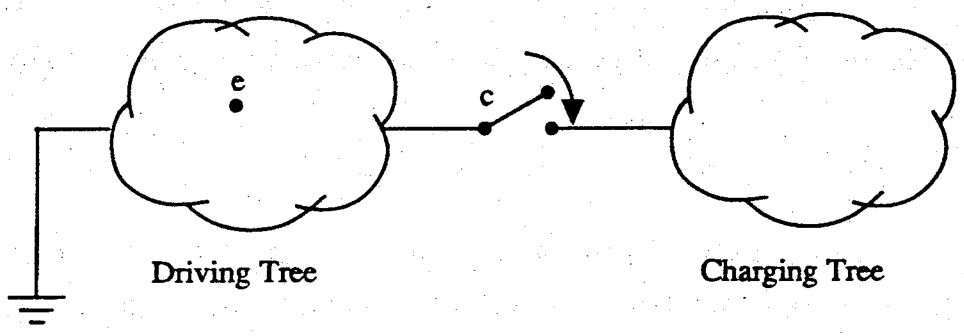


Figure 5.11: A modeling example.

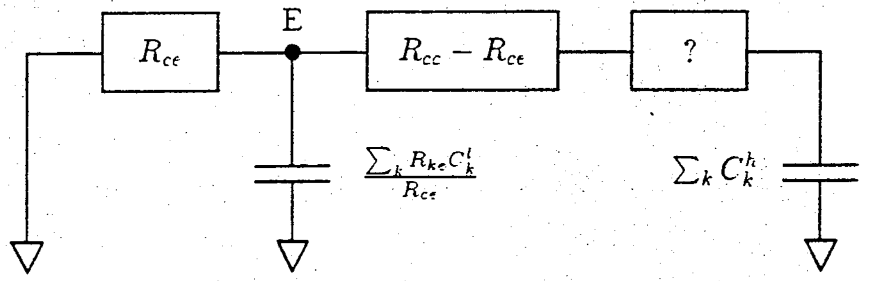


Figure 5.12: A charge-sharing model based on first-order intuitions.

waveform has the same time-domain area as that of node ϵ. This point is located at $R_{c\epsilon}$ away from the ground because the area under V_ϵ is equal to $\sum_k R_{k\epsilon} C_k^h$. and since all C_k^h's are in the charging tree, $R_{k\epsilon}$'s are all equal to $R_{c\epsilon}$. Let this point be labeled as node E; see Figure 5.12.

Node E needs some capacitance to catch the collective effect of the driving-tree capacitors being modeled. This capacitance can be approximated, to the first order, by matching the Elmore's delays at nodes ϵ and E. In this context, the Elmore's delay is the time-domain area under a waveform if all capacitors are initially charged high. In order to match the Elmore's delay $\sum_k R_{k\epsilon} C_k$ at node ϵ, the capacitance at node E has to be equal to $(\sum_k R_{k\epsilon} C_k^l)/R_{c\epsilon}$.

Up to this point, a two-capacitor-two-resistor model has been constructed. This model is obviously missing something because the resistors in the original charging tree have not been incorporated. The voltage spike at node e can be overestimated if all the charging-tree capacitors are modeled as being lumped at node c when they are actually not. This shortcoming can be improved by inserting a resistive element as shown in Figure 5.12. The size of this resistor can be determined by looking at the functionalities of the charging-tree capacitors. As far as charge sharing is concerned, charging-tree capacitors are charge suppliers. Thus, if they are modeled as one capacitor, then this capacitor should share some common charge-supplying characteristics with the original charging tree. By looking at the total charge $Q = \sum_k C_k^h V_k$ in the charging tree as a function of time, one can define charge-supplying characteristics as geometric waveform characteristics of the charge-supplying waveform. This definition is analogous to using the voltage-waveform characteristics to define the corresponding node's voltage characteristics.

The first-order charge-supplying characteristic is the time-domain area under the charge-supplying waveform; a definition similar to that of the first-order voltage-waveform characteristic. Mathematically, the area under a charge-supplying waveform is equal to

$$\int_0^\infty Q\, dt = \int_0^\infty \sum_k C_k^h V_k\, dt = R_{cc} \sum_k C_k^h + \sum_k C_k^h (\sum_j R_{jk}^c C_j^h)$$

where R_{jk}^c is the resistance of the path to node c shared by both nodes j and k. In order to match this area, a resistor of size

$$\frac{\sum_k C_k^h (\sum_j R_{jk}^c C_j^h)}{(\sum_k C_k^h)^2}$$

has to be inserted into the first-order model as shown in Figure 5.12.

This first-order model is actually very similar to the elaborate model presented in Section 4.4. The latter model is dissected in Appendix D. Comparing the two models, one can see that they are only differed by one component — a resistor. This resistor ($R_{2_2} = \frac{\sum_k R_{k\epsilon} C_k^l (R_{ck} - R_{c\epsilon})}{R_{c\epsilon} \sum_k C_k^h}$) is in Figure D.1 but it is absent from Figure 5.12. The interpretation of R_{2_2} is nonintuitive because its value is determined by the sum of second-order resistive terms.

In order to evaluate the differences between the first-order and the elaborate models. a detailed analysis of how the amplitude of a voltage spike varies with R_{2_2} is carried out in Appendix E. The conclusion is that R_{2_2} plays a relatively minor role in determining the amplitude. Thus, the first-order model is quite sufficient in modeling charge sharing in driven RC networks.

A similar first-order model can be constructed for transistor networks. It is easiest to do this in the transformed domain in which transistors are characterized by R_{eff}'s and voltages are characterized by U's. Using node ϵ of the TC network shown in Figure 5.11 as an example. one can argue that. to the first order. charging-tree capacitors act like one device which discharges through R_{cc} to the ground. In order to match the said node's U-waveform area and the Elmore's delay. the first-order model should have a capacitor equal to $(\sum_k R_{k\epsilon} C_k^l)/R_{c\epsilon}$ locating at $R_{c\epsilon}$ away from the ground just like its linear counterpart; see Figure 5.12.

Even though the capacitors of a TC network's charging tree still act as charge suppliers in the charge-sharing context. their charge-supplying characteristics are not as easy to find as that in an RC network. For instance, there is not sufficient information to find the first-order charge-supplying characteristic ($\int_0^\infty Q\, dt$) because each node k is characterized by U_k instead of V_k. However, U_k and V_k have a simple relationship, and all waveforms in the charging tree usually have approximately the same shape. Thus. by matching $\int_0^\infty \sum_k C_k^h U_k\, dt$, one can argue that $\int_0^\infty Q\, dt$ is also closely matched. As a result. the transistors in the charging tree can be modeled as one transistor with $R_{\epsilon ff} = \frac{\sum_k C_k^h (\sum_j R_{jk}^c C_j^h)}{(\sum_k C_k^h)^2}$.

In conclusion. first-order charge-sharing models for RC and TC networks have the same set of parameters. As a matter of fact, the charge-sharing model for driven

TC networks may as well use the set of parameters determined for RC networks in Section 4.4. This is because the simulation results summarized in Appendix F show that R_{2_2}, which is the only difference between the two sets of parameters, does not play an important role in the nonlinear modeling either.

The first-order and the elaborate models are applied to the read/write network of the RAM cell shown in Figure 4.4. Since the cluster of interest has only one noninput node in its driving tree, R_{2_2} of the elaborate model is equal to zero. Consequently, the two models are identical, and the result from either model can perfectly match SPICE's prediction (at level 1) of the real output.

5.5 Summary

Charge-sharing problems in MOS designs are complicated by transistors' nonlinearity. Since the conductivity of a transistor varies with the signal level being passed, it is not always reasonable to statically assign a set of resistances to a transistor. On the other hand, the voltage-current equations of transistor networks are in general somewhat intricate to formulate. Fortunately, if transistors in the same cluster are of the same type and the switching transistor has a step gate input, then these transistors operate exclusively in their linear regions.

Assuming a quadratic transistor model, Horowitz noticed that the drain-source current of a transistor operating in the linear region is linearly related to some nonlinear function of the transistor's terminal voltages. Although this observation does not change the essence of the problem, it helps to formulate the voltage of a TC circuit in a fashion similar to that of an RC circuit.

This chapter presents a pure charge-sharing model for TC networks. This model includes two functions to describe the rising and falling waveforms of a transistor network. The time constant of the model is determined by the network's circuit elements through matching waveform characteristics.

Charge-sharing problems in driven TC networks are also discussed. The conjecture is that voltage spikes in complicated TC networks share many waveform characteristics with those in a two-capacitor-two-transistor network. Thus, a first-order approximation can be made to "map" a node from a complex network to a two-capacitor-two-transistor case. Even though two-capacitor-two-transistor networks do not have closed-form solutions, numerical results can be used to determine their amplitudes.

Chapter 6

Implementation of a Switch-Level Simulator

6.1 Overview

In previous chapters, ways to improve switch-level models have been presented. This chapter describes how to efficiently implement these models.

Simulators have to handle more than the idea loop-free single-driver setting assumed throughout most of Chapters 2, 4, and 5. Terman[26] has proposed a simple scheme to randomly break a nontree network to a loop-free network. Even though his scheme is ad hoc, it is usually adequate for simulating MOS designs because nontree networks are so rare. On the other hand, networks with multiple drivers seem to be a more common and more important problem. For example, when more than one pull-down of a NOR gate is conducting simultaneously, the charge on the output capacitor can be drained through multiple paths. It is not always possible to straightforwardly merge multiple driving paths because there can be capacitors on the paths. Fortunately, multiple-driver problem is in principle very similar to the single-driver problem, and Section 6.2 suggests a simple method to transform the former problem to a more familiar single-driver form.

Section 6.3 describes the actual implementation of nRSIM — a new switch-level simulator which incorporates all the models presented in this thesis. It shows that in spite of the complexity of the models, the implementation only requires simple data structures and little computation.

6.2 Modeling Multiple Drivers

The timing and charge-sharing models presented in this thesis are based on finding the area and the first moment of an output waveform. The most widely used derivation assumes a single driving source. Lin[15] has proposed a multiple-driver derivation, which is quite general but rather complicated. This section reviews Lin's work and introduces another solution that is less general but much simpler.

Lin's LRD (Load ReDistribution) algorithm is based on the block Gauss-Seidel method for solving a system of linear equations. The algorithm can be applied to both tree and nontree networks. The basic idea behind the LRD algorithm is to carefully convert a general network to a set of single-driver tree networks such that nodes in the latter set of networks are indistinguishable, as far as their delays are concerned, from the corresponding nodes of the original network. The LRD algorithm consists of two steps. In the first step, the general network is topologically decomposed into a set of independent subnetworks. Each subnetwork is a tree and has one and only one voltage source. This step is referred to as *tree decomposition.* The decomposition process involves splitting nodes that cause loops or that connect multiple drivers together. Split nodes are referred to as *secondary nodes*, while the original nodes are called *primary nodes.* The second step determines how the capacitors associated with the split nodes are split, and it is known as *load redistribution.* Load redistribution is a relaxation process which distributes the capacitance at a primary node to the corresponding secondary nodes such that the delays at all secondary nodes are eventually equal to one another. During each relaxation step. Elmore's delays are computed for all secondary nodes. The time complexity of load redistribution is proportional to the product between the number of relaxation steps and the number of secondary nodes. The number of relaxation steps is controlled by the required accuracy. Since each split tree has its own driver. all drivers in the original network must have the same voltage: otherwise. a group of secondary nodes split from the same primary node can end up with different voltages.

Even though the LRD algorithm is capable of solving nontree as well as tree networks. most networks found in MOS designs are free of loops. It turns out that for multiple-driver tree networks, there is a scheme much simpler than the LRD algorithm to calculate timing and charge sharing. The following sections present this scheme.

6.2.1 Current Distribution in a Resistor Tree

For a single-driver resistor tree, the voltage induced at a node due to a current source at another node is a function of the resistance of their shared path towards the driver. Unfortunately, the "shared path towards the driver" is topologically ill-defined when there is more than one driver. However, there is also a nontopological way of looking at path sharing. Assume, without loss of generality, that a loop-free network has only one driver: the ground. The voltage at any node e is equal to the product between the current through that node and the resistance between that node and the ground. If there is no current passing through node e, then the node has the same voltage as its neighbors. For example, if a current source i at node k is farther away from the ground than node e as shown in Figure 6.1 (a). then all its current passes through node e. Hence, the voltage V_e at node e is equal to $R_{ee}i$.

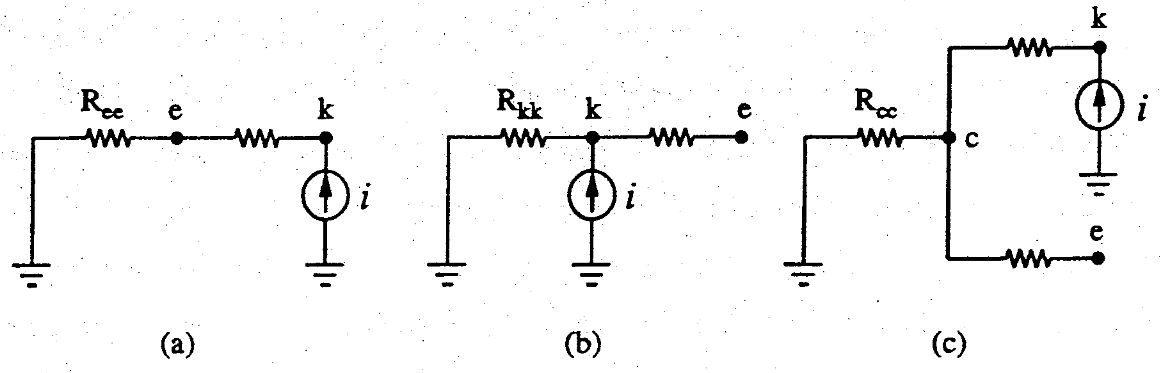


Figure 6.1: Voltage induced at node ϵ due to a current source at different positions relative to the ground.

In contrast. if the current source is closer to the ground than node ϵ as depicted in Figure 6.1 (b). then V_ϵ is equal to the voltage at node k because there is no current between nodes ϵ and k. In the third variation. when nodes ϵ and k appear on different branches as shown in Figure 6.1 (c). voltages at nodes ϵ and c are equal. and they are determined by the current discharging through resistor R_{cc}.

The voltage-current relationship can be generalized to circuits with multiple voltage sources. Assume that a resistor tree has several ground drivers and has a current source at node k. The current through each resistor can be solved by Kirchhoff's current law. and the voltage at any node is equal to the sum of current-resistance products along any path from that node to the ground.

Computing current through each resistor can be tedious: fortunately. the following transformation provides a systematic way to do that. With reference to the systems shown in Figure 6.2. the transformation states that the current source i at node Y can be replaced by a current source that has

$$i' = \frac{R_k}{R + R_k}\, i$$

at node X without changing the voltage V_X at node X.

It is worth noting that the transformation says nothing about the voltage at node

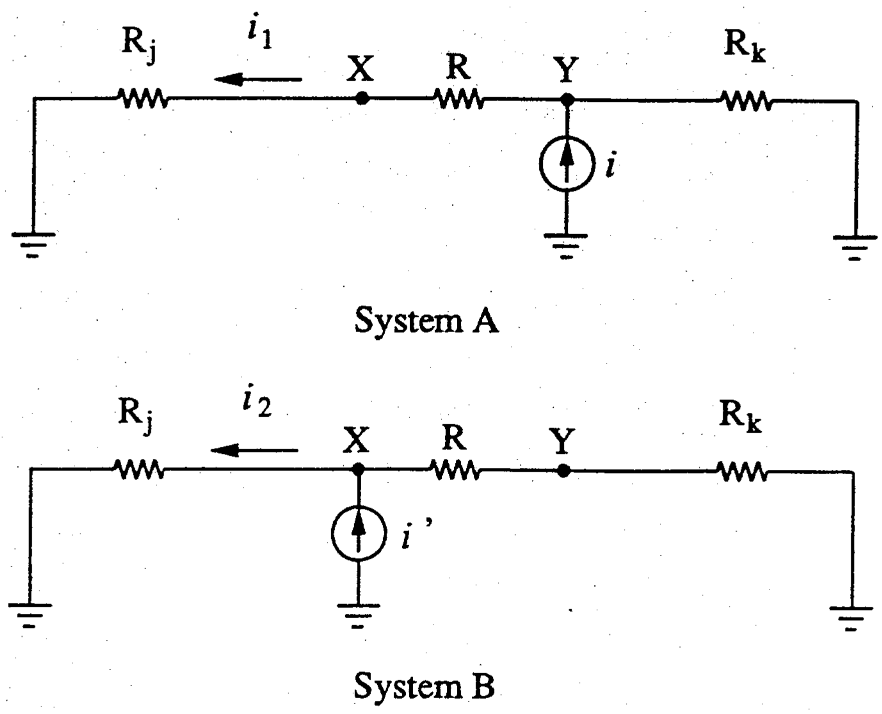


Figure 6.2: By properly adjusting its value, it is possible to move current source i from node Y to node X without changing the voltage at node X.

Y. As a matter of fact, the transformation is not transparent to node Y. Hence, systems A and B are not entirely equivalent, and system A cannot be regenerated from system B through the same transformation.

This transformation can be generalized to handle more complex circuits. For example, if R_j is replaced by n resistors R_{j_1}, R_{j_2}, ..., R_{j_n} in parallel, then as long as $R_{j_1} \parallel R_{j_2} \parallel \ldots \parallel R_{j_n} = R_j$, the replacement is transparent to both nodes X and Y. Similarly, R_k can be replaced by m resistors R_{k_1}, R_{k_2}, ..., R_{k_m} in parallel where $R_{k_1} \parallel R_{k_2} \parallel \ldots \parallel R_{k_m} = R_k$. After replacing these two resistors, the systems in Figure 6.2 look like those in Figure 6.3. According to the transformation, V_X of system A and V_X of system B are equal in Figure 6.3, hence $i_{1_1} = i_{2_1}$, $i_{1_2} = i_{2_2}$, ..., $i_{1_n} = i_{2_n}$. In other words, the transformation is not only transparent to node X but also to the network which is downstream from node X. In this example, downstream refers to the network which is to the left of node X (vice versa for upstream). Since the transformation only needs the knowledge of the upstream network, it is independent of the configuration of the downstream network.

The above transformation provides a systematic way to compute the voltage induced at one node (the destination node) due to a current source at another node (the source node): by repeatedly applying the transformation to move the current source from the source node to the destination node along the shortest path between the two nodes[1] and multiplying the resultant current by the resistance between the destination node and the ground. If there is more than one current source in the system, then the operation can be repeated for each current source one at a time according to superposition theorem. In the actual implementation of this algorithm, current sources which share their transformation paths can share their transformation information as well. This technique and its complexity will be discussed later.

6.2.2 Application of Current Distribution to RC Networks

Assume, without loss of generality, that a loop-free resistor network has multiple ground drivers. Also assume that there is a capacitor C_k at node k which is charged high initially. The current i_k coming out of the capacitor sets the voltage waveform V_ϵ at any node ϵ in the network. The relationship between V_ϵ and i_k must be linear, according to Ohm's law, and can be written as $V_\epsilon = R_{k\epsilon} i_k$. The proportional constant $R_{k\epsilon}$ can be determined using the above transformation.

To apply the transformation, the capacitor C_k has to be replaced by an independent current source with $i = -C_k \frac{dV_k}{dt}$. It has been shown that an independent

[1]It is important to do transformations along the shortest path such that the destination node is always downstream from the source node.

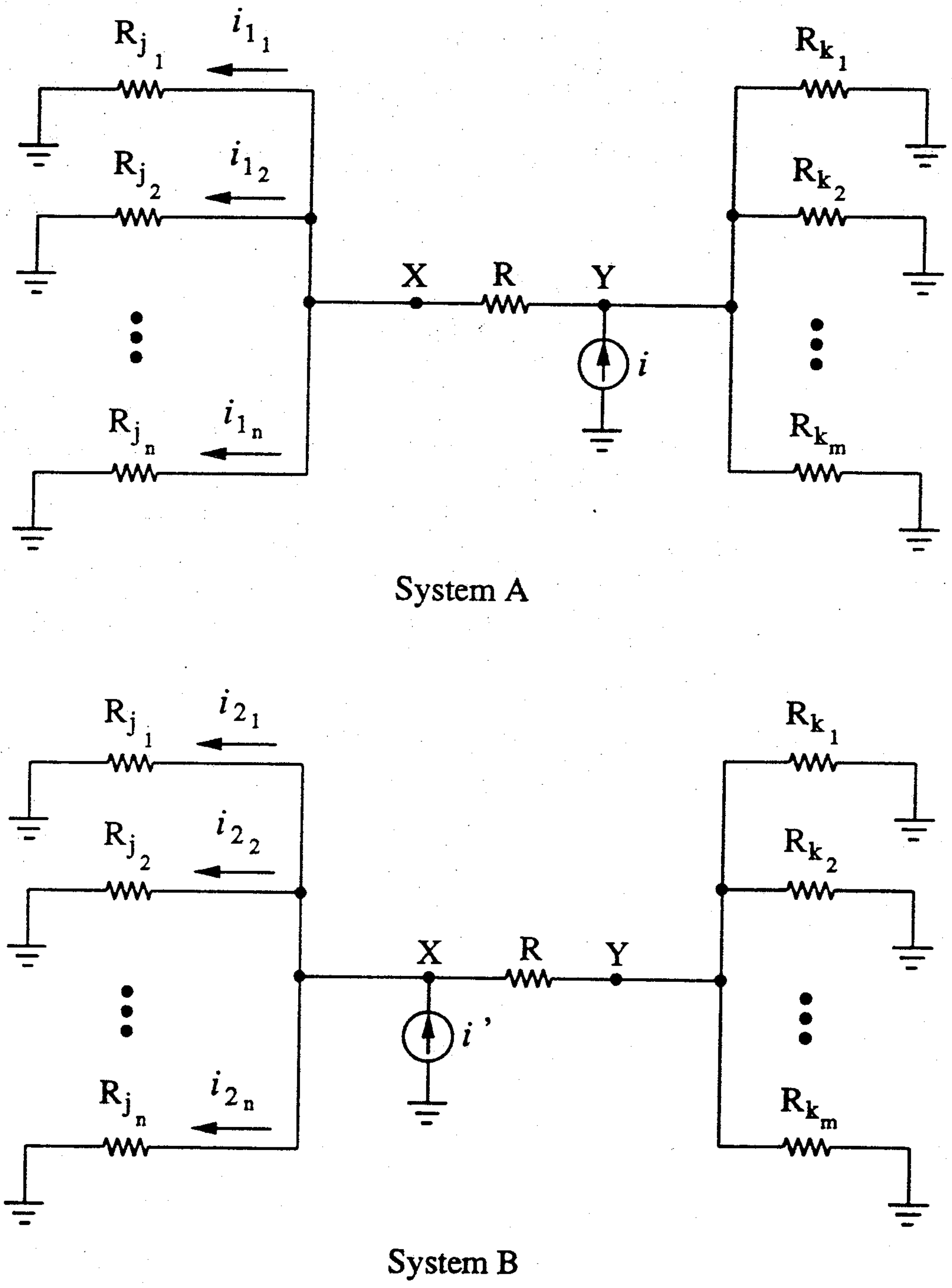


Figure 6.3: Systems generalized from those in Figure 6.2.

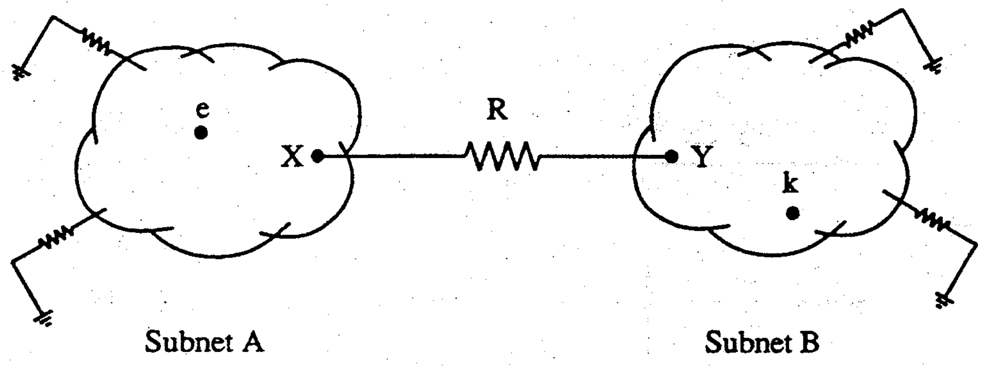


Figure 6.4: A transformation example.

current source can be transformed and moved from the source node to the destination node without disturbing the voltage at the destination node. Let ϵ be the destination node in Figure 6.4. and let X and Y be two nodes on the path between nodes k and ϵ. In order to preserve the voltage at node ϵ while moving a current source from node Y to node X. the value of the current source has to be adjusted by an adjustment factor. A_{Y-X}. where

$$A_{Y-X} = \frac{\text{Thévenin resistance at node Y contributed by subnet } B}{R + \text{Thévenin resistance at node Y contributed by subnet } B}. \tag{6.1}$$

When the current source is finally moved to node ϵ. the product of all the adjustment factors ($\prod_{j=k\longrightarrow\cdots\longrightarrow\epsilon} A_j$) together with the effective resistance between node ϵ and the ground (R_ϵ) set the proportional constant:

$$R_{k\epsilon} = R_\epsilon \prod_{j=k\longrightarrow\cdots\longrightarrow\epsilon} A_j \tag{6.2}$$

The method can be extended to networks with multiple capacitors using superposition — each capacitor can be thought of as an independent current source with the same current characteristic as the capacitor. Hence, the voltage at node ϵ is equal to

$$V_\epsilon = \sum_k R_{k\epsilon} i_k = -\sum_k R_{k\epsilon} C_k \frac{dV_k}{dt}.$$

As a matter of fact, the above derivation can be applied to TC networks as well. According to Section 5.2, transistors are linear devices in the pseudolinear transformed domain. Even though capacitors become nonlinear in the new domain, they can still be treated as independent current sources. Thus, the linear derivation is still valid, and

$$U_\epsilon = -\sum_k R_{k\epsilon} C_k \frac{dV_k}{dt}$$

where $R_{k\epsilon}$ is set by $R_{\epsilon ff}$'s instead of resistances.

6.2.3 Multiple-Capacitor Multiple-Driver Example

This section presents a step-by-step execution of the above algorithm on an example. Through this example, the complexity of the algorithm can be better understood.

Assume that all capacitors in Figure 6.5 are charged high initially. In order to compute the Elmore's delay at node ϵ, all capacitors have to be transformed and moved to node ϵ. The Elmore's delay at node ϵ, according to the multiple-driver $R_{k\epsilon}$ definition given by Equation 6.2, is

$$\tau_{D_\epsilon} = \sum_k R_{k\epsilon} C_k = R_\epsilon \sum_k \left[C_k \prod_{j=k-\cdots-\epsilon} A_j \right].$$

The expression shows that there are actually two ways to look at the delay computation. One way is to find the effective resistances ($R_{k\epsilon}$'s) of the common paths to all drivers shared by all capacitors and the node of interest (node ϵ), and the other way is to find the "effective capacitance" ($\sum_k \left[C_k \prod_{j=k-\cdots-\epsilon} A_j \right]$) seen at the node of interest. The later approach is easier to implement, and it will be followed from now on.

Since both C_1 and C_2 are on the path between nodes a and ϵ, these two capacitors can share some common transformation information. As a matter of fact, the circuit can be partitioned into three subcircuits as shown such that C_1 and C_2 are upstream

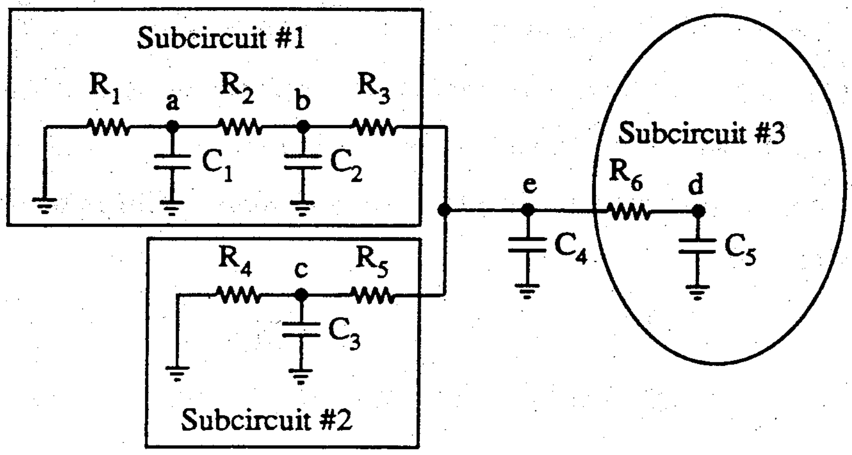


Figure 6.5: A multiple-capacitor multiple-driver example.

from node ϵ in one direction while C_3 and C_5 are upstream from node ϵ in two other directions.

Adjustment factor required to move a current source from node a to node b is equal to $A_{a\rightarrow b} = \frac{R_1}{R_1+R_2}$. Similarly, $A_{b\rightarrow\epsilon} = \frac{R_1+R_2}{R_1+R_2+R_3}$. Consequently, for subcircuit #1, the effective capacitance is equal to $(C_1A_{a\rightarrow b} + C_2)A_{b\rightarrow\epsilon}$. This information is not only useful for node ϵ, but it can also be used to compute the time constants for nodes c and d.

Subcircuit #2 can be handled the same way as subcircuit #1. Subcircuit #3 does not have a driver, hence, its effective resistance to the ground is equal to infinite. As a result, the adjustment factor $A_{d\rightarrow\epsilon}$ is equal to 1. Thus, C_5 can be moved to node ϵ without any adjustment.

While computing the adjustment factors, the effective resistance of each path to the ground is also gathered as a byproduct (for example, $R_1+R_2+R_3$ of subcircuit #1 and R_4+R_5 of subcircuit #2). This information can then be used to compute R_e, which is equal to $(R_1+R_2+R_3) \parallel (R_4+R_5)$.

This example shows that it is straightforward to collect information from a multiple-capacitor multiple-driver setting, and the information collected for one node can also be used by other nodes.

6.3 Implementation Algorithm

This section shows that all models introduced in this thesis can be implemented with algorithms that are direct extensions of those used in the original RSIM. The new implementation is called nRSIM, which adopts RSIM's event-driven skeleton and user interface in order to avoid duplicating Terman's effort.

When RSIM starts up, it reads in a circuit and a set of simulation vectors. If these vectors change the logical state of a node, then all transistors with gates connected to this node change their conductivities. As a result, the source and drain terminals of these transistors need to be reevaluated. RSIM and nRSIM are different in their evaluation algorithms, and the implementation of nRSIM's evaluation algorithm is the focus of this section. If the evaluation results in new changes, then the new changes are scheduled. The iteration terminates when the network is stablized or a prespecified simulation time limit is reached.

The smallest unit that nRSIM does its analysis is an electrically connected cluster of transistors. The analysis consists of two parts: the evaluation of final value as described in Chapter 3 and the scheduling of events as described in Chapters 2, 4, and 5. Since the implementation algorithm for a nondriven cluster is very similar to that for a driven cluster, only the latter will be discussed.

6.3.1 Evaluation of Final Value

The algorithm shown in Figure 6.6 computes the final value of a driven cluster. It calls the function final_value, which is shown in Figure 6.7, on each and every node of the cluster. Final_value implements the improved resistor-divider model

```
1.  for each node $n$ in cluster $c$ {
2.     compute $n$'s new logical state by calling final_value($n$);
3.     reset VISITED flags for all nodes;
4.     if $n$ does not have a definite path {
5.        compute $n$'s state from its charge information
6.        if $n$'s driving result $\neq$ $n$'s charging result
7.           $n$'s new logical state is X;
8.     }
9.     if a voltage spike is possible at $n$
10.       set $n$'s SPIKE flag;
11. }
```

Figure 6.6: Algorithm to compute the final value of a driven cluster.

described in Section 3.5. It returns a data structure of the following form:

$$\{type, R_{U_l}, R_{U_h}, R_{D_l}, R_{D_h}\}$$

where *type* is either the definite or the indefinite type as defined in Section 3.5, and R_{U_l}, R_{U_h}, R_{D_l}, and R_{D_h} are the corresponding parameters of the definite or indefinite block.

Function series_op in step 12 of Figure 6.7 implements the series operation as defined in Equation 3.1. Function parallel_op in step 13 implements the parallel operation as defined in Equations 3.2, 3.3, and 3.4 depending on the types of its operands. The VISITED flag in steps 9 and 11 forces the network traversal to expand outward from the starting node, hence, it can break resistor loops[26].

If a node's type is indefinite, then the node may not have a driven path. As a result, its charge information has to be taken into account (step 5 of Figure 6.6). Terman suggested an algorithm to compute the maximum and the minimum voltages from the charge stored in the cluster's capacitors. In order to find the maximum voltage due to charge sharing for node n, his algorithm assumes that all X nodes are charged high, and collects the total charge (C_H_{max}) in the cluster. Then it

```
final_value($n$)
{
  if $n$ is Vdd
    this $\longleftarrow$ {definite, 0, 0, $\infty$, $\infty$};
  else if $n$ is ground
    this $\longleftarrow$ {definite, $\infty$, $\infty$, 0, 0};
  else {
    this $\longleftarrow$ {indefinite, $\infty$, $\infty$, $\infty$, $\infty$};
    mark $n$ as VISITED;
    for each non-OFF transistor $t$ with *source* connected to $n$
      if *drain* is not marked as VISITED {
        other $\longleftarrow$ series_op(final_value(*drain*), $t$);
        this $\longleftarrow$ parallel_op(this, other);
      }
  }
  return(this);
}
```

Figure 6.7: Function to implement the improved resistor-divider model.

collects the total capacitance (C_L_{min}) from all capacitors that are in low state and that are reachable from node n through ON transistors. The ratio

$$\frac{C\_H_{max}}{C\_H_{max} + C\_L_{min}}$$

determines the maximum voltage at node n due to charge sharing. Terman's algorithm to compute the minimum voltage due to charge sharing uses the same principle.

If the charge-sharing result of an indefinite node is different from that computed by the improved resistor-divider model, then the node's new state is X (step 7 of Figure 6.6).

If node n starts and ends at the same state and there are capacitors, in the same cluster, which are charged to a different polarity, then there is a chance that node n may have a voltage spike. In this case, node n needs a spike analysis (step 10 of

```
1.  for each state $s$ in [low, high, X] {
2.     if none of the nodes in cluster $c$ has new state = $s$
3.        continue; /* Skip to the next iteration. */
4.     for each node $n$ in $c$ {
5.        compute $n$'s time constants by calling compute_tau($n$, $s$);
6.        reset VISITED flags for all nodes;
7.        if $n$'s present state $\neq$ $n$'s new state
8.           schedule $n$'s driven event;
9.     }
10.    for each node $n$ in $c$ that has (new state = $s$ and SPIKE flag set) {
11.       compute $n$'s $\tau_{P_n}$;
12.       compute the amplitude of $n$'s spike from $\tau_{A_n}$, $\tau_{D_n}$, and $\tau_{P_n}$;
13.       if $n$ has a voltage spike
14.          schedule $n$'s spike event;
15.       reset $n$'s SPIKE flag;
16.    }
17. }
```

Figure 6.8: Algorithm to schedule events for a driven cluster.

Figure 6.6).

6.3.2 Schedule of Events

The algorithm shown in Figure 6.8 schedules events according to the final states computed previously. Due to X transistors or unusual situations, there is a slim chance that nodes in the same cluster can end up with different final states. Thus, the algorithm will iterate on the three possible final states (step 1 of Figure 6.8).

Assume that some of the nodes in cluster c are driven to state s. The algorithm calls compute_tau, which is shown in Figure 6.9, to compute a set of time constants for each and every node of the cluster. Compute_tau takes the node (n) and its final state (s) as parameters, and it returns a data structure which can be used to compute τ_{A_n} and τ_{D_n} as defined in Chapters 4 and 5. Time-constant computation is complicated by X transistors, which can create more than one electrical configuration from each cluster. In order to be conservative in scheduling, the worst case timing scenario has to be identified. For example, if a node is driven to either Vdd or the ground, then the worst case scenario is that the node has the longest delay such that a potential critical path can be discovered. In contrast, if the node is driven to X, then the worst case scenario is for that node to change as soon as possible because X's are undesirable in simulation.

Three resistances,[2] R_{min}, R_{dom}, and R_{max}, are collected for each node by compute_tau. They are the effective resistances between the node and the drivers in different electrical configurations, and they are used for different purposes. R_{min} is used to compute the delay for nodes that are driven to X. Since X is an intermediate state, all voltage sources are considered as drivers. Hence, R_{min} is the resistance to all sources through non-OFF transistors (i.e. all X transistors are considered as conducting). On the other hand, R_{max} is used to compute the delay for nodes that are driven to either Vdd or the ground. Assume that node n is driven to the ground, then R_{max} is the resistance from node n to the ground through ON transistors (i.e. all X transistors are considered as nonconducting). In case there are Vdd nodes in n's cluster (for instance, if n is the output of an nMOS inverter driving low), then the Vdd nodes are considered as open circuits because they play minor roles compared to the ground (vice versa if n is driven to Vdd). In this example, the ground is called the ***dominant driver*** of node n while Vdd is called the ***secondary driver***.

Indiscriminately turning ON and OFF X transistors yields the minimum and the maximum resistances to the driving sources respectively. However, in order to conservatively compute the effective capacitances for time constants, a third

[2]In this algorithm, "resistance" refers to either the resistance of RC networks or R_{eff} of TC networks.

```
compute_tau($n$, $s$)
{
   if $n$ is an input node {
      if $s = X$ or $n$'s state $= s$
         this.$R \longleftarrow \{0, 0, 0\}$;
      else
         this.$R \longleftarrow \{0, \infty, \infty\}$;
      this.$C_{adj} \longleftarrow \{0, 0\}$;
   }
   else {
      this.$R \longleftarrow \{\infty, \infty, \infty\}$;
      if $n$'s state $\neq s$
         this.$C_{adj} \longleftarrow$ {cap($n$), cap($n$)};
      else
         this.$C_{adj} \longleftarrow$ {0, cap($n$)};
      mark $n$ as VISITED;
      for each non-OFF transistor $t$ with $source$ connected to $n$
         if $drain$ is not marked as VISITED {
            other $\longleftarrow$ transmit($drain$, compute_tau($drain$, $s$), $t$);
            this.$R \longleftarrow$ parallel(this.$R$, other.$R$);
            this.$C_{adj} \longleftarrow$ this.$C_{adj}$ + other.$C_{adj}$;
         }
   }
   return(this);
}
```

Figure 6.9: Function to compute the delay time constants.

variation of handling X transistors and secondary drivers is required to define the worst case circuit configuration from capacitors' point of view. For example, assume that nodes n and m are connected by an X transistor, and node n is also being driven to the ground through another path. If both nodes are charged high initially, then in order to compute the worst case delay for node n, the capacitor at node m should be included. In other words, the X transistor between nodes n and m should be considered as conducting for n's delay calculation. Thus, R_{max} is unsuitable for this purpose. On the other hand, secondary drivers should be considered as open circuits by nodes that are driven to either Vdd or the ground: hence, R_{min} is inappropriate either. Consequently, a third resistance, R_{dom}, is defined. R_{dom} assumes that all X transistors are conducting, but all secondary drivers are open circuits. R_{min}, R_{dom}, and R_{max} with respect to X transistors and secondary drivers are summarized in the following table:

	X transistors	Secondary drivers
R_{min}	ON	Not applicable
R_{dom}	ON	Open circuit
R_{max}	OFF	Open circuit

In order to compute τ_{A_n} and τ_{D_n}, two effective capacitances, C_A and C_D, are also collected for each node by compute_tau. C_A is defined to be the ratio between τ_{A_n} and the resistance between node n and the dominant driver. Similarly, C_D is equal to the ratio between τ_{D_n} and the same resistance. For example, if node n is driven to the ground, then $C_A = \tau_{A_n}/R_{max}$ where $\tau_{A_n} = \sum_k R_{kn}(C_k - C_k^l)$. The definition of τ_{A_n} is a little bit different from that defined in Chapters 4 and 5 because, in here, capacitors at unknown states are assumed to be charged high such that the delay estimation can be conservative. Hence,

$$C_A = \sum_k \left[(C_k - C_k^l) \prod_{j=k\longrightarrow\cdots\longrightarrow n} A_j \right].$$

```
1.  transmit($n$, accumulated, $t$)
2.  {
3.      if $t$'s state is unknown
4.          new.$R$ $\longleftarrow$ accumulated.$R$ + $\{R_{eff}(t), R_{eff}(t), \infty\}$;
5.      else
6.          new.$R$ $\longleftarrow$ accumulated.$R$ + $\{R_{eff}(t), R_{eff}(t), R_{eff}(t)\}$;
7.      if $t$'s state is unknown and accumulated.$C_{adj}.C_A = 0$
8.          new.$C_{adj}$ $\longleftarrow$ $\{0, 0\}$;
9.      else
10.         new.$C_{adj}$ $\longleftarrow$ accumulated.$C_{adj} * \frac{\text{accumulated}.R.R_{dom}}{\text{new}.R.R_{dom}}$;
11.     return(new);
12. }
```

Figure 6.10: The transmit function called by compute_tau in Figure 6.9

Similarly,

$$C_D = \sum_k \left[C_k \prod_{j=k\text{---}\cdots\text{---}n} A_j \right].$$

A data structure which consists of

$$\{R = \{R_{min}, R_{dom}, R_{max}\};\ C_{adj} = \{C_A, C_D\};\}$$

is returned by compute_tau. As shown in Figure 6.9, steps 3 through 15 initialize the data structure according to the aforementioned assignments. Then compute_tau does a depth-first traversal to collect the effective resistances and the effective capacitances. The transmit function in step 19 is listed in Figure 6.10, which handles both the resistance of a series resistor (steps 3 through 6) and the adjustment factor as described in Equation 6.1 (step 7 through 10). Step 7 of the transmit function is particularly tricky. With reference to the scenario shown in Figure 6.11, if the capacitors in subnets A and C are discharged while those in subnet B are charged high initially, then in order to compute the worst case charge-sharing scenarios for nodes in subnet A, the capacitors in subnet C should be excluded.

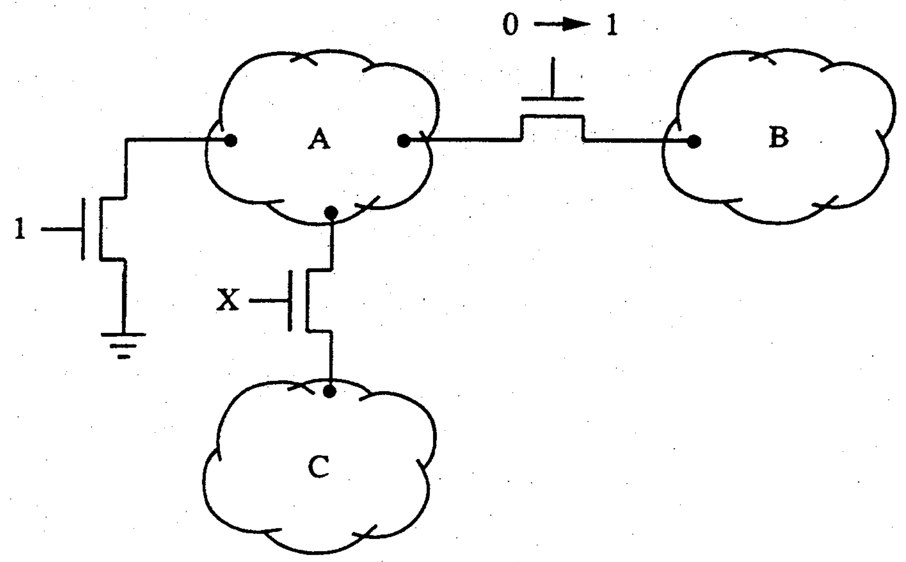


Figure 6.11: Scenario in which some capacitors should be excluded in charge-sharing calculation.

When compute_tau returns the set of R's and C's for node n, the scheduling algorithm schedules the driven event. In case a driven charge-sharing analysis is required, then τ_{P_n} will be computed. Computing τ_{P_n} is very similar to computing τ_{A_n} or τ_{D_n}, and it can be done by a function similar to compute_tau.

6.3.3 Complexity of the Algorithm

Terman[26] has done a thorough analysis of RSIM's complexity, and his result can also analyze the work in this section. In essence, final_value and compute_tau are two core routines of the simulator. Both routines are based on a recursive depth-first traversal, hence, their complexities are directly proportional to the number of nodes in the cluster. Consequently, the final-value evaluation and the delay estimation for each node can be done in linear time.

Terman also suggested a caching scheme to improve the overall complexity. He observed that the data structures collected by final_value and compute_tau during

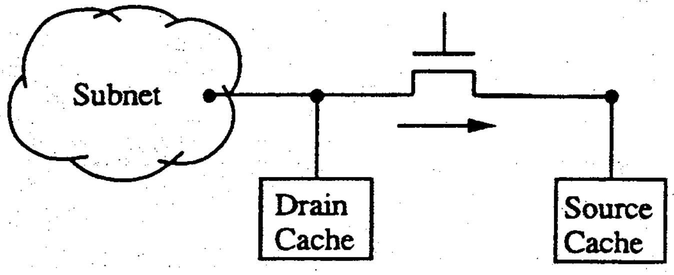


Figure 6.12: Associating a cache with a transistor to optimize computation.

each tree walk can actually be shared by other nodes. Thus. he proposed to store the information collected during each tree walk in caches which are associated with the source and drain terminals of transistors. For example, when a recursive algorithm has finished collecting parameters from the subnet shown in Figure 6.12. it stores the parameters in the cache associated with the drain of the transistor before passing it to the source. This cached information can be used by other nodes that require the same information from the same subnet. With this optimization, Terman has shown that the complete analysis of all nodes in the same cluster can be done in time that is proportional[3] to the number of transistors in the cluster. Interested readers can refer to [26] for further details.

6.4 Performance Evaluation

Even though the models used in nRSIM are much more sophisticated than those used in RSIM, they do not seem to slow down the simulator. On a per-cluster basis, nRSIM indeed takes longer to execute. Yet, it also provides more accurate predictions that eliminate fictitious events. Fictitious events, especially those due to charge sharing with a driven path, are quite common in RSIM. When not properly handled. fictitious events can severely slow down a simulator because they either

[3]More precisely, each transistor is crossed twice. One time in each direction to fill the cache.

trigger other fictitious events or put extra burden on scheduling.

The following experiment was done by people who simulated the MIPS-X processor. A functional simulator written by the members of the MIPS-X design team uses RSIM and nRSIM as the back-end to simulate the mask. In order to compare the performances of RSIM and nRSIM, Ackerman's function was run on the PC (program counter) unit of the processor. The simulation lasted for 334 MIPS-X machine cycles, and it took RSIM 49 minutes and 14 seconds (8.84 seconds/cycle) on a VAX 11/780 running Berkeley 4.3 UNIX. However, it only took nRSIM 45 minutes and 44 seconds (8.22 seconds/cycle) on the same machine. The slight speedup of nRSIM does not indicate a definite performance edge over RSIM; however, it shows that the new models are practical for switch-level simulators.

6.5 Summary

Timing and charge sharing in multiple-driver settings are not much different from those in the single-driver setting. A systematic mechanism has been proposed to transform a multiple-driver problem to the more familiar form assumed in previous chapters.

The models proposed in this thesis can be implemented with algorithms derived from RSIM. These algorithms use simple data structures and little computation. Since the new models can eliminate fictitious events with better accuracy, they do not slow down the speed of simulation despite greater complexity.

Chapter 7

Conclusions

Switch-level simulators work with high-level models instead of solving detailed differential equations. The precision of these high-level models determines the accuracy of their results. RSIM. a widely used simulator. models transistor networks as resistor-capacitor networks. and it applies simple approximations to estimate the outputs of these RC networks. Some of these approximations can be drastically improved by using slightly more complicated models. This thesis has identified two major sites for improvement, which cover both logic and timing aspects.

In the evaluation of logic, switch-level simulators are complicated by the presences of X transistors. which introduce uncertainties to the electrical configuration of a network. The logical value of a node is determined by the achievable voltages at that node. The difficulty with DC-voltage computation is how to get a conservative but non-pessimistic result. Voltage ranges and resistance ranges have always been used to specify the achievable voltages and the associated resistances at nodes; however, a closer inspection of how the relationship between voltages and resistance is represented in the existing schemes shows that the old schemes can produce undesirably pessimistic results.

A new scheme, the improved resistor-divider model, is then proposed. The new scheme is based on a simple parallel-and-series collapsing of resistor dividers. and it constructs the result at each step by minimizing the errors in the voltage-resistance solution space. The outcome of this scheme is guaranteed to be conservative and order independent.

Investigation in timing leads to better charge-sharing models. Charge sharing in a nondriven network determines not only the final state of the network but also the delay at each node. In a driven network. the charge stored in capacitors can introduce glitches before the system reaches equilibrium. Previously. both charge-sharing problems have been given low priorities because they are complex and rare. Although problems caused by improper charge sharing may only occupy a small percentage of all design problems. without accurate models. real charge-sharing problems may not be caught while fictitious charge-sharing events may be scheduled.

Two two-time-constant models have been proposed to model the charge-sharing scenarios. These models are based on the observation that voltage waveforms involving charge sharing are usually dominated by two time constants. The models determine the time constants by matching geometric waveform characteristics such as the area and the first moment. The model-by-matching-waveform-characteristic technique can be used by linear as well as nonlinear networks. However. waveforms in a nonlinear network may have different shapes from those in a linear network.

The models described in this thesis have been implemented into nRSIM. The actual implementation also handles multiple-driver configurations. With the additional accuracy provided by the new models. the new simulator can eliminate some fictitious events such that its speed performance is comparative to that of RSIM.

7.1 Future Work

The charge-sharing models described in this thesis assume that the gate input of the switching transistor is a step function, hence, transistors always operate in their linear regions. In reality, the input waveform is usually arbitrarily complex, and before it reaches some switching voltage, the switching transistor is only partially turned on. Before the switching transistor is fully turned on, the redistribution of charge is really controlled by the current conductivity of the switching transistor instead of the actual arrangement of the transistors and capacitors. The charge-sharing models in this thesis have not taken slow inputs into account.

Horowitz[12] suggested using simple ramps to model slow inputs. In his timing model for a gate driving an output network, delay consists of the gate delay due to the slow input and the intrinsic delay contributed by the output network. His model computes the gate delay by modeling the gate as a voltage-controlled current source. A similar approach may be taken here to model the switching transistor as a current source before the transistor reaches some switching point.

Another potential improvement is in delay estimation involving X transistors. The problem is fundmentally the same as that in the final-value computation — how to conservatively but non-pessimistically compute the delay. The present implementation uses the worst case transistor configuration and the worst case capacitor configuration. When these two configurations are based on very different connectivities, the delay estimation can be overly pessimistic.

The work in Chapter 3 has established a systematic way of solving a similar X-transistor problem by looking at the solution space formed by voltage and resistance. Research that follows the same line of thought may also solve the delay-computation problem.

Appendix A

Zero at the Origin

Proof is presented here to show that in a linear system, the network transfer function of a node starts and ends at the same voltage has a zero at the origin.

The voltage waveform of any node in a linear system consists of the sum of exponential functions. Assume that the steady-state voltage is V_0, the voltage V at any node can be expressed as $V = V_0 + \sum_i a_i \epsilon^{-t/\tau_i}$. Due to the steady-state condition, the sum of coefficients $(\sum_i a_i)$ is equal to 0.

The network transfer function of V can be written as

$$-\sum_i \frac{a_i}{1+\tau_i s} = -\frac{A_0 + A_1 s + A_2 s^2 + \cdots}{1 + B_1 s + B_2 s^2 + \cdots} .$$

Since A_0 is equal to $\sum_i a_i = 0$, the numerator of the network transfer function becomes $s(A_1 + A_2 s + \cdots)$. This proves that there is a zero at the origin.

Appendix B

Normalized Amplitude Function

With reference to Figure 4.8, the normalized amplitude function of V_E can be solved from two coupled linear differential equations, and it is equal to

$$f(N) = \frac{2}{1+\sqrt{1-4N}} \left(\frac{1-\sqrt{1-4N}}{1+\sqrt{1-4N}} \right)^{\frac{1-\sqrt{1-4N}}{2\sqrt{1-4N}}}$$

where

$$N = \frac{R_1 R_2 C_1 C_2}{[R_1 C_1 + (R_1 + R_2) C_2]^2} = \frac{(\tau_{P_e} - \tau_{D_e})(\tau_{D_e} - \tau_{A_e})}{\tau_{P_e}^2} .$$

Although N depends on all four circuits components, it can be represented by three variables: τ_{A_e}, τ_{D_e}, and τ_{P_e}. For physically realizable circuit components, $N \geq 0$ because R_1, R_2, C_1, and C_2 are all nonnegative. The maximum value of N is 1/4 because

$$\begin{aligned} & [R_1 C_1 - (R_1 + R_2) C_2]^2 + 4 R_1^2 C_1 C_2 \geq 0 \\ \Longrightarrow \quad & [R_1 C_1 + (R_1 + R_2) C_2]^2 - 4 R_1 R_2 C_1 C_2 \geq 0 \\ \Longrightarrow \quad & \tfrac{1}{4} \geq N . \end{aligned}$$

The normalized amplitude function is plotted in Figure B.1. Since the domain of the function is narrow, it can be pre-computed and stored in a look-up table.

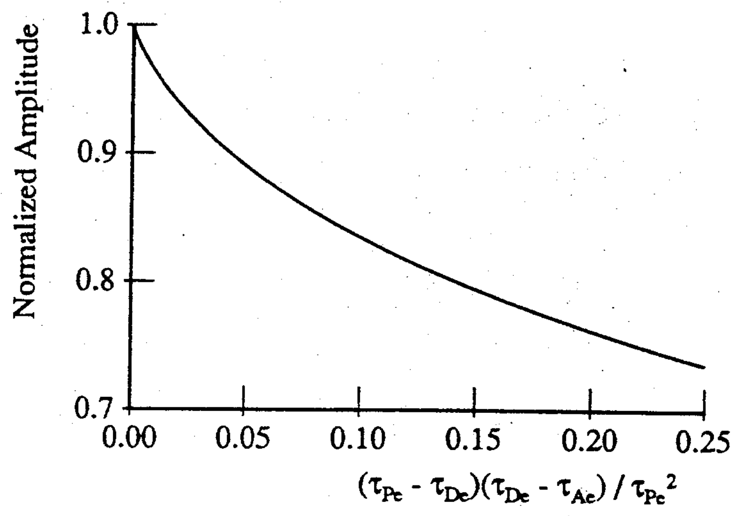


Figure B.1: Normalized amplitude as a function of τ_{A_e}, τ_{D_e}, and τ_{P_e}.

Appendix C

Proof of Waveform Similarity

This section proves that voltage V_{N_ϵ} at node N_ϵ in Figure 5.5 and voltage V_ϵ at node ϵ in Figure 5.4 are related as follows: $V_{N_\epsilon}(t') = V_\epsilon(RCt')$ where $R = R_1 + R_2$, $C = C_1 + C_2$, and t' is dimensionless.

For the network shown in Figure 5.4,

$$\begin{aligned} U_1(t) &= -R_1C_1\frac{dV_1(t)}{dt} - R_1C_2\frac{dV_2(t)}{dt} \\ U_2(t) &= -R_1C_1\frac{dV_1(t)}{dt} - (R_1+R_2)C_2\frac{dV_2(t)}{dt} \ . \end{aligned}$$

Let $t = RCt'$, then the above equations become

$$\begin{aligned} U_1(RCt') &= -\alpha\beta\frac{dV_1(RCt')}{dt'} - \alpha(1-\beta)\frac{dV_2(RCt')}{dt'} \\ U_2(RCt') &= -\alpha\beta\frac{dV_1(RCt')}{dt'} - (1-\beta)\frac{dV_2(RCt')}{dt'} \ . \end{aligned}$$

V_{N_1} and V_{N_2} of the normalized network can be formulated as

$$\begin{aligned} U_{N_1}(t') &= -\alpha\beta\frac{dV_{N_1}(t')}{dt'} - \alpha(1-\beta)\frac{dV_{N_2}(t')}{dt'} \\ U_{N_2}(t') &= -\alpha\beta\frac{dV_{N_1}(t')}{dt'} - (1-\beta)\frac{dV_{N_2}(t')}{dt'} \ . \end{aligned}$$

One can easily see from the above two sets of equations that $V_{N_\epsilon}(t') = V_\epsilon(RCt')$.

Appendix D

Dissection of the Linear Model

This section dissects the model discussed in Section 4.4, and interprets its components.

Let the two-capacitor-two-resistor circuit shown in Figure 4.8 be the reduced-network model of node ϵ in Figure 5.11. According to Equation 4.6, the components of the reduced network have the following values:

$$C_1 = \frac{\tau_{D_\epsilon} - \tau_{A_\epsilon}}{\tau_{A_\epsilon}} C_2$$

$$R_1 = \frac{\tau_{A_\epsilon}}{C_2}$$

$$R_2 = \frac{\tau_{P_\epsilon} - \tau_{D_\epsilon}}{C_2} .$$

Since the amplitude of a voltage spike in the reduced network is determined by the ***ratio*** of the two capacitors and the ***ratio*** of the two resistors, the exact value of C_2 is not important even though C_1, R_1, and R_2 are all functions of C_2. However, if a carefully selected value is assigned to C_2, the following analysis will be easier to understand. Thus, let C_2 be $\sum_k C_k^h$.

With some simple mathematical manipulations, one can show that

$$C_1 = \frac{\sum_k R_{k\epsilon} C_k^l}{R_{c\epsilon}} \text{ and } R_1 = R_{c\epsilon}$$

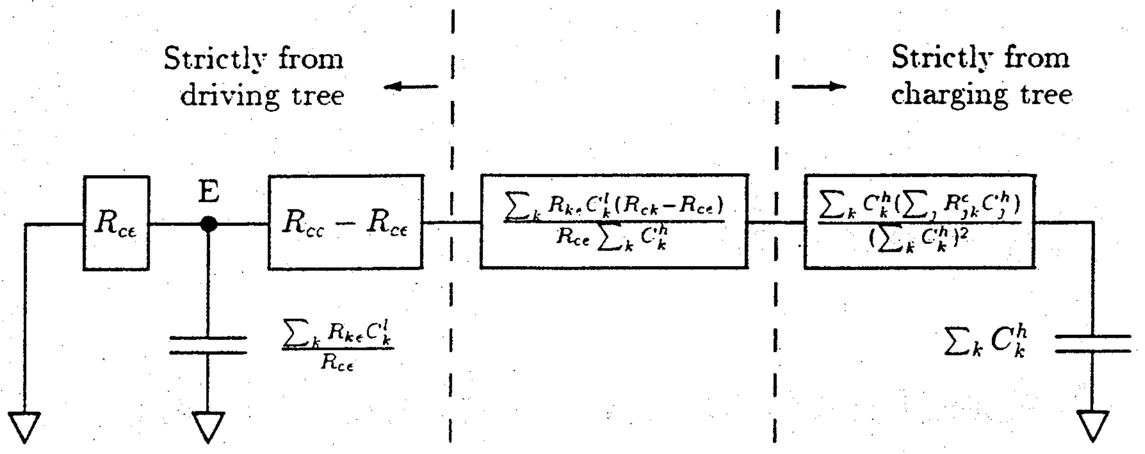

Strictly from driving tree
Strictly from charging tree
E
R_{ce}
$R_{cc} - R_{ce}$
$\frac{\sum_k R_{ke} C_k^l (R_{ck} - R_{ce})}{R_{ce} \sum_k C_k^h}$
$\frac{\sum_k C_k^h (\sum_j R_{jk}^c C_j^h)}{(\sum_k C_k^h)^2}$
$\frac{\sum_k R_{ke} C_k^l}{R_{ce}}$
$\sum_k C_k^h$

Figure D.1: The dissection of the linear charge sharing with a driven path model described in Section 4.4.

where c is the node connecting the driving tree with the charging tree. In addition, one can also show that R_2 is composed of the following three resistors in series:

$$R_{2_1} = R_{cc} - R_{ce}, \; R_{2_2} = \frac{\sum_k R_{ke} C_k^l (R_{ck} - R_{ce})}{R_{ce} \sum_k C_k^h}, \text{ and } R_{2_3} = \frac{\sum_k C_k^h (\sum_j R_{jk}^c C_j^h)}{(\sum_k C_k^h)^2}.$$

These components are shown in Figure D.1.

This dissection reveals one interesting fact: R_1, R_{2_1}, and C_1 can be derived strictly from components of the driving tree while R_{2_3} and C_2 can be derived strictly from components of the charging tree. R_{2_2} is the only component which relies on information from both trees; yet R_{2_2}'s dependency on the charging tree is merely its total capacitance. Thus, at this level of abstraction, a charging tree and its corresponding driving tree can almost be modeled separately.

Appendix E

Elaborate Model versus First-Order Model

This section compares the amplitude of a voltage spike found in Figure 5.12 to that found in Figure D.1.

R_{2_2} is the only element which is in one figure but not the other. However, the two circuits can still be identically the same if R_{2_2} is equal to zero. This can happen under many situations; for example, when the driving tree of the circuit being modeled has only one noninput node[1].

When R_{2_2} does not have a zero resistance, its value has a limited range. It is maximized when the node being modeled (node ϵ) is closer to the ground than all other nodes, and all other capacitors of the driving tree are located at node c; conversely, R_{2_2} is minimized. Mathematically,

$$\frac{\sum_k R_{k\epsilon} C_k^l (0 - R_{ce})}{R_{c\epsilon} \sum_k C_k^h} < R_{2_2} = \frac{\sum_k R_{ke} C_k^l (R_{ck} - R_{ce})}{R_{ce} \sum_k C_k^h} \leq \frac{\sum_k R_{k\epsilon} C_k^l (R_{cc} - R_{ce})}{R_{c\epsilon} \sum_k C_k^h}$$

or

$$-R_{c\epsilon} \frac{C_L}{C_H} < R_{2_2} \leq (R_{cc} - R_{ce}) \frac{C_L}{C_H}$$

[1]This includes the trivial configuration (i.e. two-capacitor-two-resistor circuits).

where $C_L = (\sum_k R_{k\epsilon} C^l_k)/R_{c\epsilon}$ and $C_H = \sum_k C^h_k$. The bounds are intentionally generous[2] such that the worst case scenarios can be evaluated more easily. If all other components in Figure D.1 are kept constants, the peak amplitude of the voltage spike decreases monotonically with R_{2_2}. Thus, the difference between the amplitudes in Figures 5.12 and D.1 is maximized when R_{2_2} has its extreme values. The two extremes are discussed separately.

If R_{2_2} is nonnegative $(0 \leq R_{2_2} \leq (R_{cc} - R_{c\epsilon})\frac{C_L}{C_H})$, then the amplitude of the spike in Figure D.1 is no greater than that in Figure 5.12. The difference becomes progressively more significant for models with a smaller R_{2_3}, and it is maximized when $R_{2_3} = 0$ (and $R_{2_2} = (R_{cc} - R_{c\epsilon})\frac{C_L}{C_H}$). In this extreme, if the circuit shown in Figure D.1 is characterized by $\ll \alpha, \beta \gg$, then the circuit shown in Figure 5.12 is characterized by $\ll \frac{\alpha}{\alpha+(1-\alpha)(1-\beta)}, \beta \gg$. Among all α-β combinations, the difference in the two circuits' amplitudes can never be more than 0.076 (out of 1), which occurs when $\alpha = 0.27$ and $\beta = 0.59$. This difference is negligibly small, so R_{2_2} does not play an important role when it is nonnegative.

On the other hand, if R_{2_2} is negative $(-R_{c\epsilon}\frac{C_L}{C_H} < R_{2_2} < 0)$, then the amplitude of the spike in Figure D.1 is greater than that in Figure 5.12. The difference is maximized when $R_{2_2} = -R_{c\epsilon}\frac{C_L}{C_H}$ and it cancels out $R_{2_1} + R_{2_3}$. In this scenario, if the circuit shown in Figure D.1 is characterized by $\ll 1, \beta \gg$, then the circuit shown in Figure 5.12 is characterized by $\ll 1 - \beta, \beta \gg$. Among all α-β combinations, the difference between the amplitudes can be as large as 0.235 (out of 1), which occurs when $\beta = 0.40$.

Although the difference in this case is significant, the scenario is rare. According

[2]A pair of tighter bounds would be

$$-R_{c\epsilon}\left(\frac{C_L}{C_H} - \frac{R_{\epsilon\epsilon}C^l_\epsilon}{R_{c\epsilon}C_H}\right) \leq R_{2_2} \leq (R_{cc} - R_{c\epsilon})\left(\frac{C_L}{C_H} - \frac{R_{\epsilon\epsilon}C^l_\epsilon}{R_{c\epsilon}C_H}\right).$$

to the analysis of Section 4.4.3. when the value of R_2 ($= R_{2_1} + R_{2_2} + R_{2_3}$) is very small or even negative. there is a good chance that the two-dominant-time-constant assumption is violated (i.e. the node being modeled really has more than two dominant poles). When this happens, neither the elaborate nor the first-order is sufficient a model: thus, the difference is not significant.

Consequently, any of the two models can be applied to most circuits. However, the elaborate model is in general more preferable because it gives a more conservative approximation when the modeling assumption is breaking down.

Appendix F

Simulation Results

This section summarizes the amplitude differences between the circuits shown in Figures D.1 and 5.12 for nMOS[1], RC, and pMOS[2] technologies.

Assume that the circuit shown in Figure D.1 is characterized by $\ll \alpha,\ \beta \gg$. The maximum differences in amplitudes (computed by subtracting the amplitude of the circuit shown in Figure 5.12 from that of the circuit shown in Figure D.1) for all possible values of R_{2_2} are summarized in the following table:

	Negative Extreme			Positive Extreme		
	Difference	α	β	Difference	α	β
nMOS	-0.080	0.36	0.60	0.310	1.00	0.31
RC	-0.076	0.27	0.59	0.235	1.00	0.40
pMOS	-0.055	0.16	0.52	0.135	1.00	0.41

As one can see from the table that an nMOS two-capacitor-two-transistor network driven to the ground is most sensitive to the variation of R_{2_2} while a pMOS

[1]An nMOS two-capacitor-two-transistor network driven to the ground (or a pMOS two-capacitor-two-transistor network driven to Vdd).

[2]A pMOS two-capacitor-two-transistor network driven to the ground (or an nMOS two-capacitor-two-transistor network driven to Vdd).

two-capacitor-two-transistor network driven to the ground is the least. Since, according to Figures 5.6, 5.8, and 5.10, the voltage spike of the nMOS configuration has the lowest amplitude among the three configurations, its sensitivity to the value of R_{2_2} is the least important.

One can also infer from the analysis in Appendix E that when the amplitude of the circuit shown in Figure D.1 is much larger than that of the circuit shown in Figure 5.12, the basis of a two-capacitor-two-transistor mapping is failing: under normal situations, the two circuits should have approximately the same amplitude. However, since the model shown in Figure D.1 is better at detecting a breakdown in the mapping basis (i.e. when $R_2 < 0$), it can be more preferable in spite of its lack of physical intuition for transistor networks.

Bibliography

[1] M. A. Breuer. A note on three-valued logic simulation. *IEEE Transactions on Computers*, 399–402, April 1972.

[2] R. E. Bryant. *Papers on a Symbolic Analyzer for MOS Circuits.* Technical Report CMU-CS-86-114, Carnegie-Mellon University, March 1986.

[3] R. E. Bryant. A switch-level model and simulator for MOS digital systems. *IEEE Transactions on Computers*, C-33(2):160–177, February 1984.

[4] R. E. Bryant. *A Switch-Level Model and Simulator for MOS Digital Systems.* Technical Report 5065:TR:83, California Institute of Technology, July 1983.

[5] R. E. Bryant. *A Switch-Level Simulation Model for Integrated Logic Circuits.* PhD thesis, Massachusetts Institute of Technology, March 1981.

[6] C.-Y. Chu and M. Horowitz. Charge sharing models for MOS circuits. In *International Conference on Computer-Aided Design*, pages 274–277. IEEE, November 1986.

[7] C.-Y. Chu and M. A. Horowitz. Charge-sharing models for switch-level simulation. *IEEE Transactions on Computer-Aided Design*, CAD-6(6):1053–1061, November 1987.

[8] W. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, 19:55–63, January 1948.

[9] D. Holt and D. Hutchings. A MOS/LSI oriented logic simulator. In *Eighteenth Design Automation Conference*, pages 280–287, ACM/IEEE, June 1981.

[10] M. Horowitz, P. Chow, D. Stark, R. T. Simoni, A. Salz, S. Przybylski, J. Hennessy, G. Gulak, A. Agarwal, and J. M. Acken. MIPS-X: a 20-MIPS peak, 32-bit microprocessor with on-chip cache. *IEEE Journal of Solid-State Circuits*, SC-22(5):790–799, October 1987.

[11] M. Horowitz, J. L. Hennessy, P. Chow, P. G. Gulak, J. M. Acken, A. Agarwal, C.-Y. Chu, S. A. McFarling, S. A. Przybylski, S. E. Richardson, A. Salz, R. T. Simoni, D. C. Stark, P. A. Steenkiste, S. W. K. Tjiang, and M. J. Wing. A 32b microprocessor with on-chip 2K byte instruction cache. In *International Solid-State Circuits Conference*, pages 30–31, 328, IEEE, February 1987.

[12] M. A. Horowitz. *Timing Models for MOS Circuits*. PhD thesis, Stanford University, December 1983.

[13] L. W. Johnson and R. D. Riess. *Numerical Analysis*. Addison-Wesley Publishing Company, 1977.

[14] N. P. Jouppi. *Timing Verification and Performance Improvement of MOS VLSI Designs*. PhD thesis, Stanford University, October 1984.

[15] T.-M. Lin. *A Hierarchical Timing Simulation Model for Digital Integrated Circuits and Systems*. PhD thesis, California Institute of Technology, August 1984.

[16] C. Mead and L. Conway. *Introduction to VLSI Systems*. Addison-Wesley Publishing Company, 1980.

[17] L. Nagel. *SPICE2: A Computer Program to Simulate Semiconductor Circuits*. Technical Report ERL Memo No. ERL-M520, University of California, Berkeley, May 1975.

[18] J. K. Ousterhout. Crystal: A timing analyzer for nMOS VLSI circuits. In *Third CalTech Conference on VLSI*, pages 57–69, March 1983.

[19] P. Penfield and J. Rubinstein. Signal delay in RC tree networks. In *Eighteenth Design Automation Conference*, pages 613–617, ACM/IEEE, June 1981.

[20] S. Przybylski. *The Implementation of MIPS*. Technical Report, Stanford University, August 1984.

[21] R. Putatunda. AUTODELAY: A program for automatic calculation of delay in LSI/VLSI chips. In *Nineteenth Design Automation Conference*, pages 616–621, ACM/IEEE, June 1982.

[22] A. Raghunathan and C. D. Thompson. Signal delay in RC trees with charge sharing or leakage. In *Asilomar Conference on Circuits, Systems & Computers*, pages 557–561, IEEE, November 1985.

[23] A. Raghunathan and C. D. Thompson. *Signal Delay in RC Trees with Charge Sharing or Leakage*. Technical Report UCB/CSD 85/243, University of California, Berkeley, June 1985.

[24] J. Rubinstein, P. Penfield, and M. A. Horowitz. Signal delay in RC tree networks. *IEEE Transactions on Computer-Aided Design*, CAD-2(3):202–211, July 1983.

[25] W. Sherwood. A MOS modelling technique for 4-state true-value hierarchical logic simulation. In *Eighteenth Design Automation Conference*, pages 775–785, ACM/IEEE. June 1981.

[26] C. J. Terman. *Simulation Tools for Digital LSI Design.* PhD thesis, Massachusetts Institute of Technology, September 1983.

[27] W. Weeks, A. Jimenez. G. Mahoney, D. Mehta, H. Qassemzadeh, and T. Scott. Algorithms for ASTAP – A network analysis program. *IEEE Transactions on Circuit Theory*, CT-20:628–634, November 1973.

[28] C. A. Zukowski. *The Bounding Approach to VLSI Circuit Simulation.* Kluwer Academic Publishers. 1986.