SIGNAL DELAY IN GENERAL RC NETWORKS WITH APPLICATION TO TIMING SIMULATION OF DIGITAL INTEGRATED CIRCUITS

by

Tzu-Mu Lin

and

Carver A. Mead

5089:TR:83

Signal Delay in General RC Networks with Application to Timing Simulation of Digital Integrated Circuits

by

Tzu-Mu Lin

and

Carver A. Mead

5089:TR:83

Computer Science Department

California Institute of Technology

Pasadena California 91125

This research is sponsored by the System Development Foundation

ABSTRACT

Modeling digital MOS circuits by RC networks has become a well accepted practice for estimating delays. In 1981, Penfield and Rubinstein proposed a method to bound the delays of the nodes in an RC tree network. In this paper, we address the problem of dynamic timing simulation under RC-based models. Based upon the delay of Elmore, a single value of delay is derived for any node in a general RC network. The effects of parallel connections and stored charges are properly taken into consideration. The algorithm can be used either as a stand-alone simulator, or as a front end for producing initial waveforms for waveform-relaxation based circuit simulators. An experimental simulator called SDS (Signal Delay Simulator) has been developed. For all the examples tested so far, this simulator runs about two to three orders of magnitude faster than SPICE, and detects all transitions and glitches at approximately the correct time.

Contents

§1 Introduction

Modeling digital MOS circuits by RC networks has become a well accepted practice for estimating delays [1,2,3,4]. In 1981, Penfield and Rubinstein (P-R) proposed a method to bound the delays of the nodes in an RC tree network [2]. Later, Horowitz (H) extended this method to allow modeling MOS transistors by nonlinear resistors [5]. The P-R-H approach has the following features. First, only tree networks with no initial charge are considered. Second, upper and lower bounds of a response are derived instead of a single value of delay. Third, These bounds are expressed in terms of a set of parameters which can be determined in a hierarchical manner. Fourth, transistors are assumed to be either fully turned on or fully turned off. Also, every node in the network is driven by one and only one source (VDD or GND) at a time.

For static timing analysis [6,7,8], a tree network with no initial charge is the only case that needs to be considered. Also, as was pointed out by P-R-H [2,3], bounds of a response offer more accurate information than a single value of delay. Therefore, the first two features of the P-R-H approach are appropriate for static timing analysis. For dynamic analysis (simulation), however, we need to deal with more general networks with parallel connections and partially stored charge. Furthermore, a single value of delay is preferable to a set of bounds (for scheduling a new event). In this paper, we address the problem of dynamic timing simulation under RC-based models. We consider general networks, and derive a single value for the delay. The third and fourth features of the P-R-H approach are also shared by ours.

Based upon the switch-level logic simulation model proposed by Bryant [9], a model for timing simulation is described in section 2. Instead of the unit delay scheme employed by Bryant, a more realistic value of delay is calculated for every logic event. The delay used in our model is based upon the delay of Elmore [10], modified to correctly treat nonmonotonic responses (section 3). This value of delay is shown to always fall within the P-R bounds for RC tree networks with zero initial charge. In section 4, transmission

matrices are used to express the transfer behavior of two-port RC networks. As far as delay is concerned, a two-port RC network is characterized by three parameters: $\overline{R}$: series resistance, $\overline{C}$: effective capacitance, and $\overline{D}$: internal delay. These three parameters can be calculated hierarchically as the corresponding two-port RC networks are composed in various ways. The composition rules agree with those described in [2], except that stored charge is properly taken into consideration. We also add composition rules for parallel connections. In general, delay calculations need to carried out independently for each individual node. However, in a tree network, the delays of all nodes can be calculated incrementally and simultaneously. In section 5, the techniques of tree decomposition and load redistribution are introduced for calculating delays in a general network. A relaxation algorithm requiring information only from neighboring nodes and transistors is presented. Under certain conditions, this algorithm is equivalent to the block Gauss-Seidel iterative method for solving a system of linear equations. The convergence of the G-S method in our case is guaranteed. The algorithm is applicable to any tree decomposition of networks. Based upon the concept of dominant path [9], one such decomposition scheme is described in section 6. In most cases, the delay estimated by the dominant path method (initial guess) is already very accurate before the relaxation process starts.

As a transistor network evolves, the topology of the approximating RC network changes, and the logic states and delay values of various nodes need to be updated accordingly. An algorithm for updating these states and values is described in section 6. The mechanism for scheduling logic events is similar to that presented by Terman[4]. Some simulation results are given in section 7.

§2 The Model

The model for timing simulation at the transistor level is based upon the switch model proposed by Bryant [9]. In this model, a network is represented by a set of transistors $T = \{t_1, t_2, \cdots, t_m\}$, and a set of nodes $N = \{n_1, n_2, \cdots, n_n\}$. With each node is associated

a capacitance and one of three different states corresponding to the node voltage: 1 (high voltage), 0 (low voltage) or X (in transition). The other end of the node capacitor is always connected to the ground, and no floating capacitors are allowed in the network. With each transistor is associated an ON-resistance (or two different values of ON-resistances: one is used in case of pull up and the other is used in case of pull down [4,5]). A transistor may be either ON or OFF depending on the state of the node controling its gate. A transistor is treated as a resistance equal to its ON-resistance if it is on or to ∞ if it is off. Instead of the order of magnitude (or logic) conductances and capacitances used in Bryant's switch model, precise values of the resistances and capacitances are kept for determining logic levels as well as for estimating delays. Although the capacitance of a node and the resistance of a transistor are voltage dependent, they are treated as constants here. This approximation is considered adequate for our purpose, since only the delay values are of interest, not the detailed waveforms. The evolution of a MOS circuit is approximated by a sequence of RC networks. Various node capacitors are charged to VDD and discharged to GND through the network. This charging-discharging process may change the state of a node which in turn will change the topology of the RC network. Under the unit delay model which is employed by Bryant and others, all such nodes change state at the same time. In our model, different nodes are charged or discharged at different rates which depend on the topology and the initial charge distribution of the RC network. When the gate node of at least one transistor changes state, a new network results. A partially charged or discharged node which connects to the gate of a transistor does not change the state of that transistor. However, the charge stored in the nodes will be taken into account when the nodes are again charged or discharged through the new network. The whole process continues until the topology of the network no longer changes.

With the approximation introduced above, the problem of estimating the delay of a MOS circuit reduces to that of an RC network. In this context, the term RC network refers only to those networks that are approximations of a MOS circuit, i.e., resistor networks

where there is a capacitor between every node and GND. A more constructive definition of RC networks is given in section 4.

At each instant of time, the approximating RC network of a MOS circuit consists of a collection of independent blocks. Two nodes are in the same block if and only if they belong to the same equivalence class induced by (the transitive closure of) the relation $\sim$:

> $A \sim B \Leftrightarrow A$ and B are connected laterally by a transistor which is ON at the present instant of time.

That is to say, two nodes are in the same block if and only if there exists a resistive path of finite resistance between these two nodes. As the size of a MOS circuit increases, that of a block remains almost a constant. In our model, each block is assumed to be driven by one and only one source (VDD or GND) which is refered to as the source of the block. The other two possibilities presented below are not considered.

1. Neither VDD nor GND is driving a block: this undriven situation may cause static charge sharing among nodes [9].
2. Both VDD and GND are driving a block: In most practical situations, one source is dominant over the other with respect to a node, otherwise a conflict condition occurs and the logic state of the node is unpredictable. Although the presence of the other source may affect the delay of the response to the dominant one, the effect is usually small, as various experiments indicate. In our model, this block is approximated by two independent blocks, one driven by VDD and the other by GND. The detail of this decomposition is described in section 6.

Further work needs to be done to include the effects of static charge sharing and multiple sources.

To measure the delay of a node in an RC network, it suffices to consider the normalized case where the node voltage starts from some initial value between 0 and 1, and is driven towards the final value 1. The results obtained in this normalized case are

easily adapted to both charging and discharging processes, and to any values of supply voltages. Normalized variables are used throughout the context, that is, V is dimensionless and therefore Q is of the same dimension as C. Moreover, as the delays of the nodes in a block are independent of those in other blocks, only one block needs to be considered at a time. The term RC network used in the context refers to only a single block.

The values of delay estimated under an RC-based model like ours are relative rather than absolute. In some sense, the values obtained are normalized with respect to the threshold voltage of certain type of transistor circuit. The effect of different values of threshold voltages of different transistor types are reflected by adjusting the values of their ON-resistances. As was introduced in [1], and further analyzed by Glasser [11], the delay time τ of an inverter is linearly related to the RC time constant, where R is the ON-resistance of the driving transistor and C is the load capacitance of the output node. The nonlinear part of the circuit behavior can be absorbed in the coefficient which can be determined from more detailed circuit analysis or simulation [12]. The delay time is also additive in that the delay of a chain of such inverters can be obtained by summing up those of the individual ones. The motivation behind our work is to extend this linear and additive property to more general transistor networks. The resistances of wires, contacts, etc., can be treated in the same manner as transistor ON-resistances. The lumped approximation of these distributed elements are investigated by Chiang[13]. Simulation results based upon this model and their comparison with SPICE outputs are given in section 7.

§3 Definition of Delay

Prior to the actual analysis, it is necessary to have a consistent and unambiguous definition of delay. There are a number of such definitions in practical use, for instance, the time required for a response to reach the threshold voltage of a MOS transistor. Although this kind of definition is useful for certain simulators whose delay calculations are based upon empirical data, it is extremely awkward for the purpose of theoretical investigation

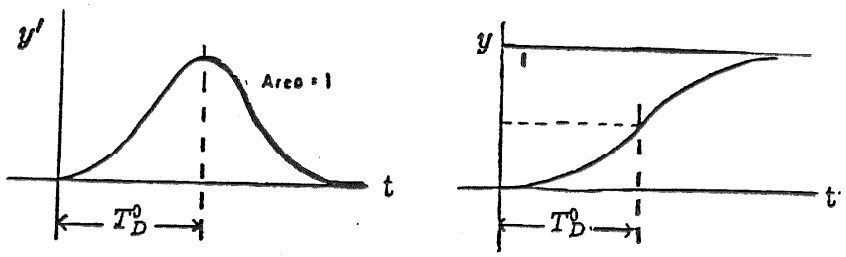


Figure 1. Curves illustrating the Elmore's delay

or symbolic analysis. On the other hand, Elmore's delay [10] is very efficient in this respect, and it is defined as

$$T_D^0 = \int_0^\infty t y'(t) dt. \tag{1}$$

where $y'(t)$ is the derivative of the transient response $y(t)$ of some node of a network. The superscript ^{0} indicates zero initial charge, which condition is always assumed by Elmore (also by P-R-H). This definition of delay is based upon the observation that, if $y(t)$ is monotonic in time, T_D^0 is the centroid of $y'(t)$, and is very close to what is commonly conceived as "delay" (Fig.1). The great usefulness of Elmore's delay lies in its close connection to the Laplace transform $\mathcal{L}$ of the response. In an RC network, $g(s) = \mathcal{L}(y(t))$ can always be expressed in the form $g(s) = \frac{1+a_1 s+a_2 s^2+\cdots+a_m s^m}{s(1+b_1 s+b_2 s^2+\cdots+b_n s^n)}$. Note that $sg(s)|_{s\to 0} = y(t)|_{t\to\infty} =$ 1 because there is no floating capacitor in the network. If there is no initial charge stored in the network, then $T_D^0 = b_1 - a_1$ [10].

Although $g(s)$ is in general a very complicated expression, $T_D^0 = b_1 - a_1$ is very easy to obtain symbolically. Penfield and Rubinstein have shown that a general expression of T_D^0 exists for any node in an RC tree network, and the expression can be determined in a hierarchical manner [2]. In this paper, we extend this result to more general RC networks with parallel connections and nonzero initial charge. To do this, a modification of Elmore's delay is necessary because the original formulation (1) only makes sense when y(t) is monotonic. In an RC tree network without any inital charge, the unit step response

of any node is guaranteed to be monotonic [3]; however, monotonicity is not true in general. To deal with general RC networks, the term delay is redefined as

$$T_D = \int_0^\infty 1 - y(t)dt. \qquad (1')$$

This expression is just the area above the response y(t), but below 1, as indicated in Fig.1. In the case of zero initial charge, T_D is equal to T_D^0. In [3], this result was proved for the case of RC trees. For general networks, the result is proved as follows:

$$\begin{aligned}\frac{1}{s} - g(s) &= \int_0^\infty [1 - y(t)] \exp^{-st} dt \\ &= T_D - s \cdot \int_0^\infty [1 - y(t)] t dt + \cdots\end{aligned}$$

so that $T_D = \lim_{s \to 0}(\frac{1}{s} - g(s)) = b_1 - a_1 = T_D^0$.

From the above discussion, T_D is consistent with and more general than T_D^0. To justify the usage of T_D, consider the case of RC tree networks with no initial charge. Referring to Fig.1, $T_D = T_D^0$ deviates from the standard visualization of delay only when the response curve is highly asymmetric. Fortunately this deviation never happens in an RC network in the following sense. Suppose that the response curve of a node is approximated by a single exponential function with time constant T_D, then the delay time t_d for the response to reach a voltage level v is $T_D \ln(\frac{1}{1-v})$. The value t_d of any node in the network always lies within the upper and lower bounds given by Penfield and Rubinstein [2] for any voltage level v, as the following theorem indicates. These bounds are far tighter than other approximations in the simulation procedure.

Theorem 1. Consider a node in an RC tree network with no initial stored charge. Let t_1, t_2, t_3, t_4 be the four bounds of time defined in [2], i.e.

$$t_1(v) = T_D - T_P(1-v)$$
$$t_2(v) = T_R \ln \frac{T_D}{T_P(1-v)}$$
$$t_3(v) = \frac{T_D}{1-v} - T_R$$
$$t_4(v) = T_P - T_R + T_P \ln \frac{T_D}{T_P(1-v)}$$

where $T_D = T_D^0$ is the Elmore's delay. T_R, and T_D are defined in [2], which satisfy $T_R \leq T_D \leq T_P$ for any node in an RC tree network. Let $t_d(v) = T_D \ln(\frac{1}{1-v})$. Then $t_d \geq t_1, t_d \geq t_2, t_d \leq t_3$ for all values of v, and $t_d \leq t_4$ for $v \geq 1 - \frac{T_D}{T_P}$.

Proof:

The following three inequalities regarding the natural log function are noted first:

1. $\ln \frac{1}{1-x} \geq x$, for $0 \leq x \leq 1$
2. $x - \ln x \geq 1$, for $x \geq 1$
3. $1 \geq x(1 - \ln x)$, for $0 \leq x \leq 1$

The proof of the theorem itself goes as follows:

$$\begin{aligned} t_d - t_1 &= T_D \ln\left(\frac{1}{1-v}\right) - T_D + T_P(1-v) \\ &\geq T_D\left(\ln \frac{1}{1-v} - v\right) \\ &\geq 0 \end{aligned}$$

$$\begin{aligned} t_d - t_2 &= (T_D - T_R) \ln \frac{1}{1-v} + T_R \ln \frac{T_P}{T_D} \\ &\geq 0 \end{aligned}$$

$$
\begin{aligned}
t_3 - t_d &= \frac{T_D}{1-v} - T_R - T_D \ln \frac{1}{1-v} \\
&= T_D\left(\frac{1}{1-v} - \ln \frac{1}{1-v}\right) - T_R \\
&\geq T_D - T_R \\
&\geq 0
\end{aligned}
$$

$$
\begin{aligned}
t_4 - t_d &= T_P - T_R + T_P \ln \frac{T_D}{T_P(1-v)} - T_D \ln \frac{1}{1-v} \\
&\geq T_P - T_R + T_D \ln \frac{T_D}{T_P} \qquad \cdots\cdots \qquad \left(\frac{T_D}{T_P(1-v)} \geq 1\right) \\
&\geq T_P\left(1 - \frac{T_D}{T_P}\left(1 - \ln \frac{T_D}{T_P}\right)\right) \\
&\geq 0
\end{aligned}
$$

T_D in case of nonzero stored charge is still consistent with the Elmore's delay T_D^0, as is discussed in section 6. The usage of T_D is also very effective in detecting glitches produced by the dynamic charge sharing effect.

In the next section, transmission matrices are used to express the transfer behavior of two-port RC networks. Based upon this formulation, three theorems are given regarding the composition of transmission matrices and subnetworks, and the abstraction of delay information from these matrices. T_D as defined in (1′) is used as our definition for delay.

§4 Delay of RC networks

A well known result from circuit theory [14] states that the (voltage-current) transfer behavior of a two-port linear network can be described by the following equation:

$$
\begin{pmatrix} V_o(s) \\ I_o(s) \end{pmatrix} = \begin{pmatrix} T_1(s) & T_2(s) \\ T_3(s) & T_4(s) \end{pmatrix} \begin{pmatrix} V_i(s) \\ I_i(s) \end{pmatrix} + \begin{pmatrix} U_1(s) \\ U_2(s) \end{pmatrix} \tag{2}
$$

where the subscripts $_o$ and $_i$ indicate the output and input ports, respectively. The matrix $\begin{pmatrix} T_1(s) & T_2(s) \\ T_3(s) & T_4(s) \end{pmatrix}$ is called the transmission matrix, or simply the T-matrix. This equation can be expressed either in the time domain or in the Laplace domain; however, it is more convenient to use the Laplace domain, as we just did in the previous section and will continue to do. Usually, the T-matrix is only a function of the network, and is independent of the initial condition. The other term $\begin{pmatrix} U_1(s) \\ U_2(s) \end{pmatrix}$ of (2) depends on both the network and the initial condition, and is refered to as the U-matrix. In general, $T_i(s)_{i=1,2,3,4}$ and $U_i(s)_{i=1,2}$ are very complicated polynomials in s, however, the delay T_D only depends on the constant and s terms of these polynomials, so higher terms can always be omitted. In what follows, the symbol $\sim$ will be used to mean that two formula are equal up to the s term. Besides the T-matrix, there are other matrices that can be used to describe the same network, for instance the G-matrix and the H-matrix. These matrices are all equivalent to one another, and any one can be easily transformed to the others. In the context of our treatment, the T-matrix turns out to be the most useful one for the following reasons. First, when several two-port networks are combined in series, the resulting T-matrix can be obtained simply by multiplying the individual matrices. More important, the coefficients of the T-matrix and U-matrix of a two-port RC network are directly related to some essential parameters that characterize the delay behavior of the network. As will be shown in theorem 2, it is possible to express the T-matrix and U-matrix of any two-port RC network up to the s term by the five parameters of the network: the series resistance R, the total capacitance C, the internal delay D due to input, the total initial charge Q, and the internal delay D^* due to stored charge. These five parameters can be determined hierarchically as the corresponding two-port RC networks are composed in various ways. Among these five parameters, R, C, and D are only functions of the network, and Q and D^* also depend on initial charge. As far as delay is concerned, the number of parameters reduces to only three, as is indicated by theorems 2′ and 5′. Prior to any further discussion, a more constructive definition of RC networks than the one described in section 2 will be given. First of all, a two-port RC network with its input and output ports is recursively defined.

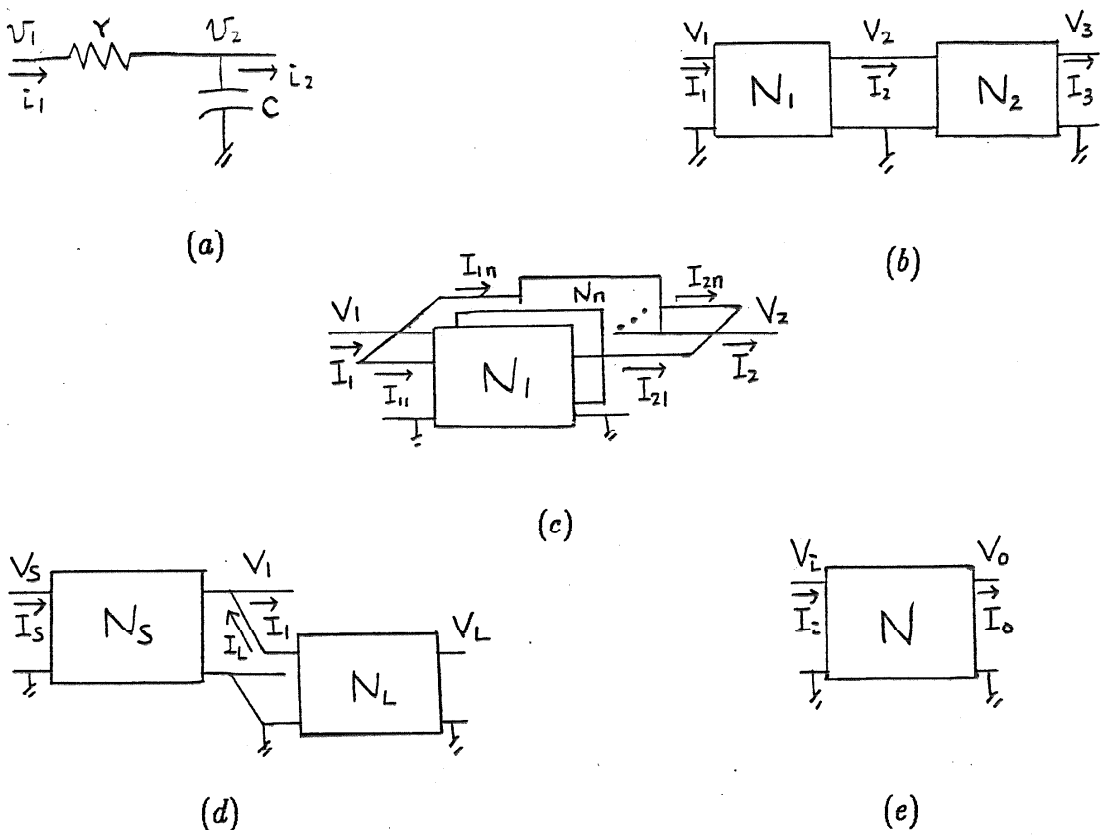


Figure 2. The five cases of a two-port RC network

A two-port RC network is one of the following:

1. a resistor in series with a capacitor: The common node of the two is the output, and the other end of the resistor is the input of the network. the other end of the capacitor is grounded (Fig.2.a).

2. a series connection of two two-port RC networks N_1 and N_2: The input of N_2 and the output of N_1 merge internally; the input of N_1 becomes the input, and the output of N_2 becomes the output of the resulting network(Fig.2.b).

3. a parallel connection of n two-port RC networks $N_{1,\cdots,n}$: The inputs and outputs of these networks merge and become the input and output of the resulting network (Fig.2.c).

4. a two-port RC network N_S with a side branch N_L of which the output is open, and the input is connected to the output of N_S (Fig.2.d).

5. a two-port RC network with input and output ports interchanged. Although this construction is not necessary in characterizing an RC network, it is very useful in practice for those networks where the directions of signal flows are dynamically changing (Fig.2.e). (3)

Finally, a "RC network" is defined as a two-port RC network with the output port open and input port connected to the source.

Remarks:

- In terms of two-port RC networks, the two cases not considered in our model (section 2) are described as follows:

  1. no driving source: a two-port RC network with output port open and input port connected to a capacitor.

  2. multiple sources: two two-port RC networks with their output ports connected together and input ports connected to VDD and GND, respectively.

- This definition of RC networks does not cover all possible network topologies because bridge connections may exist, which make the configuration neither series nor parallel. Simple bridge connections may be dealt with easily, and one such example will be given in section 5. As an extension of the $\Delta - Y$ transformation of resistor networks [14], we conjecture the existence of a transformation from a number of Y-connected networks to the same number of Δ-connected networks, and vice versa. This transformation is in terms of the (R, C, D, Q, D^*)-parameters of these networks such that, as far as delay is concerned, the two sets of networks are equivalent. If this conjecture is true, then these definitions do cover all possible network topologies. Moreover, a technique exists that is capable of dealing with general RC networks, and requires only the information of the first four cases of (3). This technique is to be described in the next section.

Consider an arbitrary node as the output. A two-port RC network between the source and the node can be constructed step by step using the process of (3). In theorem 2, the relationship between the five parameters R, C, D, Q, and D^* and the transfer equation (2) of a two-port RC network is established for each of the five cases of (3). Then in theorem 5, a formula for determining the delay of a node is derived from the two-port RC network between the source and the node. Although one such network is enough for this purpose, a more general result which also includes a explicit loading network is presented. This general result is very useful in the case of a tree network where the driving and loading networks are well-defined: they are the subtrees above and under the node, respectively. In such networks, the delay of every node can be obtained simultaneously and incrementally, as is indicated in theorem 10. The proofs of these three theorems are presented in Appendix 1.

Theorem 2. Up to the first order, the transfer equation (2) of a two-port RC network is of the following form:

$$\begin{pmatrix} V_o(s) \\ I_o(s) \end{pmatrix} \sim \begin{pmatrix} 1+s(RC-D) & -R+sb \\ -sC & 1+sD \end{pmatrix} \begin{pmatrix} V_i(s) \\ I_i(s) \end{pmatrix} + \begin{pmatrix} -D^*+sh \\ Q+sf \end{pmatrix} \tag{4}$$

The parameters[1] R, C, D, Q, and D^* for each of the five cases of (3) are determined as follows:

1. primitive case:

$$\begin{aligned} R &= r \\ C &= c \\ D &= rc \\ Q &= cv_0 \\ D^* &= 0 \end{aligned} \tag{5}$$

[1]The parameters b, h, and f are of no concern in this context because they do not appear in the formula for T_D for either simple or composite networks.

where r, c and v_0 are the values of the resistance, the capacitance and the initial voltage of the capacitor, respectively.

In the following four cases, a subscript is associated with each parameter, indicating to which network this parameter belongs. In particular, subscript T indicates the resulting network of each composition (or operation).

2. series connection of N_1 and N_2:

$$\begin{aligned} R_T &= R_1 + R_2 \\ C_T &= C_1 + C_2 \\ D_T &= D_1 + D_2 + R_1 C_2 \\ Q_T &= Q_1 + Q_2 \\ D_T^* &= D_1^* + D_2^* + R_2 Q_1 \end{aligned} \tag{6}$$

3. parallel connection of $N_{1,\cdots,n}$:

$$\begin{aligned} R_T &= \frac{1}{\sum_1^n \frac{1}{R_i}} \\ C_T &= \sum_1^n C_i \\ D_T &= R_T \left(\sum_1^n \frac{D_i}{R_i} \right) \\ Q_T &= \sum_1^n Q_i \\ D_T^* &= R_T \left(\sum_1^n \frac{D_i^*}{R_i} \right) \end{aligned} \tag{7}$$

4. N_S with side branch N_L:

$$\begin{aligned} R_T &= R_S \\ C_T &= C_L + C_S \\ D_T &= D_S + R_S C_L \\ Q_T &= Q_S + Q_L \\ D_T^* &= D_S^* \end{aligned} \tag{8}$$

5. input and output ports interchanged:

$$
\begin{aligned}
R_T &= R \\
C_T &= C \\
D_T &= RC - D \\
Q_T &= Q \\
D_T^* &= RQ - D^*
\end{aligned}
\tag{9}
$$

∎

Corollary 3. If there is no initial charge in a two-port RC network, then the parameters Q and D^* of this network are both 0. ∎

Corollary 4. If the nodes of a two-port RC network are all charged to 1 initially, then the parameters Q and D^* of this network are equal to C and $RC-D$, respectively. ∎

Theorem 5. If a node connects to the source through a network N_S with parameters R_S, C_S, D_S, Q_S, and D_S^*, and is loaded by another network N_L with parameters R_L, C_L, D_L, Q_L, and D_L^*, then the delay T_D of this node is $(D_S + D_S^* - R_S Q_S) + R_S(C_L - Q_L)$. ∎

Corollary 6. If there is no initial charge in both N_S and N_L, then $T_D = D_S + R_S C_L$. ∎

Among the five parameters of theorem 2, D^* and Q can be expressed in terms of R, C and D for special initial charge distributions (corollary 3 and 4). In fact, this reduction of parameters is possible in general, and only three parameters are necessary to represent any two-port RC network to calculate delays. Suggested by the result of theorem 5, these three reduced parameters are

$$
\begin{aligned}
\overline{R} &= R \\
\overline{C} &= C - Q \\
\overline{D} &= D + D^* - RQ
\end{aligned}
\tag{10}
$$

In case of zero initial charge, $\overline{R}$, $\overline{C}$, and $\overline{D}$ reduces to R, C, and D, respectively. In terms of these three reduced parameters, theorems 2 and 5 are restated.

Theorem 2′. The three parameters $\overline{R}$, $\overline{C}$ and $\overline{D}$ of a two-port RC network can be determined as follows:

1. primitive case

$$\begin{aligned} \overline{R} &= r \\ \overline{C} &= c(1-v_0) \\ \overline{D} &= rc(1-v_0) \end{aligned} \tag{5′}$$

The composition rules of these three parameters are the same as those of R, C and D in theorem 2, i.e.

2. series connection of N_1 and N_2:

$$\begin{aligned} \overline{R}_T &= \overline{R}_1 + \overline{R}_2 \\ \overline{C}_T &= \overline{C}_1 + \overline{C}_2 \\ \overline{D}_T &= \overline{D}_1 + \overline{D}_2 + \overline{R}_1\overline{C}_2 \end{aligned} \tag{6′}$$

3. parallel connection of $N_{1,\cdots,n}$:

$$\begin{aligned} \overline{R}_T &= \frac{1}{\sum_1^n \frac{1}{\overline{R}_i}} \\ \overline{C}_T &= \sum_1^n \overline{C}_i \\ \overline{D}_T &= \overline{R}_T\left(\sum_1^n \frac{\overline{D}_i}{\overline{R}_i}\right) \end{aligned} \tag{7′}$$

4. N_S with side branch N_L:

$$\begin{aligned} \overline{R}_T &= \overline{R}_S \\ \overline{C}_T &= \overline{C}_L + \overline{C}_S \\ \overline{D}_T &= \overline{D}_S + \overline{R}_S\overline{C}_L \end{aligned} \tag{8′}$$

5. input and output ports interchanged:

$$\begin{aligned} \overline{R}_T &= \overline{R} \\ \overline{C}_T &= \overline{C} \\ \overline{D}_T &= \overline{RC} - \overline{D} \end{aligned} \tag{9'}$$

∎

Theorem 5'. If a node connects to the source through a network N_S with parameters $\overline{R}_S$, $\overline{C}_S$, $\overline{D}_S$, and is loaded by another network N_L with parameters $\overline{R}_L$, $\overline{C}_L$, $\overline{D}_L$, then the delay T_D of this node is $\overline{D}_S + \overline{R}_S\overline{C}_L$. ∎

Corollary 7. If there is no explicit loading network N_L, then $T_D = \overline{D}_S$. ∎

Corollary 8. Consider a two-port RC network with series resistance R, total capacitance C, and total charge Q. Let T_1 and T_2 denote the delays of the output port (when the input port is driven and output port is open) and the input port (when the output port is driven and input port is open), respectively. Then $T_1 + T_2 = \overline{RC} = R(C - Q)$. ∎

Remarks:

- The separation of driving network and loading network for a given node is by no means unique. For instance, N_L in theorem 5 can be considered as a side branch of N_S so that there is no explicit loading network at all. That the value of the delay is the same for both cases is shown as follows: Let subscript $_T$ denotes the network N_S with N_L merged inside. By case 4 of theorem 2', $\overline{D}_T = \overline{D}_S + \overline{R}_S\overline{C}_L$, and $\overline{R}_T = \overline{R}_S$. Then by corollary 7, $T_D = \overline{D}_T = \overline{D}_S + \overline{R}_S\overline{C}_L$ which is the same as the result given in theorem 5'.
- In most cases, there are more than one branches (transistors) incident on a node, and these branches belong to different two-port networks. The capacitance of the node can be arbitrarily distributed among these branches without affecting the result.

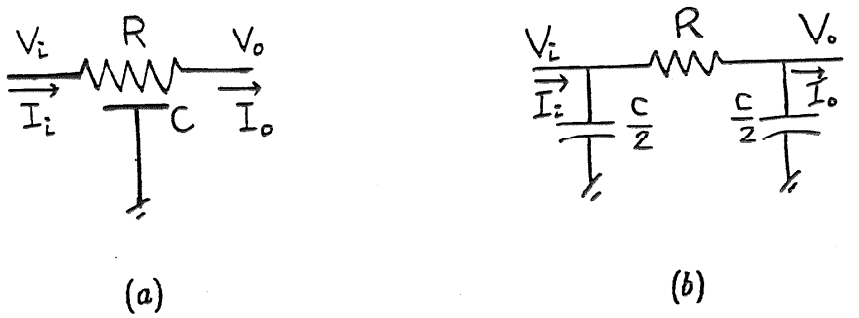


Figure 3. approximation of RC lines by lumped elements

▸ The T-matrix of a uniformly distributed RC line (Fig.3.a) is $\begin{pmatrix} \cosh\Gamma & -\frac{\sinh\Gamma}{Z} \\ -\sinh\Gamma & \cosh\Gamma \end{pmatrix}$, where $\Gamma = \sqrt{sRC}$ and $Z = \sqrt{\frac{R}{sC}}$ [15]. R and C are the total resistance and capacitance of the line, respectively. Up to the s term, the T-matrix is the same as $\begin{pmatrix} 1+\frac{sRC}{2} & -R-\frac{sR^2C}{6} \\ -sC & 1+\frac{sRC}{2} \end{pmatrix}$. That is to say, as far as delay is concerned, the RC line can be approximated by two capacitors and one resistor connected as shown in Fig.3.b [13].

Tree networks:

From theorem 5′, the delay of a node depends on both the driving and loading networks of the node. Parallel (and bridge) connections couple all nodes together so that every node is driving and loading every other node at the same time. As a result, the calculation of delays in general needs to be carried out independently for each individual node. However, no node in a tree network both drives and loads another node, so the delays of all the nodes can be calculated simultaneously and incrementally.

Theorem 9. Suppose node N_1 and node N_2 are cascaded in an RC tree network. N_1 is nearer to the source and is connected to N_2 through a resistor of value r. The total capacitance and total charge of the loading network of N_2 are C_L and Q_L, respectively. If the delay of node N_1 is T_1, then the delay of node N_2 is $T_1 + r\overline{C}_L = T_1 + r(C_L - Q_L)$. ∎

Corollary 10. The delay of a node i in a tree network is $\sum_k R_{i,k}(C_k - Q_k)$ where $R_{i,k}$ is the resistance of the (unique) path between the source and node i, that is in common with

the (unique) path between the source and the node k. C_k and Q_k are the capacitance and initial charge of node k, respectively. The summation carries over all nodes k in the network [2]. ∎

The following algorithm (TREE) calculates the delays of all the nodes in a tree network.

1. The loading information is accumulated and propagated from the loading ends towards the driving end of the tree network. To be more precise, a value C_i^L is associated with each node i, and

$$C_i^L = \begin{cases} C_i - Q_i & \text{if node i is a leaf} \\ C_i - Q_i + \sum_j C_j^L & \text{otherwise} \end{cases}$$

   where index j ranges over all the succeeding nodes of node i. C_i and Q_i are the node capacitance and stored charge of node i, respectively.

2. The delay of each node is calculated incrementally from the driving end towards the loading ends, i.e., $T_i = T_{p(i)} + r_i C_i^L$, where $p(i)$ is the parent node of node i, and r_i is the resistance between node i and node $p(i)$.

The correctness of this algorithm is guaranteed by theorem 10. The time complexity is $O(n)$, where n is the number of nodes in the tree network.

With theorem 2′ and 5′, the two-port RC network between the source and a node can be constructed, and the delay of the node can be calculated. This process is direct, constructive, and requires information regarding the global topology of the network. Presented in the next section is another approach of delay calculation that is iterative and distributive in nature [16]. Following an approach due to Chen, each node or transistor is itself a process, which only communicates with its neighboring nodes and transistors. The delays of all the nodes are determined in a collective manner. This approach is capable of

dealing with general RC networks without sacrificing the nice property of tree networks described above. Examples of both approaches are presented in the next section.

§5 Tree Decomposition and Load Redistribution

The problem of evaluating delays can be reformulated as a set of relations among neighboring nodes and branches. Associate a global index with each node. Suppose there are a_i branches incident on a node N_i. Let $r_{(i,j)}$ denote the resistance of the jth branch, and $f(i,j)$ denote the global index of the neighboring node of N_i through this branch. The idea here is to partition $\overline{C}_i = C_i - Q_i$ into these a_i branches, each of value $\overline{C}_{(i,j)}$, such that $\sum_{j=1,\cdots,a_i} \overline{C}_{(i,j)} = \overline{C}_i$, and the delays evaluated from different branches are the same. $\overline{C}_{(i,j)}$ is the equivalent load to node N_i from the jth branch. By theorem 10, $T_i = T_{f(i,j)} + r_{(i,j)}\overline{C}_{(i,j)}$. To summarize, we have the following set of relations

$$\begin{cases} T_i = T_{f(i,j)} + r_{(i,j)}\overline{C}_{(i,j)} & j = 1,\cdots,a_i,\ i = 1,\cdots,N \\ \sum_{j=1}^{a_i} \overline{C}_{(i,j)} = \overline{C}_i & i = 1,\cdots,N \end{cases} \tag{11}$$

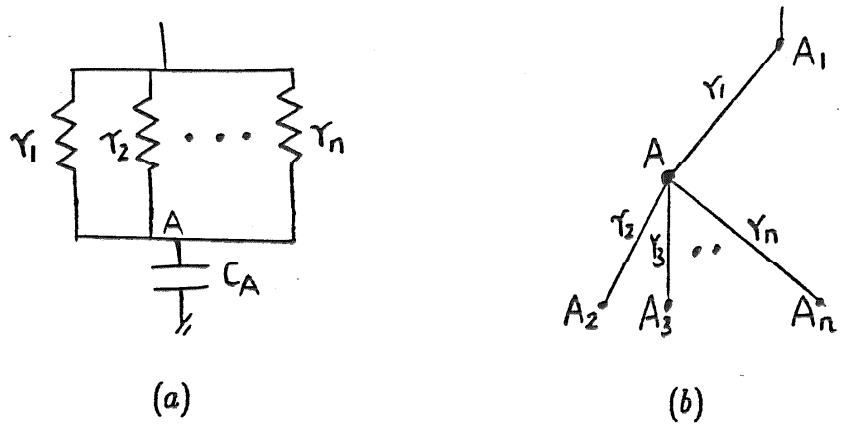
where N is the number of nodes in the network. The formal derivation of (11) is as follows. By the definition of delay, $T_i = \int_0^\infty (1 - v_i)dt$, and $T_{f(i,j)} = \int_0^\infty (1 - v_{f(i,j)})dt$.

$$\overline{C}_{(i,j)} \equiv \frac{T_i - T_{f(i,j)}}{r_{(i,j)}} = \int_0^\infty \frac{v_{f(i,j)} - v_i}{r_{(i,j)}} dt \tag{12}$$

Also, by Ohm's law and Kirchhoff's current law,

$$C_i \frac{dv_i}{dt} = \sum_j \frac{v_{f(i,j)} - v_i}{r_{(i,j)}} \tag{13}$$

Combining (12) and (13),



(a) (b)

Figure 4. Illustrations of T_i's and $C_{(i,j)}$'s

$$\begin{aligned}\sum_j \overline{C}_{(i,j)} &= \sum_j \int_0^\infty \frac{v_{f(i,j)} - v_i}{r_{(i,j)}} dt \\ &= \int_0^\infty C_i \frac{dv_i}{dt} dt \\ &= C_i - Q_i \\ &= \overline{C}_i\end{aligned}$$

Formula (11) represents a system of linear equations: $r_{(i,j)}$'s and $\overline{C}_i$'s are known, and T_i's and $\overline{C}_{(i,j)}$'s are to be determined. This system of equations is general enough to deal with all RC networks, including those with bridge connections. All nodes, however, must be driven by one and only one source. To simplify the discussion, zero initial charge is assumed throughout this section. The results obtained are directly applicable to general cases by replacing C's with $(C-Q)$'s.

Example 11. Consider the two simple circuits in Fig.4. Circuit 4.a consists of n transistors connecting the source to a node A. All transistors are ON, and are with resistances $r_{1,\cdots,n}$, respectively. From theorem 2 and corollary 7, $T_A = R_T C_A$, where $R_T = \frac{1}{\sum_{j=1}^{n} \frac{1}{r_j}}$, and C_A is the capacitance of node A. Another way to calculate delays is that, instead of combining resistances, capacitance C_A is distributed into the n incident branches: $C_{(A,j)} = \frac{R_T}{r_j} C_A$, for $j = 1, \cdots, n$. This combination of $C_{(A,j)}$'s is the only possible partition of C_A such that $\sum_{j=1}^{n} C_{(A,j)} = C_A$, plus the delays evaluated from all n branches are equal. This common value of delay equals $R_T C_A$. Both methods give the same result.

Consider an arbitrary node A inside a tree network like circuit 4.b. Suppose there are n branches incident on node A. As the network is a tree, there are also n neighboring nodes of A. Among these n neighboring nodes, one node is nearer to the source than node A (call this node A_1), and all other nodes are farther away from the source (call these nodes $A_{2,\cdots,n}$, respectively). By theorem 10, $T_{A_j} = T_A + r_j C_j^L$, for $j = 2,\cdots,n$, where r_j is the resistance between nodes A and A_j, and C_j^L is the total load capacitance of node A_j. From (12), $C_{(A,j)} = \frac{T_A - T_{(A,j)}}{r_j} = -C_j^L$, for $j = 2,\cdots,n$. Again by theorem 10, $T_A = T_{A_1} + r_1 C_A^L$, so $C_{A,1} = C_A^L$. It is easy to check that $\sum_{j=1}^{n} C_{(A,j)} = C_A^L - \sum_{j=2}^{n} C_j^L = C_A$. Note that $C_{(A,j)}$ is negative for $j = 2,\cdots,n$, indicating that node A is driving, not loading node A_j. ∎

We do not intend to solve (11) directly because of the enormous number of variables involved: $\sum_{i=1}^{N}(a_i+1)$. Note that the a_i branches incident on node N_i need not be decoupled completely as we did in the formulation of (11). These branches can be divided into any number $(b_i, 1 \le b_i \le a_i)$ of groups. Rather than fully decoupled into nodes and transistors, the network is decomposed into a smaller number of subnetworks. Delays are calculated directly and independently inside each subnetwork using the techniques discussed in the last section. The consistency of the delay of a common node shared by more than one subnetworks is checked and corrected by a procedure similar to the formulation of (11). As delays can be calculated very efficiently for a tree network, we require that all decomposed subnetworks be trees. The root of every tree must be the source of the network. For convenience, the following terminology is introduced. As node capacitance C_i is partitioned into b_i parts, each of these partitioned capacitances are considered as separated nodes. Such nodes are refered to as "secondary nodes", while the original nodes of the network are refered to as "primary nodes". If there is no ambiguity in the context, the term "node" refers to either a primary node or a secondary node. Those primary nodes with $b_i > 1$ are also called "split primary nodes". Suppose that there are P split primary nodes $(N_{1,\cdots,P})$ and $N-P$ nonsplit ones $(N_{P+1,\cdots,N})$ in the network. With every secondary node is associated an index pair (i,j), indicating the jth secondary node generated from the ith primary node

of the network. The term "equivalent secondary nodes" refers to the set of secondary nodes that correspond to a same primary node. By considering equivalent secondary nodes as disjoint, the decomposition of a network is achieved. The original network is also called the "primary network", and the decomposed network is called the "secondary network". The transformation from a primary network to a secondary network is a two step process. The first step is purely topological, while the second step concerns about the distribution of node capacitances, as well. The first step is refered to as "topological decomposition", and the second step is refered to as "load distribution". For a given RC network, a topological decomposition can always be found that separates the network into a collection of tree subnetworks. This collection of trees can be considered either as disjoint trees or as branches of one single tree that is rooted at the source of the network. Based upon the concept of dominant path [9], one such decomposition scheme is presented in the next section. The discussion in the present section applies to any tree decomposition of RC networks.

As the secondary network is a collection of "independent" tree subnetworks, the delay of each secondary node can be calculated directly. The question arises as to how the delays of these secondary nodes are related to those of the primary nodes. It is quite possible that equivalent secondary nodes have different delay values. Note that the delays of secondary nodes are dependent upon the values of $C_{(i,j)}$'s. If the capacitances C_i's are distributed incorrectly among these secondary nodes, the delay values will be different. However, if the C_i's are somehow distributed so that equivalent secondary nodes give the same delay, then it makes no difference whether these nodes are connected or not (cf. theorem 5). If connected together, the secondary network reduces to the primary network, and the delays of the primary nodes are equal to the common delays of the corresponding set of equivalent secondary nodes. In what follows, we show that for any given tree (topological) decomposition of an RC network, such a load distribution always exists and is unique. Via this distribution, we also present an algorithm to find the delays of all nodes of an arbitrary

network.

From corollary 10, the delay $T_{(i,j)}$ of node $N_{(i,j)}$ is equal to $\sum_{u=1}^{N}\sum_{v=1}^{b_u} R_{(i,j)}^{(u,v)}C_{(u,v)}$, where $R_{(i,j)}^{(u,v)}$ is the resistance of the (unique) path between the source and node $N_{(i,j)}$, that is in common with the (unique) path between the source and node $N_{(u,v)}$. If node $N_{(u,v)}$ and node $N_{(i,j)}$ are not in a same tree subnetwork, then $R_{(i,j)}^{(u,v)} = 0$. Equating $T_{(i,j)}$'s for equivalent secondary nodes, one gets $\sum_{u=1}^{N}\sum_{v=1}^{(b_u)} R_{(i,1)}^{(u,v)}C_{(u,v)} = \sum_{u=1}^{N}\sum_{v=1}^{(b_u)} R_{(i,2)}^{(u,v)}C_{(u,v)} = \cdots = \sum_{u=1}^{N}\sum_{v=1}^{(b_u)} R_{(i,b_i)}^{(u,v)}C_{(u,v)}$, or $\sum_{u=1}^{N}\sum_{v=1}^{(b_u)}(R_{(i,j)}^{(u,v)} - R_{(i,b_i)}^{(u,v)})C_{(u,v)} = 0$, for $j = 1,\cdots,b_i-1$, and $i = 1,\cdots,P$. Since $\sum_{v=1}^{b_u} C_{(u,v)} = C_u$, or $C_{(u,b_u)} = C_u - \sum_{v=1}^{b_u-1} C_{(u,v)}$, the above set of equations can be reduced to

$$\sum_{u=1}^{P}\sum_{v=1}^{b_u-1}\left(R_{(i,j)}^{(u,v)} - R_{(i,b_i)}^{(u,v)} - R_{(i,j)}^{(u,b_u)} + R_{(i,b_i)}^{(u,b_u)}\right)C_{(u,v)} = \sum_{u=1}^{N}\left(R_{(i,b_i)}^{(u,b_u)} - R_{(i,j)}^{(u,b_u)}\right)C_u \qquad (14)$$

(14) represents a system of linear equations with $\sum_{u=1}^{P}(b_i-1)$ variables: $C_{(1,1)},\cdot,C_{(1,b_1-1)}$, $C_{(2,1)},\cdot,C_{(2,b_2-1)},\cdots,C_{(p,1)},\cdot,C_{(p,b_p-1)}$. (14) can also be written in a matrix form $Ax = b$, where A is a $\sum_{i=1}^{p}(b_i-1)\times\sum_{u=1}^{p}(b_i-1)$ matrix with elements $a_{(i,j),(u,v)} = R_{(i,j)}^{(u,v)} - R_{(i,b_i)}^{(u,v)} - R_{(i,j)}^{(u,b_u)} + R_{(i,b_i)}^{(u,b_u)}$. Both b and x are $\sum_{i=1}^{p}(b_i-1)$- vectors with elements $b_{(i,j)} = \sum_{u=1}^{N}(R_{(i,b_i)}^{(u,b_u)} - R_{(i,j)}^{(u,b_u)})C_u$, and $x_{(i,j)} = C_{(i,j)}$. Given a tree decomposition, all $a_{(i,j),(u,v)}$'s and $b_{(i,j)}$'s are fixed. This matrix equation can also be expressed in the following block form:

$$\begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,p} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,p} \\ \vdots & \vdots & & \vdots \\ A_{p,1} & A_{1,2} & \cdots & A_{p,p} \end{pmatrix}\begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_p \end{pmatrix} = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_p \end{pmatrix} \qquad (15)$$

where $A_{i,u} = \begin{pmatrix} a_{(i,1),(u,1)} & \cdots & a_{(i,1),(u,b_u-1)} \\ \vdots & & \vdots \\ a_{(i,b_i-1),(u,1)} & \cdots & a_{(i,b_i-1)(u,b_u-1)} \end{pmatrix}$, $C_u = \begin{pmatrix} x_{(u,1)} \\ \vdots \\ x_{(u,b_u-1)} \end{pmatrix}$, and $B_i = \begin{pmatrix} b_{(i,1)} \\ \vdots \\ b_{(i,b_i-1)} \end{pmatrix}$.

The block Gauss-Seidel method [17] can be used to solve (15), i.e.

$$C_i^{(m+1)} = A_{i,i}^{-1}\left(B_i - \sum_{j=1}^{i-1} A_{i,j} C_j^{(m+1)} - \sum_{j=i+1}^{P} A_{i,j} C_j^{(m)} \right) \qquad (16)$$
$$i = 1, \cdots, P, m \geq 0$$

where superscript $^{(m)}$ indicates the mth step of relaxation. Starting from any initial guess of $C_i^{(0)}$, this method always converges, as is indicated in the following theorems. The proofs of these theorems are given in Appendix 2.

Theorem 12. The $\sum_{i=1}^{N} b_i \times \sum_{i=1}^{N} b_i$ matrix R with elements $R_{(i,j),(u,v)} = R_{(i,j)}^{(u,v)}$ is symmetric and positive-definite [18]. ∎

Theorem 13. The matrix A in (15) is symmetric and positive-definite. ∎

Corollary 14. The matrix A is nonsingular, so the solution of (15) exists and is unique. ∎

Theorem 15. Let A be an $n \times n$ real symmetric matrix. Then the block Gauss-Seidel Method is convergent for all initial $x_i^{(0)}$'s if and only if A is positive-definite. [17] ∎

Corollary 16. The scheme (16) converges for all initial $C_i^{(0)}$'s. ∎

The system of linear equations (15) can be solved by another algorithm which only uses local information during the relaxation process. Given an initial load distribution for a tree decomposition of an RC network, the delays of the secondary nodes are calculated using algorithm TREE (section 4). The relaxation process (RELAX) starts by scanning through the split primary nodes $N_{1,\cdots,p}$, and checking if equivalent secondary nodes give the same values of delay. If they do not, node capacitances are distributed improperly somewhere in the network. Although nothing is known as to where this improper distribution happens, we can always adjust the local distribution of $C_{(i,j)}$'s so that the delays of equivalent secondary nodes are equal for the primary node presently under investigation. The adjustment is

done as follows. Suppose N_i is the current node under investigation, and $T_{(i,1)},\cdots,T_{(i,b_i)}$ are not all equal. Based upon the result of case 3 of theorem 2, the delay of node N_i at this step of relaxation is given by

$$T_i = \frac{\sum_{j=1}^{b_i} \frac{T_{(i,j)}}{R_{(i,j)}}}{\sum_{j=1}^{b_i} \frac{1}{R_{(i,j)}}} \tag{17}$$

where $R_{(i,j)}$ is the source resistances of node $N_{(i,j)}$, and remains fixed during the relaxation process. For the dominant-path decomposition scheme to be described in the next section, the $R_{(i,j)}$ values are determined at the time when the network is decomposed. Let $\Delta_{C_{(i,j)}}$ be the amount of load adjustment for the secondary node $N_{(i,j)}$. Then

$$\Delta_{C_{(i,j)}} = \frac{T_i - T_{(i,j)}}{R_{(i,j)}} \tag{18}$$

The constraint $\sum_{j=1}^{b_i} C_{(i,j)} = C_i$ is satisfied automatically since $\sum_{j=1}^{b_i} \Delta_{C_{(i,j)}} = \sum_{j=1}^{b_i} \frac{T_i - T_{(i,j)}}{R_{(i,j)}} = \left(\sum_{j=1}^{b_i} \frac{1}{R_{(i,j)}}\right) T_i - \sum_{j=1}^{b_i} \frac{T_{(i,j)}}{R_{(i,j)}} = 0$. To maintain consistency, this adjustment of $C_{(i,j)}$'s must be propagated to other nodes in the same tree so that their delays can be updated accordingly $\left(\Delta_{T_{(u,v)}}|_{(i,j)} \equiv \Delta_{T_{(u,v)}}|_{\Delta_{C_{(m,n)}}=0,(m,n)\neq(i,j)} = R^{(i,j)}_{(u,v)} \Delta_{C_{(i,j)}}\right)$. Consider the following two conditions:

1. Before a node is combined with other nodes using (17), the delay of the node is fully updated.
2. No two equivalent secondary nodes lie in a same tree, i.e., $R^{(i,v)}_{(i,j)} = 0$ if $j \neq v$, for $j, v = 1,\cdots,b_i$, and $i = 1,\cdots,P$.

(19)

Theorem 17. If both conditions of (19) are satisfied, then the relaxation process based upon (17) and (18) is equivalent to the block Gauss-Seidel method of (16), the convergence of which is guaranteed. ∎

The proof of this theorem is also given in Appendix 2. Condition 1 of (19) can always be satisfied if, whenever there is a change in $C_{(i,j)}$, this information is propagated to all the nodes in the same tree. However, this is a very time-consuming process. A more efficient approach is to accumulate the changes as the scan process goes along. The delay of a node is not updated until it is scanned. Instead of scanning through split primary nodes, the corresponding secondary nodes are visited in a depth-first manner [20] for each tree subnetwork. This algorithm, called RELAX, is described in a top down manner by the following pigeon ICL[21] code:

```
TYPE PRIMARY_NODE=[SECONDARY_LIST:{SECONDARY_NODE} DONE:BOOL];
    "secondary_list: the corresponding set of equivalent secondary nodes
     done: to make sure that the node is combined only once per relaxation step"


TYPE SECONDARY_NODE=[PRIMARY:PRIMARY_NODE C2:CAPACITANCE
    R:RESISTANCE T,ΔT:DELAY SONS:{SECONDARY_NODE}];
    "primary: the corresponding primary node
     r: the source resistance
     sons: the immediate succeeding nodes"

VAR SOURCE=SECONDARY_NODE; "the source of the network"
    PRIMARY_LIST= {PRIMARY_NODE}; "the primary nodes of the network"
    ERROR=REAL;     "error tolerance"
    CONVERGE=BOOL;

DEFINE RELAX:
    CONVERGE:=FALSE;
    DO RELAXSTEP; WHILE NOT CONVERGE;
ENDDEFN "relax"

DEFINE RELAXSTEP:
   BEGIN VAR N=PRIMARY_NODE; S=SECONDARY_NODE;
      CONVERGE:=TRUE;
      DO @(N).DONE:=FALSE;
      FOR N ∈ PRIMARY_LIST;
      DO SCAN(S,0.0);
      FOR S ∈ SOURCE.SONS;
   END
```

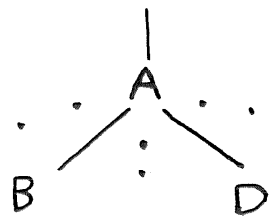


Figure 5. Illustration of the idea behind procedure SCAN

```
ENDDEFN "relaxstep"

DEFINE SCAN(A:SECONDARY_NODE T0:DELAY)=CAPACITANCE:
  BEGIN VAR ∑T; ∑C,C1=CAPACITANCE;  S=SECONDARY_NODE;
    DO @(A).T::=+T0;                                        "a"
       IF NOT A.PRIMARY.DONE THEN COMBINE(A.PRIMARY); FI
    GIVE IF A.SONS = nil THEN A.ΔT/A.R
         ELSE DO ∑T:=T0+A.ΔT;                               "b"
                 ∑C:=0.0;
                 DO C1:=SCAN(S,∑T+A.R*S.C2);
                    ∑C::=+C1;
                    @(S).C2:=∑C;                            "c"
                    ∑T::=+C1*A.R;                           "d"
                    @(A).T::=+C1*A.R;                       "e"
                 FOR S ∈ A.SONS;
                 DO @(S).C2:=∑C-S.C2;                       "f"
                 FOR S ∈ A.SONS;
              GIVE A.ΔT/A.R+∑C
         FI
  END
ENDDEFN "scan"

DEFINE COMBINE(N:PRIMARY_NODE):
  BEGIN VAR T=DELAY; S=SECONDARY_NODE;
    T:=(+S.T/S.R FOR S ∈ N.SECONDARY_LIST;)/
          (+1/S.R FOR S ∈ N.SECONDARY_LIST;);
    DO @(S).ΔT:=T-S.T;
       @(S).T:=T;
       CONVERGE::=AND (ABS(S.ΔT)/S.T < ERROR);
    FOR S ∈ N.SECONDARY_LIST;
    @(N).DONE:=TRUE;
  END
ENDDEFN "combine"
```

Procedure COMBINE implements (17) and (18). The idea behind procedure SCAN is indicated by the following relationship among the three nodes A, B and D of Fig.5:

1. $\Delta_{T_B}|_{A,D}\ (\equiv \Delta_{T_B}|_{\Delta_{C_i}=0, i\neq A,D}) = R_{B,A}\Delta_{C_A} + \Delta_{B,D}\Delta_{C_D} = R_A(\Delta_{C_A} + \Delta_{C_D}) = \Delta_{T_A}|_{A,D}$.

2. $\Delta_{T_A}|_{B,D} = R_{A,B}\Delta_{C_B} + R_{A,D}\Delta_{C_D} = R_A(\Delta_{C_B} + \Delta_{C_D})$.

The first equation above suggests the accumulation and propagation of Δ_T (parameter $T0$ of procedure SCAN) from the driving end towards the loading ends as the nodes of a tree is scanned in a depth-first manner. The second equation suggests the accumulation of Δ_C (returned by procedure SCAN) from the loading ends towards the driving end. If branch B is scanned before branch D, then $\Delta_{T_D}|_B$ is in effect at the present scan. On the other hand, the value of Δ_{C_D} is stored at variable B.C2 to update T_B when node B is scanned at the next relaxation step.

To discuss this algorithm in more detail, the following terminology is used. Let $C(X)$ be the set of nodes along the path from the source to node X, excluding node X itself. If a node A is in $C(B)$, then A is called an "ancestor" of B, and this relation is denoted by $A \prec B$. A is called the "parent" of B if A is adjacent to B, and $A \prec B$. Every node N except the source has a unique parent which is denoted by $p(N)$. Let $\mathrm{com}(A,B)$ denote the common ancestor of A and B such that every other common ancestor of A and B is also an ancestor of $\mathrm{com}(A,B)$. Given two nodes A and B, $\mathrm{com}(A,B)$ always exists, and is unique. Also $R_{A,B} = R_{\mathrm{com}(A,B)}$. Let $S(A) = \{X \mid p(X) = A\}$, and $D(A) = \{X \mid A \preceq X\}$. With $S(A)$ is associated an ordering $O_A : S(A) \rightarrow \{1, \cdots, |\, S(A)\,|\}$, where $|\, S(A)\,|$ is the size of $S(A)$. A.SONS in the above code corresponds to the set $S(A)$, and O_A indicates the ordering of the nodes in A.SONS. Let $E(A) = \{X \mid X \in S(p(A)),\ O_{p(A)}(X) < O_{p(A)}(A)\}$, and $F(A) = \{X \mid X \in S(p(A)),\ O_{p(A)}(A) < O_{p(A)}(A)\}$. Define a total ordering $\prec_s$ among the nodes in a tree network as follows. For any two distinct nodes X and Y, let $Z = \mathrm{com}(X,Y)$,

1. If $X \prec Y$, i.e. $Z = X$, then $X \prec_s Y$.

2. If $Y \prec X$, then $Y \prec_s X$.

3. If $X, Y \in S(Z)$, then $X \prec_e Y$ if $O_Z(X) < O_Z(Y)$, and vice versa.

4. Otherwise, let X_1 and Y_1 be the two nodes such that $X \preceq X_1$, $Y \preceq Y_1$, and $X_1, Y_1 \in S(Z)$. Then $X \prec_e Y$ if $X_1 \prec_e Y_1$, and vice versa.

That is to say, if node X is scanned before node Y in the above code, then $X \prec_e Y$, and vice versa. Let $Q(A) = \{X \mid X \prec_e A\}$. The following relations among the sets defined above are observed.

Lemma 18. $D(A) = \{A\} \cup \left(\cup_{x \in S(A)} D(x)\right)$. ∎

Lemma 19. If A is the parent of S, i.e. $A = p(S)$, then $Q(S) = Q(A) \cup \{A\} \cup \left(\cup_{x \in E(S)} D(x)\right)$. ∎

As $T_A = \sum_x R_{A,x} C_x = \sum_x R_{\mathrm{com}(A,x)} C_x$,

$$
\begin{aligned}
\Delta T_A &= \Delta\left(\sum_x R_{\mathrm{com}(A,x)} C_x\right) \\
&= \sum_x R_{\mathrm{com}(A,x)} \Delta C_x \\
&= \sum_{N \in C(A)} R_N \left(\sum_{x \in B_{A,N}} \Delta C_x\right) + R_A \sum_{x \in D(A)} \Delta C_x \\
&= \sum_{N \in C(A)} R_N \left(\sum_{x \in G_{A,N}} \Delta C_i + \sum_{x \in H_{A,N}} \Delta C_i\right) + R_A \sum_{x \in D(A)} \Delta C_x \\
&= \sum_{N \in C(A)} \left(R_N \sum_{x \in G_{A,N}} \Delta C_x\right) + R_A \sum_{x \in D(A)} \Delta C_x + \sum_{N \in C(A)} \left(R_N \sum_{x \in H_{A,N}} \Delta C_x\right) \\
&= \Delta_1 T_A + \Delta_2 T_A + \Delta_3 T_A
\end{aligned}
\tag{20}
$$

where $B_{A,N} = \{X \mid \mathrm{com}(A,X) = N\}$ is the set of nodes that are in the side branches of $C(A)$, and connect to $C(A)$ at node N, including node N. $G_{A,N} = B_{A,N} \cap Q(A)$, and $H_{A,N} = B_{A,N} - G_{A,N}$. $\Delta_1 T_A = \sum_{N \in C(A)} R_N (\sum_{x \in G_{A,N}} \Delta C_x)$, $\Delta_2 T_A = R_A (\sum_{x \in D(A)} \Delta C_x)$, and $\Delta_3 T_A = \sum_{N \in C(A)} R_N (\sum_{x \in H_{A,N}} \Delta C_x)$.

Lemma 20. $Q(S) = \bigcup_{N \in C(S)} G_{S,N}$. Moreover, if $A = p(S)$, then

$$\bigcup_{N \in C(S)} G_{S,N} = \left(\bigcup_{N \in C(A)} G_{A,N} \right) \bigcup \{A\} \bigcup \left(\bigcup_{x \in E(S)} D(x) \right)$$

$$\bigcup_{N \in C(S)} H_{S,N} = \left(\bigcup_{N \in C(A)} H_{S,N} \right) \bigcup \left(\bigcup_{x \in F(S)} D(x) \right)$$

∎

Theorem 21. As procedure SCAN(A,T0) is called and executed at the mth relaxation step, the following statements are true.

1. The value of $\sum_{x \in D(A)} \Delta C_x^{(m)}$ is returned by the procedure.
2. When the procedure returns, variable $A.C2$ contains the value of $\sum_{x \in F(A)} \sum_{i \in D(x)} \Delta C_i^{(m)}$.
3. The value of the parameter $T0$ equals $\Delta_1 T_A^{(m)} + \Delta_3 T_A^{(m-1)}$.
4. Before entering procedure COMBINE, T_A is updated to be consistent with all the changes of $C_i^{(k)}$ for $k < m, \forall i$, and for $k = m$, $i \prec_s A$.

Proof:

1. (1) is proved by induction. If a node A is a leaf, i.e., $S(A) = \phi$, then $D(A) = \{A\}$. In this case, SCAN returns $A.\Delta_T / A.R = \Delta C_A^{(m)}$. Suppose $S(A) \neq \phi$, and (1) is true for all nodes $X \in D(A)$. In this case, SCAN returns $A.\Delta_T / A.R + \sum_C = \Delta C_A^{(m)} + \sum_{S \in S(A)} \text{SCAN}(S, \cdot) = \Delta C_A^{(m)} + \sum_{S \in S(A)} \sum_{i \in D(S)} \Delta C_i^{(m)}$, which equals $\sum_{i \in D(A)} \Delta C_i^{(m)}$, by lemma 18.
2. By (1), SCAN$(S1, \cdot)$ returns $\sum_{i \in D(S1)} \Delta C_i^{(m)}$. When instruction c marked in the code is executed, $S.C2 = \sum_{S1 \in E(S) \bigcup \{S\}} \text{SCAN}(S1, \cdot)$. Then at the time when instruction f is executed, $\sum_C = \sum_{S1 \in S(A)} \text{SCAN}(S1, \cdot)$ where $A = p(S)$. Thus $S.C2 = \sum_{S1 \in F(S)} \text{SCAN}(S1, \cdot) = \sum_{S1 \in F(S)} \sum_{i \in D(S1)} \Delta C_i^{(m)}$.
3. (3) is proved by induction. If node S is connected to the source directly, then $C(S) = \phi$, and $\Delta_1 T_S^{(m)} = \Delta_3 T_S^{(m-1)} = 0$. Suppose node S is not connected directly to the source,

and (3) is true for $p(S) = A$. Then by lemma 20, $\Delta_1 T_S^{(m)} = \Delta_1 T_A^{(m)} + R_A(\Delta C_A^{(m)} + \sum_{x\in E(S)} \sum_{i\in D(x)} \Delta C_i^{(m)})$, and $\Delta_3 T_S^{(m-1)} = \Delta_3 T_A^{(m-1)} + R_A(\sum_{x\in F(S)} \sum_{i\in D(x)} \Delta C_i^{(m-1)})$. Thus

$$\Delta_1 T_S^{(m)} + \Delta_3 T_S^{(m-1)} = \Delta_1 T_A^{(m)} \Delta_3 T_A^{(m-1)} + R_A \Delta C_A^{(m)} + R_A \sum_{x\in E(S)} \sum_{i\in D(x)} \Delta C_i^{(m)} + R_A \sum_{x\in F(S)} \sum_{i\in D(x)} \Delta C_i^{(m-1)}$$

By (1) and (2), $\Delta_1 T_S^{(m)} + \Delta_3 T_S^{(m-1)} = \Delta_1 T_A^{(m)} + \Delta_3 T_A^{(m-1)} + R_A \Delta C_A^{(m)} + R_A \sum_{x\in E(S)} \text{SCAN}(x, \cdot) + R_A(S.C2)$. $\Delta_1 T_A^{(m)} + \Delta_3 T_A^{(m-1)}$ is entered as the parameter $T0$. $T0$ and $R_A \Delta C_A^{(m)}$ are added up to $\sum_T$ by instruction b. $R_A \sum_{x\in E(S)} \text{SCAN}(S, \cdot)$ is added to $\sum_T$ at instruction d. Finally when SCAN(S,T) is called, the parameter T equals $\sum_T + R_A(S.C2) = \Delta_1 T_S^{(m)} + \Delta_3 T_S^{(m-1)}$.

4. (4) is proved by induction on m. When SCAN(A,T0) is called at the first relaxation step, T0=$\Delta_1 T_A^{(1)}$ is added to $A.T$, or T_A, by instruction a before calling procedure COMBINE. Suppose (4) is true for $k = 1$ up to $m-1$. Then in procedure COMBINE ($(m-1)$th step), $R_A \Delta C_A^{(m-1)}$ is added to $A.T$. At instruction e, $R_A \text{SCAN}(S, \cdot) = R_A \sum_{i\in D(S)} \Delta C_i^{(m-1)}$ is added to A.T one by one for $S \in S(A)$. Thus all terms of $\Delta_2 T_A^{(m-1)}$ are added to $A.T$ before the end of this relaxation step. As procedure SCAN(A,T0) is called at the mth relaxation step, by (3), $T0 = \Delta_1 T_A^{(m)} + \Delta_3 T_A^{(m-1)}$ which is added to $T0$ before procedure COMBINE is called. ∎

If we consider the collection of tree subnetworks as branches of one single tree rooted at the source, then all the secondary nodes in the network are totally ordered by relation $\prec_s$. Define a mapping f from the set of primary nodes to the set of secondary nodes by $f(p) = x$ such that $x \in S_p$, and $y \prec_s x$ for $\forall y \in S_p$, $y \neq x$, where S_p is the corresponding set of equivalent secondary nodes of p. To put it differently, $f(p)$ is the first secondary node in S_p that is scanned. The nodes in S_p are combined when and only when node $f(p)$ is scanned. Induced by $\prec_s$ is the totally ordering $\prec_p$ among primary nodes: $p_1 \prec_p p_2$ iff

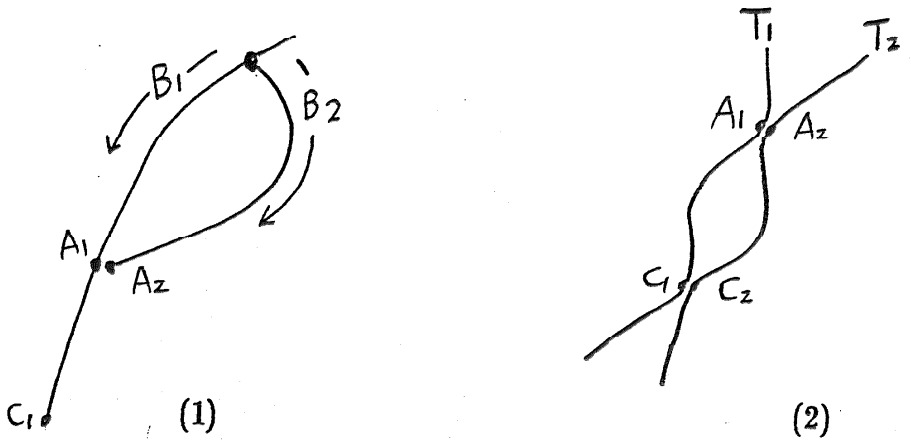


Figure 6. Two situations in which condition 1 of (19) is not satisfied

$f(p_1) \prec_s f(p_2)$. Implied in (16) of theorem 17 is exactly the ordering $\prec_p$ among split primary nodes. Condition 1 of (19) is satisfied if the following relation holds for all $p_1, p_2 = 1, \cdots, P$:

$$f(p_1) \prec_p f(p_2) \rightarrow (x \prec_s y \text{ or } R_{x,y} = 0, \quad \forall x \in S_{p_1}, \forall y \in S_{p_2})$$

Condition 1 of (19) is satisfied in most cases, except the two situations shown in Fig.6. In case 1, nodes A_1 and A_2 are equivalent, and they are in the same tree as node C_1. Suppose branch B_1 is scanned before branch B_2. Then, at the time when node C_1 is scanned, $T_{(C,1)}$ is only partially updated (Since $A_1 \prec_s C_1 \prec_s A_2$, the adjustment of $T^{(m)}_{(C,1)}$ due to $\Delta^{(m)}_{C_{(A,1)}}$ is in effect, but that due to $\Delta^{(m)}_{C_{(A,2)}}$ is not). In case 2, nodes A_1 and C_1 are in a same tree, while A_2 and C_2 are in another. Suppose tree T_1 is scanned before tree T_2. Then, at the time when C_1 is scanned, node C_2 is not fully updated yet ($A_1 \prec_s C_1 \prec_s A_2 \prec_s C_2$, so the adjustment of $T_{(C,2)}$ due to $\Delta_{C_{(A,2)}}$ is not in effect).

Algorithm RELAX is applied regardless whether the two conditions of (19) are satisfied or not. Convergence of this algorithm has always been observed, although the convergence rate is slower when condition 2 is not satisfied. Note that only secondary nodes that correspond to split primary nodes need to be considered in the relaxation process. The delays of other nodes can be calculated after the relaxation process terminates.

Theorem 22. The time complexity of algorithm RELAX is $O(l \cdot Q)$, where l is the number of relaxation steps used and $Q = \sum_{i=1, b_i \neq 1}^{N} b_i = \sum_{i=1}^{P} b_i$ is the number of secondary nodes corresponding to split primary nodes.

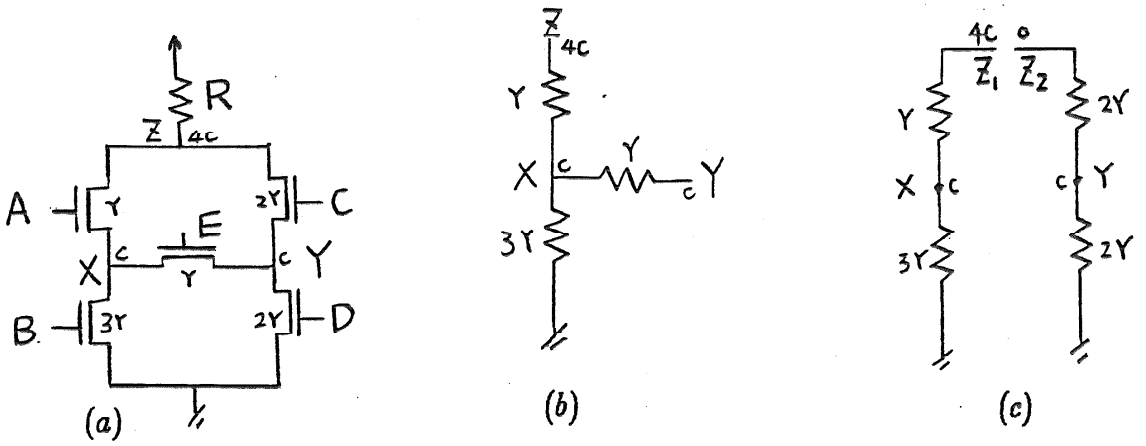


Figure 7. An NMOS Circuit to illustrate algorithms TREE and RELAX

Proof: COMBINE(p) is of time complexity $O(|\ S_p\ |)$ and, at each relaxation step, COMBINE is called exactly once for each split primary node. The other part of the code SCAN(A,$T0$) takes time $O(|\ D_A\ |)$, which can be easily checked by induction. The theorem follows from the fact that $\sum_p(|\ S_p\ |) = Q$, and $|\ D_{source}\ | = Q$. ∎

The number of relaxation steps required depends on the accuracy aimed at. Usually four or five steps are enough to bring the error down to within 10%.

Example 23. Consider the NMOS circuit shown in Fig.7.a. Let A, B, C, D, E indicate the input nodes (also transistors) of the circuit, and X, Y, Z indicate the internal nodes. The values marked by the transistors are their ON-resistances and those by the nodes their capacitances. In order for this circuit to function properly, the condition $R \gg 6r$ must be satisfied. Initially, transistors D and B are OFF, and transistors A and C are ON, so all the internal nodes are at voltage level VDD. At time 0^+, the topology of the network changes, and various internal nodes are pulled down towards a voltage level very close to GND. The delays of these nodes are determined using algorithms TREE and RELAX. As a shorthand, let $(b_1, b_2, b_3, b_4, b_5)$ denote the logic level of the five input nodes A, B, C, D, E, where $b_{1,\cdots,5} = 0, 1$.

1. $(1, 1, 0, 0, 1)$: The corresponding RC network in this case is a tree (Fig7.b). Therefore, the dealys can be calculated directly using algorithm TREE, and no relaxation process is required.

   ▸ phase 1 (backward): $C_y^L = c$, $C_z^L = 4c$, and $C_x^L = C_y^L + C_z^L + c = 6c$.

- phase 2 (forward): $T_x = 3r \cdot 6c = 18rc$, $T_y = T_x + r \cdot c = 19rc$, and $T_z = T_x + r \cdot 4c = 22rc$.

2. $(1,1,1,1,0)$: The network in this case is not a tree, so some decomposition is necessary. Say, node capacitance C_z is split into two parts: $C_{(z,1)}$, and $C_{(z,2)}$ (Fig.7.c). The source resistances of various nodes are $R_x = 3r$, $R_y = 2r$, $R_{(z,1)} = R_x + r = 4r$, and $R_{(z,2)} = R_y + 2r = 4r$. Assume that $C_{(z,1)} = 4c$, and $C_{(z,2)} = 0$ initially.

a. Initialization of delays using TREE:

- phase 1: $C^{L(0)}_{(z,1)} = 4c$, $C^{L(0)}_x = 5c$ (tree 1), $C^{L(0)}_{(z,2)} = 0$, and $C^{L(0)}_y = c$ (tree 2).
- phase 2: $T^{(0)}_x = 15rc$, $T^{(0)}_{(z,1)} = 19rc$ (tree 1), $T^{(0)}_y = 2rc$, and $T^{(0)}_{(z,2)} = 2rc$ (tree 2).

b. RELAX, step 1:

- Check node Z (forward): $T^{(1)}_z = \frac{\frac{T^{(0)}_{(z,1)}}{R_{(z,1)}} + \frac{T^{(0)}_{(z,2)}}{R_{(z,2)}}}{\frac{1}{R_{(z,1)}} + \frac{1}{R_{(z,2)}}} = \frac{10+2}{2} rc = 10.5rc$. Then, $\Delta^{(1)}_{C_{(z,1)}} = \frac{10.5-19}{4} c = -2.125c$, and $\Delta^{(1)}_{C_{(z,2)}} = \frac{10.5-2}{4} c = 2.125c$.
- Correct T_x and T_y due to the changes of $C_{(z,1)}$ and $C_{(z,2)}$ (backward): $T^{(1)}_x = T^{(0)}_x + R_x \Delta_{C_{(z,1)}} = (15 - 3 \cdot 2.125)rc = 8.625rc$, and $T^{(1)}_y = T^{(0)}_y + R_y \Delta_{C_{(z,2)}} = (2 + 2 \cdot 2.125)rc = 6.25rc$.

One relaxation step gives us the exact delays of all the nodes. This is in general true if there is only one node to split, and condition 2 of (19) is satisfied. As a comparison, the delays of node X, Y, and Z are calculated again using the techniques presented in the last section. The triple (R, C, D) is used to represent a two-port RC network, where R, C, D are the parameters described in theorem 2. Let $-$ and $\|$ denote series and parallel connections, respectively, and $-$ has precedence over $\|$. The RC networks between nodes X, Y, Z and GND can be individually represented and reduced as follows:

$$\begin{aligned}(Z): \ & (3r, c, 3rc) - (r, 2c, 2rc) \| (2r, c, 2rc) - (2r, 2c, 4rc) \\ =& (4r, 3c, 11rc) \| (4r, 3c, 10rc) \\ =& (2r, 6c, 10.5rc)\end{aligned}$$

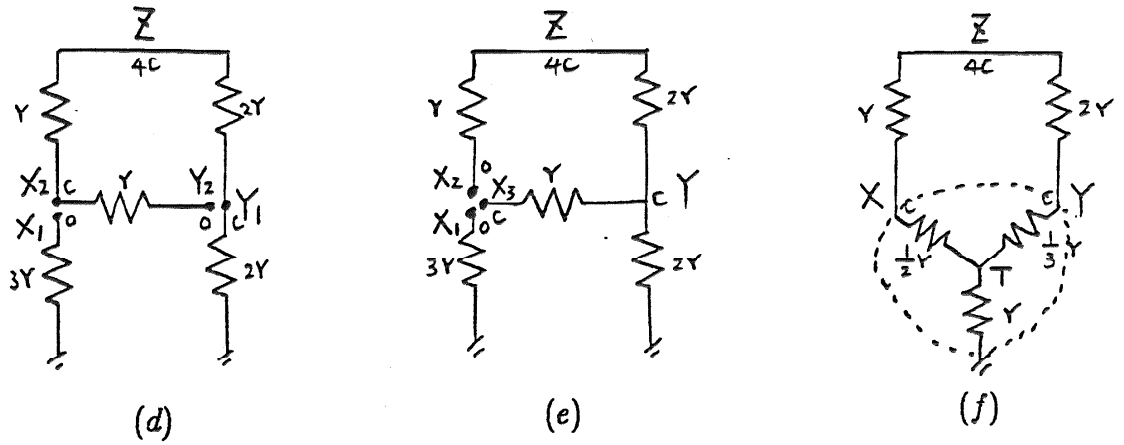


Figure 7(continued). An NMOS Circuit to illustrate algorithms TREE and RELAX

$$(X): (3r, c, 3rc) \| (2r, c, 2rc) - (2r, 4c, 8rc) - (r, 0, 0)$$
$$= (3r, c, 3rc) \| (5r, 5c, 18rc)$$
$$= (1.875r, 6c, 8.625rc)$$

$$(Y): (2r, c, 2rc) \| (3r, c, 3rc) - (r, 4c, 4rc) - (2r, 0, 0)$$
$$= (2r, c, 2rc) \| (6r, 5c, 19rc)$$
$$= (1.5r, 6c, 6.25rc)$$

By corollary 7, $T_z = 10.5rc$, $T_x = 8.625rc$, and $T_y = 6.25rc$.

3. $(1, 1, 1, 1, 1)$: Suppose the tree decomposition of Fig.7.d is selected. Both node X and Y are split, which results in two chains: GND - Y_1 - Z - X_2 - Y_2, and GND - X_1, respectively. The source resistances of various nodes are $R_{(y,1)} = 2r$, $R_z = 4r$, $R_{(x,2)} = 5r$, $R_{(y,2)} = 6r$, and $R_{(x,1)} = 3r$. Assume that $C_{(x,1)} = 0$, $C_{(x,2)} = c$, $C_{(y,1)} = c$, and $C_{(y,2)} = 0$ initially.

a. Initialization of delays using TREE:

- $C_{y,2}^{L(0)} = 0$, $C_{x,2}^{L(0)} = c$, $C_z^{L(0)} = 5c$, $C_{(y,1)}^{L(0)} = 6c$ (tree 1), and $C_{(x,1)}^{L(0)} = 0$ (tree 2).
- $T_{(y,1)}^{(0)} = 12rc$, $T_z^{(0)} = 22rc$, $T_{(x,2)}^{(0)} = 23rc$, $T_{(y,2)}^{(0)} = 23rc$ (tree 1), and $T_{(x,1)}^{(0)} = 0$ (tree 2).

b. RELAX, step 1: Suppose tree 1 is scanned before tree 2.

- Check node Y_1: $T_y^{(1)} = \frac{\frac{T_{(y,1)}}{R_{(y,1)}} + \frac{T_{(y,2)}}{R_{(y,2)}}}{\frac{1}{R_{(y,1)}} + \frac{1}{R_{(y,2)}}} = \frac{\frac{12}{2} + \frac{23}{6}}{\frac{1}{2} + \frac{1}{6}} rc = 14.75rc$. Then $\Delta_{C_{(y,1)}}^{(1)} = \frac{14.35 - 12}{2} c = 1.375rc$, $\Delta_{C_{(y,2)}} = \frac{14.75 - 23}{6} c = -1.375c$. Also $\sum_T = 14.75rc - 12rc = 2.75rc$.
- Check node X_2 (forward): $T_{(x,2)}$ is first updated to $23rc + \sum_T = 25.75rc$ (the effect of $\Delta_{C_{(y,1)}}^{(1)}$ on $T_{(x,2)}$). Then $T_x^{(1)} = \frac{\frac{25.75}{5} + \frac{0}{3}}{\frac{1}{5} + \frac{1}{3}} rc = 9.66rc$. Thus, $\Delta_{C_{(x,1)}}^{(1)} = \frac{9.66 - 0}{3} c = 3.22c$,

and $\Delta^{(1)}_{C_{(x,2)}} = \frac{9.66-25.75}{5}c = -3.22c$. Also $\sum_T = (2.75 + (9.66 - 25.75))rc = -13.34rc$ (or $(9.66 - 23)rc$).

- Update $T_{(y,2)}$ (forward): $T_{(y,2)}$ is first updated to $14.75rc + \sum_T = 1.41rc$. As node Y is already checked, nothing needs to be done further. On the other hand, an end of the tree is reached, so the backtrack phase starts with $\sum_C = \Delta_{C_{(x,2)}} = -1.375c$.
- Update $T_{(x,2)}$ (backward): $T^{(1)}_{(x,2)}$ is updated to $9.66rc + R_{(x,2)}\sum_C = (9,66 - 5 \cdot 1.375)rc = 2.78rc$. Also $\sum_C$ is accumulated up to $-1.375c + \Delta^{(1)}_{C_{(x,2)}} = -4.595c$.
- Update $T_{(y,1)}$ (backward): $T^{(1)}_{(y,1)}$ is updated to $14.75rc + R_{(y,1)}\sum_C = (14.75 - 2 \cdot 4.595)rc = 5.56rc$.
- Nothing is done for tree 2 since node X has already been scanned.

In summary, after the first step of relaxation, $C_{(x,1)} = 3.22c$, $C_{(x,2)} = -2.22c$, $C_{(y,1)} = 2.38c$, $C_{(y,2)} = -1.38c$, $T_{(x,1)} = 9.66rc$, $T_{(x,2)} = 2.78rc$, $T_{(y,1)} = 5.56rc$, and $T_{(y,2)} = 1.41rc$. The delay of every node is fully updated.

step	0	1	2	3	4	5	10	>12
$C_{(x,1)}$	0	3.22	2.23	2.58	2.49	2.54	2.56	2.56
$C_{(x,2)}$	1	-2.22	-1.23	-1.58	-1.49	-1.54	-1.56	-1.56
$T_{(x,1)}$	0	9.66	6.69	7.75	7.48	7.62	7.68	7.69
$T_{(x,2)}$	23	2.78	9.29	7.19	7.86	7.65	7.69	7.69
$C_{(y,1)}$	1	2.38	1.86	1.97	1.89	1.88	1.83	1.83
$C_{(y,2)}$	0	-1.38	-0.86	-0.97	-0.89	-0.88	-0.83	-0.83
$T_{(y,1)}$	12	5.56	7.54	6.84	7.01	6.92	6.88	6.87
$T_{(y,2)}$	23	1.41	8.43	6.22	6.97	6.76	6.86	6.87

Table 1. Values of C's and T's at the end of some relaxation steps

The results at the end of some relaxation steps are summarized at table 1. Note that node Z is not split, and thus is not involved in the relaxation process. The value of T_z is determined after the process terminates. In this case, T_y is $6.87rc$, and $C^L_z = C_{y,2} + C_{x,2} + C_z = (-1.56 - 0.83 + 4)c = 1.61c$, so $T_z = (6.87 + 2 \cdot 1.61)rc = 10.09rc$. Consider another tree

decomposition shown in Fig.7.e: node X is split into three parts, and tree 1 is not a simple chain. The source resistances of various nodes are $R_{(x,1)} = 3r$, $R_{(x,2)} = 5r$, $R_{(x,3)} = 3r$, and $R_y = 2r$. Assume $C_{(x,1)} = 0$, $C_{(x,2)} = 0$, and $C_{(x,3)} = c$ initially.

a. Initialization of delays using TREE: $T_y^{(0)} = 12rc$, $T_{(x,2)}^{(0)} = 20rc$, $T_{(x,3)}^{(0)} = 13rc$ (tree 1), and $T_{(x,1)}^{(0)} = 0$ (tree 2).

b. RELAX, step 1: Assume that tree 1 is scanned before tree 2, and branch 1 of tree 1 is scanned before branch 2.

- Check node X_2: $T_x^{(1)} = \frac{\frac{T_{(x,1)}}{R_{(x,1)}} + \frac{T_{(x,2)}}{R_{(x,2)}} + \frac{T_{(x,3)}}{R_{(x,3)}}}{\frac{1}{R_{(x,1)}} + \frac{1}{R_{(x,2)}} + \frac{1}{R_{(x,3)}}} = \frac{0 + \frac{20}{5} + \frac{13}{3}}{\frac{1}{3} + \frac{1}{5} + \frac{1}{3}} = 9.61rc$. Then $\Delta_{C_{(x,2)}}^{(1)} = \frac{9.61 - 20}{5} c = -2.08c$, $\Delta_{C_{(x,3)}}^{(1)} = -1.13c$, and $\Delta_{C_{(x,1)}}^{(1)} = 3.21c$.

- Back to node Y: $\sum_C = \Delta_{C_{(x,2)}}^{(1)} = -2.08c$, and $\sum_T = R_y \cdot \sum_C = -4.16rc$.

- Update node X_3: $T_{(x,3)}^{(1)}$ is updated to $9.61rc + \sum_T = 5.45rc$. Nothing is done further, except that $\Delta_{C_{(x,3)}}^{(1)}$ is recorded in branch 1 of tree 1 to update $T_{(x,2)}$ at step 2.

After step 1, $C_{(x,1)} = 3.21c$, $C_{(x,2)} = -2.08c$, $C_{(x,3)} = -0.13c$, $T_{(x,1)} = 9.61rc$, $T_{(x,2)} = 9.61rc$, and $T_{(x,3)} = 5.45rc$. $T_{(x,2)}$ is only partially updated (the actual value is $9.61rc - R_y \cdot \Delta_{C_{(x,3)}} = 7.35rc$).

c. RELAX, step 2:

- Check node X_2: $T_{(x,2)}$ is updated to $7.35rc$ first. Then $T_x^{(2)} = \frac{\frac{9.61}{3} + \frac{7.35}{5} + \frac{5.45}{3}}{\frac{13}{15}} rc = 7.49rc$. Thus $\Delta_{C_{(x,2)}}^{(2)} = 0.03c$, and $\Delta_{C_{(x,3)}}^{(2)} = 0.68c$.

- Update node X_3: $T_{(x,3)}$ is updated to $7.49rc + R_y \cdot \Delta_{C_{(x,2)}}^{(2)} = 7.55rc$. $T_{(x,2)}$ is not fully updated, and $\Delta_{C_{(x,3)}}^{(2)}$ is recorded for step 3.

step	0	1	2	3	4	5	6	>10
$C_{(x,1)}$	0	3.21	2.50	2.61	2.57	2.57	2.57	2.56
$C_{(x,2)}$	0	-2.08	-2.05	-2.25	-2.31	-2.35	-2.37	-2.38
$C_{(x,3)}$	1	-0.13	0.55	0.64	0.74	0.78	0.80	0.82
$T_{(x,1)}$	0	9.61	7.50	7.83	7.72	7.72	7.70	7.69
$T_{(x,2)}$	20	9.61	7.50	7.83	7.72	7.72	7.70	7.69
$T_{(x,3)}$	13	5,46	7.55	7.42	7.60	7.64	7.67	7.69
$T'_{(x,2)}{}^{*}$	20	7.36	8.85	8.02	7.91	7.79	7.75	7.69

*$T'_{(x,2)}$ is the fully updated value of $T_{(x,2)}$

Table 2. Values of C's and T's at the end of some relaxation steps

Table 2 summarizes the results at the end of some relaxation steps. T_y and T_z are calculated after the process terminates: $T_y = T_{(x,3)} - r \cdot C_{(x,3)} = 6.87rc$, and $T_z = T_{(x,2)} - r \cdot C_{(x,2)} = 10.09rc$. Note that (11) is applied for the above calculation. To calculate the delay of node Z directly, a $\Delta - Y$ transformation among the three nodes GND, X, Y is necessary. One phantom node T is generated by this transformation, and the resulting network is shown in Fig.7.f. Note that this transformation is not necessary for node X and Y. The network between nodes X, Y, Z and GND are as follows:

$$
\begin{aligned}
(Z): &\ (r,0,0) - \{(0.5r,c,0.5rc) - (r,2c,2rc) \| (0.33r,c,0.33rc) - (2r,2c,4rc)\} \\
(X): &\ (3r,c,3rc) \| \{(2r,c,2rc) - ((2r,4c,8rc) - (r,0,0) \| (r,0,0))\} \\
(Y): &\ (2r,c,2rc) \| \{(3r,c,3rc) - ((r,4c,4rc) - (2r,0,0)) \| (r,0,0)\}
\end{aligned}
$$

It can be easily checked that $T_x = 7.69rc$, $T_y = 6.87rc$, and $T_z = 10.09rc$. ∎

Before finishing this section, we give an example to indicate that a Jacobi-like method [17] (update the delays simultaneously after all the nodes are scanned) may lead to divergence of the relaxation scheme.

Example 24. Consider the circuit shown in Fig.8.a. There are 101 branches incident on node N. Paths $P_{1,\cdots,100}$ all look identical: P_i connects to node N_i, which in turn connects the source. Path 101 connects to the source directly. Suppose that all nodes $N_{1,\cdots,100}$ are

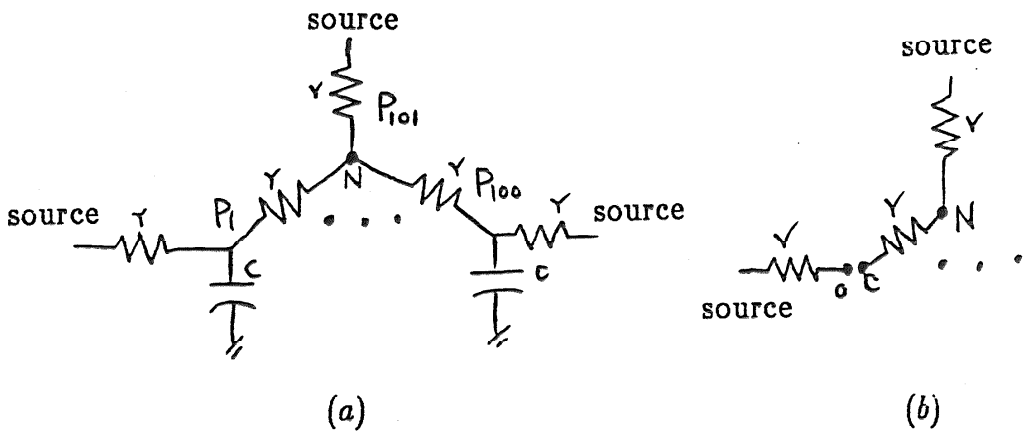


Figure 8. Example for which the Jacobi method diverges

split into two parts (Fig.8.b), and the network is decomposed into 101 tree subnetworks. If the initial guess of the load distribution is $C_{(N_i,1)} = c$, and $C_{(N_i,2)} = 0$ for $i = 1, \cdots, 100$, then $C_N^{L(0)} = 101c$, $T_N^{(0)} = 101rc$, and $T_{N_i,1}^{(0)} = 102rc$. On the other hand, $T_{(N_i,2)} = 0$. At the first step of relaxation, $T_{N_i}^{(1)} = \frac{102}{2}rc = 51rc$, and $\Delta_{C_{(N_i,1)}}^{(1)} = -25.5c$. Reflecting all these changes of capacitances back to node N, $T_N^{(1)}$ becomes $101rc - 100 \cdot r \cdot 25.5c = -2449rc$. At the second step of relaxation, $T_{(N_i,1)}^{(2)} = -2473.5rc$, $T_{N_i}^{(2)} = \frac{-2473.5+102}{2}rc = -1185.75rc$. Then $\Delta_{C_{(N_i,1)}}^{(2)} = 643.875c$. Reflecting these changes back to node N, $T_N^{(2)}$ becomes $-2449rc + 100 \cdot r \cdot 643.875c = 61938.5rc$. It can be easily seen the absolute values of $\Delta_{C_{N_i}}$'s, $T_{N_i,1}$'s and T_N's will grow indefinitely, and the process diverges. On the other hand, both conditions of (19) are satisfied by this example, so algorithm RELAX converges. ∎

§6 Transistor Networks - The Dominant Path Scheme

The concept of dominant path has been successfully used in logic simulation for digital MOS circuits [9,22]. We show, in this section, that this concept also leads to a natural tree decomposition of RC networks. In most cases, the delays estimated from each decomposed tree subnetwork are already very accurate. For other cases, the technique of load redistribution introduced in the last section can be used to calculate the exact delay of every node. The processes of logic simulation and delay estimation are combined into one unified process. The mechanism for scheduling logic events is quite similar to that presented by Terman [4]. As each new event is invoked, the delay values as well as the logic states of the nodes of the network are updated incrementally.

Dominant Path

Consider a transistor network driven by both VDD and GND. There are certain number of paths that lead a node in the network to either one of the sources. Among all these paths, the ones with the smallest series resistance (the source resistance) are called the dominant paths of the node (in general, there are more than one such path). In most practical cases, all the dominant paths of a node lead to a same source, and the logic state of the node is set to that of the source, 1 for VDD and 0 for GND. Similar to the formulation of (11), the source resistances of the nodes and the series resistance of the paths are related as

$$\begin{cases} R_i^j = R_j + r_{i,j} & j = 1, \cdots, a_i,\ i = 1, \cdots, N \\ R_i = \min_{j=1,\cdots,a_i} R_i^j & i = 1, \cdots, N \end{cases} \tag{21}$$

where R_i is the source resistance of node N_i, $r_{i,j}$ is the resistance between node N_i and its jth neighbor, and R_i^j is the series resistance of the jth path of node N_i. The determination of the R_i and R_i^j values are equivalent to the single-source shortest path problem with positive costs. There exist quite a few algorithms for solving this problem, for example, Dijkstra's algorithm [20]. The solution of (21) exists and is unique, and the time complexity of the algorithm is $O(n^2)$, where n is the number of nodes in the network. Equation (21) needs to be solved only once and, as the transistor network evolves, the dominant paths and source resistances of the nodes and the series resistances of the paths can be updated incrementally.

The dominant paths of the nodes in a network lead to a very efficient scheme for estimating delays. Recall, in section 2, that any block of an RC network is driven by one and only one source. If a block B is driven by both VDD and GND, then this block is decomposed into two (or more) blocks: B_{VDD} and B_{GND} which consist of the nodes of B whose dominant paths lead to VDD and GND, respectively. A resistor between two nodes in a same block (B_{VDD} or B_{GND}) is considered to be a resistor of the block. A resistor

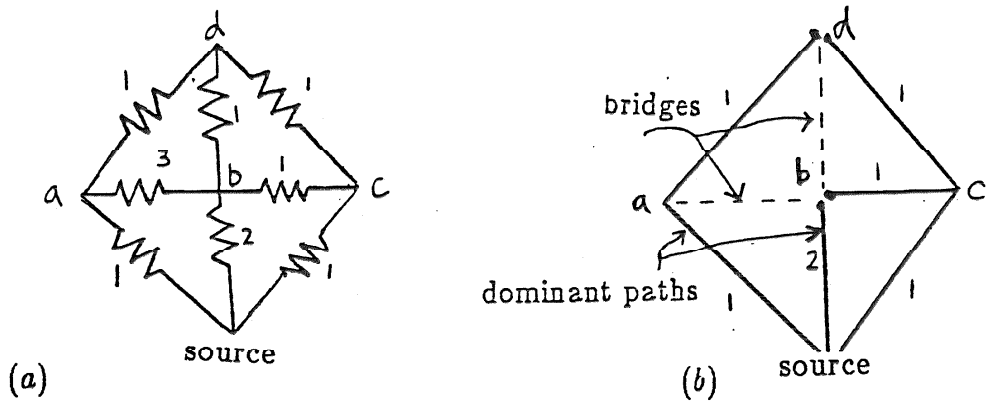


Figure 9. Illustration of the dominant path decomposition scheme

between one node in B_{VDD} and another node in B_{GND} is considered as if it were not present. Such resistors are called the bridges between B_{VDD} and B_{GND}. The above scheme can be generalized to decompose a block even if the nodes are driven by the same source. That is to approximate the load distribution such that a node only loads those nodes that are along its dominant paths, and has no effect on the nodes along other paths, whether they lead to the same source or not. This scheme, referred to as the dominant path (DP) scheme, decomposes a network into a collection of trees since a node A cannot be in the dominant path of another node B if node B is in the dominant path of A. If a node has more than one dominant path, then the load is equally distributed among these paths.

Example 25. Consider the network of Fig.9.a. which consists of four nodes: a, b, c and d. There is only one source driving the network. The dominant paths of these nodes decompose the network into three trees, which are indicated in Fig.9.b. Branches $b-d$ and $a-b$ do not belong to any of these trees, and are called the bridges among these trees. The delays of various nodes are calculated independently for each tree. There are two dominant paths incident on node b, so the node capacitance is split into two equal parts. The delay of this node is approximated by the average of the delays evaluated from the two dominant paths. The same for node d. If the difference between the delays of every pair of nodes that are connected by a bridge is within certain error bound, then no relaxation process is necessary. Otherwise, every bridge needs to be bound to a tree to start the relaxation process. Note that the load redistribution technique only applies to a collection of trees that are driven by the same source. Two trees that are driven by different sources are not allowed to be connected by bridges. ∎

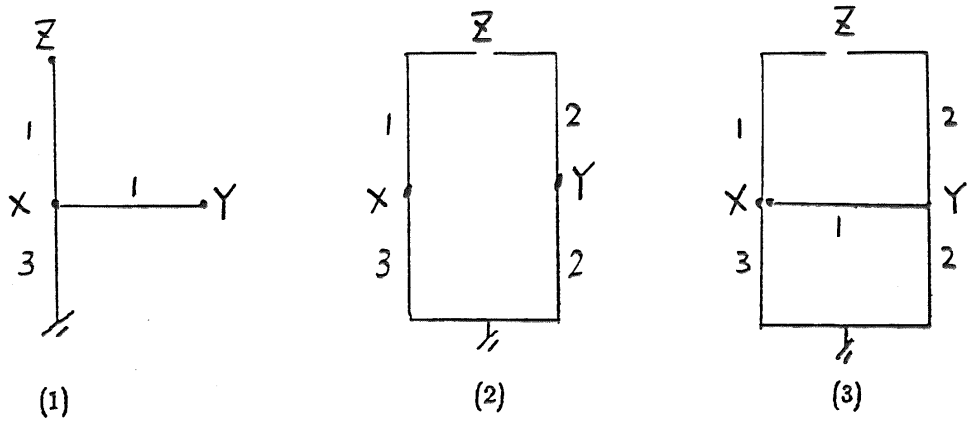


Figure 10. The DP decomposition in each case of example 23

As an aid to detect conflict conditions in a network, the source resistance of a node to VDD (the smallest resistance among the paths that lead to VDD) is compared with that to GND. A conflict condition is detected when the difference between the two resistances is smaller than a threshold value.

Example 26. The NMOS circuit presented in example 23 of the last section is considered again. The dominant paths of the nodes in each of the three cases are shown in Fig.10. The delays evaluated by the DP scheme (T') and the exact values (T) are listed in Table 3.

case	1	2			3		
	T_x, T_y, T_z	T_x	T_y	T_z	T_x	T_y	T_z
T'	–	9	6	10.5	9	7.5	11.25
T	–	8.625	6.25	10.5	7.69	6.87	10.09
$\frac{T'-T}{T}$	0%	4.2%	−4.2%	0%	−2.5%	31%	11%

Table 3. Comparison of the exact delays and the delays calculated by the DP scheme

In case 1, the network is a tree, so the delays evaluated by the DP scheme are all exact. In case 2, both paths incident on node Z are dominant, and the delay calculated is also exact. Independent of the size of the network, the relative errors of case 3 and case 2 are typical for nontree networks with and without bridge connections, respectively. Such error percentages are quite acceptable for most practical applications. ∎

To a node, there are two kinds of nondominant paths: one kind leads to a source (nondominant driving paths), and the other kind leads to an open end (loading paths). A pair of nondominant paths may also result in an phantom path, i.e. a loop with no

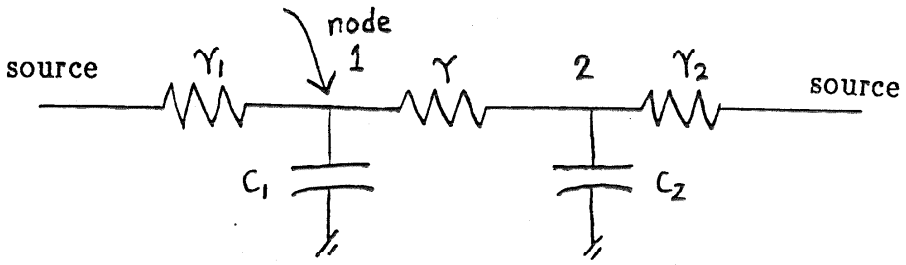
side branch that connects to a source. Such paths are no different from those which lead to an open end. Note that all dominant paths must lead to the source. The presence of nondominant driving paths causes an error in evaluating delays, while that of loading paths does not. On the other hand, the following two conditions are sufficient for the exact calculation of the delay of a node N using the DP scheme:

a. All the driving paths of node N are dominant.

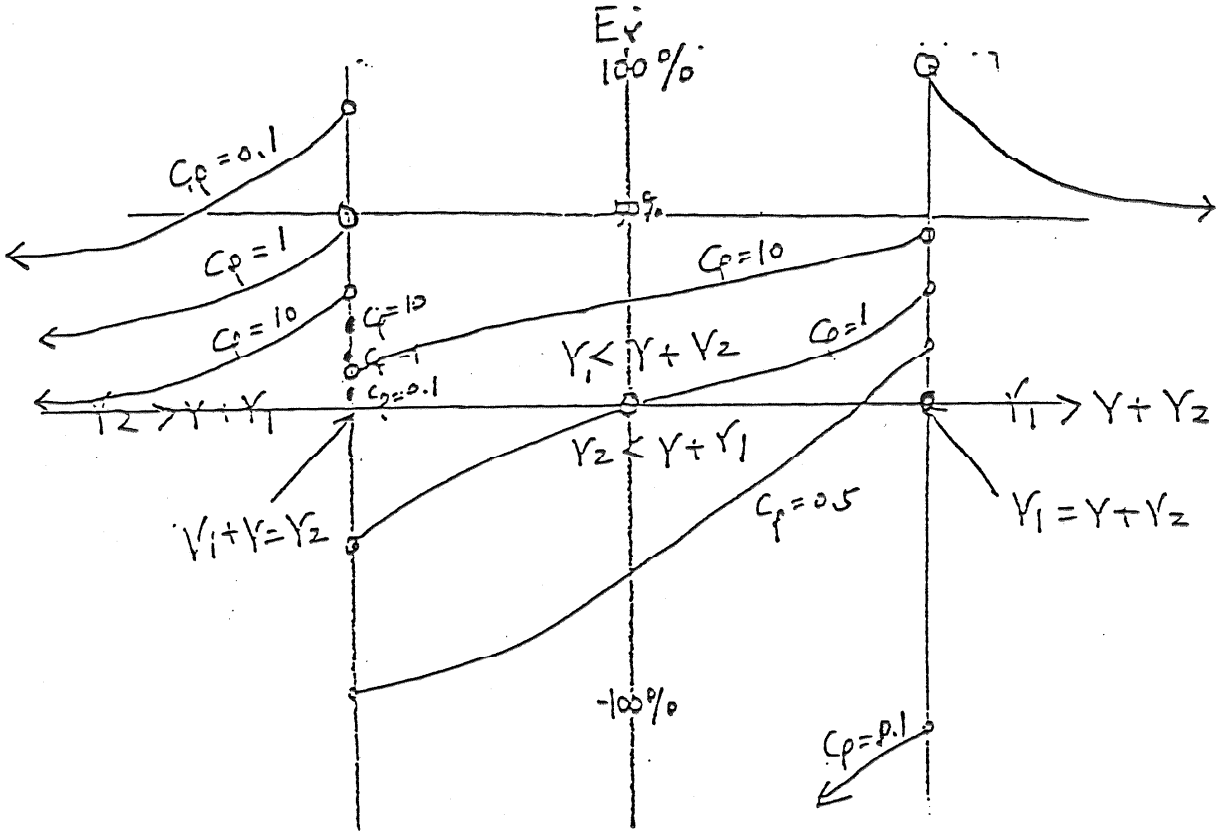
b. For those nodes that are along a driving path of N, all their driving paths but the one that passes through N are dominant.

Another factor that affects the accuracy of the DP scheme is the realtive magnitude of the node capacitances. Note that only the values of resistances participate in the determination of dominant paths, not the values of capacitances. Roughly speaking, the more uniformly the capacitances are distributed in the network, the more accurate the DP scheme is. The following example illustrates the effect of the capacitance ratio on the accuracy of the DP scheme.

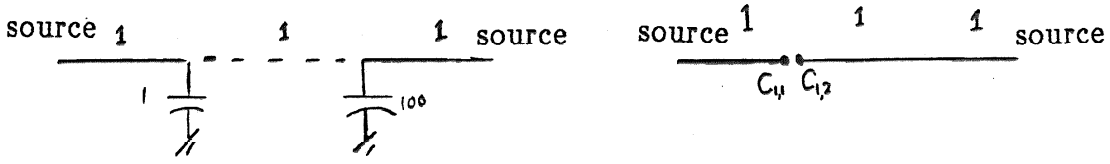
Example 27. Consider the simple circuit of Fig.11.a which is driven by a source from both ends. the exact delay of node 1 of the circuit is $T = \frac{r_1(r+r_2)}{r_1+(r+r_2)}c_1 + \frac{r_1 r_2}{r_1+(r+r_2)}c_2$. On the other hand, the delay T' estimated by the DP scheme depends on the relative magnitude of the three resistors: r, r_1, and r_2. There are five possible cases, of which the values of T' and the relative errors E_r are listed in Table 4. The term E_r is $\frac{T'-T}{T}$ if $T' > T$, and $\frac{T'-T}{T'}$ otherwise. Shown in Fig.11.b is E_r ploted as a function of $\log r_\rho$, with c_ρ as a parameter, where $r_\rho = \frac{r_1}{r_2}$, and $c_\rho = \frac{c_1}{c_2}$. The following remarks refer to Table 4 and Fig.11.b.



(a)



(b)



(c) (d)

Figure 11. the effect of capacitance ratios on the accuracy of the DP scheme

		T'	$\frac{T'-T}{T}$
1	$r_1 + r < r_2$	$r_1(c_1 + c_2)$	$\frac{r_1c_1+(r+r_1)c_2}{(r+r_2)c_1+r_2c_2}$
2	$r_2 = r_1 + r$	$r_1(c_1 + \frac{c_2}{2})$	$\frac{r_1c_1}{(r+r_2)c_1+r_2c_2}$
3	$r_1 + r > r_2$, $r_2 + r > r_1$	r_1c_1	$\frac{r_1c_1-r_2c_2}{c_1(r_1+r+r_2)}$
4	$r_1 = r + r_2$	$\frac{r_1c_1+r_2c_2}{2}$	0
5	$r_1 > r + r_2$	$(r + r_2)c_1 + r_2c_2$	$\frac{r+r_2}{r_1}$

Table 4. T' and E_r for different ratios of r_1, r_2, and r

- If $r_2 > r_1 + r$, then the approximation is made as if the resistor r_2 were not present. As a result, T' is larger than T. E_r increases as r_ρ gets larger, or c_ρ gets smaller.

- If $r_1 > r + r_2$, then the approximation is made as if r_1 were not present. In this case, E_r is independent of the value of c_ρ. With r_1 and r_2 interchanged, this error is always larger than that in the previous case. As $c_\rho \to 0$, the two errors become identical, and the two curves become symmetric.

- If $r_1 < r + r_2$, and $r_2 < r + r_1$, then the approximation is made as if the resistor r did not exist. Node 1 is fully loaded (overloaded) by c_1, and is not loaded at all (underloaded) by c_2. Although these two errors are of opposite sign, they do not always cancel out exactly. As the c_ρ varies, the error E_r changes tremendously. As two extremes, $E_r \to 50\%$ as $c_\rho \to \infty$, and $E_r \to -\infty$ as $c_\rho \to 0$. Fortunately in a practical circuit, such highly asymmetric situations do not occur very often.

- If $r_1 = r + r_2$, then $E_r = 0$; if $r_2 = r + r_1$, then $E_r = \frac{c_\rho}{5c_\rho+3}$. Compared with the errors of the other cases, these two values are considerably smaller. Moreover, a small perturbation of the values of r's at these points will result in a drastic increase of the error. In fact, the smaller the perturbation, the larger the increase of the error. This phenomenon, due to the discontinuous (with respect to r) approximation of load distribution, is common to various networks. One possible modification of the DP scheme is to consider paths with slightly larger resistance to be dominant as well.

▸ Take $r_1 = r = r_2 = 1$, $c_1 = 1$, and $c_2 = 100$ to show how algorithm RELAX corrects the errors caused by the DP scheme. The exact delays of nodes 1 and 2 are: $T_1 = 34$ and $T_2 = 67$, respectively. The dominant paths of these two nodes are shown in Fig.11.c. and, under the DP scheme, their delays are $T'_1 = 1$ and $T'_2 = 100$, respectively. To start the relaxation process, bridge $1-2$ is linked to either node 1 or node 2. If it is linked to node 2, then the network is decomposed as indicated in Fig.11.d and node 1 is split into two parts. At the first relaxation step, $T_1^{(1)} = \frac{T'_1 + \frac{T'_2}{(r+r_2)}}{\frac{1}{r_1} + \frac{1}{r+r_2}} = 34$, $\Delta_{C_{1,2}} = \frac{T_1^{(1)} - T'_2}{r+r_2} = -33$, and $T_2^{(1)} = T'_2 + r_2 \cdot \Delta_{C_{1,2}} = 67$. In this example, one relaxation step is enough to correct the values of the delays. ∎

Transistor Networks

The evolution of a transistor network is represented by a sequence of logic events, sorted by time. Each event corresponds to a change of the logic state of a node, from 1 to 0, or vice versa. At the time when an event is scheduled, say to change node A from L to $\overline{L}$, the logic state of node A is set to X first, indicating it is in transition. Not until the event is activated after certain delay does the state of A switches to $\overline{L}$. The activation of this event will affect other nodes in the network through the transistors whose gates are controled by node A. Consider the circuit of Fig.12.a. Initially, transistor M is turned off, and networks $N1$ and $N2$ are independent of each other. Let R_D and R_S denote the source resistances of the two lateral nodes D and S (drain and source) of transistor M, respectively. At time T_c, an event is invoked to switch the gate node of M and turn the transistor on. As a result, the series resistance of path $P1$ of node D changes from ∞ to $R_S + R_M$, where R_M is the ON-resistances of M. Likewise, the series resistance of path $P2$ becomes $R_D + R_M$. One and only one of the following five conditions is satisfied.

a. $R_S + R_M > R_D$ and $R_D + R_M > R_S$: $P1$ and $P2$ are both nondominant paths of node D and S, respectively. Under the DP scheme, neither the logic state nor the delay value of any node of the network is changed.

b. $R_S + R_M < R_D$: $P1$ becomes the dominant path of node D. Take the network of

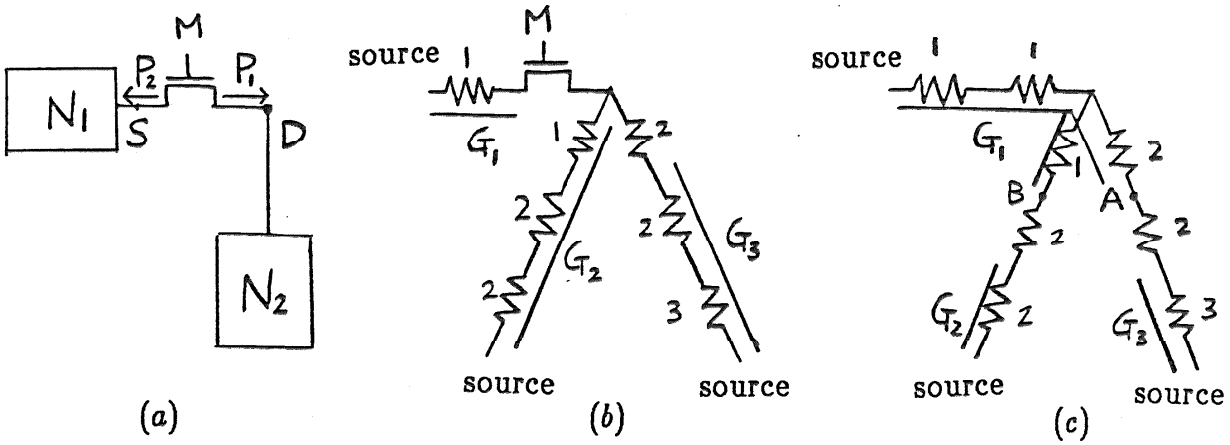


Figure 12. The status of the network before and after transistor M is turned on

Fig.12.b as an example. The dominant paths of the nodes before and after transistor M switches are indicated in Fig.12.b and Fig.12.c, respectively. With M turned off, node A belongs to tree G_1, so its logic state equals, or is scheduled to be, the state of the source driving G_1. As M is turned on, A becomes a part of tree G_3, and its logic state will be driven towards the source of G_3 which may be the same or different from that of G_1. The above argument also applies to node B. Trees G_1, G_2 and G_3 are perturbed by transistor M turning on. The logic states of the other nodes in these perturbed trees do not change, however, their delay values may be affected. The most primitive approach to updating the delays is to evaluate the stored charge of each node of these trees and use algorithm TREE to recalculate the delays. If there is no event scheduled for a node, then the stored charge of the node equals 0 or the node capacitance, depending on whether its logic state is the same as or opposite to that of the source driving it. The simplest way to estimate the stored charge of a node when an event is being scheduled is by way of linear interpolation: $Q = Q_0 + \frac{T_1+T_0-T_2}{T_1}(C - Q_0)$, where Q_0 and T_1 are respectively the charge and delay calculated at time T_0 when the event is scheduled. More accurate estimation by using an exponential function is also possible, but unwarranted considering the level of approximation being used.

The other three cases ($R_D + R_M < R_S$, $R_S + R_M = R_D$, and $R_D + R_M = R_S$) can be dealt with in a similar way.

It was pointed out, in section 3, that the definition of delay (1) is equal to Elmore's

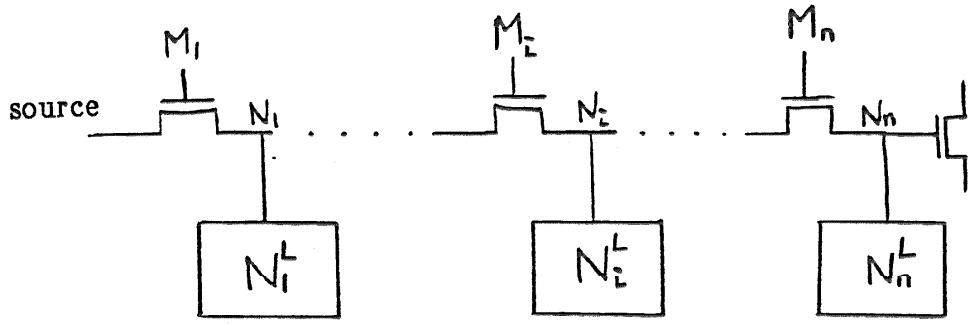


Figure 13. A chain of transistors in a tree network

delay in the case of zero initial charge. In what follows, the consistency between these two definitions is discussed for the cases of nonzero initial charge. This discussion also suggests another simulation algorithm that is very efficient and gives the exact delay value for the end node of a chain of transistors. Consider the chain of transistor in the tree network shown in Fig.13. Initially, all the transistors in the chain are tuned off, and all the transistors in the side branches are turned on. All internal nodes are without initial charge. Compare the following two cases:

1. All transistors in the chain are turned on at the same time. This is a case in which Elmore's definition can be applied.

2. The transistors in the chain are turned on one after another, starting from M_1, then $M_2, \cdots, M_n$, successively. M_i is not turn on until the nodes $N_{1,\cdots,i-1}$ all settle down. Because of nonzero initial charge, this is a case where Elmore's definition cannot be applied.

In case 1, T_D, the delay for the end node N_n, equals $\sum_{i=1}^{n}(r_i \sum_{k=i}^{n} C_k^L)$, where r_i is the ON-resistance of transistor T_i, and C_k^L is the total load capacitance at node N_k, including side branches. In case 2, there are n time intervals to be considered. The ith time interval starts when transistor M_i is turned on, and ends when the nodes $N_{1,\cdots,i}$ all settle down. T_i, the length of the ith time interval, is equal to $(\sum_{k=1}^{i} r_k)C_i^L$. Note that the stored charge in the nodes $N_{1,\cdots,i-1}$ have been taken into consideration. It is easy to check that $\sum_{i=1}^{n} T_i$ is

equal to the T_D in case 1. ∎

In the case of nonzero initial conditions, T_D is still consistent with the Elmore's delay, as the above example indicates. In fact, $\sum_{i=1}^{n} T_i = T_D$ even if transistor M_i is turned on before the nodes $N_{1,\ldots,i}$ settle down. As theorem 10 shows, the value of delay only depends upon the amount of charge yet to be supplied for each node. Regardless how the charge is actually supplied, the overall delay should always be the same. Another thing to be noted is that, when the network topology changes, a nonzero delay may be associated with a node which has been settled previously. This delay corresponds to the settling of the glitches produced by the dynamic charge sharing effect. Recall that T_D is equal to the area between 1 and the response curve. The larger value of T_D always implies a bigger glitch. In practical circuits, small glitches do not tend to produce transitions at the next stages. We set a threshold value and ignore all glitches that are smaller than this value. This filtering action prevents circuit events from over-propagation, and makes our algorithm more efficient. On the other hand, the occurrence of a sizable glitch is very useful information for the designer, and our algorithm is capable of detecting these glitches without any extra cost.

The result of the above example suggests a modification of the DP simulation scheme. That is to consider a node not directly driven by the source, but by one of its neighboring nodes through a dominant path. Take case 1 of the above example. Instead of driving all nodes $N_{1,\ldots,n}$ at the same time, the source only drives N_1 because only N_1 is adjacent to the source. After delay time $T_1 = r_1 c_1$, node N_1 will change its logic state, and be able to drive next node N_2, which takes time $T_2 = (r_1 + r_2)c_2 = R_2 c_2$, where $R_2 = r_1 + r_2$ is the source resistance of N_2. Any other nodes that are also adjacent to N_1 will be driven by it, as well. However, each node is driven independently (an additional approximation), and the delay is not affected by the presence of other nodes. Note that c_i is the node capacitance, as opposed to C_i^L which is the total load capacitance of node i. In general, node N_i is driven by node N_{i-1}, which takes time $T_i = R_i c_i = (r_1 + \cdots + r_i)c_i$.

The advantages of this modified scheme over the original one are as follows:

- Among the three parameters R, C and D for evaluating delay values, only R (source resistance) needs to be calculated. The determination of the other two parameters does not require any computation at all. Under this modified scheme, the total load capacitance is replaced by the node capacitance for parameter C. On the other hand, parameter D is implicit in the overall delay before the activation of the event corresponding to its driving node. Note that the values of resistances are also essential for the determination of logic levels, and this modified scheme is almost as efficient as a pure logic simulator.
- Under the original scheme, all nodes $N_{1,\ldots,n}$ are inserted into the event queue at the same time. Under the modified scheme, however, each node will not be inserted until the event corresponding to its driving node is evoked and removed. The average length of the queue is in general much shorter for the modified scheme.

The disadvantages of the modified scheme are as follows:

- This modified scheme only calculates correct delay for a node that is at the open end of a chain of transistors. An error occurs if some intermediate node is also of concern, or the network is more complicated than just a chain.

In summary, the modified scheme is more efficient than the original one, however, it is less accurate. Depending on individual applications, this simulator can be easily tuned to fulfill the requirement. An experimental simulator called SDS (Signal Delay Simulator) using the above algorithms has been developed. Some simulation results are presented in the next section.

§7 Simulation Results

To calibrate the time unit of SDS, an NMOS inverter chain (Fig.14.a) with rise time to fall time ratio equal to 4 is considered. The simulation results of SDS and SPICE

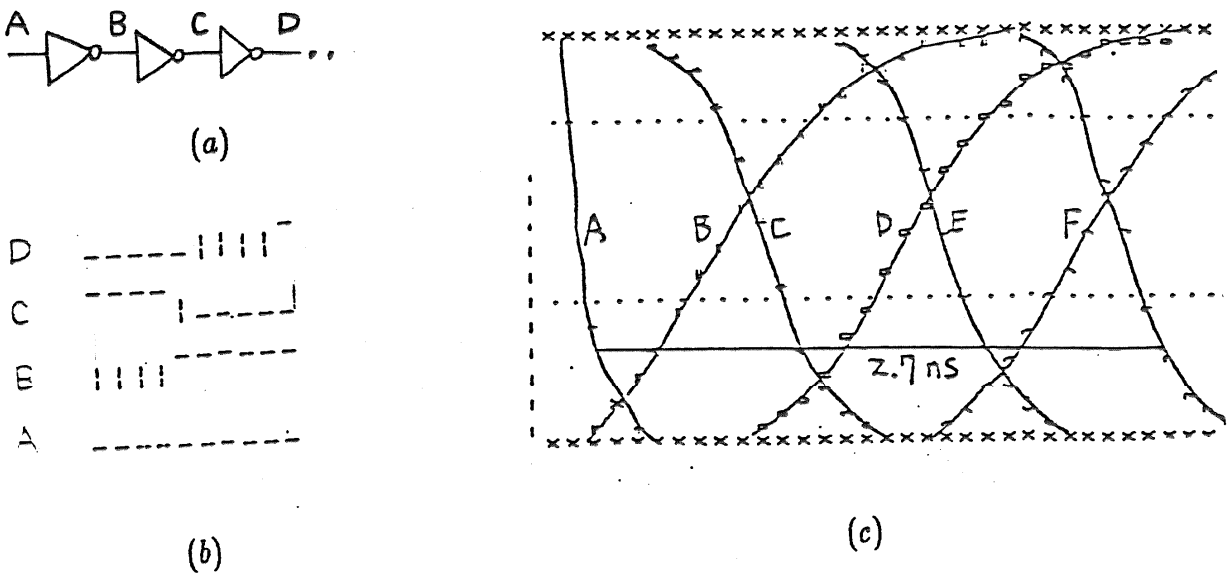


Figure 14. Calibration of time units by SPICE

are shown in Fig.14.b and c, respectively. The time taken for two consecutive inverters to switch, one up and one down, is five time units (τ) in SDS, and $\frac{2.7}{3} = 0.9ns$ in SPICE. Therefore, τ equals $\frac{0.9}{5} = 0.18ns$.

Shown in Fig.15.a. is an NMOS circuit of one-bit full adder used in a student project at Caltech. To speed up the circuit, the resistance ratio among pass-transistors, pull down transistors and pull up transistors is chosen to be 1:30:120. The simulation result generated by SPICE for the case that the three input bits change from (1,0,1) to (1,0,0) is shown in Fig.15.b. Simple as it is, this example serves as a good test case of SDS because there is a lot of feedback and multiplexed paths in the circuit. The simulation result produced by SDS is shown in Fig.15.c. Calibrated by the constant $\tau = 0.18ns$ obtained above, the time intervals of some transitions and glitches estimated by SDS are compared with the waveforms generated by SPICE (Fig.15.d). All the glitches and transitions are detected at approximatedly the correct time. Both programs run on a DEC-20 computer. SPICE takes about 40sec (CPU), and SDS takes about 0.15 sec (CPU). The difference of two to three orders of magnitude is typical for examples tested so far.

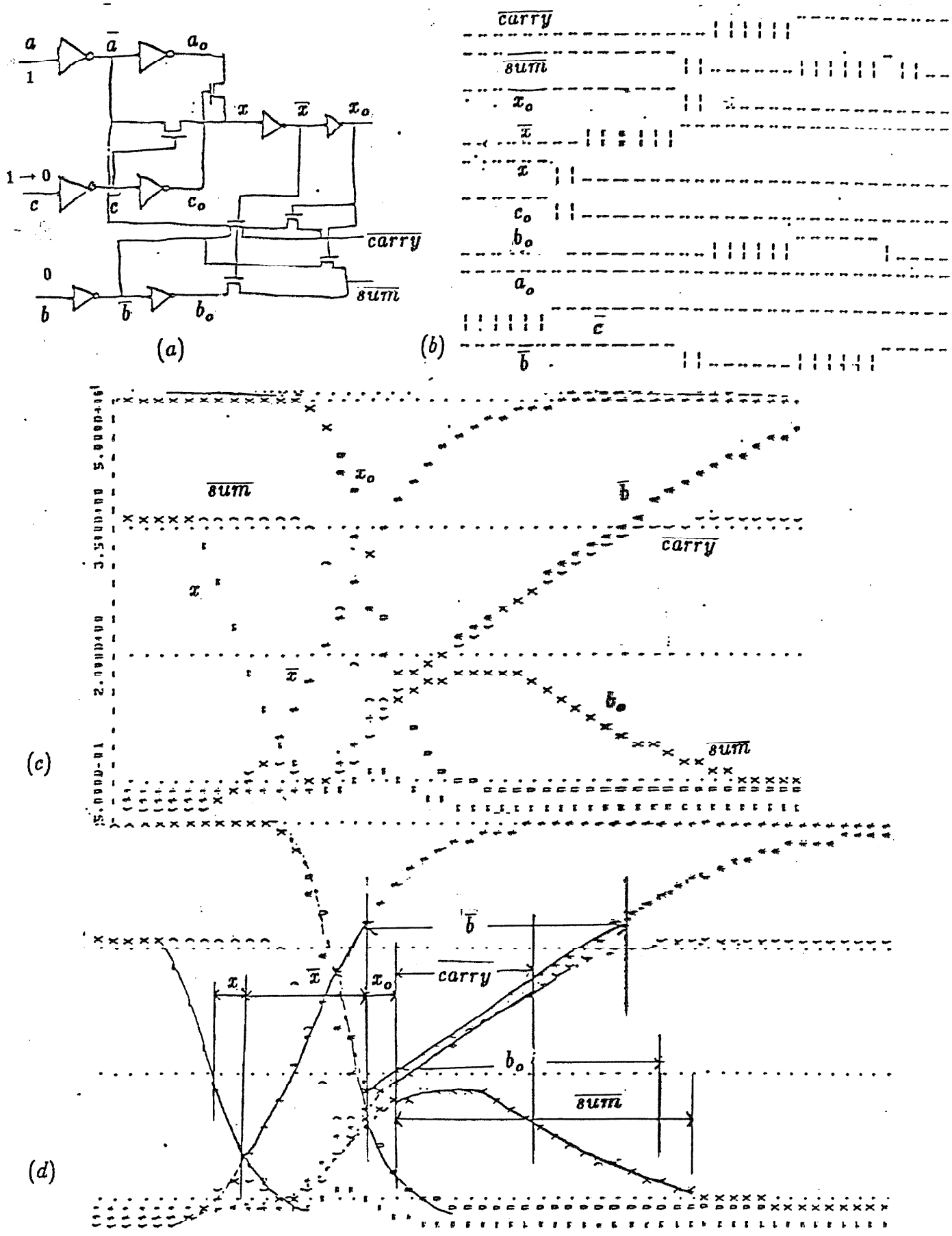


Figure 15. Comparison of a simulation result of SPICE and SDS

§8 Conclusions

The area criterion of (1′) is used throughout this paper as the definition of delay. For any RC network driven by a single source, the delay value of any node can be determined precisely. The effects of parallel connections and stored charge are properly taken into consideration. As an application, an experimental simulator called SDS was developed. For all the examples we have tested so far, this simulator runs two to three orders of magnitude faster than SPICE, and detects all the transitions and glitches at approximately the correct time. The algorithm can be used either as a stand-alone simulator, or as a front end for producing initial waveforms for waveform-relaxation based circuit simulators [23]. Further work needs to be done to include the effects of multiple-sources and static charge sharing. More efficient simulation algorithms are also being investigated.

A1 Proofs of Theorems in Section 4

Theorem 2. Up to the first order, the transfer equation (2) of a two-port RC network is of the following form:

$$\begin{pmatrix} V_o(s) \\ I_o(s) \end{pmatrix} \sim \begin{pmatrix} 1+s(RC-D) & -R+sb \\ sC & 1+sD \end{pmatrix} \begin{pmatrix} V_i(s) \\ I_i(s) \end{pmatrix} + \begin{pmatrix} -D^* + sh \\ Q+sf \end{pmatrix} \tag{4}$$

The parameters R, C, D, Q, and D^* for each of the five cases in (21) are determined as follows:

1. primitive case: $R = r$, $C = c$, $D = rc$, $Q = cv_0$, $D^* = 0$. (5)

2. series connection of N_1 and N_2: $R_T = R_1 + R_2$, $C_T = C_1 + C_2$, $D_T = D_1 + D_2 + R_1 C_2$, $Q_T = Q_1 + Q_2$, $D_T^* = D_1^* + D_2^* + R_2 Q_1$. (6)

3. parallel connection of $N_{1,\cdots,n}$: $R_T = \frac{1}{\sum_1^n \frac{1}{R_i}}$, $C_T = \sum_1^n C_i$, $D_T = R_T\left(\sum_1^n \frac{D_i}{R_i}\right)$, $Q_T = \sum_1^n Q_i$, $D_T^* = R_T\left(\sum_1^n \frac{D_i^*}{R_i}\right)$. (7)

4. N_S with side branch N_L: $R_T = R_S$, $C_T = C_L + C_S$, $D_T = D_S + R_S C_L$, $Q_T = Q_S + Q_L$, $D_T^* = D_S^*$. (8)

5. input and output ports interchanged: $R_T = R$, $C_T = C$, $D_T = RC - D$, $Q_T = Q$, $D_T^* = RQ - D^*$. (9)

Proof:

1. primitive case: From Fig.2.a, v_1, v_2, i_1, and i_2 are related by the following equations:

$$\begin{aligned} v_1(t) - v_2(t) &= r i_1(t) \\ i_1(t) - i_2(t) &= c\frac{d}{dt} v_2(t) \end{aligned} \tag{22}$$

Expressed in the matrix form in the Laplace domain, (22) becomes

$$\begin{pmatrix} V_2(s) \\ I_2(s) \end{pmatrix} = \begin{pmatrix} 1 & -r \\ -sc & 1+src \end{pmatrix} \begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} + \begin{pmatrix} 0 \\ cv_0 \end{pmatrix}$$

This is of the form (4) with parameters given in (5).

2. series connection: Assume that

$$\begin{pmatrix} V_2(s) \\ I_2(s) \end{pmatrix} \sim \begin{pmatrix} 1+s(R_1C_1-D_1) & -R_1+sb_1 \\ -sC_1 & 1+sD_1 \end{pmatrix} \begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} + \begin{pmatrix} -D_1^* + sh_1 \\ Q_1 + sf_1 \end{pmatrix}$$

$$\begin{pmatrix} V_3(s) \\ I_3(s) \end{pmatrix} \sim \begin{pmatrix} 1+s(R_2C_2-D_2) & -R_2+sb_2 \\ -sC_2 & 1+sD_2 \end{pmatrix} \begin{pmatrix} V_2(s) \\ I_2(s) \end{pmatrix} + \begin{pmatrix} -D_2^* + sh_2 \\ Q_2 + sf_2 \end{pmatrix}$$

then

$$\begin{aligned} \begin{pmatrix} V_3(s) \\ I_3(s) \end{pmatrix} &\sim \begin{pmatrix} 1+s(R_2C_2-D_2) & -R2+sb_2 \\ -sC_2 & 1+sD_2 \end{pmatrix} \begin{pmatrix} V_2(s) \\ I_2(s) \end{pmatrix} + \begin{pmatrix} -D_2^* + sh_2 \\ Q_2 + sf_2 \end{pmatrix} \\ &\sim \begin{pmatrix} 1+s(R_2C_2-D_2) & -R_2+sb_2 \\ -sC_2 & 1+sD_2 \end{pmatrix} \begin{pmatrix} 1+s(R_1C_1-D_1) & -R_1+sb_1 \\ -sC_1 & 1+sD_1 \end{pmatrix} \begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} \\ &\quad + \begin{pmatrix} 1+s(R_2C_2-D_2) & -R_2+sb_2 \\ -sC_2 & 1+sD_2 \end{pmatrix} \begin{pmatrix} -D_1^* + sh_1 \\ Q_1 + sf_1 \end{pmatrix} + \begin{pmatrix} -D_2^* + sh_2 \\ Q_2 + sf_2 \end{pmatrix} \\ &\sim \begin{pmatrix} 1+s(R_TC_T-D_T) & -R_T+sb_T \\ -sC_T & 1+sD_T \end{pmatrix} \begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} + \begin{pmatrix} -D_T^* + sh_T \\ Q_T + sf_T \end{pmatrix} \end{aligned}$$

This is of the form (4). The corresponding parameters may be easily checked against those in (6).

3. parallel connection: Assume that

$$\begin{pmatrix} V_2(s) \\ I_{2,i}(s) \end{pmatrix} \sim \begin{pmatrix} 1+s(R_iC_i-D_i) & -R_i+sb_i \\ -sC_i & 1+sD_i \end{pmatrix} \begin{pmatrix} V_1(s) \\ I_{1,i}(s) \end{pmatrix} + \begin{pmatrix} -D_i^* + sh_i \\ Q_i + sf_i \end{pmatrix}$$

This equation may be transformed into the following G-matrix form [14]:

$$\begin{pmatrix} I_{1,i}(s) \\ I_{2,i}(s) \end{pmatrix} \sim \begin{pmatrix} -\frac{1+s(R_iC_i-D_i)}{-R_i+sb_i} & \frac{1}{-R_i+sb_i} \\ -sC_i - \frac{(1+sD_i)(1+s(R_iC_i-D_i))}{-R_i+sb_i} & \frac{1+sD_i}{-R_i+sb_i} \end{pmatrix} \begin{pmatrix} V_1(s) \\ V_2(s) \end{pmatrix} + \begin{pmatrix} -\frac{-D_i^*+sh_i}{-R_i+sb_i} \\ -\frac{(1+sD_i)(-D_i^*+sh_i)}{-R_i+sb_i} + Q_i + sf_i \end{pmatrix}$$

$$\sim \begin{pmatrix} \frac{1}{R_i} + s\left(C_i - \frac{D_i}{R_i} + \frac{b_i}{R_i^2}\right) & -\left(\frac{1}{R_i} + s\frac{b_i}{R_i^2}\right) \\ \frac{1}{R_i} + s\frac{b_i}{R_i^2} & -\left(\frac{1}{R_i} + s\left(\frac{D_i}{R_i} + \frac{b_i}{R_i^2}\right)\right) \end{pmatrix} \begin{pmatrix} V_1(s) \\ V_2(s) \end{pmatrix} + \begin{pmatrix} -\frac{D_i^*}{R_i} + sm_i \\ -\frac{D_i^*}{R_i} + Q_i + sl_i \end{pmatrix}$$

where $m_i = \left(\frac{h_i}{R_i} - \frac{b_iD_i^*}{R_i^2}\right)$, and $l_i = \frac{-D_iD_i^*}{R_i} + \frac{b_iD_i^*}{R_i^2} + \frac{h_i}{R_i} + f_i$. Since $I_1(s) = \sum_1^n I_{1,i}(s)$ and $I_2(s) = \sum_1^n I_{2,i}(s)$,

$$\begin{pmatrix} I_1(s) \\ I_2(s) \end{pmatrix} \sim \begin{pmatrix} \sum_1^n\left(\frac{1}{R_i} + s\left(C_i - \frac{D_i}{R_i} + \frac{b_i}{R_i^2}\right)\right) & -\sum_1^n\left(\frac{1}{R_i} + s\frac{b_i}{R_i^2}\right) \\ \sum_1^n\left(\frac{1}{R_i} + s\frac{b_i}{R_i^2}\right) & -\sum_1^n\left(\frac{1}{R_i} + s\left(\frac{D_i}{R_i} + \frac{b_i}{R_i^2}\right)\right) \end{pmatrix} \begin{pmatrix} V_1(s) \\ V_2(s) \end{pmatrix} + \begin{pmatrix} \sum_1^n\left(-\frac{D_i^*}{R_i} + sm_i\right) \\ \sum_1^n\left(-\frac{D_i^*}{R_i} + Q_i + sl_i\right) \end{pmatrix}$$

Transforming from the G-matrix formulation back to the T-matrix formulation,

$$\begin{pmatrix} V_2(s) \\ I_2(s) \end{pmatrix} \sim \begin{pmatrix} \frac{\sum_1^n\left(\frac{1}{R_i} + s\left(C_i - \frac{D_i}{R_i} + \frac{b_i}{R_i^2}\right)\right)}{\sum_1^n\left(\frac{1}{R_i} + s\frac{b_i}{R_i^2}\right)} & -\frac{1}{\sum_1^n\left(\frac{1}{R_i} + s\frac{b_i}{R_i^2}\right)} \\ \sum_1^n\left(\frac{1}{R_i} + s\frac{b_i}{R_i^2}\right) - \frac{\sum_1^n\left(\frac{1}{R_i} + s\left(\frac{D_i}{R_i} + \frac{b_i}{R_i^2}\right)\right)\sum_1^n\left(\frac{1}{R_i} + s\left(C_i - \frac{D_i}{R_i} + \frac{b_i}{R_i^2}\right)\right)}{\sum_1^n\left(\frac{1}{R_i} + s\frac{b_i}{R_i^2}\right)} & \frac{\sum_1^n\left(\frac{1}{R_i} + s\left(\frac{D_i}{R_i} + \frac{b_i}{R_i^2}\right)\right)}{\sum_1^n\left(\frac{1}{R_i} + s\frac{b_i}{R_i^2}\right)} \end{pmatrix} \begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix}$$

$$+ \begin{pmatrix} \frac{\sum_1^n\left(-\frac{D_i^*}{R_i} + sm_i\right)}{\sum_1^n\left(\frac{1}{R_i} + s\frac{b_i}{R_i^2}\right)} \\ -\frac{\sum_1^n\left(\frac{1}{R_i} + s\left(\frac{D_i}{R_i} + \frac{b_i}{R_i^2}\right)\right)\sum_1^n\left(-\frac{D_i^*}{R_i} + sm_i\right)}{\sum_1^n\left(\frac{1}{R_i} + s\frac{b_i}{R_i^2}\right)} + \sum_1^n\left(-\frac{D_i^*}{R_i} + Q_i + sl_i\right) \end{pmatrix}$$

$$\sim \begin{pmatrix} 1 + s\left(\frac{\sum_1^n C_i}{\sum_1^n \frac{1}{R_i}} - \frac{\sum_1^n \frac{D_i}{R_i}}{\sum_1^n \frac{1}{R_i}}\right) & -\frac{1}{\sum_1^n \frac{1}{R_i}} + s\frac{\sum_1^n \frac{b_i}{R_i^2}}{\left(\sum_1^n \frac{1}{R_i}\right)^2} \\ -s\sum_1^n C_i & 1 + s\frac{\sum_1^n \frac{D_i}{R_i}}{\sum_1^n \frac{1}{R_i}} \end{pmatrix} \begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} + \begin{pmatrix} -\frac{\sum_1^n \frac{D_i^*}{R_i}}{\sum_1^n \frac{1}{R_i}} + sh_T \\ \sum_1^n Q_i + sf_T \end{pmatrix}$$

This is of the form (4) with parameters given in (7).

4. with open side branch N_L: Assume that

$$\begin{pmatrix} V_1(s) \\ I_1(s) - I_L(s) \end{pmatrix} \sim \begin{pmatrix} 1 + s(R_S C_S - D_S) & -R_S + sb_S \\ -sC_S & 1 + sD_S \end{pmatrix} \begin{pmatrix} V_S(s) \\ I_S(s) \end{pmatrix} + \begin{pmatrix} -D_S^* + sh_s \\ Q_S + sf_s \end{pmatrix}$$

$$\begin{pmatrix} V_L(s) \\ 0 \end{pmatrix} \sim \begin{pmatrix} 1 + s(R_L C_L - D_L) & -R_L + sb_L \\ -sC_L & 1 + sD_L \end{pmatrix} \begin{pmatrix} V_1(s) \\ I_L(s) \end{pmatrix} + \begin{pmatrix} -D_L^* + sh_L \\ Q_L + sf_L \end{pmatrix}$$

After a few steps of simplification,

$$\begin{pmatrix} V_1(s) \\ I_1(s) \end{pmatrix} \sim \begin{pmatrix} 1 + s(R_T C_T - D_T) & -R_T + sb_T \\ -sC_T & 1 + sD_T \end{pmatrix} \begin{pmatrix} V_S(s) \\ I_S(s) \end{pmatrix} + \begin{pmatrix} -D_T^* + sh_T \\ Q_T + sf_T \end{pmatrix}$$

with the set of parameters shown in (8).

5. input and output ports interchanged:

$$\begin{pmatrix} V_o(s) \\ I_o(s) \end{pmatrix} \sim \begin{pmatrix} 1 + s(RC - D) & -R + sb \\ -sC & 1 + sD \end{pmatrix} \begin{pmatrix} V_i(s) \\ I_i(s) \end{pmatrix} + \begin{pmatrix} -D^* + sh \\ Q + sf \end{pmatrix}$$

$$\begin{pmatrix} V_i(s) \\ I_i(s) \end{pmatrix} \sim \begin{pmatrix} 1 + s(RC - D) & -R + sb \\ -sC & 1 + sD \end{pmatrix}^{-1} \begin{pmatrix} V_o(s) + D^* - se \\ I_o(s) - Q - sf \end{pmatrix}$$
$$\sim \begin{pmatrix} 1 + sD & R - sb \\ sC & 1 + s(RC - D) \end{pmatrix} \begin{pmatrix} V_o(s) \\ I_o(s) \end{pmatrix} + \begin{pmatrix} D^* - RQ + sh_T \\ -Q - sf_T \end{pmatrix}$$

Finally,

$$\begin{pmatrix} V_i(s) \\ -I_i(s) \end{pmatrix} \sim \begin{pmatrix} 1 + sD & -R + sb \\ -sC & 1 + s(RC - D) \end{pmatrix} \begin{pmatrix} V_o(s) \\ -I_o(s) \end{pmatrix} + \begin{pmatrix} -(RQ - D^*) + sh_T \\ Q + sf_T \end{pmatrix}$$

∎

Theorem 5. If a node connects to the source through a network N_S with parameters R_S, C_S, D_S, Q_S, and D_S^*, and is loaded by another network N_L with parameters R_L, C_L, D_L, Q_L, and D_L^*, then the delay T_D of this node is $(D_S + D_S^* - R_S Q_S) + R_S(C_L - Q_L)$

Proof:

From the hypothesis and from theorem 2,

$$\begin{pmatrix} V_L(s) \\ 0 \end{pmatrix} \sim \begin{pmatrix} 1+s(R_LC_L-D_L) & -R_L+sb_L \\ -sC_L & 1+sD_L \end{pmatrix} \begin{pmatrix} V(s) \\ I(s) \end{pmatrix} + \begin{pmatrix} -D_L^* + sh_L \\ Q_L + sf_L \end{pmatrix}$$

$$\begin{pmatrix} V(s) \\ I(s) \end{pmatrix} \sim \begin{pmatrix} 1+s(R_SC_S-D_S) & -R_S+sb_S \\ -sC_S & 1+sD_S \end{pmatrix} \begin{pmatrix} s^{-1} \\ I_S \end{pmatrix} + \begin{pmatrix} -D_S^* + sh_S \\ Q_S + sf_S \end{pmatrix}$$

After a few steps of simplification,

$$V_L(s) \sim \frac{1+s(R_TC_T - R_S(C_T-Q_S-Q_L) - (D_S^* + R_LQ_S + D_L^*))}{s(1+sD_T)}$$

where $R_T = R_S + R_L$, $C_T = C_S + C_L$, and $D_T = D_S + D_L + R_SC_L$. Reflecting back from node L to node 1,

$$\begin{aligned} \begin{pmatrix} V(s) \\ I(s) \end{pmatrix} &\sim \begin{pmatrix} 1+s(R_LC_L-D_L) & -R_L+sb_L \\ -sC_L & 1+sD_L \end{pmatrix}^{-1} \begin{pmatrix} V_L + D_L^* - sh_L \\ -Q_L - sf_L \end{pmatrix} \\ &\sim \begin{pmatrix} 1+sD_L & R_L - sb_L \\ sC_L & 1+s(R_LC_L-D_L) \end{pmatrix} \begin{pmatrix} V_L + D_L^* - sh_L \\ -Q_L - sf_L \end{pmatrix} \end{aligned}$$

Finally,

$$\begin{aligned} V(s) &\sim (1+sD_L)V_L(s) + (1+sD_L)(D_L^* - sh_L) - (R_L - sb_L)(Q_L + sf_L) \\ &\sim \frac{(1+sD_L)(1+s\Delta) + s(1+sD_T)(D_L^* - R_LQ_L)}{s(1+sD_T)} \\ &\sim \frac{1+s(D_L + \Delta + D_L^* - R_LQ_L)}{s(1+sD_T)} \end{aligned}$$

where $\Delta = R_TC_T - R_S(C_T - Q_S - Q_L) - (D_S^* + R_LQ_S + D_L^*)$. As a result, $T_D = D_T - (D_L + \Delta + D_L^* - R_LQ_L) = D_S + R_S(C_L - Q_L) + (D_S^* - R_SQ_S)$. ∎

Theorem 10. Suppose node N_1 and node N_2 are cascaded in an RC tree network. N_1 is nearer to the source and is connected to N_2 through a resistor of value r. The total capacitance and total charge of the loading network of N_2 are C_L and Q_L, respectively. If the delay of node N_1 is T_1, then the delay of node N_2 is $T_1 + r\overline{C}_L = T_1 + r(C_L - Q_L)$. ∎

Proof:

Let subscripts $_{Li}$ and $_{Si}$ denote the loading and the driving networks with respect to the node $N_i, i = 1, 2$, respectively.

From theorem 2,

$$\begin{aligned}\overline{D}_{S2} &= \overline{D}_{S1} + \overline{R}_{S2}\overline{C}_2 \\ \overline{R}_{S2} &= \overline{R}_{S1} + r \\ \overline{C}_{L2} &= \overline{C}_{L1} - \overline{C}_2\end{aligned}$$

From Theorem 5′,

$$\begin{aligned}T_1 &= \overline{D}_{S1} + \overline{R}_{S1}\overline{C}_{L1} \\ T_2 &= \overline{D}_{S2} + \overline{R}_{S2}\overline{C}_{L2}\end{aligned}$$

Thus, $T_2 - T_1 = (\overline{D}_{S2} - \overline{D}_{S1}) + (\overline{R}_{S2}\overline{C}_{L2} - \overline{R}_{S1}\overline{C}_{L1}) = r\overline{C}_{L1} = r(C_{L1} - Q_{L1})$. ∎

A2 Proofs of Theorems in Section 5

The following lemmas are used in the proofs of theorems 12 and 13.

Lemma A1. Suppose that A and D are two real symmetric $n \times n$ matrices, and X is an $n \times n$ nonsingular matrix. If $A = XDX^T$, then A is positive-definite if and only if D is positive-definite [19].

Lemma A2. The pricipal minors of a positive-definite matrix are also positive-definite [19].

Theorem 12'. Suppose there are N nodes in a collection of tree networks. Let R be an $N \times N$ matrix with elements $R_{i,j}$ equal to the resistance of the path between the source of node i, that is in common with the path between the source and node j. Then the matrix R is symmetric and postive-definite.

Proof:

It suffices to consider a tree network since the R matrix associated with a collection of tree networks is just the direct sum of the R matrices associated with individual trees. It is easy seen that $R_{i,j} = R_{j,i}$, so matrix R is symmetric. As the network is a tree, there are same number of branches as there are nodes (the source, or the root, is not considered as a node in this case). With each node i is associated a branch $b(i)$ connecting the node to its parent node $p(i)$. This node-to-branch mapping is clearly one to one and onto. Let matrix X be defined as follows: If the path from the source to node i passes through branch $b(j)$, then $x_{i,j} = 1$; otherwise $x_{i,j} = 0$. Note that X can be obtained from $I_{N\times N}$ by a sequence of row operations $O(i)$: adding row i to all rows j such that $p(j) = i$. Starting from the source, this operation proceeds in a top down manner until the leaves of the tree are met. $O(i), \forall i$ preserves the determinant of the matrix, so $\det(X) = \det(I) = 1$, and X is nonsingular. Let D denote the diagonal matrix with diagonal element $d_{i,i} = r_{b(i)} > 0$. It is obvious that D is a positive-definite matrix. Check that $XDX^T = R$. Finally, by Lemma A1, R is positive-definite. ∎

Theorem 12. The $\sum_{i=1}^{N} b_i \times \sum_{i=1}^{N} b_i$ matrix R with elements $R_{(i,j),(u,v)} = R_{(i,j)}^{(u,v)}$ is symmetric and positive-definite.

Proof: immediate from theorem 12'. ∎

Theorem 13 The matrix A in (15) is symmetric and positive-definite.

Proof:

Let E be an $Q \times N$ matrix with elements

$$e_{(i,j),u} = \begin{cases} -1 & \text{if } i = u, \text{ for } j = 1, \cdots, b_i - 1, i = 1, \cdots, P \\ 0 & \text{otherwise} \end{cases}$$

where $Q = \sum_{i=1}^{P}(b_i - 1)$. Consider the $\sum_{i=1}^{N} b_i \times \sum_{i=1}^{N} b_i$ matrix $X = \begin{pmatrix} I_{Q\times Q} & E \\ E^T & I_{N\times N} + E^T E \end{pmatrix}$.

As $\det(X) = \det\begin{pmatrix} I & 0 \\ E^T & I \end{pmatrix} = 1$, X is nonsingular. Let $A' = XRX^T$, where R is the matrix of theorem 12. Then by lemma A1, A' is positive definite. Check that matrix A is the Qth pricipal minor of A'. By lemma *A2*, A is positive-definite. ∎

The following lemmas are used in the proof of theorem 17.

Lemma A3. Let $A_{i,i}$ be the ith diagonal block of the matrix A in (15), If $R_{(i,j)}^{(i,v)} = 0$ for $j \neq v$, $j, v = 1, \cdots, b_i$, then the determinant of $A_{i,i}$ equals

$$\left(\Pi_{k=1}^{b_i} R_{(i,k)}\right)\left(\sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}\right) \tag{23}$$

where $R_{(i,k)}$ is the source resistance of node $N_{(i,k)}$.

Proof:

As $R_{(i,j)}^{(i,v)} = 0$ for $j \neq v$, the (j,v)-element of $A_{i,i}$, $a_{(i,j),(i,v)}$, equals $R_{(i,b_i)} + \delta_{j,v} R_{(i,j)}$. This lemma is proved by induction on the order of the matrix $A_{i,i}$: $n = b_i - 1$. For $n = 1$, $A_{i,i} = [R_{(i,1)} + R_{(i,2)}] = [R_{(i,1)} R_{(i,2)} (\frac{1}{R_{(i,1)}} + \frac{1}{R_{(i,2)}})]$

Suppose (23) is true for $n = m - 1$. Then for $n = m (= b_i - 1)$,

$$A_{i,i} = \begin{pmatrix} R_{(i,1)} + R_{(i,b_i)} & R_{(i,b_i)} & \cdots & R_{(i,b_i)} \\ R_{(i,b_i)} & R_{(i,2)} + R_{(i,b_i)} & \cdots & R_{(i,b_i)} \\ \vdots & \vdots & & \vdots \\ R_{(i,b_i)} & R_{(i,b_i)} & \cdots & R_{(i,b_i-1)} + R_{(i,b_i)} \end{pmatrix}$$
$$= \begin{pmatrix} R_{(i,1)} & R_{(i,b_i)} & \cdots & R_{(i,b_i)} \\ -R_{(i,2)} & R_{(i,2)} + R_{(i,b_i)} & \cdots & R_{(i,b_i)} \\ 0 & R_{(i,b_i)} & \cdots & R_{(i,b_i)} \\ \vdots & \vdots & & \vdots \\ 0 & R_{(i,b_i)} & \cdots & R_{(i,b_i-1)} + R_{(i,b_i)} \end{pmatrix} \tag{24}$$

Expanded along the first column, the determinant of $A_{i,i}$ equals $R_{(i,1)}\left(\Pi_{k=2}^{b_i} R_{(i,k)}\right)\left(\sum_{k=2}^{b_i} \frac{1}{R_{(i,k)}}\right) + R_{(i,2)}\left(\Pi_{k=3}^{b_i} R_{(i,k)}\right) = \left(\Pi_{k=1}^{b_i} R_{(i,k)}\right)\left(\sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}\right)$ ∎

Lemma A4. Let $A_{i,i}$ be the ith diagonal block of the matrix A in (15), and $D_{i,i}$ is the inverse of $A_{i,i}$. If $R_{(i,j)}^{(i,v)} = 0$ for $j \neq v$, $j, v = 1, \cdots, b_i$, then

$$d_{(i,j),(i,v)} = \begin{cases} \Delta_{(i,j)}\left(\sum_{k=1, k\neq j}^{b_i} \frac{1}{R_{(i,k)}}\right) & \text{if } j = v \\ \frac{-\Delta_{(i,j)}}{R_{(i,v)}} & \text{otherwise} \end{cases} \tag{25}$$

where $d_{(i,j),(i,v)}$ is the (j,v)-element of $D_{i,i}$, and $\Delta_{(i,j)} = \frac{1}{R_{(i,j)} \sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}}$.

Proof: Immediate from inverting (24) using cofactors. ∎

Theorem 17. If both conditions of (19) are satisfied, then algorithm RELAX is equivalent to the block Gauss-Seidel method of (16).

Proof:

Algorithm RELAX, plus condition 1 of (19), implies that $T_{(i,j)}^{(m+1)} = \sum_{u=1}^{i-1} \sum_{v=1}^{b_u} R_{(i,j)}^{(u,v)} C_{(u,v}^{(m-}$ $+ \sum_{u=i}^{N} \sum_{v=1}^{b_u} R_{(i,j)}^{(u,v)} C_{(u,v)}^{(m)}$, and $T_i^{(m+1)} = \frac{\sum_{k=1}^{b_i} \frac{T_{(i,k)}^{(m+1)}}{R_{(i,k)}}}{\sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}}$. Thus

$$
\begin{aligned}
C_{(i,j)}^{(m+1)} &= C_{(i,j)}^{(m)} + \Delta_{C_{(i,j)}}^{(m+1)} \\
&= C_{(i,j)}^{(m)} + \frac{T_i^{(m+1)} - T_{(i,j)}^{(m+1)}}{R_{(i,j)}} \\
&= C_{(i,j)}^{(m)} + \frac{1}{R_{(i,j)}} \left(\frac{\sum_{k=1}^{b_i} \frac{T_{(i,k)}^{(m+1)} - T_{(i,j)}^{(m+1)}}{R_{(i,k)}}}{\sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}} \right) \\
&= C_{(i,j)}^{(m)} + \Delta_{(i,j)} \sum_{k=1}^{b_i} \left(\sum_{u=1}^{i-1} \sum_{v=1}^{b_u} \frac{R_{(i,k)}^{(u,v)} - R_{(i,j)}^{(u,v)}}{R_{(i,k)}} C_{(u,v)}^{(m+1)} + \sum_{u=i}^{N} \sum_{v=1}^{b_u} \frac{R_{(i,k)}^{(u,v)} - R_{(i,j)}^{(u,v)}}{R_{(i,k)}} C_{(u,v)}^{(m)} \right) \qquad (26) \\
&= C_{(i,j)}^{(m)} + \Delta_{(i,j)} \sum_{k=1}^{b_i} \left(\sum_{u=1}^{i-1} \sum_{v=1}^{b_u - 1} \frac{R_{(i,k)}^{(u,v)} - R_{(i,j)}^{(u,v)} - R_{(i,k)}^{(u,b_u)} + R_{(i,j)}^{(u,b_u)}}{R_{(i,k)}} C_{(u,v)}^{(m+1)} + \right. \\
&\quad \left. \sum_{u=i}^{P} \sum_{v=1}^{b_u - 1} \frac{R_{(i,k)}^{(u,v)} - R_{(i,j)}^{(u,v)} - R_{(i,k)}^{(u,b_u)} + R_{(i,j)}^{(u,b_u)}}{R_{(i,k)}} C_{(u,v)}^{(m)} + \sum_{u=1}^{N} \frac{R_{(i,k)}^{(u,b_u)} - R_{(i,j)}^{(u,b_u)}}{R_{(i,k)}} C_u \right)
\end{aligned}
$$

where $\Delta_{(i,j)} = \frac{1}{R_{(i,j)} \sum_{k=1}^{b_i} \frac{1}{R_{(i,k)}}}$. On the other hand, the block Gauss-Seidel method of (16), plus condition 2 of (19), implies

$$
\begin{aligned}
C_{(i,j)}^{(m+1)} &= \sum_{k=1}^{b_i - 1} d_{(i,j),(i,k)} \left(b_{(i,k)} - \sum_{u=1}^{i-1} \sum_{v=1}^{b_u - 1} a_{(i,k),(u,v)} C_{(u,v)}^{(m+1)} - \sum_{u=i+1}^{P} \sum_{v=1}^{b_u - 1} a_{(i,k),(u,v)} C_{(u,v)}^{(m)} \right) \\
&= \Delta_{(i,j)} \left(\left(\sum_{k=1, k \neq j}^{b_i} \frac{1}{R_{(i,k)}} \right) \left(\sum_{u=1}^{N} \left(R_{(i,b_i)}^{(u,b_u)} - R_{(i,k)}^{(u,b_u)} \right) C_u - \sum_{u=1}^{i-1} \sum_{v=1}^{b_u - 1} a_{(i,j),(u,v)} C_{(u,v)}^{(m+1)} - \right. \right. \\
&\quad \left. \sum_{u=i+1}^{P} \sum_{v=1}^{b_u - 1} a_{(i,j),(u,v)} C_{(u,v)}^{(m)} \right) + \sum_{l=1, l \neq j}^{b_i} - \frac{\Delta_{(i,j)}}{R_{(i,l)}} \left(\sum_{u=1}^{N} \left(R_{(i,b_i)}^{(u,b_u)} - R_{(i,l)}^{(u,b_u)} \right) C_u - \right. \qquad (27) \\
&\quad \left. \left. \sum_{u=1}^{i-1} \sum_{v=1}^{b_u - 1} a_{(i,l),(u,v)} C_{(u,v)}^{(m+1)} - \sum_{u=i+1}^{P} \sum_{v=1}^{b_u - 1} a_{(i,l),(u,v)} C_{(u,v)}^{(m)} \right) \right)
\end{aligned}
$$

where $a_{(i,k),(u,v)}$ is the $(i,k),(u,v)$-element of the matrix A in (15). The coefficients of some terms of (26) and (27) are reduced as follows:

- Term C_u in (27), for $u = 1, \cdots, N$:

$$
\begin{aligned}
&\Delta_{(i,j)} \sum_{k=1,k\neq j}^{b_i} \sum_{u=1}^{N} \frac{R_{(i,b_i)}^{(u,b_*)} - R_{(i,j)}^{(u,b_*)}}{R_{(i,k)}} - \Delta_{(i,j)} \sum_{l=1,l\neq j}^{b_i} \sum_{u=1}^{N} \frac{R_{(i,b_i)}^{(u,b_*)} - R_{(i,l)}^{(u,b_*)}}{R_{(i,l)}} \\
&= \Delta_{(i,j)} \sum_{k=1,k\neq j}^{b_i} \sum_{u=1}^{N} \frac{R_{(i,b_i)}^{(u,b_*)} - R_{(i,j)}^{(u,b_*)} - R_{(i,b_i)}^{(u,b_*)} + R_{(i,k)}^{(u,b_*)}}{R_{(i,k)}} \\
&= \Delta_{(i,j)} \sum_{k=1,k\neq j}^{b_i} \sum_{u=1}^{N} \frac{R_{(i,k)}^{(u,b_*)} - R_{(i,j)}^{(u,b_*)}}{R_{(i,k)}} \\
&= \Delta_{(i,j)} \sum_{k=1}^{b_i} \sum_{u=1}^{N} \frac{R_{(i,k)}^{(u,b_*)} - R_{(i,j)}^{(u,b_*)}}{R_{(i,k)}}
\end{aligned}
$$

▸ Term $C_{(u,v)}$ in (27), for $v = 1, \cdots, b_u - 1$, $u = 1, \cdots, P$, and $u \neq i$:

$$
\begin{aligned}
&\Delta_{(i,j)} \sum_{k=1,k\neq j}^{b_i} -\frac{a_{(i,j),(u,v)}}{R_{(i,k)}} - \Delta_{(i,j)} \sum_{l=1,l\neq j}^{b_i} -\frac{a_{(i,l),(u,v)}}{R_{(i,l)}} \\
&= \Delta_{(i,j)} \sum_{k=1,k\neq j}^{b_i} \frac{a_{(i,k),(u,v)} - a_{(i,j),(u,v)}}{R_{(i,k)}} \\
&= \Delta_{(i,j)} \sum_{k=1}^{b_i} \frac{R_{(i,k)}^{(u,v)} - R_{(i,b_i)}^{(u,v)} - R_{(i,k)}^{(u,b_*)} + R_{(i,b_i)}^{(u,b_*)} - R_{(i,j)}^{(u,v)} + R_{(i,b_i)}^{(u,v)} + R_{(i,j)}^{(u,b_*)} - R_{(i,b_i)}^{(u,b_*)}}{R_{(i,k)}} \\
&= \Delta_{(i,j)} \sum_{k=1}^{b_i} \frac{R_{(i,k)}^{(u,v)} - R_{(i,j)}^{(u,v)} - R_{(i,k)}^{(u,b_*)} + R_{(i,j)}^{(u,b_*)}}{R_{(i,k)}}
\end{aligned}
$$

▸ Term $C_{(i,v)}$ in (26), for $v = 1, \cdots, b_i$, $v \neq j$:

$$
\begin{aligned}
&\Delta_{(i,j)} \sum_{k=1}^{b_i} \frac{R_{(i,k)}^{(i,v)} - R_{(i,j)}^{(i,v)} - R_{(i,k)}^{(i,b_i)} + R_{(i,j)}^{(i,b_i)}}{R_{(i,k)}} \\
&= \Delta_{(i,j)} \left(\sum_{k=1}^{b_i} \frac{R_{(i,k)}^{(i,v)}}{R_{(i,k)}} - \sum_{k=1}^{b_i} \frac{R_{(i,k)}^{(i,b_i)}}{R_{(i,k)}} \right) \\
&= \Delta_{(i,j)} \left(\frac{R_{(i,v)}^{(i,v)}}{R_{(i,v)}} - \frac{R_{(i,b_i)}^{(i,b_i)}}{R_{(i,b_i)}} \right) \\
&= 0
\end{aligned}
$$

▸ Term $C_{(i,j)}$ in (26):

$$
\begin{aligned}
&1+\Delta_{(i,j)}\sum_{k=1}^{b_i}\frac{R^{(i,j)}_{(i,k)}-R^{(i,j)}_{(i,j)}-R^{(i,b_i)}_{(i,k)}+R^{(i,b_i)}_{(i,j)}}{R_{(i,k)}}\\
&=1-\Delta_{(i,j)}\left(\sum_{k=1}^{b_i}\frac{R^{(i,j)}_{(i,j)}}{R_{(i,k)}}\right)+\Delta_{(i,j)}\left(\sum_{k=1}^{b_i}\frac{R^{(i,j)}_{(i,k)}}{R_{(i,k)}}-\sum_{k=1}^{b_i}\frac{R^{(i,b_i)}_{(i,k)}}{R_{(i,k)}}\right)\\
&=1-1+0\\
&=0
\end{aligned}
$$

All the corresponding coefficients of (26) and (27) are eqaul, so the two methods are equivalent. ∎

Bibliography

[1] Mead, C.A. & Conway, L.A., Introduction to VLSI Systems, Chapter 1, Addison Wesley, 1980.

[2] Penfield, P. & Rubinstein, J., "Signal Delay in RC Tree Networks", Proc. of the 2nd Caltech VLSI Conference, pp.269-283, March 1981.

[3] Penfield, P., Rubinstein, J., & Horowitz, M., "Signal Delays in RC Tree Networks", IEEE Trans. on Computer Aided Design (to be published 1983).

[4] Terman, C.J., Simulation Tools for Digital LSI Design, MIT/LCS, Dec. 1981.

[5] Horowitz, M., "Timing Models for MOS Pass Networks", International Symposium on Circuits and Systems (ISCAS), pp.198-201, 1983.

[6] Hitchcook, R.B., "Timing Verification and the Timing Analysis Program", Proc. of the 19th Design Automation Conference, pp.594-604, 1982.

[7] Ousterhout, J.K., "Crystal: A Timing Analyzer for nMOS VLSI Circuits", Proc. of the 3rd Caltech VLSI Conference, pp.57-70, March 1983.

[8] Jouppi, N.P., "TV: An nMOS Timing Analyzer", Proc. of the 3rd Caltech VLSI Conference , pp.71-86, March 1983.

[9] Bryant, R.E., A Switch-level Simulation Model for Integrated Logic Circuits, Doctoral Dissertation, MIT/LCS/TR-259, MIT, March 1981.

[10] Elmore, W.C., "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers", Journal of Applied Physics, vol.19, No.1, pp.55-63, Jan. 1948.

[11] Glasser, L.A., "The Analogy Behavior of Digital Integrated Circuits", Proc. of the 18th Design Automation Conference, pp.603-612, June 1981.

[12] Nagel, L.W., SPICE2: A Computer Program to Simulate Semiconductor Circuits

, ERL Memo, No. ERL-M520, Electronics Research Laboratory, University of California, Berkeley, 1975.

[13] Chiang, C., Distributed RC Delay Line Model and MOS PLA Timing Estimation , Technical Report, Computer Science, Caltech, in preparation.

[14] Desoer, C.A. & Kuh, E.S., Basic Circuit Theory , Chapter 17, McGraw-Hill, 1969.

[15] Ghausi, M.S. & Kelly, J.J., Introduction to Distributed-Parameter Networks: with Application to Integrated Circuits, Chapter 1, Holt, Rinehart and Winston Inc., 1968.

[16] Chen, M.C., Space-Time Algorithms: Semantics and Methodology , Doctoral Dissertation, Computer Science, Caltech, 5090:TR:83, May 1983.

[17] Varga, R.S., Matrix Iterative Analysis , Prentice-Hall series in automatic computation, 1962.

[18] Rysor, H.J., private communication.

[19] Strang, G., Linear Algebra and its Applications, Chapter 6, 2nd. ed., Academic Press, 1980.

[20] Aho, A.V., Hopcroft, J.E. & Ullman, J.D., The Design and Analysis of Computer Algorithms , Chapter 5, Addison-Wesley, 1974.

[21] Ayres, R., A Language Processor and a Sample language, Doctoral Dissertation, TR#2276, Computer Science, Caltech, June 1978.

[22] Bryant, R.E., Schuster, M. & Whiting, D., MOSSIM II: A Switch-Level Simulator for MOS LSI: User's Manual , 5033:TR:82, Computer Science, Caltech, 1982.

[23] Lelarasmee, E., Ruehli, A.E. & Sangiovanni-Vincentelli, A.L., "The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD-1, No.3, pp.131-145, July 1982.