[4] H. Fujiwara, T. Shimono, "On the acceleration of test generation algorithm," *IEEE Trans. Comput.*, vol. C-32, pp. 1137-1144, Dec. 1983.
[5] S. Patel and J. Patel, "Effectiveness of heuristics for automatic test pattern generation," in *Proc. 23rd IEEE Design Automation Conf.*, June 1986, pp. 547-552.
[6] L. H. Goldstein, "Controllability/observability analysis of digital circuit," *IEEE Trans. Circuit, Syst.*, CAS-26, pp. 685-693, Sept. 1979.
[7] A. Ivanov and V. K. Agarwal, "Dynamic testability measures for ATPG," *IEEE Trans. on Computer-Aided Design*, pp. 598-608, Dec. 1988.
[8] F. Brglez and H. Fujiwara, "A neutral netlist of the combinational benchmark circuits and a target translator in FORTRAN," presented at IEEE Int. Symp. on Circuits and Systems, Kyoto, Japan, June 5-7, 1985.

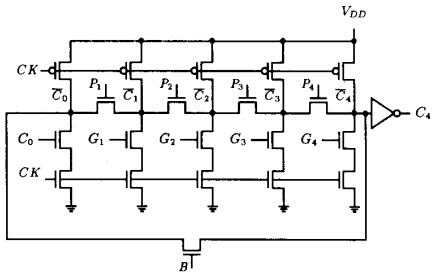

Fig. 1. Manchester adder with carry-bypass.

Computing Signal Delay in General *RC* Networks by Tree/Link Partitioning

PAK K. CHAN AND KEVIN KARPLUS

Abstract—**Most *RC* simulators handle only tree networks, not arbitrary networks. We present an algorithm for computing signal delays in general *RC* networks using the *RC*-tree computation as the primary operation. We partition a given network into a spanning tree and links. Then we compute the signal delay of the spanning tree, and update the signal delay as we incrementally add the links back to reconstruct the original network. If m is the number of links, this algorithm requires $m(m+1)/2$ updates and $m+1$ tree delay evaluations. All the tree delay evaluations involve computing signal delays with the same resistive spanning tree, but with different values for the capacitors.**

I. Introduction

The linear *RC* model has become an acceptable and pragmatic approach for modeling digital MOS circuits in the past decade. Research has been carried out both in bounding the waveforms [17], [22], [24] and in estimating the signal delays [3], [7], [13], [14], [19] of *RC* networks. In particular, Elmore's notion of signal delay [8] has been used widely to approximate the time taken for a signal to start from an initial value and reach half of its final value.

If $\boldsymbol{G}$ is the node conductance matrix of a given *RC* network and $\boldsymbol{C}$ is the capacitance matrix of the network, calculating Elmore's delay, t_d, can be as simple as evaluating the product of $\boldsymbol{G}^{-1}$, $\boldsymbol{C}$, and unit vector $\mathbf{1}$. Since $\boldsymbol{G}$ and $\boldsymbol{C}$ are given, the delay estimation problem amounts to computing the resistance matrix, $\boldsymbol{R} \equiv \boldsymbol{G}^{-1}$. Thus delay estimation in *RC* networks has been viewed as a numerical problem: inverting $\boldsymbol{G}$ [13]. The computational requirement of this numerical approach limits its applicability to general problems. However, if the *RC* network is a tree, then $\boldsymbol{R}$ can be determined by inspection, and t_d can be computed in linear time. Almost all MOS timing level simulators treat networks as if they were trees, trading off accuracy for simplicity [3], [14], [19].

Signal delays in tree networks are easy to evaluate; unfortunately, many practical MOS circuits are not trees. The Manchester adder with carry-bypass circuitry, as depicted in Fig. 1, is an example [20]. Since the bypass transistor B is connected to $\overline{C}_0$ and $\overline{C}_4$, when this transistor is ON and all the P_i are high, they form a closed loop of conducting transistors, and cannot be modeled as an *RC* tree.

The goal of this paper is to provide an efficient way of computing signal delays in *RC* networks that do not necessarily form trees. We partition a given network into a spanning tree and links. We then start with the signal delay of the *spanning tree*, and gradually update the signal delay while incrementally piecing the *links* back to reconstruct the original network. The concept of circuit partitioning will be explained in Section III. A delay computation algorithm and its complexity will be explained in Section IV. Here, we shall consider only *RC* networks with grounded capacitors at each node of the network and no floating capacitors. Extensions to handle floating capacitors are discussed in [5].

A note on related work: Lin and Mead invented the technique of "tree decomposition" and "load redistribution" to calculate signal delay in general *RC* networks [13]. Their algorithm is relaxation-based, and the number of relaxation steps depends on the required accuracy. The algorithm that we are presenting here is based on dynamic programming, and terminates in a predetermined number of steps with the exact result.

II. Signal Delay in *RC* Networks

The *delay estimation problem* aims to find the time interval that it would take a signal to start from an initial value and reach a prescribed value. The most meaningful such value for digital circuits is the threshold voltage where the two logic states cross. However, locating this delay time precisely can be as hard as finding the exact waveform. A notion of delay defined in terms of the first moment of the impulse response was introduced by Elmore [8]. Many researchers have used the normalized Elmore delay to approximate such a delay time [2], [3], [13], [23]:

$$t_d \equiv \frac{\int_0^\infty h(t)\,t\,dt}{v(\infty) - v(0)} \tag{1}$$

where $v(\infty)$ and $v(0)$ are the final and initial voltages. If $v(t)$ is is the voltage response due to a unit step input, then an equivalent definition of Elmore delay is [17], [22], [23]:

$$t_d \equiv \frac{\int_0^\infty [v(\infty) - v(t)]\,dt}{v(\infty) - v(0)}. \tag{2}$$

Based on this notion of signal delay, closed-form delay expressions can be derived for *RC* networks without floating capacitors. We note that for a simple *RC* circuit with only one resistor and one grounded capacitor, Elmore delay is the same as the time constant of the circuit.

For a given *RC* network with n grounded capacitors, let

- $G_{i,j}$ be the branch conductance between nodes i and j (and by reciprocity, $G_{i,j} = G_{j,i}$);

Manuscript received August 28, 1988; revised July 1, 1989 and September 6, 1989. The work of P. K. Chan was supported in part by the University of California MICRO Program under Grant 89-136. This paper was recommended by Associate Editor R. K. Brayton.

The authors are with the Department of Computer Engineering, University of California, Santa Cruz, CA 95064.

IEEE Log Number 9035469.

- $G_{i,i}$ be the sum of all branch conductances connected to node i;
- C_i denote the capacitance at node i.

The node conductance matrix of the network has the following form:

$$\boldsymbol{G} = \begin{bmatrix} G_{1,1} & -G_{1,2} & \cdots & -G_{1,n} \\ -G_{1,2} & G_{2,2} & & -G_{2,n} \\ \vdots & & & \vdots \\ -G_{1,n} & -G_{2,n} & \cdots & G_{n,n} \end{bmatrix}.$$

With this, Elmore's delay for the ith node in the RC network is

$$t_{d,i} \equiv \frac{\sum_{j=1}^{n} R_{i,j} C_j [v_j(\infty) - v_j(0)]}{v_i(\infty) - v_i(0)} \tag{3}$$

or in matrix form

$$\boldsymbol{T}_d \equiv \frac{\boldsymbol{RC}[\boldsymbol{v}(\infty) - \boldsymbol{v}(0)]}{\boldsymbol{v}(\infty) - \boldsymbol{v}(0)} \tag{4}$$

where $\boldsymbol{R} \equiv \boldsymbol{G}^{-1}$, $\boldsymbol{C}$ is a diagonal matrix with entries $(C_1, C_2, \cdots, C_n)$ and $\boldsymbol{v}(\infty)$ and $\boldsymbol{v}(0)$ are the final and initial node voltages.

Speaking in terms of the components of the $\boldsymbol{R} = (R_{i,j})$ matrix, $R_{i,i}$ is the driving-point resistance between node i and the input node, and $R_{i,j}$ is the *transfer resistance* between nodes i and j. If the given RC network is a tree, then it has only one spanning tree and $R_{i,j}$ is the resistance of the portion of the unique path between the input and the jth node that is common with the unique path between the input and node i. In particular, $R_{i,i}$ is the resistance between the input and node i [17]. Hence, the resistance matrix $\boldsymbol{R}$ can readily be found by inspection. Evaluating delays for all n nodes of a tree network requires only n multiplications.

We shall focus on treatments of RC networks which are not necessarily trees for the rest of the paper.

III. Circuit Partitioning

Tearing is a means of partitioning circuits into several smaller, more manageable subcircuits to enhance computational efficiency. A large circuit is "torn" into simpler subcircuits, we solve the subcircuits, then we piece together the subcircuit solutions with a formal mechanism to yield the composite result.

An RC circuit is abstracted as a circuit graph $\mathcal{G} = (N, E)$, consisting of a set of nodes $N = \{n_0, n_1, \cdots, n_n\}$ and a set of edges $E = \{e_1, \cdots, e_b\}$. Each node in the RC circuit has a grounded capacitor, and each edge has a resistor. Without loss of generality, the node n_0 designates the input node. Furthermore, we can partition $\mathcal{G}$ into two parts: a *spanning tree* consisting of n edges; and $b - n$ edges that do not belong to the tree, called *links*.

3.1. Tree/Link Partitioning

Given a circuit graph, we can partition it into a spanning tree and links using a simple depth-first search of the graph. We start by solving the spanning tree, and update the solution by incrementally adding the links back to reconstruct the original network. The mechanism that we use for updating the solution is based on a formula in [16] where Kron's idea of circuit partitioning was advocated [12].[1] Rohrer shows the consequence of adding a resistor to a circuit that has been solved. Adding more than one resistor can be treated inductively by adding one resistor at a time. The idea of tree/link partitioning has previously been applied to computing the voltage drops along a power distribution net [11], to a piecewise linear circuit simulator [10], [15], and to solving linear equations by tree relaxation [18].

[1]Similar ideas have been explored in different contexts, see [4], [9].

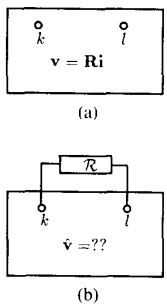

Fig. 2. Updating a resistive circuit.

3.2. Adding a Single Resistor

Suppose that we have already computed all the node voltages $\boldsymbol{v}$ of a resistive circuit $\boldsymbol{R}$ driven by the current source vector $\boldsymbol{i}$ (Fig. 2(a)), and we wish to ascertain the consequences of adding a resistor of value $\mathcal{R}$ between its kth and lth nodes, as illustrated in Fig. 2(b).

Rohrer [16] shows that the effect on the circuit node voltages of the addition of a resistor $\mathcal{R}$ between nodes k and l is

$$\hat{\boldsymbol{v}} = \boldsymbol{v} - \frac{v_k - v_l}{\mathcal{R} + R_{k,k} + R_{l,l} - 2R_{k,l}} \boldsymbol{R}\xi_{k,l} \tag{5}$$

where $\xi_{k,l}$ is a column vector with a "+1" in the kth row, a "−1" in the lth row, and zeros everywhere else. We note that the connection vector $\xi_{k,l}$ indicates that the resistor to be added is connected from node k to node l.

3.3. Applying Tree/Link Partitioning to Signal Delay Estimation

First we consider networks with only one driving-source (either V_{DD} or ground), such that all node voltages $v_i(\infty)$ attain the same final value. The effect of multiple driving-sources and different initial and final node voltages will be considered in Section III-3.4. Assuming that all initial and final node voltages are the same, from (5) we obtain

$$\begin{aligned} &\hat{\boldsymbol{v}}(\infty) - \hat{\boldsymbol{v}}(t) \\ &\quad = \boldsymbol{v}(\infty) - \boldsymbol{v}(t) \\ &\qquad - \frac{(v_k(\infty) - v_k(t)) - (v_l(\infty) - v_l(t))}{\mathcal{R} + R_{k,k} + R_{l,l} - 2R_{k,l}} \boldsymbol{R}\xi_{k,l}. \end{aligned}$$

Then we apply Elmore's definition of delay to obtain the following formula for updating delay values:

$$\hat{\boldsymbol{t}}_d = \boldsymbol{t}_d - \frac{t_{d,k} - t_{d,l}}{\mathcal{R} + R_{k,k} + R_{l,l} - 2R_{k,l}} \boldsymbol{R}\xi_{k,l}. \tag{6}$$

The strategy for computing the signal delay for arbitrary RC networks is clear: we remove all the links until we have a spanning tree, compute the signal delay of the tree, and then gradually add back the previously deleted links using (6).

This approach has the following desirable properties: first, the order of the removal of the links is arbitrary, which makes implementation easier. Second, it is not necessary to compute the signal delay of all nodes—the computation involved can be limited to the nodes for which signal delays are required, for instance, the primary output nodes. Third, the complexity of this approach depends on the number of links. Since VLSI circuits have few links, this approach lends itself well to VLSI applications.

Most existing RC timing simulators handle only tree networks [1], [19]. To facilitate the incorporation of our idea into these simulators, we shall formulate our method using the tree delay evaluation as the primary operation. Referring to (6), we note that $\boldsymbol{R}\xi_{k,l}$ can be conceived of as a delay calculation with the grounded capacitances of the kth and lth nodes set to +1 and −1, respectively,

and the rest set to zeros. Furthermore, if we define $\boldsymbol{r} = (r_i) \equiv \boldsymbol{R}\xi_{k,l}$, then the delay expression can be reformulated as

$$\hat{t}_d = t_d - \frac{t_{d,k} - t_{d,l}}{\Re + r_k - r_l}\boldsymbol{r}. \tag{7}$$

This formula, and the algorithm to be presented in Section IV, constitute the major result of this article.

Example: Fig. 3 shows an instance (with $P_i = 1$, $G_i = 0$, and $C_0 = 1$ for $i = 1, \cdots, 4$) of the *RC* model for the transistor loop in the Manchester adder as shown in Fig. 1. With the switch (*CK*) closed, deleting any one of the resistors $R_1, \cdots, R_5$ from the circuit will result in a tree, so we arbitrarily remove R_3 and compute the signal delay of the remaining spanning tree, as shown in Fig. 4(a):

$$\boldsymbol{t}_d = \begin{bmatrix} R_6(C_1 + C_2 + C_3 + C_4 + C_5) \\ R_6(C_1 + C_2 + C_3 + C_4 + C_5) + R_1(C_2 + C_3) \\ R_6(C_1 + C_2 + C_3 + C_4 + C_5) + R_1(C_2 + C_3) + R_2C_3 \\ R_6(C_1 + C_2 + C_3 + C_4 + C_5) + R_5(C_4 + C_5) + R_4C_4 \\ R_6(C_1 + C_2 + C_3 + C_4 + C_5) + R_5(C_4 + C_5) \end{bmatrix}$$

$$= \begin{bmatrix} 1000 \\ 1200 \\ 1300 \\ 1500 \\ 1400 \end{bmatrix}. \tag{8}$$

Let's call this an *RC* tree delay operation. To compute $\boldsymbol{r}$, we simply set C_3 to $+1$, C_4 to -1, and the rest of the capacitors to zeros in (8), as depicted in Fig. 4(b). This gives us

$$\boldsymbol{r} = \begin{bmatrix} 0 \\ R_1 \\ R_1 + R_2 \\ -(R_4 + R_5) \\ -R_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \\ 20 \\ -30 \\ -20 \end{bmatrix}.$$

Now, we *merge* $\boldsymbol{t}_d$ and $\boldsymbol{r}$ using (7) to obtain the Elmore signal delay of the original network:

$$\hat{\boldsymbol{t}}_d = \boldsymbol{t}_d - \frac{t_{d,3} - t_{d,4}}{R_3 + r_3 - r_4}\boldsymbol{r} = \begin{bmatrix} 1000.0 \\ 1233.3 \\ 1366.7 \\ 1400.0 \\ 1333.3 \end{bmatrix}.$$

For a connected network with $n + 1$ nodes and n edges, only one tree delay evaluation is needed. We have just shown that it takes two tree delay evaluations to compute the signal delay of the carry-bypass circuit with a single loop. In general, it takes $b - n + 1$ tree delay evaluations for a network consisting of $n + 1$ nodes, b edges, and $b - n$ links. A general analysis will be given in Section IV.

3.4. RC Networks with Two Driving Sources

To compute signal delays in a network with two driving sources, we need to know the final node voltages. We partition the network into subnetworks, each driven by a single source. We use (5) to compute the final node voltages (with the renaming $\boldsymbol{r} = \boldsymbol{R}\xi_{k,l}$), namely:

$$\hat{\boldsymbol{v}}(\infty) = \boldsymbol{v}(\infty) - \frac{v_k(\infty) - v_l(\infty)}{\Re + r_k - r_l}\boldsymbol{r}. \tag{9}$$

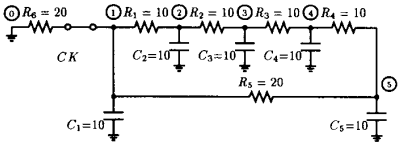

Fig. 3. Manchester adder *RC* model.

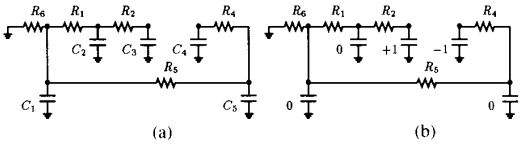

Fig. 4. Two tree evaluations as a result of partitioning.

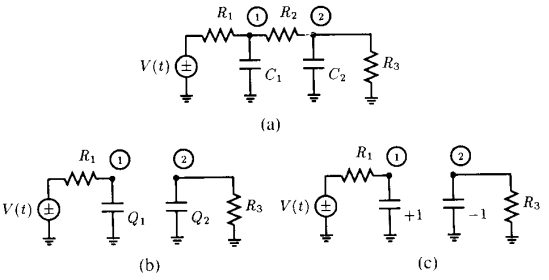

Fig. 5. Tree evaluations as a result of partitioning.

The effect of initial and final node voltages on signal delays can be easily accommodated by the following trick. Referring to (3), imagining that each node capacitance is of value $Q_j = C_j[\hat{v}_j(\infty) - \hat{v}_j(0)]$, we compute the "signal delays" of each subnetwork with these "capacitances." The "signal delay" values computed are merged by using (7); then are divided by $\hat{v}_i(\infty) - \hat{v}_i(0)$ to obtain the actual signal delay. The following simple example will illustrate the idea.

Fig. 5(a) shows an *RC* network with a leakage path to ground. Let $V(\infty) = 1$ V, and $v_1(0) = v_2(0) = 0$ V. The signal delay of this network can be computed by considering the subnetworks (forest) shown in Fig. 5(b) and 5(c), obtained by deleting R_2 from Fig. 5(a). First, we obtain $\boldsymbol{r}$ by considering the subnetworks shown in Fig. 5(c)

$$\boldsymbol{r} = \begin{bmatrix} R_1 \\ -R_3 \end{bmatrix}.$$

By (9), the final node voltages are

$$\hat{\boldsymbol{v}}(\infty) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \frac{1}{R_2 + R_1 - (-R_3)}\begin{bmatrix} R_1 \\ -R_3 \end{bmatrix}$$

$$= \frac{1}{R_1 + R_2 + R_3}\begin{bmatrix} R_2 + R_3 \\ R_3 \end{bmatrix}.$$

Next, we compute the "signal delays" of Fig. 5(b), which yields

$$\boldsymbol{t}_d = \begin{bmatrix} R_1C_1\hat{v}_1(\infty) \\ R_3C_2\hat{v}_2(\infty) \end{bmatrix}.$$

Finally, we combine (4) and (7) into

$$\hat{\boldsymbol{t}}_d = \frac{\boldsymbol{t}_d - \dfrac{t_{d,1} - t_{d,2}}{R_2 + r_1 - r_2}\boldsymbol{r}}{\hat{\boldsymbol{v}}(\infty)}$$

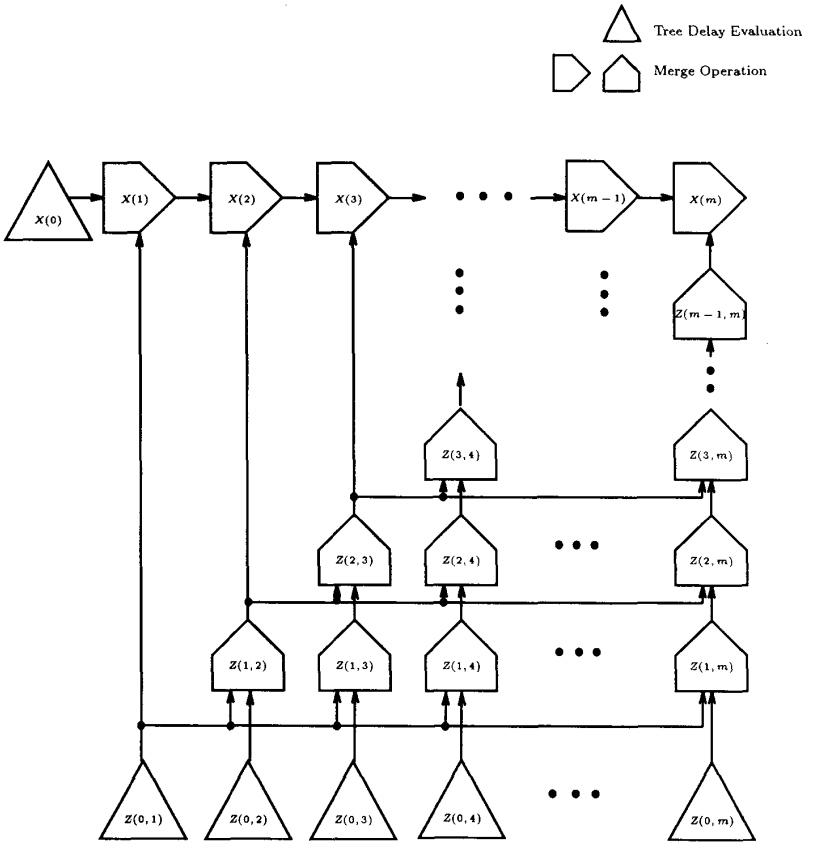

Fig. 6. Combining subcircuits.

to obtain the result

$$\hat{t}_d = \frac{1}{R_1 + R_2 + R_3} \begin{bmatrix} R_1(R_2 + R_3)C_1 + R_1 \dfrac{R_3^2}{R_2 + R_3} C_2 \\ R_1(R_2 + R_3)C_1 + R_3(R_1 + R_2)C_2 \end{bmatrix}.$$

IV. Complexity

Let $n + 1$ be the number of nodes and b be the number of edges in a given network. Let $m = n - b$ be the number of links. We show that the algorithm uses $m + 1$ tree evaluations and $m(m + 1)/2$ merges. Notice that the update operation in (7) can be applied only to two circuits with the same structure $\boldsymbol{R}$. After the update, the new delays correspond to a different resistance structure. To add another link to the circuit, we first have to compute the delay for that branch with the modified resistance structure.

Given a network with circuit graph $\mathcal{G} = (N, E)$, let:

- $s \leq n$ be the number of specific nodes for which signal delays are required;
- $L = \{b_1, \cdots, b_m\}$ be any set of links, and (p_j, q_j) be the nodes to which the link b_j is connected;
- $X(i)$ be a circuit with the original capacitors but with links b_{i+1} to b_m deleted from $\mathcal{G}$;
- $Z(i, j)$ be a contrived circuit with links b_{i+1} to b_m deleted from $\mathcal{G}$, and with the $+1$ and -1 capacitors connected to nodes p_j and q_j, respectively, and the rest of capacitors zeroed; defined only for $j > i$.

The $m + 1$ tree evaluations are performed on $X(0)$ (the spanning tree with the original capacitors), and $\{Z(0, j); j = 1, \cdots, m\}$ (the spanning trees with the $+1$ and -1 capacitors connected to nodes p_j and q_j). Each tree evaluation requires n operations.

Starting with the trees $X(0)$, $Z(0, 1)$, $\cdots$, $Z(0, m)$, the algorithm for combining the circuits can be expressed by the recurrences:

$$\begin{cases} Z(i, j) = \text{merge}\big(Z(i-1, i), Z(i-1, j)\big), \\ \qquad i = 1, \cdots, m; j > i \\ X(i) = \text{merge}\big(X(i-1), Z(i-1, i)\big), \\ \qquad i = 1, \cdots, m \end{cases}$$

where *merge* denotes the operation of computing delay using (7); with the first argument providing $\boldsymbol{t}_d$ and the second providing $\boldsymbol{r}$. Because the merge operation that involves link b_j needs the signal delays of nodes (p_j, q_j), the merge operation takes $O(n)$ additions and multiplications. Overall, the time complexity of the algorithm is $O(nm^2)$, and the space complexity is $O(nm^2)$.[2]

The structure of the computation is illustrated in Fig. 6, clearly showing the $m(m + 1)/2$ merges that are required. Each triangle denotes a tree delay evaluation of the labeled circuit, and each diamond denotes a merge operation using (7). We reiterate the important point that all $m + 1$ tree delay evaluations use the same resistive spanning tree, but with different values for the capacitors.

If $O(nm^2)$ operations are too many for some applications, faster approximate answers can be obtained by restoring only a few of the links. The circuits $X(0)$, $\cdots$, $X(m - 1)$ can be considered

[2]If m is much smaller than n and the delay of only s nodes is required, then each merge can be computed in $O(m + s)$ time giving a time complexity of $O(n(m + 1) + m^3 + sm^2)$.

as approximations to $X(m)$. Premature termination of the algorithm will deliver the signal delay of an approximated circuit.

If $\mathcal{G}$ is a connected simple planar graph, the maximum number of edges is $3(n+1)-6$ [21]. Therefore, the time complexity of the algorithm for a planar circuit is $O(n^3)$. The performance on real circuits is expected to be much better—nearly linear in n, since VLSI circuits are tree-dominant.

V. Remarks and Conclusions

We have presented a simple technique for computing delays in arbitrary networks of resistors with grounded capacitors. The technique allows rapid computation for *RC* networks that have a few links. It is, therefore, particularly appropriate for MOS timing simulators.

Not only is our method able to handle arbitrary *RC* networks for timing simulation, but it also provides the basis for the analysis of CMOS circuits with transistor loops. For example, in designing a Manchester adder with variable carry-skip [6], we need to determine the carry-ripple and carry-skip delays of a k-bit CMOS Manchester adder block (Fig. 1 shows a 4-b Manchester adder block). Using our technique we derived analytical *RC* delay models for carry-ripple and carry-skip, and determined the optimal configuration for the adder. This would not have been possible with the more numerical nature of Lin and Mead's method [13].

References

[1] *TIMEMILL User Manual.* EPIC Design Technology, Santa Clara, CA 1988.

[2] K. G. Ashar, "The method of estimating delay in switching circuits and the figure of merit of a switching transistor," *IEEE Trans. Electron Devices*, vol. ED 11, pp. 497–506, Nov. 1964.

[3] R. Bauer, P-C. Ng, A. Raghunathan, M. W. Saake, and C. D. Thompson, "Simulating MOS VLSI circuits using supercrystal," in *Proc. Int. Conf.*, VLSI 87, IFIP, 1987.

[4] M. L. Brock, "Algebraic and graph-theoretic aspects of networks," Master's thesis, Massachusetts Instit. Tech., Dec. 1984.

[5] P. K. Chan, "Signal delay in *RC* networks with floating capacitors," in *Proc. Int. Symp. Circuits and Systems*, Espoo, Finland, June 1988.

[6] P. K. Chan and M. D. F. Schlag, "Analysis and design of CMOS Manchester adders with variable-skip," in *Proc. 9th Computer Arithmetic Symp.*, Santa Monica, Los Angeles, Sept. 1989, pp. 86–95.

[7] ——, "Bounds on signal delay in *RC* mesh networks," *IEEE Trans. Computer-Aided Design*, vol. 6, pp. 581–589, June 1989.

[8] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, Jan. 1948.

[9] A. S. Householder, *The Theory of Matrices in Numerical Analysis.* New York: Dover, 1964.

[10] X. Huang, L. T. Pillage, and R. Rohrer, "TALISMAN: A piecewise linear simulator based on tree/link repartitioning," in *Proc. IEEE Int. Conf. Computer-Aided Design ICCAD-87*, Santa Clara, CA, Nov. 9–12, 1987, pp. 98–101.

[11] F. H. Branin, Jr., "The analysis and design of power distribution nets on LSI chips," in *Proc. Int. Conf. Circuits and Computers*, New York, Sept. 28–Oct. 1, 1980, pp. 785–790.

[12] G. Kron, *Tensor Analysis of Networks.* New York: Wiley, 1939.

[13] T.-M. Lin and C. A. Mead, "Signal delay in general *RC* networks," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 331–349, Oct. 1984.

[14] J. K. Ousterhout, "A switch-level timing verifier for digital MOS VLSI," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 336–348, July 1985.

[15] L. T. Pillage, X. Huang, and R. Rohrer, "Tree/link partitioning for the implicit solution of circuit equations," in *Proc. Int. Symp. Circuits and Systems*, San Jose, CA, June 1987, pp. 1072–1075.

[16] R. A. Rohrer, "Circuit partitioning simplified," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 2–5, Jan. 1988.

[17] J. Rubinstein, P. Penfield, Jr., and M. Horowitz, "Signal delay in *RC* tree networks," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 202–211, July 1983.

[18] C. Shi and K. Zhang, "Tree relaxation: A new iterative solution method for linear equations," in *Proc. Int. Symp. Circuits and Systems*, Espoo, Finland, June 1988, pp. 2355–2358.

[19] C. J. Terman, "Simulation tools for digital LSI design," Ph.D. dissertation, Massachusetts Instit. Tech., Sept. 1983.

[20] N. Weste and K. Eshraghian, *Principles of cMOS VLSI Design—A Systems Perspective.* Reading, MA: Addison-Wesley, 1985.

[21] R. J. Wilson, *Introduction to Graph Theory.* New York: Academic, 1979.

[22] J. L. Wyatt, Jr., "Signal delay in *RC* mesh networks," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 507–510, May 1985.

[23] R. A. Zakarevicius, "The time delay of a network: Characterization and measurement," in *Proc. IREE*, pp. 556–560, Dec. 1972.

[24] C. A. Zukowski, "Relaxing bounds for linear *RC* mesh circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 305–312, Apr. 1986.