RICE: Rapid Interconnect Circuit Evaluation Using AWE

Curtis L. Ratzlaff and Lawrence T. Pillage, *Member, IEEE*

***Abstract*— This paper describes the Rapid Interconnect Circuit Evaluator (RICE) software developed specifically to analyze RC and RLC interconnect circuit models of virtually any size and complexity. RICE focuses specifically on the passive interconnect problem by applying the moment-matching technique of Asymptotic Waveform Evaluation (AWE) and application-specific circuit analysis techniques to yield large gains in run-time efficiency over circuit simulation without sacrificing accuracy. Moreover, this focus of AWE on passive interconnect problems permits the use of moment-matching techniques that produce stable, pre-characterized, reduced-order models for RC and RLC interconnects. RICE is demonstrated to be as accurate as a transient circuit simulation with hundreds or thousands of times the efficiency. The use of RICE is demonstrated on several VLSI interconnect and off-chip microstrip models.**

I. Introduction

THE CAPABILITY to accurately predict the delay of signal paths in today's high-speed digital integrated circuits (IC's) has never been more important. With clock speeds continuing to increase at all levels of integration, the safety margin for timing errors is decreasing. Moreover, competitive pressures in the fast-moving world of high technology have forced producers of electronic equipment to tighten their timing tolerances and to push their specifications to the limits of current technology. This means that accurate timing analysis of critical signal paths has become crucial to maintaining quality and remaining competitive.

The delay in signal paths can be attributed to two primary components—the gate delay and the interconnection delay. Popular timing analyzers/simulators [2]–[4] have typically assumed that the on-chip delay was attributed primarily to the gate driving the total capacitance of the loads and interconnect. However, designers are discovering that more of the signal delay is being attributed to the RC effects of the interconnection. In fact, up to 70 percent of the total delay may now be attributed to the interconnection alone [1].

The problem of accurate interconnect characterization is not limited to just VLSI designers. After all, the very fast devices being developed by VLSI designers are ultimately being incorporated into board-level designs and/or multichip modules (MCM's). In these areas, not only are RC effects evident, but inductive effects also begin to appear as anomalies of overshoot, undershoot, ringing, and signal reflections. With these effects present, the problem now is to predict the shape of the signal waveform rather than the delay alone. With high-speed ECL and GaAs devices often used in the MCM and board-level designs, interconnect and packaging effects are dominating factors [4]. Transmission line effects on high-speed digital MCM's are the norm and traditional timing analyzers are not equipped to handle these or related effects.

The remainder of this paper focuses on our techniques for rapid and accurate evaluation of linear interconnect circuit models composed of R, L, and C components. These techniques have been incorporated into the Rapid Interconnect Circuit Evaluator (RICE) software developed specifically for characterization of interconnect circuit models. RICE generally yields results within 1 percent of a circuit simulator, such as SPICE [5], but with considerably more efficiency (over $1000\times$ better for moderate to large circuit models). The basic technique used in RICE is the moment-matching method of Asymptotic Waveform Evaluation (AWE) [6]–[8], which is becoming a popular technique for timing analysis. What distinguishes RICE from other implementations of AWE is that it focuses specifically on the passive interconnect problem, thereby taking full advantage of the regularity of the circuits. Specifically, path-tracing techniques are used to analyze the circuit, thus exploiting the simple topologies often encountered for interconnect models. Furthermore, the passive nature of interconnect circuits is exploited to ensure stable results.

Section II reviews the various methods that have been used to characterize the delay in RC interconnects. Section III briefly reviews the basic AWE theory and technique followed by a presentation in Section IV describing, in general, how circuit moments may be calculated from analysis of a simple equivalent dc circuit. Section V expands upon this discussion with a specific example of how RICE computes the moments for R(L)C tree-like topologies, while Sections VI and VII introduce extensions to the basic path-tracing method for handling any RLC topology, including those with loops of resistors and inductors and floating nodes. Section VIII briefly discusses how the circuit moments are matched to the dominant poles and residues that comprise the reduced interconnect model. Section IX presents the accuracy and performance results for various circuit topologies.

II. Methods of Interconnect Characterization

A variety of methods for the characterization of interconnect circuit models have been investigated [9]–[11]. Obviously the

Manuscript received April 6, 1992; revised December 20, 1993. This work was supported in part by the National Science Foundation under MIP # 9157263, the Semiconductor Research Corporation under contracts 91-DP-142 and 93-DJ-343, and International Business Machines Corporation.This paper was recommended by Associate Editor A. E. Ruehli.

The authors are with the Electrical and Computer Engineering Department, University of Texas at Austin, Austin, Texas 78712 USA.

IEEE Log Number 9210624.

most accurate solution would be to use a circuit simulator such as SPICE, but this is unreasonably slow and, for very large interconnect models, may not even be possible. A solution that has been applied in industry is to develop a specialized circuit simulator that applies a fixed time-step approach so that only one circuit-equation factorization is required [12]. The time-step used in the integration may be varied to control the accuracy and/or execution-time. This is significantly faster than a more generalized SPICE approach [13], but it is still unacceptably slow for moderate-sized interconnects. Moreover, both SPICE and the fixed time-step approaches are very sensitive to the stiffness of the circuit equations, and it is well known that interconnect circuit models tend to be stiff.

The most common technique encountered for bypassing the inefficiency of the circuit simulator approach is to model the interconnect in LSI/VLSI technologies with an RC tree or RC mesh and then to calculate the *Elmore delay* [14] for this model. The Elmore delay corresponds to an approximation of the time at which the output response of an interconnect to a step input reaches 50 percent of its final value. The Elmore delay was the first *moment-method* to be applied to the interconnect problem. The Elmore delay, T_D, is the first time-moment of the impulse response for a unit-step driving an RC tree or RC mesh:

$$T_D = \int_0^\infty t\dot{v}\; dt \tag{1}$$

The Elmore delay value may be easily found for an RC tree with a linear complexity algorithm described in [9]. The value of T_D, however, may be either pessimistic or optimistic as an approximation of the interconnect delay. The delay can be bounded as described in [10], but the bounds are sometimes too wide.

The Elmore delay may be considered to be a one-pole or dominant time-constant approximation for RC trees and meshes where the dominant pole is expressed as T_D^{-1}. To improve the accuracy and address the problem of nonequilibrium initial conditions, a two-pole model for analysis of RC meshes was developed in [15], [16]. It required the calculation of the first three moments of the impulse response. This method is generally adequate for RC trees and meshes, but it could not be applied to circuits with floating capacitances or inductors.

The next extension to the moment-matching techniques was Asymptotic Waveform Evaluation (AWE) presented in [6]–[8]. This is a generalized n-th order moment-matching technique that allows a linear(ized) circuit to be analyzed for its dominant poles and corresponding residues. The general idea is to calculate a certain number of time-moments for the circuit model and then match these moments to the reduced order model. For example, a linear circuit model containing hundreds of energy storage elements will in general possess hundreds of poles in the frequency domain. However, the transient response of the circuit at any given node is generally dominated by a very small number (<10) of approximate poles. The goal in AWE is to locate this set of dominant poles that may be used to characterize the circuit response(s).

III. A Brief Overview of AWE

The basic process in AWE is to approximate the transfer function of a linear circuit model, $H(s)$, by a reduced set of approximate poles and residues of the form

$$\hat{H}(s) = \frac{k_1}{s-p_1} + \frac{k_2}{s-p_2} + \cdots + \frac{k_q}{s-p_q} \tag{2}$$

where p_i and k_i are the poles and residues, respectively, and q is the order of approximation (number of approximate poles). This is easily cast to the time-domain equivalent

$$\hat{h}(t) = k_1 e^{p_1 t} + k_2 e^{p_2 t} + \cdots + k_q e^{p_q t} \tag{3}$$

which is what we seek for timing analysis.

AWE uses moment-matching to uniquely specify the k's and p's in (2) and (3). The moments of $\hat{H}(s)$ are forced to match the actual circuit moments. In general, we refer to the i-th moment of any function of time, $f(t)$, as

$$f(t) = \frac{(-1)^i}{i!} \int_0^\infty t^i f(t)\; dt \tag{4}$$

Applying the definition in (4) to $\hat{h}(t)$ in (3), the approximate moments are of the form:

$$\hat{m}_i = \frac{k_1}{p_1^{i+1}} + \frac{k_2}{p_2^{i+1}} + \cdots + \frac{k_q}{p_q^{i+1}} \tag{5}$$

Thus, for a q-th pole approximation there will be q unknown poles $(p_1 \cdots p_q)$ and q unknown residues $(k_1 \cdots k_q)$ for a total of $2q$ unknowns. The $2q$ equations are

$$\begin{array}{ccccccccc}
k_1 & + & k_2 & + & \cdots & + & k_q & = & m_0 \\
\frac{k_1}{p_1} & + & \frac{k_2}{p_2} & + & \cdots & + & \frac{k_q}{p_q} & = & m_1 \\
\frac{k_1}{p_1^2} & + & \frac{k_2}{p_2^2} & + & \cdots & + & \frac{k_q}{p_q^2} & = & m_2 \\
\vdots & & \vdots & & & & \vdots & & \vdots \\
\frac{k_1}{p_1^{2_q-1}} & + & \frac{k_2}{p_2^{2_q-1}} & + & \cdots & + & \frac{k_q}{p_q^{2_q-1}} & = & m_{2_q-1}
\end{array} \tag{6}$$

Note that the right-hand side of these equations are the $2q$ actual time-moments for a *specific* response variable in the circuit. A response variable may be any voltage or current, and only those responses of interest need to be solved.

Evaluating (6) to determine the poles and residues can be done as described in [6]. However, since such a mapping has been shown to be equivalent to a Padé approximation [18], it is well known that positive dominant poles can result even for stable circuits. In RICE, however, the nature of interconnect circuit models is once again exploited by forcing stable poles since the circuits are guaranteed to be stable. Three different approaches were tried and each is briefly covered in Section XIII. The details on instability and approaches to avoiding it are too extensive for a thorough coverage here. More information can be found in [18]–[21].

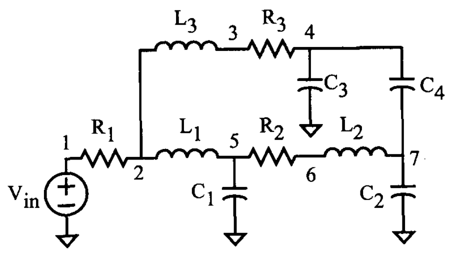
IV. General Method for Calculating the Circuit Moments

The AWE moment-matching technique depends on two critical assumptions: 1) the time-moments of the circuit model can be accurately and efficiently calculated and 2) these moments can be matched to the reduced-order approximation. The elegance of AWE is attributed to the manner in which moments for a linear circuit are calculated. It requires only a few successive dc analyses of the circuit model where capacitors are replaced by dc current sources and inductors are replaced by dc voltage sources. The process begins by replacing the input driver with a dc source set equal to the final value, all capacitors with zero-valued current sources, and all inductors with zero-valued voltage sources. The resultant voltages across each capacitor-current source represent the first generation of capacitor moments. Similarly, the currents through each inductor-voltage source comprise the first generation of inductor moments. The succeeding generations of moments are calculated by setting the driver to zero and replacing each capacitor and inductor source with the product of its previous moment and respective value of capacitance or inductance.

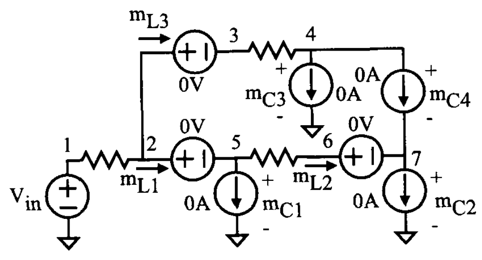
For example, consider the RLC circuit in Fig. 1(a). The first generation of moments may be calculated by first transforming this circuit into the dc equivalent shown in Fig. 1(b) and then performing a dc analysis. The voltages across each capacitor and currents through the inductors are the first set of moments. Next, the source values are modified as shown in Fig. 1(c) and the circuit re-solved. The next set of moments are again the resulting voltages and currents of the capacitors and inductors, respectively. The procedure is repeated until the desired number of moments is calculated.

The reason AWE is inherently much faster than a transient circuit analysis is because only a few dc analyses are required to compute all the moments and matching the moments to the actual poles and residues is efficient for a small number of poles. A transient analysis, on the other hand, requires the circuit to be solved at a large number of time steps. The most competitive transient simulation technique we have encountered is when a fixed-time step integration algorithm is applied to a relatively small circuit that is not overly stiff. Even in this case, AWE out-performs the transient simulation by a large factor. It is pointless, however, to make a direct comparison since the numerical integration methods will depend on the number of time steps taken and the error due to local truncation. But it is safe to say that the AWE circuit is generally a simpler circuit to solve and the analytical transfer function model produced is more valuable that a single time-domain waveform.

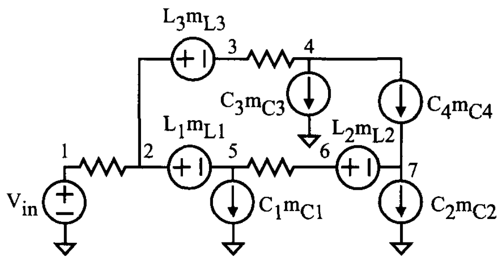
From this discussion it is clear that the primary factor in guaranteeing efficiency is to employ an efficient dc analysis algorithm. In general, any sound circuit analysis technique can be employed. For instance, the implementation of AWE described in [7] uses modified nodal analysis to solve the dc circuit. Other techniques are also possible, such as the use of sparse-tableau, tree/link analysis, and others. However, we chose to exploit the generally simple topology of most interconnect circuit models by using a circuit path-tracing technique. Most interconnect model topologies follow a tree-like structure such as the RC tree that are ideal for a path-tracing method. Terman [9] demonstrated how effective it was for computing the Elmore delay in RC trees. We have extended this to R(L)C trees and similar topologies. Internodal or coupling capacitances may be added to the tree structures without disturbing our definition of a tree-like topology. Furthermore, extensions have been developed which allow deviations from a strict tree structure. Instances of resistor loops, floating



(a)



(b)



(c)

Fig. 1. (a) RLC circuit example, (b) its equivalent dc circuit for the first moment generation, and (c) the circuit with source values changed for the second moment generation.

nodes or inductor loops usually have only a minor effect on performance. In the limit, as the circuit topologies deviate from the desired tree structure, the algorithms degenerate to an efficient implementation of nodal analysis. In short, the overall approach taken in the development of RICE is to allow tree-like structures to be evaluated in the most efficient manner possible, while allowing models that deviate from this to degrade performance by an amount more or less proportional to the topological complexity.

V. Moment Computation for R(L)C Trees

Before describing the basic path-tracing algorithm for R(L)C trees, we must first define our notion of a tree-like topology. If a spanning-tree of the circuit model can be constructed that includes *all* voltage sources, inductors, and resistors and *excludes* all capacitors and current sources, then the circuit model is strictly tree-like. This is possible for many interconnect circuit models, particularly those used for modeling RC delay effects in VLSI circuits. RICE *does* handle circuits that do not satisfy the stated conditions, but it must invoke extensions to the basic algorithm, all of which are discussed in succeeding sections.

A spanning tree, T, of the connected circuit graph, G, is defined as a connected subgraph of G that contains all nodes of G, but contains no loops [23]. The branches in T are called *tree-branches*, while all other branches of G (not contained in T) are called links. By our definition of a strictly tree-like interconnect circuit, all capacitors (current-sources) must be links, while all resistors and inductors must be tree-branches.

A. The Basic Path-Tracing Algorithm

Consider the dc equivalent circuit in Fig. 1(b) and (c) to calculate the time-moments for an AWE analysis. Once the capacitor-current sources and inductor-voltage sources are assigned their values from previous moment calculations, a spanning tree of the circuit graph may be traversed to solve for the currents and voltages of the dc circuit. A circuit graph and spanning tree for the circuit is revealed in Fig. 2. A spanning-tree for the circuit can be efficiently constructed in linear time using a standard algorithm such as that found in [32]. One complete traversal of the circuit graph is required to compute all tree-branch currents and another traversal is required to yield all node voltages. A similar approach was used in [9] to compute the Elmore delay for RC trees.

For example, beginning at any leaf node of the graph in Fig. 2, each node is visited by performing a *reverse* depth-first traversal of the spanning tree. As each node is visited, its incident tree-branch and link currents are summed, excluding the current of the tree branch from the predecessor node. This sum becomes the total current for the tree branch from the predecessor node. The use of a reverse depth-first traversal guarantees that a node is not visited until the currents for all branches from descendant nodes are known. The process is completed when the ground node is encountered. For each inductor tree-branch, the resulting current is the new inductor moment to be used in the next moment generation. The currents of tree-branch resistors and their corresponding

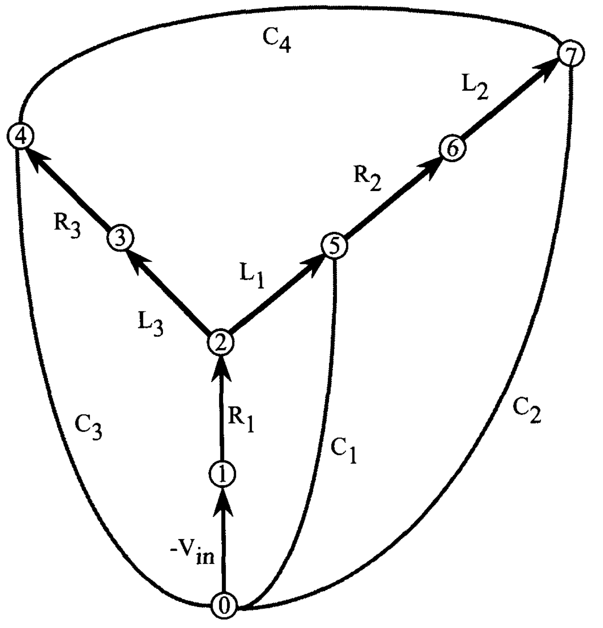


Fig. 2. Graph and spanning tree representation for the RLC circuit in Fig. 1.

resistance values are then used to compute the resistor branch voltages.

This application of the reverse path-trace is equivalent to solving the KCL equations $\mathbf{A}i_b = 0$, where $\mathbf{A}$ is the reduced nodal-incidence matrix [23]. The $\mathbf{A}$-matrix is subdivided into the tree-branch sub-matrix, $\mathbf{A}_t$, and the link-current sub-matrix, $\mathbf{A}_l$. Dummy entries are added to $\mathbf{A}_l$ to force it to be the identity matrix. This is equivalent to adding zero-valued (capacitor) current sources at each node that have no capacitors, which simplifies the subsequent analysis. With A organized as described, the following describes the reverse path-trace:

$$\begin{aligned} \mathbf{A}_t i_t + \mathbf{A}_l i_l = 0 \rightarrow i_t &= -\mathbf{A}_t^{-1}\mathbf{A}_l i_l \\ \text{but } \mathbf{A}_l = 1 \text{ so } i_t &= -\mathbf{A}_t^{-1} i_l \end{aligned} \tag{7}$$

$\mathbf{A}_t$ is always upper-triangular since tree-branches are ordered as they would be encountered during a forward traversal of the spanning tree. Therefore, solution for i_t can be achieved by a back-substitution using the link currents, i_l, as the right-hand-side (RHS). This is, in fact, what is achieved during the reverse path-trace. We do not explicitly build the $\mathbf{A}$ matrix; however, we do construct the circuit graph and find a spanning tree that can be considered equivalent to constructing and ordering $\mathbf{A}$ as described.

Next, beginning at the ground node, a *forward* depth-first traversal of the tree is performed to visit each node. The voltage of each node is computed by subtracting the voltage of the predecessor tree-branch from the predecessor node voltage. The use of the forward traversal guarantees a node is not visited until the voltage of its predecessor is known. The node voltages are then used to compute the voltage of each capacitor, which becomes the new moment for each capacitor used in the next moment generation. This application of the forward path-trace is equivalent to solving the KVL equations

$$\mathbf{A}^T v_n = v_b \text{ where } \mathbf{A}^T = \begin{bmatrix} \mathbf{A}_t^T \\ \text{- -} \\ \mathbf{A}_l^T \end{bmatrix} \text{ and } v_b = \begin{bmatrix} v_t \\ \text{- -} \\ v_l \end{bmatrix}$$

$$\text{so } \begin{bmatrix} \mathbf{A}_t^T \\ \text{- -} \\ \mathbf{A}_t^T \end{bmatrix} v_n = \begin{bmatrix} v_t \\ \text{- -} \\ v_l \end{bmatrix} \rightarrow \mathbf{A}_t^T v_n = v_t \text{ and } \mathbf{A}_l^T v_n = v_l$$

$$\text{but } \mathbf{A}_l^T = \mathbf{1} \text{ so } v_n = v_l \tag{8}$$

where v_n are the node voltages and v_l the (capacitor) link voltages. $\mathbf{A}_t^T$ is guaranteed to be lower-triangular, therefore a forward substitution operation may be applied to the lower-triangular system to solve for the node voltages. Again, the matrix is not explicitly built and performing the forward path-trace is equivalent to a forward-substitution procedure.

B. Virtual Path-Tracing

Rather than implementing the basic path-tracing scheme just described, a more efficient *virtual path-tracing* scheme was employed. The virtual path-tracing technique improves the efficiency by eliminating the graph traversal overhead. This is important since the graph is traversed many times to produce all required moment generations. Moreover, the technique also improves upon the usage of run-time memory. It does not reduce the amount of memory, but it does minimize the memory page-faults that occur during virtual path traces. When circuits are so large that they no longer fit into main memory, minimization of memory page faults become much more important than all other processing overhead since they significantly effect the actual elapsed time.

To understand the basis of virtual path-tracing, it must be recognized that a dc circuit need only be (actually) path-traced one time regardless of the number of moment generations required. Since neither the circuit graph nor spanning tree changes between each generation, the nodes will be visited in the same order during each moment generation. Moreover, the amount of information at each node (links and tree branches) does not change between generations. This suggests that instead of executing an actual traversal of the graph and tree for each analysis, the traversal should be performed only one time to *memorize* the location and order of the links and tree branches. This memorization is performed by re-organizing the graph into a set of numeric data vectors and compressed instruction lists. The data vectors contain the values of tree-branch and link elements organized in order of a depth-first traversal of the spanning tree. The (integer) instruction lists describe what arithmetic operations must be performed on the vectors to produce the results of the path-tracing algorithm. One instruction list describes the reverse trace while the other list describes the forward trace. These lists are compressed for maximum efficiency. For example, a nonbranching RC tree would be described by a single two-word integer instruction, regardless of the number of nodes in the tree. A complete virtual path-trace can be achieved by reading the instruction and then looping across the floating point vectors in forward and reverse order.

Uncompressed Tree-Current Mnemonics		Compressed Tree-Current Mnemonics	
Node	Instruction(s)	Node(s)	Instruction(s)
4	(Push)Load	4	(Push)Load
3	AddSto	3	AddSto
7	PushLoad	7	PushLoad
6	AddSto	6,5	Rep 2 AddSto
5	AddSto	2	PopAdd
2	PopAdd	2,1	Rep 2 AddSto
	AddSto		
1	AddSto		

Instruction Definitions

```
PushLoad    (S) <-- A;        // store accumulator at stack pointer
            S <-- S + 1;      // increment stack pointer
            A <-- (L);        // load accumulator with next link current
            L <-- L - 1;      // decrement link current pointer

AddSto      A <-- A + (L);    // add link current to accumulator
            (L) <-- A;        // and replace it
            L <-- L - 1;      // decrement link current pointer

PopAdd      S <-- S - 1;      // decrement stack pointer
            A <-- A + (S);    // add top-of-stack to accumulator
```

Operand Definitions

A	Floating-point accumulator
S	Stack pointer
(S)	Floating-point value at stack-pointer location
L	Link-current vector pointer initially set to point to total link current of last node in the nodelist.
(L)	Floating-point value at link-current pointer location

After all instructions have been executed, L will point to the forward list of tree-branch current totals.

Fig. 3. An illustration of the virtual-path tracing generation process for the (circuit) graph of Fig. 2. Both the uncompressed and compressed instruction-lists are shown.

The information contained in a given instruction list is the same information that would be obtained by performing a depth-first search of the spanning tree, but the information is optimized so repeated use of it results in a large performance improvement. If an actual path-trace were used to compute the dc solution of the circuit, then not only would floating point operations be required at each node to compute currents and voltages, but traversal of each tree-branch and recognition of links would be required at every node. Virtual path-tracing avoids the traversal of tree-branches and links for every moment generation. The virtual path-trace for the Fig. 1 (graph in Fig. 2) circuit can be expressed as shown in Fig. 3. The odd-looking mnemonics are referred to as the virtual path-tracing *pseudo-instructions* and are defined in the illustration. These pseudo-instructions, stored as integer words, are interpreted to control the reverse and forward traces. The only information required to generate the primitives are the degrees of each respective node in the spanning tree. Note the use of the REPeat primitive in the instructions. This allows similar operations to be compressed into a single word, which allows them to be processed in a very tight loop, thus yielding very high throughput.

The use of stack primitives is not required, but it is highly desirable to maintain the highest level of data vectorization. If a stack were not used, then the partial results would have to be accessed randomly, thus causing a higher level of memory page-faults. The cost of these primitives pays off if even a single page fault is avoided. The example in Fig. 3 illustrates the reverse (tree-current) trace only. A similar procedure is used to handle the forward trace. The instructions are different, but the concept is the same.

The primitives have been called pseudo-instructions since they are not instructions mapped to a given computer architecture. However, this could be done. In other words, the list of pseudo-instructions could be translated to instructions of the target processor. This would remove the last bit of overhead from the path-trace, so the only remaining overhead would be that attributed to a given architecture. This is not done in RICE since it compromises the portability of the code and would not significantly increase performance. As it is, the virtual path-tracing represents one of the least expensive operations in RICE.

VI. Extension for Resistor Loops

One type of circuit topology heavily studied was that containing resistor loops or links (R-loops or R-links). Resistors that cause loops in the circuit graph are classified as resistor links during construction of the spanning tree. Resistor links are undesirable since their currents are not known *a priori.* Instances of R-loops occur for certain types of RC and RLC interconnect topologies. For example, loops of metal are often added to clock lines to reduce the skew among gates. Use of the additional metal run(s) is often more desirable than increasing the metal pitch. There is also a growing tendency to route clock lines with a mesh or grid pattern rather than a tree. Lossy transmission lines that include dielectric loss also have R-loops to ground at every node. Moreover, transmission lines are often terminated with resistors to ground, thus introducing R-loops.

A. Solving for R-Loop Currents by Branch Tearing

One way to handle link resistors in the AWE dc circuit is by *branch tearing* or Kron's method [24]. For a circuit with a single link resistor, the circuit is first solved to obtain the open circuit voltage, v_{oc}, across an open R-link. From Kron's method, the current that would flow through the link, were it not opened, is

$$I_R = \frac{v_{\text{oc}}}{R_{\text{link}} + R_{\text{th}}} \tag{9}$$

where R_{th} is the Thevenin resistance seen by R_{link}. The link resistor is then replaced by a current source of value I_R, thus restoring the tree structure of the dc equivalent circuit and allowing the circuit to be solved by path-tracing.

In the case where there are multiple, m, R-links the approach is still basically the same. We need to calculate a value of current for each R-link that would flow were they not opened. Therefore, we need to extend (9) to the general case of m R-links. The Kron method also supports this case, where I_R and v_{oc} become m-vectors and $(R_{\text{link}} + R_{\text{th}})^{-1}$ is replaced by Z^{-1} as follows:

$$I_R = Z^{-1} v_{\text{oc}} \tag{10}$$

given that

$$Z = F R_t F^T + R_i \tag{11}$$

where the loop/cutset matrix, $\mathbf{F}$, is $m \times N$, where m is the number of R-links and N is the number of tree branch resistors. $\mathbf{R}_t$ is an$N \times N$ diagonal matrix of tree-branch resistor values and Rl is an $m \times m$ diagonal matrix of R-link values. The m-vector, v_{oc}, is the *torn* or open-circuit voltages across the torn branches, which are easily obtained by an initial path-trace. The entry $\mathbf{F}_{ij}$ is ± 1 if the j-th resistor tree-branch is in the loop caused by the i-th R-link, otherwise the entry is zero. The sign is positive if the reference directions for the R-link and tree branch agree with one another, otherwise it is negative.

Solving the system for the R-link currents, I_R, requires the inversion of $\mathbf{Z}$, which is $m \times m$ and symmetric for resistive circuits but tends to be very dense. This, of course, depends on $\mathbf{F}$, which in turn depends on the spanning tree that is selected for the circuit graph. The F-matrix was not explicitly constructed. Instead, a path-tracing algorithm described in [25] was used to build Z directly from the circuit graph. Next, $\mathbf{Z}$ is LU factored while another path-trace yields the open-circuit voltages (i.e., with the R-links not present), and forward and back-substitution of these voltages yields the R-link currents. Finally, these currents are substituted for the torn branches and the circuit path-traced one last time to yield the final circuit solution.

Both the construction of Z and its subsequent inversion tend to be very expensive. Our empirical results suggest that this method is suitable for circuits with few resistor loops ($m < 100$), but tends to be overwhelming for circuits containing a large number of loops.

B. Solving for R-Loop Currents by Circuit Compaction

The inefficiency of branch-tearing for circuits with many R-loops motivated the development of a more efficient method—circuit compaction. The goal of this method is to formulate a smaller equivalent circuit that may be solved to obtain the R-link currents. The circuit is re-formulated such that original resistor loops are left intact, but Norton-equivalent sub-circuits are substituted for remaining sections of the circuit between the loops. These Norton equivalents are easily constructed during the initial path-trace with R-loops opened (R-links removed). A similar technique was used in the analysis of VLSI power-bus networks [26]. The technique introduced here is similar, with a primary difference being the allowance of voltage sources in the model and the ability to handle zero-resistance tree-branch subsections.

The compaction scheme replaces all long tree-branch sections between successive R-loops with a *super tree-branch* (STB). This is demonstrated by the circuit sub-section shown in Fig. 4. The STB resistance is simply the sum of all resistances between the ends of the STB. The Thevenin voltage is the voltage difference across the STB with all R-links opened. This is illustrated in the last sub-figure in Fig. 4 where the The venin voltage shown is $V(x) - V(y)$ and the Thevenin resistance is $R_1 + R_2$. The total downstream current through the STB is the sum of the capacitor-current sources ($\mathbf{I}_c$) and the unknown R-link currents. The capacitor current ($\mathbf{I}_c$) will be known after path-tracing.

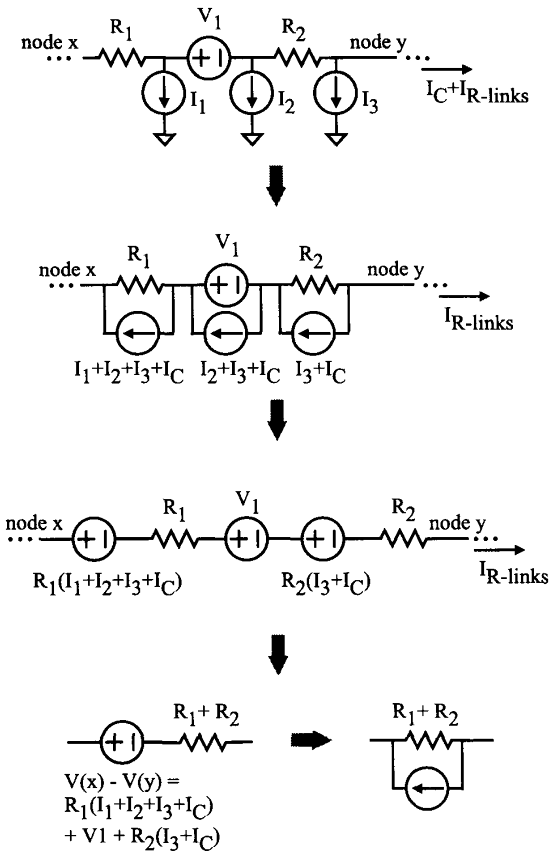
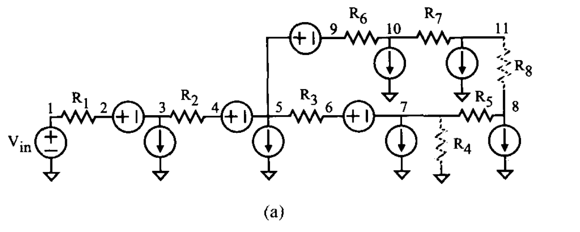


Fig. 4. Compaction of a tree-branch segment (free of R-links) into a super-tree-branch (STB).



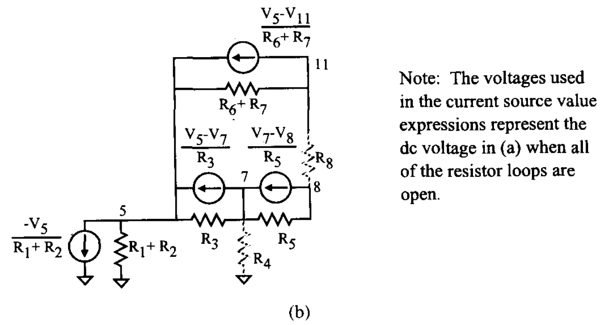

V5-V11 / R6+R7
R6+R7
V5-V7 / R3
V7-V8 / R5
R8
-V5 / R1+R2
R1+R2
R3
R5
R4
(b)
Note: The voltages used in the current source value expressions represent the dc voltage in (a) when all of the resistor loops are open.

Fig. 5. (a) Original dc (AWE) circuit and (b) its compacted equivalent.

A Norton equivalent is used in lieu of the Thevenin model to facilitate the use of nodal analysis to solve the compact circuit. An initial path-trace is required to calculate the Norton resistances. The starting and ending nodes of the STB are retained in the compact circuit equivalent, while the intermediate nodes are deleted. The *folding up* of the current sources shown in Fig. 4 is allowed by source transportation [23] and occurs naturally during the path-trace. Conceptually, the values of the current sources are modified so the net current for each branch is not changed. This figure is used only to illustrate the concept. We do not actually modify the current values, but we jump immediately to the model in the last sub-figure, which is easily obtained during path-tracing.

What may not be immediately obvious is how the compact circuit can be efficiently formulated. The first step is to decide which nodes in the main circuit must be retained in the compact circuit. A node in the main circuit is designated as an *m-node*, while the corresponding node that it maps to in the compact circuit is known as a *c-node*. The most obvious m-nodes in the circuit that must be retained are those with one or more incident resistor links. Other m-nodes that must be retained are those that root two or more subtrees, each containing at least one resistor link. This requirement guarantees that the mutual effects of resistor links on one another are the same in the compact circuit as in the main circuit. An m-node that roots two or more subtrees, where only one subtree contains one or more R-links, *need not* be retained since the effects of these subtrees will be reflected in the Norton currents. Likewise, an m-node that roots two or more subtrees with no resistor links may be discarded. After determining the set of c-nodes, the Norton resistances may be found by summing all resistor values between mapped nodes. This is accomplished with a single depth-first traversal of the spanning-tree.

An example of circuit compaction is illustrated in Fig. 5. It is assumed that Fig. 5(a) is the AWE dc circuit obtained from an RLC circuit. Note that the compact circuit contains only four of the original eleven nodes. Node 5 was retained in the compact circuit since it roots two subtrees that contain R-links. The topology of the compact circuit does not vary between moment generations, but the currents of the Norton sources, shown in Fig. 5(b), must be recomputed between each generation. This is easily accomplished with a single virtual path-trace. A more dramatic example of the compaction is shown in Fig. 6, which illustrates a grid clock line model that was compacted from about 12,000 nodes to 9 nodes.

C. Formulating the Compact Circuit Equations

The dc solution for the compact circuit is obtained by formulating and solving the node voltage equations

$$G_{cc}v_{cc} = i_{cc} \tag{12}$$

where G_{cc} is the $n \times n$ node-conductance matrix for the compact circuit, v_{cc} the unknown n-vector of compact node voltages, i_{cc} is an n-vector that contains the sums of Norton currents at each node, and n is the number of nondatum nodes in the compact circuit.

The use of a Norton equivalent was selected over the Thevenin version since it avoids the introduction of voltage-

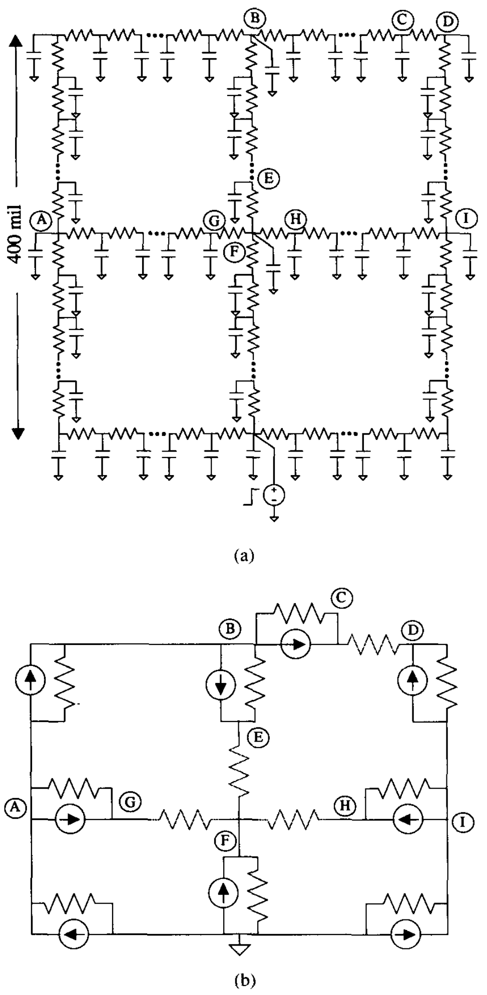


Fig. 6. (a) Grid clock distribution net model consisting of about 12,000 segments and (b) compacted equivalent dc circuit. Shaded resistors are the links.

source current variables in G_{cc}. This is desirable so that G_{cc} will be symmetric and positive-definite, thus allowing a Cholesky decomposition [31] to be performed. This accomplishes the decomposition

$$G_{\mathrm{cc}} = LL^T, \tag{13}$$

which is inherently more efficient than LU decomposition. The higher efficiency of Cholesky decomposition over LU is primarily for two reasons. First, due to symmetry, only about one-half of the usual number of floating-point operations are required. Secondly, no pivoting strategy is required, thus allowing the use of very efficient matrix storage and ordering algorithms. RICE uses the reverse Cuthill-McGee technique [31] for ordering the equations and a vector-based format for storing and manipulating the matrix.

D. Getting the Final Solution

The formulation of the compact circuit and associated equations and decomposition is a one-time cost associated with circuit setup and is performed before any moments are computed. To generate a set of moments, the circuit is initially solved via a virtual path-trace (R-links opened). Next, as illustrated in Fig. 5(b), the Norton currents are calculated, summed to the right-hand-side of (12), and then a Cholesky forward- and back-substitution (using L and L^T in (13)) yields the compact circuit node voltages. The node voltages in the compact circuit correspond to voltages in the original circuit, so they may be used to directly compute the current for each R-loop. These currents are substituted into the original circuit and a second virtual path-trace is performed to produce the final dc solution.

E. Zero-Resistance STB's

There is a problem when zero-resistance STB's are encountered, which occurs when one or more voltage sources and/or inductors is found between successively mapped m-nodes with no intervening resistance. This often occurs at the input of a net where one side of the voltage-input is grounded and the other is connected to a c-node. In such cases, no valid Norton model can be formulated. One alternative is to use the Thevenin models, but this destroys the positive-definite characteristic of G_{cc} and adds an additional variable to the system. A better alternative is to apply voltage-source transportation [23] to *transport* the total voltage of the zero-resistance STB forward into other incident STB's and incident R-links. This transportation causes the downstream Norton models to be modified.

VII. Extension for Floating Nodes & Inductor Loops

There are some interconnect models in which floating nodes or loops of inductors may be encountered. A floating node has no dc path to ground, thus causing a pure cutset of capacitors in the circuit graph. A loop of inductors is caused when a closed path of inductors is specified with no intervening resistance or capacitance. These problems can be solved by the application of charge and flux conservation, respectively [27]. Also, it is desirable to solve these types of circuits in a manner that is decoupled from the solution of resistor loops, thus preserving the efficiency of the R-loop solution.

A. Floating Nodes

For an interconnect model, a floating node suggests that two interconnect nets are capacitively-coupled and one net has no active driver. This may be true for a line that is driven by a three-state device in its high-impedance state or for an interconnect circuit model that is constructed to measure clock feedthrough. The input to the device is still coupled by parasitic capacitance to the inactive output, which is represented by the model shown in Fig. 7(a). The problem with this circuit topology is that when each capacitor is replaced by a current source (as required in AWE), an illegal cutset of current sources is formed, causing one of the capacitor (current) sources to be placed in the spanning-tree. This situation can be resolved by applying charge conservation to the tree-capacitor. This technique for allowing floating nodes

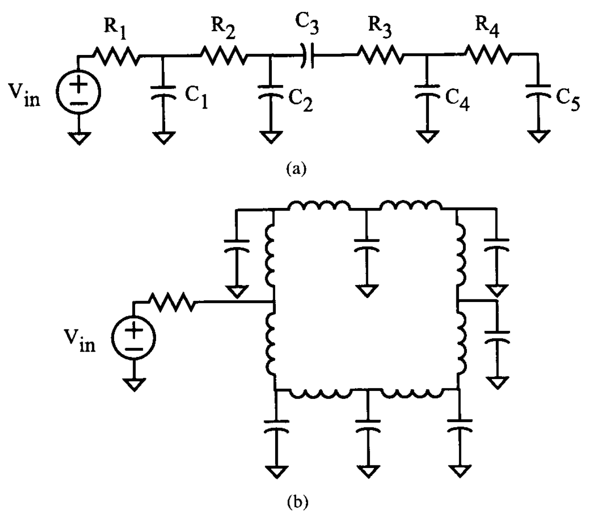


Fig. 7. (a) Interconnect model containing floating nodes dueto a tree-capacitor and (b) interconnect model with a loop of inductors.

in AWE was initially presented in [27] with a path-tracing version presented in [25].

In general, one charge conservation equation must be formulated for each tree-capacitor. So for a model with m tree-capacitors, a linear $m \times m$ system results. For example, consider the example in Fig. 7(a) where C_3 is selected as the tree-capacitor. The voltage across C_3 in the dc equivalent circuit can be resolved by solving the following set of equations:

$$C_3V_{C_3} - C_4V_{C_4} - C_5V_{C_5} = 0$$
$$V_{C_3} + V_{C_4} = V_{\text{in}} - V_{R_1} - V_{R_2} - V_{R_3}$$
$$V_{C_3} + V_{C_5} = V_{\text{in}} - V_{R_1} - V_{R_2} - V_{R_3} - V_{R_4}$$

Note that there are three equations, but only one tree capacitor (i.e., $m = 1$). The last two (KVL) equations are easily eliminated in a path-trace by recognizing that C_4 and C_5 have been replaced by constant current sources, thus the voltage drop from C_3 to C_4 and C_4 to C_5 is constant, regardless of any change in the voltage across C_3. Mathematically, this means that the variables V_{C4} and V_{C5} can be eliminated by rewriting the last two equations above in terms of only V_{C3}, thereby effectively eliminating the equations. This can be accomplished in a single path-trace, thus producing the $m \times m$ system of equations. Remember that the charge conservation equations are written *only* for the tree capacitors. This implies that the tree has already been selected by RICE, therefore it knows which capacitors are in the tree. The tree that is chosen by RICE is arbitrary and unimportant, though we tend to give ground capacitors preferential treatment as links since their voltages map directly to node voltages.

This system is solved by LU-decomposition to solve for the tree-capacitor voltages, formulated and LU-factored prior to computing any moments. To compute a generation of moments requires two (virtual) path-traces. In the initial trace, the tree-capacitors are treated as zero-valued voltage sources. A subset of the results of this path-trace are used to set up the right-hand-side of the m linear equations. Next, these equations are solved via a forward- and back-substitution step, resulting in the actual tree-capacitor voltages. Finally, these voltages are substituted in place of the zero-values and the circuit path-traced a second time to yield the final dc solution.

B. Loops of Inductors

Another situation that can occur in certain interconnect models is a pure loop of inductances. This type of situation has been encountered in the modeling of backplane or pc-board interconnects where two devices drive a line simultaneously and the metal between the drivers is modeled as a lossless inductive line. An example of a model with inductor loops is shown in Fig. 7(b). Even though there is some dc resistance present, it is sometimes considered negligible and excluded from the model. If the resistance were not excluded, loops of inductors would be avoided, therefore this is considered a pathological case. Even so, the solution for loops of inductors is simply the dual of the floating node (tree capacitor) problem. A loop of inductors will cause one of the inductors to be excluded from the spanning-tree (i.e., an inductor link). Since inductors in AWE are modeled as voltage sources, this again presents a problem. There is no well-defined dc solution for a loop of voltage sources. In this case, flux conservation must be applied to resolve the unknown current of each inductor link. As in the capacitor case, for m inductor links there will be an $m \times m$ linear system of equations. These are formulated in a manner similar to the capacitor case, except that we initially replace each link inductor with a zero-valued current source and path-trace the circuit to get the preliminary tree-inductor currents. We use these to formulate the right-hand side of the equations, solve them, and then substitute the resulting link-inductor currents in place of the zero values. Finally, the second path trace yields the final solution.

C. Mixing Loops of Resistors, Loops of Inductors, and Floating Nodes

If a circuit contains a combination of the three types of troublesome elements, then they must be solved in a specific order to guarantee a valid solution. Link inductors must be solved first, followed by tree capacitors, and then resistor loops. A loop of inductors can only contain other inductors and a cutset of capacitors may contain only other capacitors, so these do not affect each other and are not affected by changes of current in loop resistors. This decoupling approach accounts for much of the efficiency and the ability to gracefully degrade as the topology of interconnect models becomes more complex.

VIII. Matching Moments to Approximate Poles

Once the moments have been generated, they must be matched to the reduced-order pole/residue model as discussed in Section III. Various versions of RICE have included three different methods for this problem. Consult [19]–[21] for detailed treatment of the various methods.

A. The Original Technique

In [6] the moment equations in (6) are analyzed by first solving the following set of linear equations: to get the characteristic polynomial

$$\begin{bmatrix} m_0 & m_1 & \cdots & m_{q-1} \\ m_1 & m_2 & \cdots & m_q \\ & \vdots & & \vdots \\ m_{q-1} & m_q & \cdots & m_{2q-2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{q-1} \end{bmatrix} = \begin{bmatrix} -m_q \\ -m_{q+1} \\ \vdots \\ -m_{2q-1} \end{bmatrix} \tag{15}$$

$$a_0 + a_1\tau + a_2\tau^2 + \cdots + a_{q-1}\tau^{q-1} + \tau^q = 0 \tag{16}$$

followed by finding the roots of (16) to yield the dominant time constants (reciprocal poles) of the system. The residues can then be obtained from the dominant poles by solving the equations in (6).

A serious problem with this classical moment-matching approach is that it may yield unstable models for a large number of practical interconnect circuits. In many cases (10–20 percent), positive poles were found, which is invalid for passive interconnect problems. In many other cases, invalid complex (conjugate) dominant poles for RC circuit models were found. These problems are often caused by the inherent instability associated with moment-matching techniques and numerical noise in the moment values.

B. Constrained Optimization

To bypass this problem, one approach was to force the poles to have negative real parts since the circuit models were strictly passive. Initial work along this line lead to the development of a constrained optimization method to match moments to the dominant poles [19]. This was done by first performing a variable substitution, $\tau = -e^x$, where τ is the reciprocal pole. This substitution guarantees that any solution will yield negative poles. After substitution, an unconstrained nonlinear descent algorithm was used to find the values of the new variable.

This technique works well for RC problems where all poles are real, but it was not easily cast into the complex domain for RLC circuits. Furthermore, it was relatively inefficient for high orders of approximation ($>$ 4 poles).

C. Moment Shifting

A second approach to avoiding instability involves a *moment-shifting* algorithm [20] that applies (13) and (6) to a shifted set of moments. It first attempts a solution by solving for the poles as detailed in part A. If it fails to yield a stable result, then the moments are *shifted* such that the lowest-order moment is dropped from the system and the next higher-order moment is added to the system. In [20], this is shown to be equivalent to exciting the circuit with less high frequencies and thereby eliminating the effects of high-frequency poles, which cause instability. One shift typically yields a stable solution, but if does not, shifting is allowed to continue until a solution is found or until shifting is no longer possible (due to numerical problems). Shifting doesn't necessarily guarantee stable results, but so far no circuit models (RC or RLC) have been encountered for which this method produces positive poles. Furthermore, empirical results obtained so far suggests that the method is more accurate than the constrained approximation technique. More details can be found in [20] and [21].

D. Selecting the Order of Approximation

One issue that arises in any moment-matching scheme is the difficulty in quantifying the time-domain waveform errors and selecting an appropriate order of approximation *a priori*. Empirically, for RC interconnect models, it has been observed that second or third order approximations yield results within 1–2 percent of a transient simulation in over 95 percent of all cases. There are similar empirical results for RLC circuits in that no more than 8 poles are required to accurately characterize a large category of RLC circuit models. However, this is very dependent on the nature of the RLC model. Low-loss lines, for example, often require more dominant poles due to the high frequencies present in the response.

For applications that require higher model accuracy, it is possible to select the approximation order based on the highest frequency of interest. This maximum frequency will usually be known, at least conservatively, for any VLSI, multi-chip module or board-level design. This maximum frequency can also be approximately determined from the fastest signal transition-time possible for the given interconnect. It is reasonable to use the fastest signal present on the target design, since only a conservative estimate is required. Once this transition time is known, it may be used to determine the maximum frequency as described in [28].

Armed with this maximum frequency, it is possible to determine the order of approximation. It has been observed that as the order of approximation is increased, higher frequency dominant poles are captured. Therefore, it is reasonable to assume that increases in order may be halted when a new pole that is *greater* in magnitude than the maximum frequency of interest is found. More detail on this technique and related material may be found in [28].

IX. Accuracy and Performance Results

To demonstrate the performance and accuracy, a variety of circuit models and topologies were tested, including examples from industry. Both RC and RLC circuit topologies were considered, including circuits containing resistor loops. When possible, run times from a SPICE simulator [29] are shown for a rough comparison. Plots illustrating the accuracy are revealed for some of the responses. All run-time and memory statistics presented are from a SUN SPARCstation 1 equipped with 16 MB of physical memory. Both large and small circuit models have been tested to show how run time varies with circuit size. Unless otherwise noted, parsing time has been excluded from the run times.

A. RC Model Results

Results from three basic types of RC interconnect models are presented. The first type is the well-known RC-tree topology (RC ladder) shown in Fig. 8(a). The second type, shown

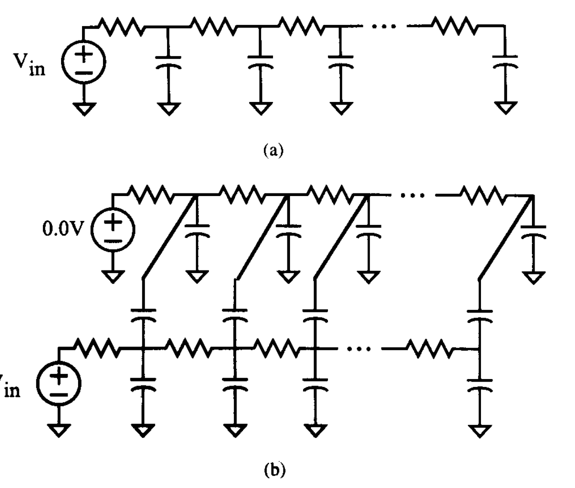


Fig. 8. Examples of the two basic types of RC interconnect circuit models.

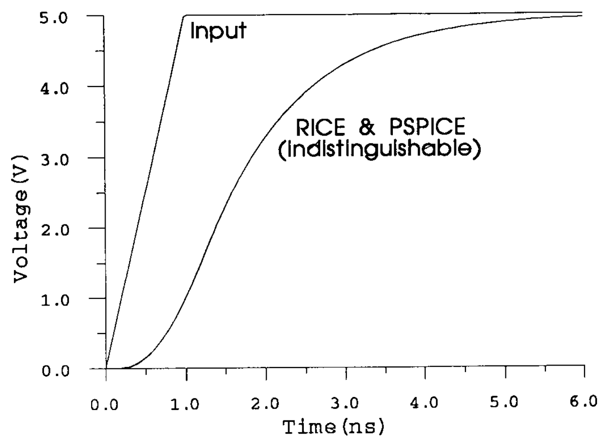


Fig. 9. Comparison of the PSPICE waveform and a second-order RICE approximation for the smaller, coupled RC tree circuit in Table I. The two waveforms are indistinguishable in this plot.

TABLE I
RUN TIMES FOR TWO TYPES OF RC CIRCUIT MODELS. PSPICE RUN TIMES EXCLUDE PARSING

Circuit Type	Total Nodes	Total Branches	RICE CPU-Sec				PSPICE
			Parse	Setup	Compute	Total	cpu-sec
RC Tree	2,000	4,000	0.49	0.11	0.07	0.67	98.15
	8,000	16,000	2.33	0.48	0.28	0.28	1170.67
Coupled RC Trees	16,00	4,000	0.58	0.11	0.07	0.76	97.60
	64,00	16,000	3.38	0.44	0.30	3.02	908.7

in Fig. 8(b), is two RC lines coupled by crosstalk capacitance at every node. Table I displays the run times for each model for both RICE 3.2 [30] and SPICE [29].

Considering only the setup and compute times, the speed-up for the larger model was over $1,200\times$ as compared to the circuit simulation. Furthermore, though memory statistics are not shown, RICE required less than 20 percent of the memory required for a circuit simulation. Also note that, as expected, the run time for the larger model increased linearly as a function of circuit size.

The waveform accuracy is also excellent as evidenced by the plots shown in Fig. 9. This plot compares the transient waveform from PSPICE for a 1-ns input rise-time ramp to the same for RICE. The plots are indistinguishable at this resolution. Delay accuracy better than one percent is common for RC circuit models. In fact, for large circuit models it is imperative to decrease the maximum allowable time step in PSPICE, so the results match those from AWE. This is particularly true for RLC lines where the cumulative numerical integration truncation error exceeds the AWE time-domain error for stiff circuits.

The third type of RC circuit is an IBM clock-line model extracted from a custom 1-m CMOS design. It contains 968 uniform RC (URC) segments with 150 fan-out points along the net, as well as four loops of metal that RICE recognizes as resistor loops. A reduced representation of this net is shown in Fig. 10(a). RICE automatically breaks the URC's into

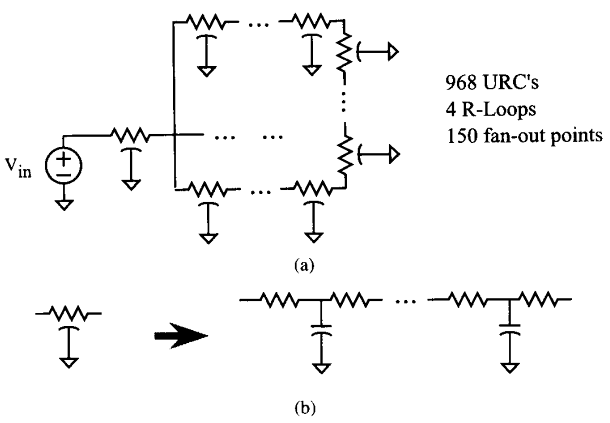


Fig. 10. (a) Reduced representation of an IBM clock distribution net model and (b) replacement used for distributed RC segments.

lumped T-segments as illustrated in Fig. 10(b). The number of segments is a function of signal rise time, but no more than five T segments are required, even for unrealistically fast rise times (e.g., 10 ps) [28]. The plot shown in Fig. 11 compares the RICE generated waveform with that of PSPICE at a selected fan-out point, while Table II compares the run time and memory required for each. Again, the waveforms are indistinguishable in this plot.

RICE was installed into PALADIN, a proprietary delay analysis tool of Texas Instruments. PALADIN analyzes the RC-interconnect models produced by a post-layout extractor and evaluates the delay to all fan-out points. Prior to integration of RICE, PALADIN was requiring over 70 hours of elapsed time to analyze the many RC nets for a large gate array. After integration of RICE, run time was reduced to less than 20 minutes. These are total times that include the database manipulation and other overhead. Moreover, accuracy was significantly improved over the old version of PALADIN. Without RICE, the PALADIN results were off by as much

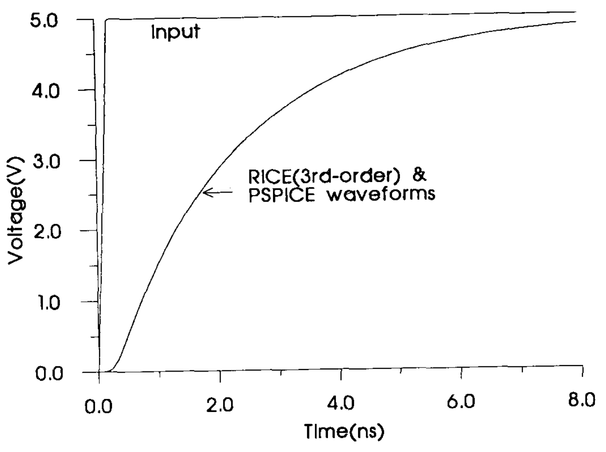


Fig. 11. Comparison of the RICE and PSPICE transient waveforms for the clock line model in Fig. 10.

TABLE II
EXECUTION STATISTICS FOR IBM CLOCK NET SHOWN IN FIG.11.

	CPU Time (sec)	Memory (kB)
RICE 3.1	0.39	401
PSPICE	176.53	1,386

as 20 percent from SPICE; with RICE installed, the results were generally within a few picoseconds of SPICE.

B. Performance on RLC Circuit Models

RICE may be used to analyze RLC circuit models, but the analysis is much more sensitive to the number of poles approximated. RC models rarely require more than two or three dominant poles for high accuracy, independent of the RC topology. In the RLC case, however, parameters such as the line loss, capacitive and inductive coupling, and the transition time of the input, all affect the required approximation order and sometimes require more robust moment-matching techniques. Some RLC lines may be accurately approximated with two or three poles while others may require as many as 14 poles.

Fig. 12 illustrates an RLC model of a section of backplane (Fig. 12(a)) obtained from Raytheon. Fig. 12(b) shows the lumped RLC circuit that was provided by the company to model this section of line. The model consists of a total of 120 RCL segments and ends at an AC terminator. The plot in Fig. 13 reveals a comparison of a RICE third-order approximation with the PSPICE prediction. The waveforms are almost indistinguishable. Total run time (excluding parsing) for RICE was 0.04 cpu-seconds, while PSPICE required 23.05 seconds (over 500x speedup). The poles for the model were obtained via the moment-shifting algorithm [20].

The next RLC model consists of two lines coupled by crosstalk capacitance as shown in Fig. 14. Each line is modeled by 28 π sections with the values shown in the figure. One line is held at logic low, while the other line is switched. The result of the RICE and PSPICE simulations is shown in Fig. 15. In this case, an eighth-order approximation was required to

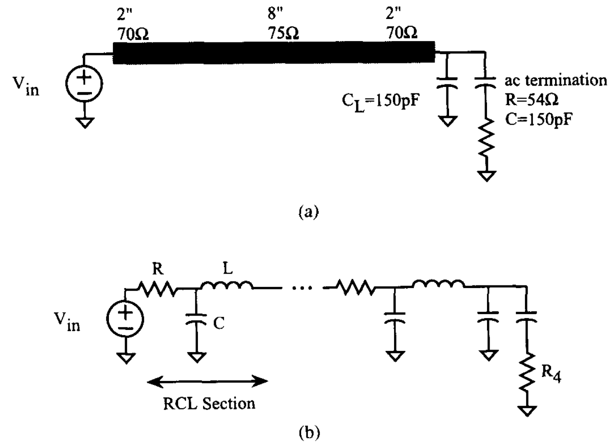


Fig. 12. (a) Section of backplane modeled by 120 RCL sections. (b) Lumped model. Each 2-in. setion is modeled by 20 RCL sections ($R = 0.02\Omega, C = 2.06\text{pF}, L = 1.009\text{nH}$). The 8-in. section is modeled by 80 RCL sections($R = 0.02\Omega, C = 2.31\text{pF}, L = 1.30\text{nH}$).

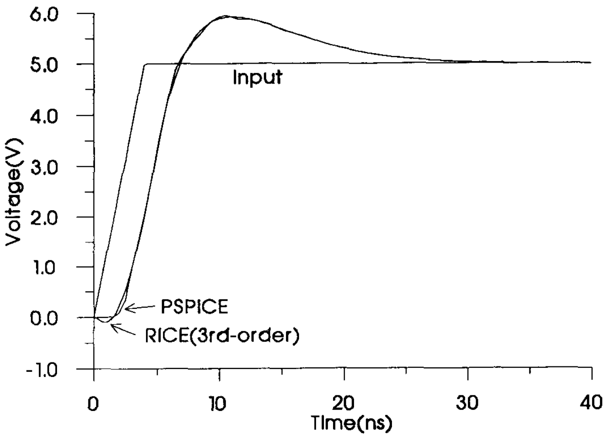


Fig. 13. Comparison of RICE and PSPICE transient waveforms for the RCL lumped circuit model inFig. 12.

adequately match the PSPICE results. RICE correctly predicts signals on the line being switched and the crosstalk on the other line.

C. Performance on Models With Resistor Loops

The last circuit model topology considered is similar to that shown in Figure 8(b), except resistor loops have been added to the model. The first model does not contain any resistor loops, while all other models contain 5000 branches, 2000 nodes, and 500 resistor loops. The placement of the loops in all other models is varied to demonstrate degradation in performance that occurs due to increasing circuit complexity. Run-time results for RICE, PSPICE, and a nodal analysis version of AWE are shown in Table III. The nodal analysis version of AWE is a special version of RICE that performs the dc analyses on the complete uncompacted circuit using the Cholesky decomposition described earlier. The *relative complexity* column is an estimate of the relative topological

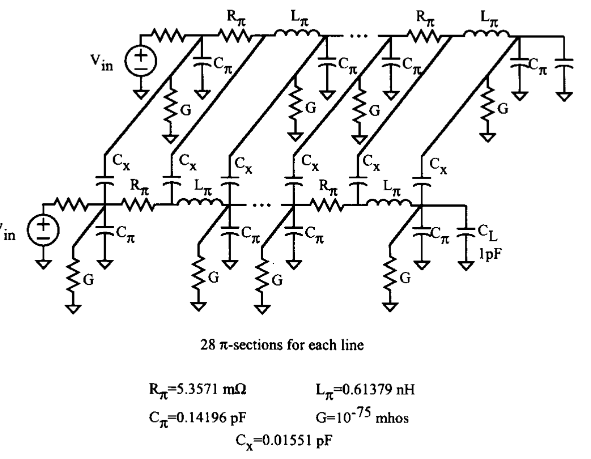


Fig. 14. Two RLCG lines coupled by capacitance at every node. 28 π sections for each line.

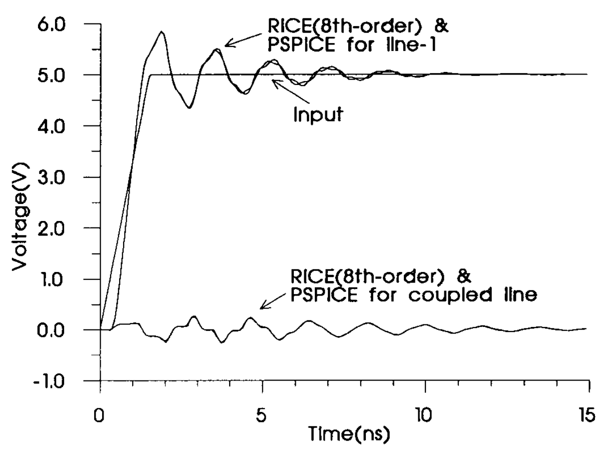


Fig. 15. Comparison of the RICE and PSPICE transient waveforms for the coupled lines in Fig. 14.The response for both lines is illustrated.

complexity of each circuit normalized against the second circuit. The randomness in placement of the loops is increased with each circuit. The second circuit in the table is similar to Fig. 8(b), except it contains regularly-spaced resistors (loops) at 500 nodes. The remaining circuits have the 500 resistors distributed in a more random fashion. The randomness and degree of nonplanarity of the loops is increased with each circuit.

In all cases, RICE's path-tracing and compaction performance is better than the nodal analysis AWE version and the circuit simulation approach. The performance of all degrades as the randomness of the loops increases. This is expected since all are solving a circuit matrix, however RICE's matrix is much smaller since it compacts the circuit prior to formulation. The performance advantage of RICE over the nodal analysis version for RC trees with no loops is always about 2.5 to 1 for all cases we tested. Both nodal analysis and path-tracing for trees can be done with linear

TABLE III
Run-Time Results (in CPU-Secs) for RC Circuits Similar to Topology in Fig 8(b), Except All Circuits Contain 5,000 Branches and 2,000

Circuit	Relative Complexity	RICE w/Path Tracing & Compaction	Nodal Analysis AWE	Transient Analysis
No R-Loops	o	0.19	0.48	126.57
Regular R-loops	1	0.55	0.57	113.15
Both Regular and Random	7	2.06	3.07	98.48
More Random	14	5.00	6.05	3126.82
Very Random	36	14.07	51.77	32137.28

complexity, but nodal analysis involves inherently more floating point operations than path-tracing.

The performance advantage of circuit compaction for a large number of regularly spaced resistor loops is not significant. The reason for this is neither the compacted circuit matrix nor the nodal analysis matrix suffer from significant fill-ins, since the matrices are banded in nature. For this type of regular topology, the performance gained from solving a compacted matrix is offset by the two path traces required in circuit compaction. For more complex topologies there is a significant advantage. This is primarily due to the avoidance of performing operations on fill-ins in the larger circuit matrix. Circuit compaction eliminates many fill-ins. Remember that the nodal analysis used in the nodal analysis version of RICE performs Cholesky decomposition and therefore does not perform any pivoting operations. It is therefore inherently more efficient than a standard matrix decomposition. The initial matrix is obtained from the same tree RICE selected in the path-tracing version.

X. Conclusion

By focusing AWE specifically on the passive interconnect problem, large gains in speed, accuracy, and model stability have been achieved. RICE's application to numerous industry examples have demonstrated its effectiveness for a wide variety of timing analysis problems. Furthermore, while RICE is efficient in comparison to a circuit simulation, it is more important to point out that it produces analytical models for interconnect circuits that completely characterize the transfer and driving-point functions of the circuit. This transfer function may then be used to produce the output waveform with any type of input.

Acknowledgment

The success of RICE is due to the efforts of several people. The theoretical results by Nanda Gopal, Demos Anastasakis, and Seok-Yoon Kim, relating to the production of accurate and stable models, was crucial to the success of RICE. Nanda is also the developer of RICE V4.0. Another important contributor to the overall quality of RICE was John Croix, who installed the initial version of RICE into Texas Instrument's PALADIN timing tool.

REFERENCES

[1] S. Khanna, "Sorting out signal integrity," *Electron. Engin. Times*, pp. 66–69, June 10, 1991.

[2] J. K. Ousterhoust, "CRYSTAL: A timing analyzer for NMOS VLSI circuits," in *Proc. 3rd Caltech Conf. on VLSI*, Mar. 1983, pp. 57–69.

[3] C. J. Terman, "Simulation Tools for Digital LSI Design." Ph.D. thesis, Massachusetts Institute of Technology, Sept. 1983.

[4] N. P. Jouppi, "Timing Analysis and Performance Improvement of MOS VLSI Designs," *IEEE Trans. Computer-Aided Design*, vol. 6, pp. 650–665, 1987.

[5] T. A. Lane, F. J. Belcourt, and R. J. Jensen, "Electrical Characteristics of Copper/Polyimide Thin-Film Multilayer Interconnects," in *Multichip Modules: System Advantages, Major Constructions, and Materials Technologies*. New York: IEEE Press, 1991.

[6] L. W. Nagel, "SPICE2, A computer program to simulate semiconductor circuits," Tech. Rep. ERL-M520, University of California, Berkeley, May 1975.

[7] L. Pillage and R. Rohrer, "Asymptotic Waveform Evaluation for timing analysis," *IEEE Trans. Computer-Aided Design*, pp. 352–366, April 1990.

[8] X. Huang, V. Raghavan, and R. A. Rohrer, "AWEsim: A program for the efficient analysis of linear(ized) circuits," *Int'l Conf. on Computer-Aided Design*, Nov. 1990.

[9] L. T. Pillage, "Asymptotic Waveform Evaluation for Timing Analysis," Ph.D. thesis, Carnegie Mellon University, April 1989.

[10] C. J. Terman, "Simulation Tools for Digital LSI Design," PhD thesis, Massachusetts Institute of Technology, Sept. 1983.

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

[12] R. Putatunda, "Auto-delay: A program for automatic calculation of delay in LSI/VLSI chips," in *Proc. 19th Design Automation Conf.*, June 1981, pp. 616–621, .

[13] A. E. Ruehli, *Circuit Analysis, Simulation, and Design*. Amsterdam, The Netherlands: North Holland, vol. 3, pt. 1, 1986.

[14] Personal communications with developers of proprietary timing analyzers at various companies.

[15] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 155–63, 1948.

[16] C. Chu and M. Horowitz, "Charge-Sharing Models for Switch-Level Simulation," *IEEE Trans. Computer-Aided Design*, vol. 6, no. 6, pp. 1053–1060, 1987.

[17] M. A. Horowitz, "Timing models for MOS circuits," Ph.D. dissertation, Stanford University., Jan. 1984.

[18] Y. Shamash, "Stable reduced-order models using padś-type approximations," *IEEE Trans. on Auto. Control*, vol. 19, pp. 615–616, 1974.

[19] Xiaoli Huang, "Padé Approximation of Linear(ized) Circuit Responses," Ph.D. thesis, Carnegie Mellon University, Nov. 1990.

[20] N. Gopal, C. Ratzlaff, and L. Pillage, "Constrained approximation of dominant time constants in rc circuit delay models," in *Proc. Int'l Mathematics and Computation Symposium* (Invited Paper), July 1991.

[21] D. F. Anastasakis, N. Gopal, S. Y. Kim, and L. T. Pillage, "On the stability of moment-matching approximations in Asymptotic Waveform Evaluation," in *Proc. 29th Design Automation Conf.*, June 1992.

[22] D. F. Anastasakis, "Moment Matching Stabilization in Asymptotic Waveform Evaluation," Master's thesis, University of Texas at Austin, May 1992.

[23] C. Ratzlaff and L. T. Pillage, "The architecture of RICE-3: Fast and accurate evaluation of RLC interconnect models," Tech. Rep. UT-CERC-TR-LTP91-04, Computer Eng. Research Ctr., University of Texas at Austin, July 1991.

[24] C. A. Desoer and E. S. Kuh, *Basic Circuit Theory*. New York: McGraw-Hill, 1969.

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

[26] C. Ratzlaff, "A Fast Algorithm for Computing the Time Moments of RLC Circuits," Master's thesis, University of Texas-Austin, May 1991.

[27] D. Stark and M. Horowitz, "Techniques for calculating the currents and voltages in VLSI power supply networks," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 2, pp. 126–132, 1990.

[28] L. T. Pillage, Xiaoli Huang, and R. A. Rohrer, "Asymptotic waveform evaluation for circuits containing floating nodes," *IEEE Int'l Symposium on Circuits and Systems*, May 1990.

[29] N. Gopal, D. P. Neikirk, and L. T. Pillage, "Evaluating RC-interconnect using moment-matching approximations," in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, pp. 74–77, Nov. 1991.

[30] *PSPICE Users Manual*, Microsim Corporation, Version 4.03, Jan., 1990.

[31] A. Chakraborty and C. Ratzlaff, *RICE User's Guide*, Release 3.2, University of Texas at Austin, Computer Engineering Research Center, April 1992.

[32] S. Pissanetksy, *Sparse Matrix Technology*. London: Academic Press, 1984.

[33] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*. New York: Prentice-Hall, 1974.

C. L. Ratzlaff, photograph and biography not available at the time of publication.

L. T. Pillage (S'87–M'89) received his Ph.D. in electrical and computer engineering from Carnegie Mellon University (CMU) in 1989. Priot to joining CMU, he worked as an integrated circuit and system designer at Westinghouse Research and Development from 1984 through 1986.. During that time he acquired three patents, and in 1986 he was recognized with the corporation's highest engineering achievement award. Since 1989 he has been an assistant professor at the University of Texas at Austin, where he holds a Temple Endowed Faculty Fellowship in Engineering. In 1991 he received the IEEE Transactions on Computer-Aided Design Best Paper Award. Also in 1991, he received a Presidential Young Investigator Award from the National Science Foundation. In 1992, he received a Technical Excellence Award from the Semiconductor Research Corporation (SRC). In 1993 he received a Technical Invention Award from the SRC. Since 1992 he has served as a member of the technical program committee for the International Conference of Computer-Aided Design.