New Implicit Integration Method for Efficient Latency Exploitation in Circuit Simulation

PAUL F. COX, MEMBER, IEEE, RICHARD G. BURCH, PING YANG, SENIOR MEMBER, IEEE, AND DALE E. HOCEVAR, MEMBER, IEEE

Abstract—**To exploit time-domain latency in circuit simulation using direct methods, an accurate, computationally efficient model for slowly moving, dormant portions of the circuit is required. A new, implicit integration method, the overdetermined polynomial method (ODPM), has been developed which permits the formulation of an accurate latent model. Using the ODPM integration method, the Jacobian of a dormant subcircuit need not be reevaluated over a large number of time steps of varying size. An accurate Norton equivalent circuit that emulates the impedance and current characteristics of the subcircuit can be obtained without reevaluation of the Jacobian or nonlinear charge computations. This new approach for utilizing latency produces significant improvements in circuit simulation speed with no decrease in accuracy or generality. We have demonstrated speed gains of 3 to 20*X* over TISPICE for several large circuits.**

I. INTRODUCTION

ACCURATELY verifying the performance and functionality of circuit designs prior to fabrication is an essential task in producing IC's in a timely and cost effective manner. The SPICE circuit simulator has played a key role in making this verification a reality since the 1970's [1]. With the addition of accurate device models [2], and refinements in the algorithms, SPICE has evolved to provide an extremely accurate and widely used simulation tool. Unfortunately, the efficiency of SPICE needs to be much greater than it is presently. Today, cost effective simulation of VLSI circuits is impossible, even with greatly optimized versions of SPICE such as TISPICE. Typically, it is only practical to simulate small, critical sections of a design for a few input patterns. The designer then attempts to determine the functionality of the circuit from these limited circuit simulations and other less accurate simulations, such as timing and logic simulations. More circuit simulation is nearly always desired, but prohibited because of cost and time. It is unlikely that it will ever be practical to simulate entire large chips with the accuracy of SPICE. However, approaches which significantly increase the efficiency of circuit simulation provide the designer with a tremendous advantage, and are highly desired. Advanced techniques for faster and more accurate circuit simulation are being pursued vigorously in many industrial and academic research groups.

Most attempts to improve circuit simulation capabilities are compromises which trade computation speed for accuracy and/or general applicability. The largest compromises are in switch level and timing simulators, [3]–[6], where only a bare minimum of accuracy is maintained, but tremendous speed gains are possible. Other works, [7]–[9], have simplified the model calculations by using table models and interpolation; however, for large circuits the speed gains have been limited since solving the matrix equation dominates the execution time. Mixed-mode simulation, [10], is receiving renewed interest since it can utilize accurate solution techniques on critical sections while solving the remaining circuitry with faster, but less accurate algorithms. Though very advantageous for large circuits with small critical sections, it does not address the problem of accurate solution of large circuits.

One physical aspect of circuits that can be utilized to gain efficiency while incurring little or no loss of accuracy is latency. Time-domain latency occurs any time one section of the circuit is changing at a different rate than the rest of the circuit. The sections, or subcircuits, that are changing slowly need not be simulated as often as those that are changing faster, provided that some approach can be found to model the effect of the "latent" subcircuit on the rest of the circuit. In large circuits, many of the sections are latent for long periods of time during simulation and thus a great potential exists for increasing simulation efficiency. Iterative methods, such as relaxation based methods, [11]–[15], can exploit time-domain latency to attain significant speed gains with SPICE-like accuracy; however, they have proven to be successful only for special classes of circuits and perform poorly for many other types of circuits.

The direct method solution technique, as implemented in SPICE, offers accuracy and robust characteristics for almost all circuits; this makes it highly desirable. There are many techniques which can be utilized to increase the speed of the direct method without any loss of accuracy. Recent work has focused on the utilization of parallel [16]–[20] and vector [21]–[23] processing machines and algorithms. These approaches have demonstrated significant, cost effective improvements in speed. Our approach in the SUPPLE simulator is to utilize the direct method

Manuscript received June 8, 1988; revised November 30, 1988 and May 9, 1989. The review of this paper was arranged by Associate Editor D. Rose.

The authors are with the VLSI Design Laboratory, Texas Instruments, Inc., Dallas, TX 75265.

IEEE Log Number 8929570.

with a block partitioned matrix approach and a combination of techniques to improve efficiency. This bordered block approach makes it possible to simultaneously exploit parallel processing, vector processing, and latency. We have previously reported results on parallel processing and some forms of latency [19], [20]. In this paper we will concentrate on exploiting time-domain latency. A novel method will be presented which allows effective latency exploitation with the direct method solution technique for circuit simulation. In the past, circuit latency exploitation with direct methods has been explored, [24]–[26], though with less success. The key to success is to find an approach to accurately model the effect of the "latent" subcircuit on the rest of the circuit. Techniques developed thus far in the literature have not solved this modeling problem with sufficient accuracy to allow latency to be effectively utilized. Our new technique accurately solves this problem and allows time-domain latency to be effectively used to increase simulation speed significantly.

II. Background Information

During circuit simulation, latent sections of the circuit have smaller changes in their state variables than other, more active sections. Latent sections can take longer time steps than active sections and incur the same truncation error as the active sections. If a uniform time step is used for all sections, then the latent sections will require fewer iterations to solve the nonlinear equations at a time point. Any approach which uses longer time steps for the latent sections of the circuit utilizes "time-domain latency;" while, any approach which uses the increased convergence rate of latent sections is utilizing "iteration latency." Both types of latency are similar; however, iteration latency is easier to identify and utilize. Unfortunately, the gains from iteration latency are very limited. Typically, convergence is achieved in 2–4 iterations in a transient analysis for any circuit section and at least 2 iterations are necessary to determine convergence; thus a latent section saves only 1 or 2 iterations. SLATE [24] was one of the first programs to effectively utilize iteration latency; although SLATE's ability to utilize time-domain latency was severely limited.

Although other schemes are possible, most direct simulation methods which attempt to utilize latency are based on either node tearing or branch tearing. These methods separate the circuit into pieces and form specific structures with the matrix equations. In node tearing [24], the circuit is divided into subcircuits separated by interconnect or tearing nodes (Fig. 1). This leads to the matrix structure shown in Fig. 2, which is called a bordered block diagonal matrix. This is mathematically equivalent to a reordering of the circuit equations. Each diagonal block contains the internal equations for one subcircuit. Entries for the interconnect nodes are found in the border and also in the lower-right submatrix, called the interconnect matrix.

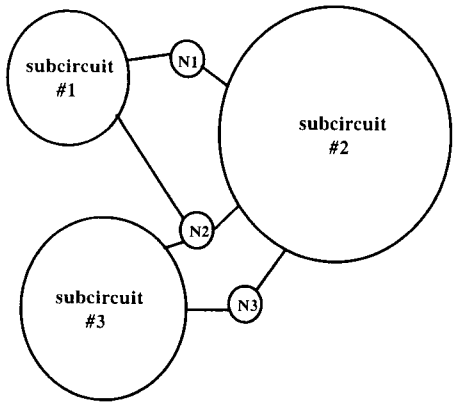


Fig. 1. A circuit can be separated into subcircuits connected by tearing or interconnect nodes. In this example, an arbitrary circuit is divided into three subcircuits with $N1$, $N2$, and $N3$ as interconnect nodes.

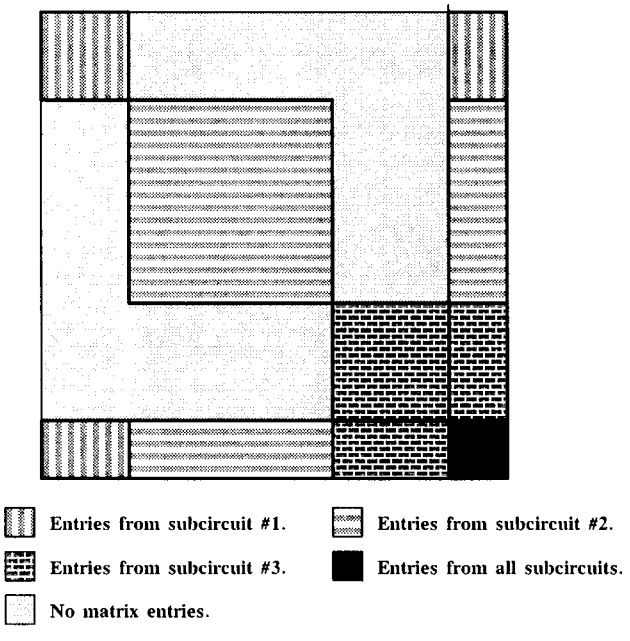


Fig. 2. A bordered block matrix showing individual subcircuit matrices, border, and interconnect matrix.

In the actual implementation of node tearing, separate data structures are used for each subcircuit's internal equations and its contribution to the interconnect equations. The subcircuit matrix structures are shown in Fig. 3. A partial LU decomposition and forward substitution are then used to eliminate terminal subcircuit variables from each subcircuit's contribution to the interconnect matrix. This process converts the subcircuit matrix to a Norton equivalent expressed in external variables only (now stored in the lower right corner of each of the subcircuit structures in Fig. 3). The contributions of each subcircuit to the interconnect are summed to form the interconnect matrix, which is solved (LU factored, forward substituted, and backward substituted) to obtain the interconnect voltages and currents. Each of the subcircuit matrices is then backward substituted to obtain the internal subcircuit variables. Since this is a direct LU matrix so-

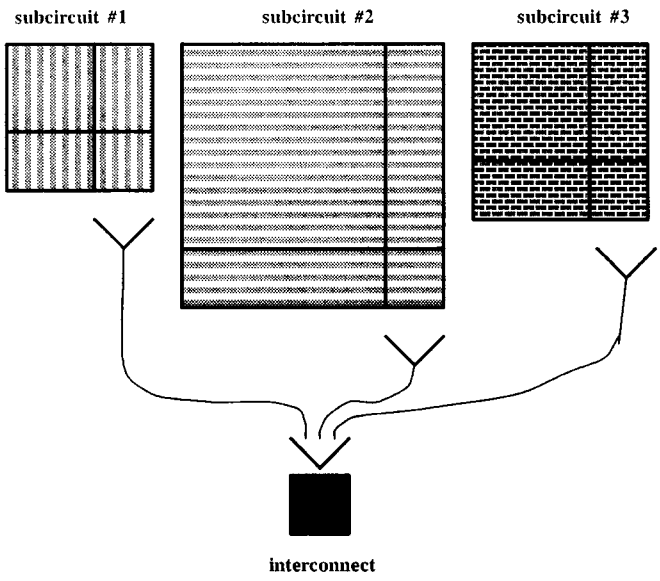


Fig. 3. Partitioned matrix structure. Each subcircuit is reduced to a Norton equivalent circuit in terms of the interconnect nodes. These are combined to form the interconnect which is solved to obtain interconnect voltages. These voltages can now be used during backward substitution to obtain state variables of each subcircuit.

lution method, not an iterative technique, an accurate solution is obtained from one pass of this process. The lack of convergence robustness and slow rates of convergence typical of iterative matrix methods are avoided when using a direct method.

To utilize time-domain latency, the active subcircuits must take shorter time steps than the latent subcircuits. To solve the equations at these time points the contributions to the interconnect matrix from the latent subcircuits are necessary. To gain efficiency from latency, these contributions must be accurately computed in a very cost effective manner. Such a model for computing these terms is the crucial element in developing an effective technique for exploiting latency with direct methods. If none of the state variables in a subcircuit are changing at all, for a period of time, then the subcircuit is completely latent. In this case its contributions to the interconnect matrix are simply those from the last full solution of the subcircuit. This approach for time-domain latency is taken in SLATE; however, the gains are limited since most latent subcircuits are not completely dormant, but are moving slowly. For slowly moving subcircuits, it is not possible to simply use the last full evaluation to provide the necessary contributions; this is very inaccurate. In fact, techniques which simply estimate the contributions by extrapolation introduce far too much error to be useful, especially considering that latent subcircuits may have time steps 10 times larger than the active time step.

SAMSON [25], [26] uses a partitioned circuit approach and attempts to exploit time-domain latency. It models the effects of latent subcircuits with polynomial extrapolation of the voltages of input nodes and the currents of output nodes. This approach is severely limited since polynomial extrapolation is inherently inaccurate for predicting the exponential curves typical in circuits and only applies to a restricted class of circuits.

An ideal latent subcircuit model must provide an accurate Norton equivalent representation of the subcircuit during its dormant period. To exploit time-domain latency, the computational requirements for dormant subcircuits must be much less than for active subcircuits. The model used in SLATE is invalid if any changes occur in either state variables or the time step after the model is evaluated. The method used in SAMSON is accurate for a slowly moving subcircuit only when the latent time step is not much larger than the active time step. The ideal subcircuit model must be valid for both changes in time step and moderate changes in state variables.

III. Ideal Latent Subcircuit Model

To develop the concepts used by our latent subcircuit model, it is necessary to lay a little groundwork. In transient circuit simulation, the electrical circuit is modeled by a system of nonlinear algebraic-differential equations:

$$F(x, \dot{x}, t) = 0, \qquad x(0) = x_0 \tag{1}$$

where x is an m-dimensional vector of voltages and currents, $\dot{x}$ is its time derivative, t is time, and x_0 is the value of x at $t = 0$. In transient circuit analysis, a numerical solution of (1) is required at a sequence of times, t_1, t_2, t_3, $\cdots$, t_n. At each time point, $t = t_i$, the value of x is represented by $x(t_i)$, (or x_i), and the corresponding approximate numerical solution by $\hat{x}(t_i)$ (or $\hat{x}_i$). At each of these time points, three major numerical operations are performed to obtain the desired solution. First, a suitable discretization function is used to replace the time derivatives $\hat{\dot{x}}_{n+1}$ in (1) with a divided difference approximation:

$$\hat{\dot{x}}_{n+1} = g_k(\hat{x}_{n+1}). \tag{2}$$

At time t_{n+1}, (1) then becomes

$$F(\hat{x}_{n+1}, g_k(\hat{x}_{n+1}), t_{n+1}) = 0. \tag{3}$$

The discretization step converts the system of nonlinear algebraic-differential equations to a system of nonlinear algebraic equations.

An iterative solution technique is then used to solve the discretized circuit equations (3). Most direct method circuit simulation algorithms utilize the Newton–Raphson (NR) method for solution. Each iteration of the NR method is described by the equation [1], [25]:

$$\nabla F(\hat{x}^i_{n+1})\hat{x}^{i+1}_{n+1} = -F(\hat{x}^i_{n+1}) + \nabla F(\hat{x}^i_{n+1})\hat{x}^i_{n+1} \tag{4}$$

where $\nabla F(\hat{x}^i_{n+1})$ is the Jacobian matrix of F, and i is an iteration counter. Starting with an arbitrary guess $\hat{x}^0_{n+1}$ and subject to specific conditions, (4) will converge to the desired solution, $\hat{x}_{n+1}$. For linear networks, only one iteration is required for convergence.

After convergence is obtained with the NR method, an estimate must be made of the error obtained in the numerical solution. Because truncation of the series approximations used in obtaining g_k is the principal source of

error in obtaining $\hat{x}_{n+1}$, this phase of the simulation process is usually termed truncation error analysis. From an estimate of the truncation error, the solution is either accepted at this time point, or it is rejected and a smaller time step taken. The truncation error estimate is also used to select an appropriate time step for the next solution.

The Jacobian of (3) can be shown to be a dc Jacobian, which contains dc conductance terms, added to the product of a scalar $f(h, IM)$ and a matrix C of branch capacitances values. The scalar $f(h, IM)$ is a function of the time step, h, and the integration method (IM) used for the discretization step. For any particular implicit integration method:

$$\dot{x}_{n+1} = \Sigma_{i=-1}^{p} a_i x_{n-i} + \Sigma_{i=0}^{p} b_i \dot{x}_{n-i}$$
$$f(h, IM) = a_{-1}. \tag{5}$$

At many time points in the transient simulation, the Jacobian would not need to be reloaded, if the time step did not change. Both the dc conductance and the capacitances are sufficiently linear that the matrix would converge in a single Newton iteration. These sections of the circuit are latent and would have small truncation error and large time steps. If a method can be found to use the previous matrix formulation at new time points for these latent subcircuits, an ideal latent subcircuit model would result and significant savings from time-domain latency would be possible.

Since the dc Jacobian terms and the capacitances are constant for this case, reusing the Jacobian requires that $f(h, IM)$ be held constant for changing h. Unfortunately, traditional integration methods require a one-to-one correspondence between $f(h, IM)$ and h. It is possible to use two integration methods that have a common $f(h, IM)$ at different values of h; however, this is of limited usefulness since only one such point exists for each pair of integration formulas. To reuse the Jacobian for any value of h, a new integration method is required that allows $f(h, IM)$ to be held constant. This will allow the Jacobian to be reused whenever the old linearization is sufficiently valid.

The requirements for keeping $f(h, IM)$ constant in a dormant subcircuit model are different from the problem addressed by Jackson and Sacks-Davis [27]–[29]. They have described an integration procedure which permits $f(h, IM)$ to be held constant by interpolating the state variables at past times to match the time intervals required for integration using standard integration methods. This method should be useful for incremental changes in the global time step. However, in exploiting multirate behavior in circuit simulation, the effective time step for a dormant subcircuit must often be varied by more than an order of magnitude. The integration method described in the following section does not require interpolation to obtain the past values of the state variable and should be more efficient. In addition, the new method uses the past, calculated values of the dependent variable directly and eliminates possible interpolation errors. To minimize integration errors it is important to take full use of the most recent calculated values of the state variables.

IV. Overdetermined Polynomial Integration

This section will present an integration method that allows the value of $f(h, IM)$ to be held constant while h is varied. This is an important innovation in the modeling of latent subcircuits. Any multistep integration formula, with constant h, can be expressed as

$$I_{n+1} = \Sigma_{i=-1}^{p} a_i Q_{n-i} + \Sigma_{i=0}^{p} b_i I_{n-i} \tag{6}$$

where Q is the charge and I is the current, the time derivative of charge, the two variables of primary interest in circuit simulation. If the $2p + 3$ coefficients (a_{-1}, a_0, $\cdots$, a_p, b_0, b_1, $\cdots$, b_p) are chosen so that the formula is exact for any polynomial of degree k, then $k + 1$ of the coefficients are derived formulas that are determined by the time steps used in the integration. The remaining coefficients are fixed when the formula is derived. Traditionally in the Adams–Bashforth algorithms or the Gear algorithms, these $2p + 2 - k$ coefficients are set to zero to minimize the complexity of the formula and the amount of data required to implement it. In the A-contractive methods [30]–[33], more than $k + 1$ non-zero coefficients are utilized to improve the stability and damping properties of the numerical integration. We propose a different choice. Since $f(h, IM)$ is a_{-1}, set a_{-1} to a constant, non-zero value. Now set $2p + 1 - k$ coefficients to zero and solve for the remaining $k + 1$ coefficients in terms of a_{-1}. We call this approach "overdetermined polynomial integration method" (ODPM).

Now we will derive ODPM_2, an ODPM which utilizes charge at the three previous time points:

$$I_{n+1} = a_{-1} Q_{n+1} + a_0 Q_n + a_1 Q_{n-1} + a_2 Q_{n-2}. \tag{7}$$

The goal here is to hold a_{-1} constant for any given time step $h = (t_{n+1} - t_n)$. If a_{-1} is held constant, then the coefficients for the ODPM at time t_{n+1} can be determined by requiring the formula to be exact to a second-order polynomial and solving for the undetermined coefficients. The coefficients are:

$$a_2 = \left(\frac{(2h + h_1) - a_{-1}(h^2 + hh_1)}{(h_1 + h_2)h_2} \right)$$
$$a_1 = \left(\frac{a_{-1}h - a_2(h_1 + h_2) - 1}{h_1} \right)$$
$$a_0 = -a_2 - a_1 - a_{-1}$$

where $h_1 = (t_n - t_{n-1})$, and $h_2 = (t_{n-1} - t_{n-2})$. This formula is an implicit integration formula since both sides of the equation depend on Q_{n+1}. While this integration method is valid for any constant value for a_{-1}, the use intended presents a logical choice. If a step h' is predicted for the latent subcircuit, a_{-1} should be chosen so that the

coefficients a_{-1}, a_0, a_1, and a_2 match the second-order Gear's coefficients when $h = h'$. This requires that a_{-1} be the normal second-order Gear's coefficient for h' and h_1:

$$a_{-1} = \left(\frac{2h' + h_1}{(h')^2 + h'h_1}\right).$$

The other coefficients will automatically go to the second-order Gear's coefficients when $h = h'$.

To be useful to form a latent subcircuit model, the accuracy of the integration formula is important. The accuracy of any integration formula is measured by its truncation error [34]. The formulas for calculating truncation error are well presented by Chua and Lin [35]. Their formulas can be directly utilized if the integration method is written in a more traditional form:

$$Q_{n+1} = \Sigma_{i=0}^{p} A_i Q_{n-i} + h \Sigma_{i=-1}^{p} B_i I_{n-i}. \quad (8)$$

If h is held constant so that t_{n+1}, t_n, t_{n-1}, and t_{n-2} are evenly spaced and the coefficient a_{-1} is set to an arbitrary constant, the formula derived for ODPM_2 can be rewritten in this form. The new integration coefficients A_0, A_1, A_2, and B_{-1} can be calculated; all other coefficients are zero:

$$A_0 = \left(\frac{-a_0}{a_{-1}}\right) = 3 - \left(\frac{5}{2a_{-1}h}\right)$$

$$A_1 = \left(\frac{-a_1}{a_{-1}}\right) = \left(\frac{4}{a_{-1}h}\right) - 4$$

$$A_2 = \left(\frac{-a_2}{a_{-1}}\right) = 1 - \left(\frac{3}{2a_{-1}h}\right)$$

$$B_{-1} = \left(\frac{1}{a_{-1}h}\right)$$

$$Q_{n+1} = A_0 Q_n + A_1 Q_{n-1} + A_2 Q_{n-2} + hB_{-1} I_{n+1}. \quad (9)$$

The standard form for expressing the truncation error (see [35, p. 458]) of an integration method:

$$Q_{n+1} = \Sigma_{i=0}^{p} A_i Q_{n-i} + h \Sigma_{i=-1}^{p} B_i I_{n-i} \quad (10)$$

which is exact to any polynomial of degree less than or equal to k, given by

$$\epsilon_T = C_k \hat{X}^{(k+1)}(\hat{\tau}) h^{k+1} = O(h^{k+1}) \quad (11)$$

where

$$-ph < \hat{\tau} < h$$

and

$$C_k = \left(\frac{1}{(k+1)!}\right)\left((p+1)^{k+1} - \left(\sum_{i=0}^{p-1} A_i (p-i)^{k+1} + (k+1) \sum_{i=-1}^{p-1} B_i (p-i)^k\right)\right).$$

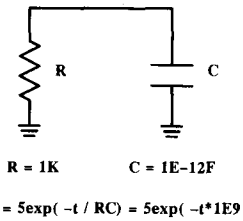


Fig. 4. Simple *RC* circuit used for analysis of truncation error.

Utilizing this theorem, it is easy to calculate the truncation error formula for ODPM_2:

$$Q_{n+1} = a_0 Q_n + a_1 Q_{n-1} + a_2 Q_{n-2} + hb_{-1} I_{n+1}. \quad (12)$$

In this method $p = 2$ and $k = 2$. The general form of the truncation error formula is

$$\epsilon_T = C_2 \hat{X}^{(3)}(\hat{\tau}) h^3 = O(h^3) \quad (13)$$

and the value of C_2 is

$$C_2 = 1 - \frac{11}{6a_{-1}h}.$$

A simple example will provide another method of looking at the truncation error of ODPM. The truncation error for a simple *RC* circuit (Fig. 4) is examined for the ODPM_2 just derived. In this example, h_1, h_2, and h' were set to the same value ($h' = 0.1\ RC = 0.1$ ns). The value of t_{n-2}, t_{n-1}, t_n, and $t_n + h'$ were arbitrarily chosen to be 0.8, 0.9, 1.0, and 1.1 ns. The value of h was then varied from 0.001 to 0.10 ns and the value of V_{n+1} was calculated using ODPM_2 and a simple nodal approach. Correct values for V_n, V_{n-1}, and V_{n-2} were obtained analytically and used to predict V_{n+1} using ODPM_2 with a_{-1} calculated for second-order Gear's with constant time step, $h_1 = h_2 = h'$. V_{n+1} was predicted for many values of t_{n+1} between 1.0 and 1.1 ns. The answers obtained for V_{n+1} as h is stepped from 0.001 to 0.1 ns are not saved and do not affect the next computation of V_{n+1}. This is exactly the way that ODPM_2 will be used to provide a dormant subcircuit model. The state variables calculated with ODPM_2 will not be saved, since the values at t_n, t_{n-1}, and t_{n-2} are used each time the values at t_{n+1} are calculated for a different t_{n+1}. Fig. 5 compares the estimates using ODPM_2 with the correct answer and the answer if non-constant time step, second-order Gear's integration was used to predict V_{n+1}. It is nearly impossible to differentiate the three curves; Fig. 6 plots the percentage error of ODPM_2 and second-order Gear's. The error with ODPM is greater than the Gear's error for h equal to small fractions of h'. In normal use, a time step h' would have been selected for which the second-order Gear's truncation error was satisfactory. Since the error at any h less than h' is never much larger than this error, ODPM_2 is sufficiently accurate to be used as part of a model for a latent subcircuit.

The choice of coefficients used when deriving the ODPM just presented was purely arbitrary. To illustrate

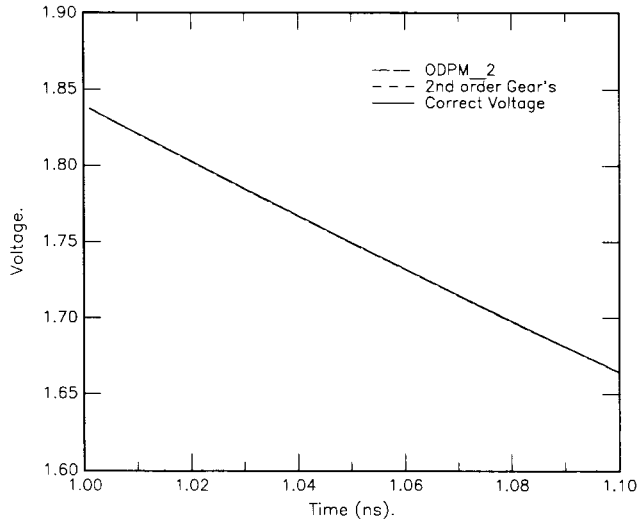


Fig. 5. Comparison of estimated voltage for the *RC* circuit shown in Fig. 4 using ODPM_2 integration and second-order Gear integration with the true voltage obtained analytically.

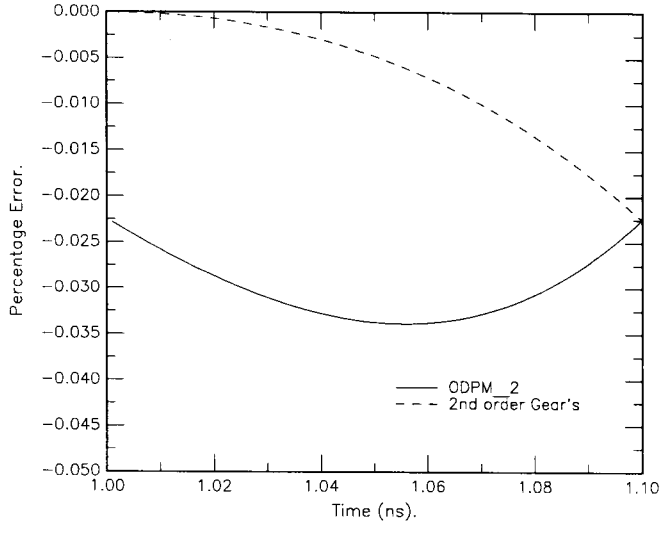


Fig. 6. Percentage error in voltages of Fig. 5.

this, we will derive ODPM_1, a method exact for first-order polynomials, using previous current information:

$$I_{n+1} = a_{-1}Q_{n+1} + a_0Q_n + b_0I_n \tag{14}$$

If we hold a_{-1} constant and require the formula to be exact to first order, the values for a_{-1}, a_0, and b_{-1} are easy to determine:

$$a_0 = -a_{-1}$$

$$b_0 = 1 - a_{-1}h$$

where $h = (t_{n+1} - t_n)$. These can be rewritten in standard form:

$$A_0 = 1$$

$$B_{-1} = \left(\frac{1}{a_{-1}h}\right)$$

$$B_0 = 1 - \left(\frac{1}{a_{-1}h}\right)$$

$$Q_{n+1} = A_0Q_n + hB_{-1}I_{n+1} + hB_0I_n. \tag{15}$$

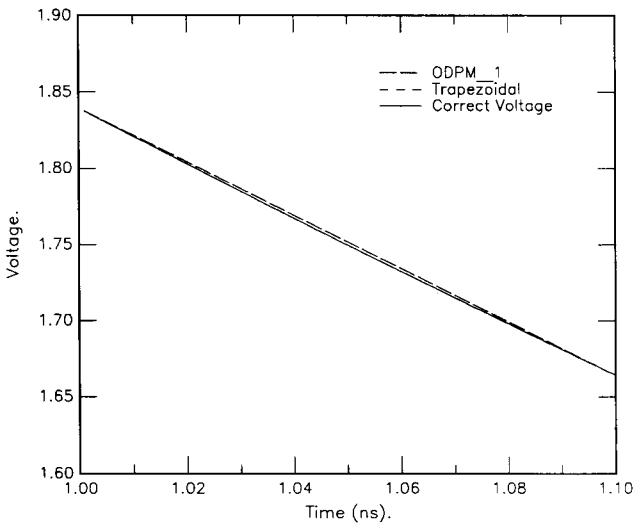


Fig. 7. Comparison of estimated voltage for the *RC* circuit shown in Fig. 4 using ODPM_1 integration and trapezoidal integration with the true voltage obtained analytically.

The logical choice for a_{-1} would require the equation to match trapezoidal integration for the predicted dormant timestep h':

$$a_{-1} = \frac{2}{h'}.$$

The truncation error can be calculated with the same procedure used to calculate it for the ODPM_2 method. In the ODPM_1 method, $p = 0$ and $k = 1$. The truncation error formula is

$$\epsilon_T = C_1\hat{X}^{(2)}(\hat{\tau})h^2 = O(h^2) \tag{16}$$

and the value of C_1 is

$$C_1 = 0.5 - \frac{1}{a_{-1}h}.$$

When h equals h', C_1 is zero and the method will be exact for second-order polynomials.

To examine the accuracy of this approach, we apply it to the same simple *RC* circuit (Fig. 4) that we used for ODPM_2. Similar voltage and error curves are shown in Figs. 7 and 8. This formula has a much larger truncation error in the center of the dormant region. This is expected since it is only exact for first-order polynomials at this point. Since $h\lambda = 0.1$, we would expect the error in this region to be approximately one order of magnitude greater than the trapezoidal error. The results substantiate this expectation.

V. Implicit Subcircuit Model

In circuit simulation, the state variables for each of the subcircuits are often changing at very different rates and require different time steps to maintain acceptable truncation errors. To exploit time-domain latency, we would like to simulate each subcircuit with its own, independent time step. The global simulation time step must be determined by the most active part of the circuit. However, the simulation step for less active subcircuits would be the

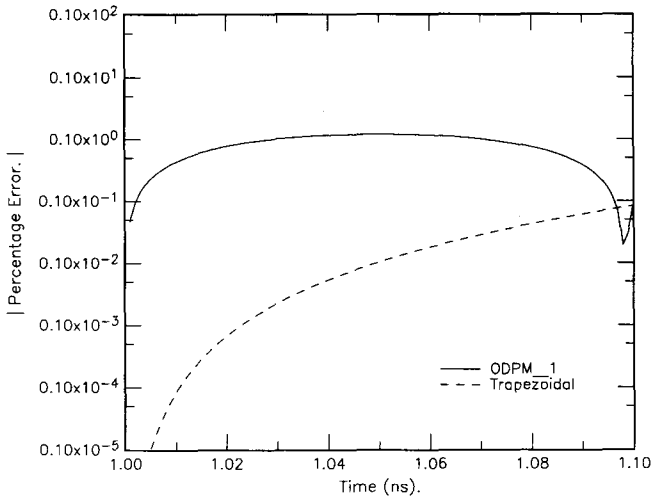


Fig. 8. Percentage error in voltages of Fig. 7.

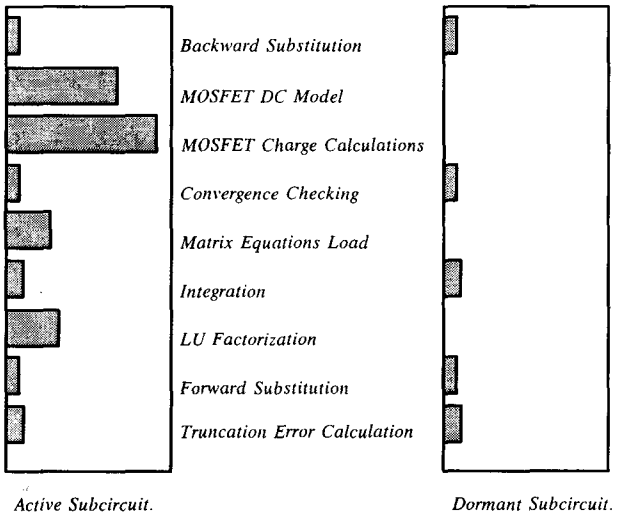


Fig. 9. This figure illustrates the savings of the dormant subcircuit model when compared to the simulation time for an active subcircuit. The bars illustrate the approximate relative work of the different tasks required to simulate the subcircuit. In most cases, a dormant subcircuit can be simulated, using the dormant subcircuit model, with less than 10 percent of the effort required for an active subcircuit.

step required to maintain its own local truncation error within specified tolerances. With the iterative methods, this subcircuit independent time step control can be easily implemented. However, the resulting simulators lack the robustness and generality of SPICE; moreover, cumbersome schemes are needed to allow the backup of time points to maintain simulation accuracy [15]. However, with the direct methods, no good schemes were previously available to implement this subcircuit independent time step control. Overdetermined polynomial integration provides us with the tools to develop an accurate model for a latent, slowly moving subcircuit at intermediate global time points using the direct methods. This model is actually the Norton equivalent of a valid linearization of the subcircuit for the latent time step, (h). As an actual equivalent circuit, the model is sensitive to changes in the operating conditions of the circuit.

To comprehend the value of this model, it is necessary to review the steps required to generate the Norton equivalent of an active subcircuit, which describes the contribution of that subcircuit to the interconnect matrix. During each NR iteration:

1) linearized conductance and currents must be calculated for all nonlinear devices at the present operating point,

2) the contributions from all devices must be loaded into the Jacobian and the right hand side of the matrix,

3) the charge must be integrated and the appropriate currents summed into the right hand side,

4) the subcircuit matrix must be partially LU factored and forward substituted to remove internal subcircuit variables from the equations which represent the subcircuit's contribution to the interconnect.

After the contributions from each subcircut have been combined to form the interconnect matrix, it is solved to determine the interconnect voltages and currents. Each subcircuit matrix is then backward substituted to determine the subcircuit's state variables. The branch voltages and currents are then checked for each nonlinear device to determine if the Newton iterations have converged. If not, then another iteration of this process is required. A typical time chart for one iteration of a circuit containing over 2400 MOSFET's and 1400 equations is shown on the left of Fig. 9.

Overdetermined polynomial integration allows the use of the same Jacobian to model subcircuits at dormant time points. First, the Jacobian is either obtained from the last iteration at the previous subcircuit time point or created for the time step h' predicted for the latent subcircuit. At intermediate, latent time points, the effort to generate the Norton equivalent of the subcircuit is greatly reduced. To generate this implicit subcircuit model it is only necessary to:

1) integrate the node charges using ODPM integration and update the right hand side of the matrix equation,

2) forward substitute the subcircuit matrix using the new current vector.

Since the linearity of all subcircuit elements has not changed, it is not necessary to evaluate any nonlinear models, load the Jacobian, or LU factorize it. Two inexpensive steps, integration and forward substitution, are all that is required to generate the Norton equivalent of a latent circuit.

These equivalent circuits are then combined to form the interconnect matrix and solved to determine the interconnect voltages. An inexpensive step, backward substitution, is then done for all subcircuits, except the truly inactive ones, to obtain internal node voltages. These voltages are not saved; however, they must be monitored to determine if the linearization of any device has changed and the subcircuit must be activated. Another advantage of this approach is that since all the external and internal

node voltages are computed, truncation error analysis can be performed at every global time point. If a subcircuit is active, the truncation error is used to determine if the current time point is acceptable and to estimate a new time step for the subcircuit. If a subcircuit is slowly varying, the truncation error is used to determine if the current time point is acceptable, but is not used to compute the next time step. If the truncation error of any subcircuit is unacceptable, the time point is rejected, and the simulation backs up only one time point. Since the truncation error of all subcircuits was acceptable at the previous time point, no "unzipping effect" effect occurs when the truncation error of a slowly varying subcircuit is unacceptable. Most approaches must back all subcircuits up to the last time when the dormant subcircuit was active, to assure overall accuracy. This eliminates the most persistent problem with time-domain latency schemes. The speed advantage of this dormant subcircuit model is shown on the right of Fig. 9.

Our implicit subcircuit model has some similarity to the NR method described by Bank and Rose [36] at the Newton iteration level, but is much more general. In the NR method, at the Newton iteration level, the Jacobian is approximated by reusing the previous one. With the traditional integration methods, the Jacobian changes with a changing step size. Thus at a new time point, this approximation is not very good and the evaluation of a new Jacobian is normally needed to ensure rapid convergence. In our approach, a new integration scheme, ODPM, is developed to allow the Jacobian to stay constant with a changing step size. However, in the region for which our dormant subcircuit model is intended, the changes in voltage are small enough that the linearization of circuit elements is not significantly altered. The Jacobian terms and all terms in the right hand side of the matrix equation are constant with the exception of currents estimated from the change in charge at each node. A single integration step is all that is required to update the right-hand side.

VI. Stability

One important issue when considering any simulation approach that modifies the integration algorithms is stability. One of the ODPM methods described in Section IV will be shown in this section to be stiffly stable. In addition, the A-contractive methods can be considered as a special class of ODPM; their stability properties are well known [30]–[33]. Therefore, there are at least a few OPDM's which can be used in our implicit subcircuit model with guaranteed stability. Other ODPM integration methods derived by using all possible combinations of defining coefficients cannot be generally guaranteed to be stable. Following the stability analysis procedure described by Chua and Lin [35], it can be shown that some choices of ODPM methods may have limited ranges of stability and would not be useful as general integration schemes. However, we contend that they can still be used in our dormant subcircuit model without causing stability problems for the following reason.

If an integration method is stable, then errors made at one time step do not get amplified, but actually decrease with time. It can be shown that some ODPM alternatives are not stable by this definition. This can only be a problem if the state variables for the dormant subcircuit were saved and used to compute values at later time points; the errors could then be amplified. However, as we discussed when considering truncation error, our dormant subcircuit model is only used to provide a Norton equivalent of the dormant subcircuit to the rest of the circuit; the results affected by ODPM are not saved. If the dormant subcircuit is used to provide a Norton equivalent subcircuit at t_{n+h} and t_{n+2h}, the same information about previous charges and currents that are used for the ODPM integration at t_{n+h} is used at t_{n+2h} (provided that the dormant subcircuit was not active between t_{n+h} and t_{n+2h}). Any error in a charge, Q_{n+h}, cannot be amplified by ODPM at t_{n+2h}, since Q_{n+h} is discarded and not used during integration at t_{n+2h}. Since variables in the active portions of the circuit are all that is saved and they are using stable integration approaches, the use of ODPM in our dormant subcircuit model does not affect the stability of the circuit. Some simple *RC* circuits have been analyzed to verify this result.

Stability of ODPM is not necessary for our dormant subcircuit model. However, if desired, stable ODPM methods are available for the dormant subcircuit model. Here we will show that one of the ODPM approaches derived in Section IV is stiffly stable in a significant region of operation. One approach, ODPM_1, was derived that used previous current information:

$$Q_{n+1} = A_0 Q_n + hB_{-1} I_{n+1} + hB_0 I_n \qquad (17)$$

where

$$A_0 = 1$$

$$B_{-1} = \left(\frac{1}{a_{-1}h}\right)$$

$$B_0 = 1 - \left(\frac{1}{a_{-1}h}\right)$$

$$a_{-1} = \frac{2}{h'}.$$

This approach is stiffly stable as long as the time step, h, since the dormant subcircuit was last simulated does not exceed the time step, h', that was used in formulating the dormant subcircuit's Jacobian.

This property can be shown by following the approaches taken by Chua and Lin [35]. Assume a simple exponential, where the current (I) is the derivative of the charge (Q) with respect to time:

$$I = -\lambda Q, \quad Q(0) = 1.$$

The equation

$$Q_{n+1} = A_0 Q_n + hB_{-1} I_{n+1} + hB_0 I_n \qquad (18)$$

can be recast as the following difference equation:

$$(1 + \sigma B_{-1}) Q_{n+1} - (A_0 - \sigma B_0) Q_n = 0 \qquad (19)$$

where

$$\sigma = h\lambda.$$

The solution to this equation is given by

$$Q_n = c_1 z_1^n$$

where z_1 is the root of the polynomial equation

$$P(z) = (1 + \sigma B_{-1}) z - (A_0 - \sigma B_0) = 0. \qquad (20)$$

To be stable for a certain value of σ, $|z| \leq 1$ must be true. In general σ can be complex and for the approach to be stiffly stable several criteria must apply. One good quick stability test is to consider the stability of the integration approach for $\sigma = 0$ and $\sigma = +\infty$.

If $\sigma = 0$ then

$$z - 1 = 0$$

$$z = 1.$$

This implies that this version of ODPM is stable at $\sigma = 0$ for all choices of a_{-1}.

If $\sigma = +\infty$ then

$$B_{-1} z + B_0 = 0$$

$$z = \left(\frac{-B_0}{B_{-1}}\right).$$

Using the values calculated for B_{-1} and B_0

$$z = \left(\frac{\left(\frac{1}{a_{-1}h}\right) - 1}{\left(\frac{1}{a_{-1}h}\right)}\right)$$

$$z = 1 - a_{-1}h.$$

To be stable $|z| \leq 1$, which implies

$$0 \leq a_{-1}h \leq 2.$$

Since the Jacobian was set up for the trapezoidal method with a time step of h'

$$0 \leq \left(\frac{2h}{h'}\right) \leq 2$$

$$0 \leq h \leq h'.$$

Thus the approach is stable for $\sigma = +\infty$ as well as $\sigma = 0$ as long as $h \leq h'$. Furthermore, the approach is stiffly stable as long as $h \leq h'$. This can be verified by plotting the regions of stability for complex σ as a function of h. Stability plots for all values of a complex σ with four values of h are shown in Fig. 10.

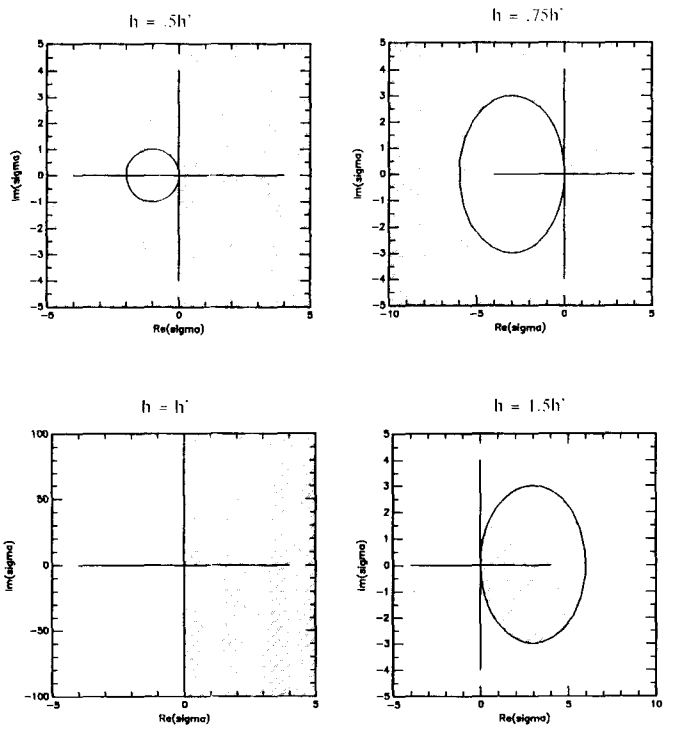


Fig. 10. This figure shows the stability properties ODPM_1. This method is exact for first-order polynomials. For $h < h'$, the algorithm is stiffly stable. For $h > h'$, it is not stiffly stable. For $h = h'$, the stability properties are exactly this for trapezoidal integration, since the formulas are equivalent. For the four choices of h presented here, the shaded regions indicate where the algorithm is stable.

VII. Results

This implicit subcircuit model for exploiting time-domain latency has been implemented in SUPPLE [19], [20], [37], a general purpose circuit simulation program based on TISPICE. Benchmark results using several large digital, memory, and analog circuits have shown that SUPPLE achieved a 3–20X speed up, while producing accurate results equivalent to TISPICE. In the following, we will describe these results. For the ease of demonstration, we will first start with two simple RC circuit examples to examine the performance of this model, then we will discuss the benchmark results obtained from the large industrial circuits.

The first simple RC circuit we will examine is shown in Fig. 11, which has but one time constant. If the circuit is divided into two subcircuits, with C_1 and R_1 in the first subcircuit and C_2 and R_2 in the second, the time step predicted by each subcircuit would depend solely on the value of C_1 and C_2. If $C_1 = nC_2$, where n is any integer, then the time step chosen for subcircuit two can be expected to be approximately n times longer than the time step for subcircuit one. We will use this to determine how long the latent time steps should be for subcircuit two when subcircuit one is simulated with constant time step h. During the latent time steps, subcircuit two will be approximated using the implicit subcircuit model. To perform this simulation, we will use difference equations for this particular circuit. The circuit can be discretized (Fig. 12),

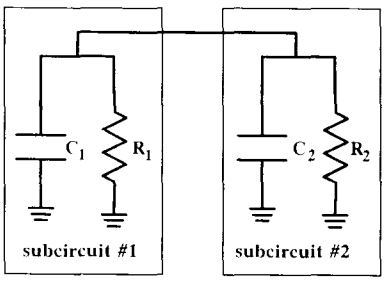


Fig. 11. Single time constant circuit partitioned into two subcircuits.

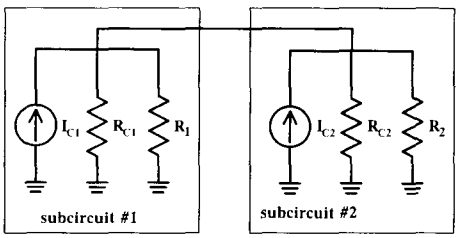


Fig. 12. Discretized version of circuit in Fig. 11.

and the nodal equation written:

$$\left(\left(\frac{1}{R_1}\right) + \left(\frac{1}{R_{C_1}}\right) + \left(\frac{1}{R_2}\right) + \left(\frac{1}{R_{C_2}}\right)\right) V_{n+1} = I_{C_1} + I_{C_2} \tag{21}$$

where R_{C_1}, R_{C_2}, I_{C_1}, and I_{C_2} are determined by the integration method. If second-order Gear's integration is used for subcircuit one and ODPM_2 is used for subcircuit two:

$$I_{C_1} = \left(\frac{2C_1}{h}\right) V_n - \left(\frac{C_1}{2h}\right) V_{n-1}$$

$$I_{C_2} = -XC_2 V_n - YC_2 V_{n-1} - ZC_2 V_{n-2}$$

$$R_{C_1} = \left(\frac{2h}{3C_1}\right)$$

$$R_{C2} = WC_2.$$

The simulation was performed with $C_1 = nC_2$, for $n = 1$, 4, 8, and 16. These values were chosen to allow the latent subcircuit's time step h' to be h, $4h$, $8h$, and $16h$. The value of h was held constant so that $h\lambda = 0.01$, a reasonable value for simulation. The voltage curves for these simulations are contained in Fig. 13. Since it is nearly impossible to distinguish the curves, Fig. 14 was included to plot the error of each simulation. The error in voltage is around 0.5 percent for normal second-order Gear's integration ($h' = h$) and is never greater than one precent for ODPM. A small increase in error was expected since overdetermined polynomial integration has a slightly higher error coefficient and the latent subcircuit has more error due to its increased time step.

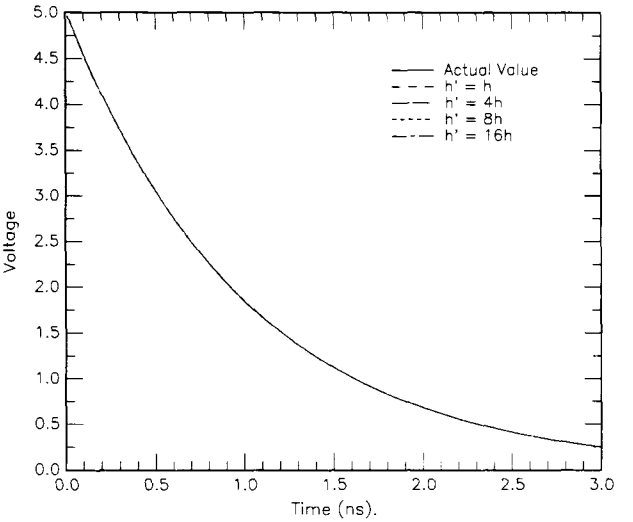


Fig. 13. Simulated voltages curves for circuit shown in Fig. 11 with 4 values of dormant time step h'.

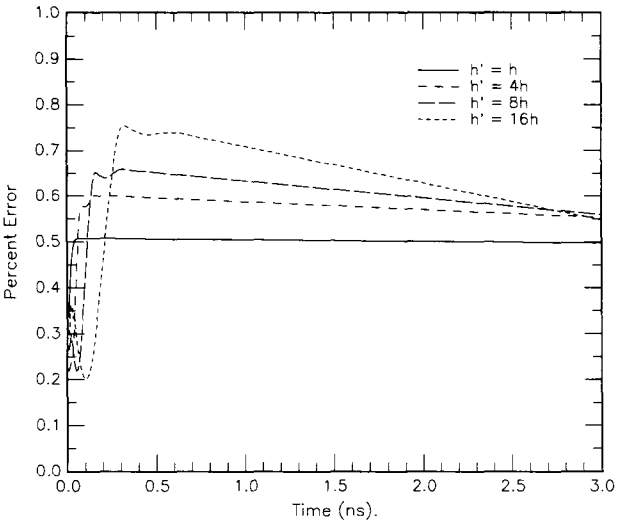


Fig. 14. Error curves for voltages curves in Fig. 13.

The overall increase in error is not significant because the error in the active subcircuit dominates. This is clearly demonstrated in Fig. 15. When the entire circuit is allowed to move at the large time step of the dormant subcircuit, the error increases by an order of magnitude. If these individual RC subcircuits were actually Norton equivalent subcircuits of two equal, large subcircuits, then the majority of simulation time would be required to generate the Norton equivalents. If the time step of the entire circuit was allowed to increase, the truncation error would increase to an unacceptable level; however, if only the latent subcircuit's time step was increased, the maximum speedup due to time-domain latency exploitation, roughly half the execution time savings, could be obtained with no significant error increase.

A slightly more complex example makes this generalization more clear. A two node RC circuit (Fig. 16) has similar capacitance, but the voltage at each node changes at a different rate. This can be discretized (Fig. 17) to

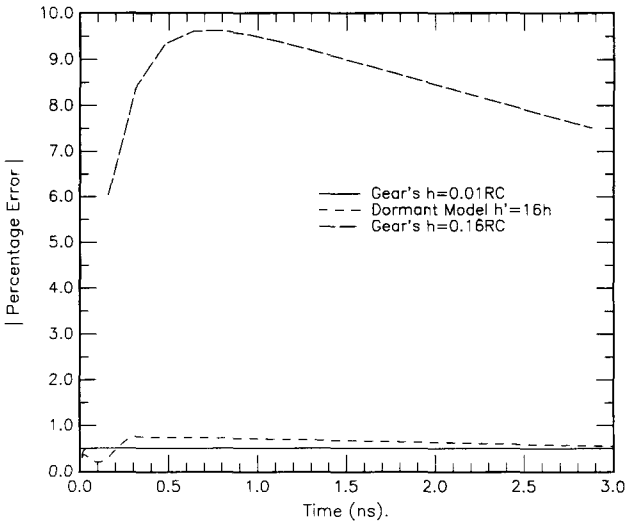
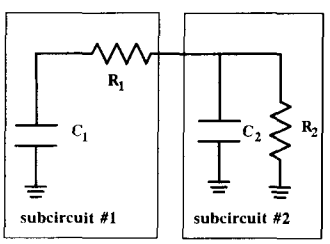


Fig. 15. Comparison of error in dormant model to error if large time step was used for simulation of active subcircuit.



$R_1 = R_2 = 2K$

$C_1 = 2E\text{-}12 \qquad C_2 = 1E\text{-}12$

Fig. 16. More complex, two node *RC* circuit partitioned into two subcircuits.

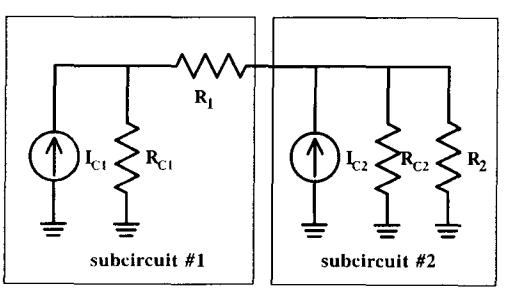


Fig. 17. Discretized version of two node *RC* circuit in Fig. 16.

obtain nodal difference equations similar to the approach used above. The implicit subcircuit model can then be used to predict subcircuit one in the region where it is moving slower than subcircuit two. The circuit was examined for a particular choice of capacitor and resistance values where subcircuit one is moving much slower than subcircuit two. Figs. 18 and 19 compare the voltage curves for various latent time steps for subcircuit one with the voltage curve if the overall timestep is increased to match the largest latent time step. The value of the latent time step makes no discernible difference in the voltage curves for node 2; however, moving the entire circuit at the large time step introduces maximum errors approaching 100 percent. The most interesting curves are those for node 1. Even though the latent subcircuit is only simulated at each latent time point, its voltage is accurate for large latent time steps. However, when the whole circuit is simulated for large time steps, the error in the voltage at node 2 causes a dramatic increase in the error of the voltage at node 1.

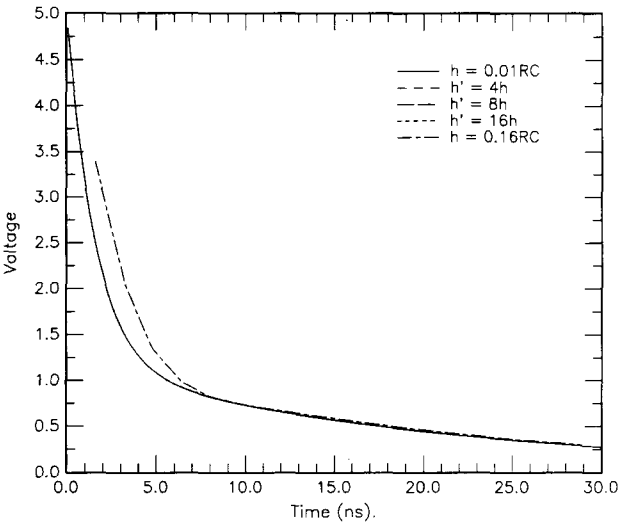


Fig. 18. Transient voltage of Node 2 of the two node *RC* circuit as function of h'.

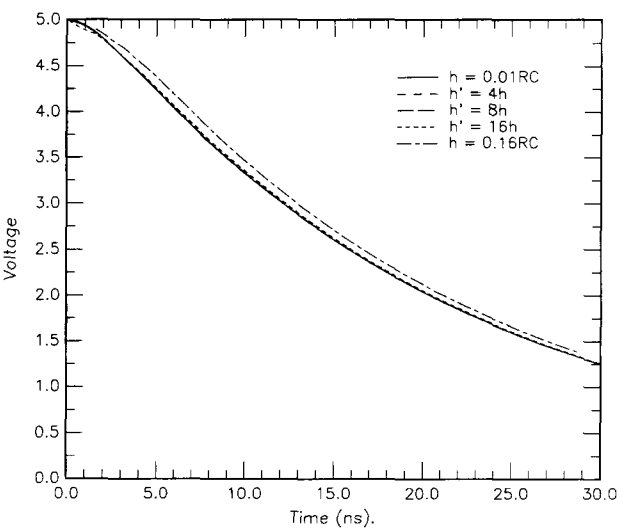


Fig. 19. Transient voltage of Node 1 of the two node *RC* circuit as function of h'.

Now let us examine the benchmark results obtained from large industrial circuits. The circuits tested include digital, memory, and analog circuits. The voltage waveforms obtained with SUPPLE are virtually identical to those obtained with TISPICE. One set of simulated waveforms are given in Fig. 20. The circuit example used in this figure is ROW15, a large section of a 4-Mbyte dRAM circuit. Waveforms from two nodes are specifically chosen to demonstrate the accuracy and robustness of our approach. The first node has a strong coupling from nearby subcircuits and has a very noisy waveform with glitches

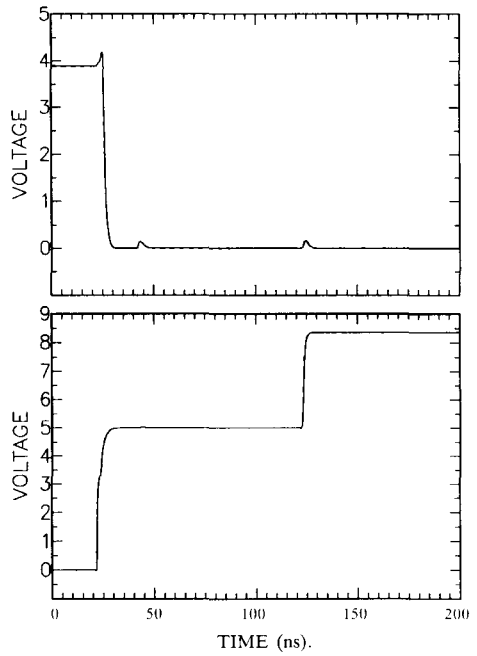


Fig. 20. This figure illustrates the close agreement in results between SUPPLE (the dashed waveform) and TISPICE (the solid waveform) for two nodes of *Row15*, a large section of a memory circuit. These nodes are of particular interest since they do not have simple logical waveforms. The first node is a noisy logic signal and the second is a bootstrap node. Even for these complex waveforms, the results of SUPPLE and TISPICE are indistinguishable.

and crosstalks. The second node is a bootstrap node with ungrounded capacitors and normally presents a major problem for circuit simulators based on iterative methods. It can be seen from Fig. 20 that SUPPLE accurately simulated this circuit and the waveforms of SUPPLE and TISPICE are indistinguishable. However, utilizing the new implicit subcircuit model, the present version of SUPPLE executes 17 times faster than TISPICE.

Results for some of the large circuits we have tested are given in Table I. The reported execution time was obtained with all Fortran source code on a CONVEX C-240 system. The CONVEX C-240 is an advanced parallel/vector processor but neither the parallel nor vector features were utilized in obtaining these timings. ADDENA and TREE are digital circuits. The 34-bit ALU circuit is a critical path from a large LISP microprocessor. ROW15, as stated earlier, is a large section of a 4-Mbyte dRAM circuit. An inverter chain example is also included for comparison purpose. In general, SUPPLE runs about 3 to $20X$ faster than TISPICE (Table I). The two programs, TISPICE and SUPPLE, are identical except for the utilization of latency by SUPPLE; the speed-up factor for a given circuit is determined by the amount of time-domain latency which can be exploited. For example, the speed up for the 34-bit ALU is about $3X$; it is a dynamically clocked circuit with very little time-domain latency. In obtaining these results, the circuits were partitioned using both user-defined subcircuits and an automatic circuit partitioner which we have described elsewhere [37]. Since a direct solution method is used in SUPPLE, the partitioning can effect the efficiency of latency exploitation, but not the accuracy of the results, or the convergence of the matrix solution. A number of issues such as the effect of circuit partitioning on efficiency, truncation error analysis, and individual subcircuit time step control are important considerations in circuit simulation. However, these are complex problems worthy of papers in their own right and will not be discussed here.

TABLE I
COMPARISON OF EXECUTION TIMES FOR TISPICE AND SUPPLE. SUPPLE RUNS FROM 3 TO 20 TIMES FASTER THAN TISPICE DEPENDING ON THE DEGREE OF LATENCY AVAILABLE IN THE CIRCUITS. ALL SIMULATION TIMES ARE IN SECONDS AND CIRCUIT SIZE IS SPECIFIED IN # MOSFET'S/# EQUATIONS

Circuit	Circuit Size	TISPICE	SUPPLE	GAIN $\left(\frac{\text{TISPICE}}{\text{SUPPLE}}\right)$
addena	301 / 249	136.6	19.0	7.1X
tree	419 / 490	61.0	17.8	3.4X
34bit alu	1839 / 1014	733.0	252.9	2.9X
row15	2483 / 1589	2908.1	173.0	16.8X
inv chain	256 / 136	21.7	1.1	19.7X

VIII. CONCLUSION

Accurate and efficient algorithms for circuit simulation are essential for VLSI design. There exists a critical need for simulation techniques which produce a significant improvement in execution speed while maintaining the accuracy and robustness of SPICE. This paper addressed the topic of exploiting time-domain latency within the circuit while using the traditional direct method solution techniques. Our simulator used a block partitioned matrix approach to allow latency to be exploited on a subcircuit by subcircuit basis. The key element for success was in our development of an important new model for latent subcircuits which is both efficient and accurate. Dramatic reductions in simulation time, by factors of $3X$ to $20X$ over TISPICE, have been demonstrated on large circuits from commercial VLSI designs, without sacrificing accuracy. This is significant because it is the first method which allows time-domain latency to be exploited to produce substantial speed improvement while maintaining accuracy for general circuits.

The new model provided an accurate Norton equivalent of the latent subcircuit's impedance and currents which was used in the circuit equations at intermediate time points. These calculations are computationally efficient as compared to performing a complete evaluation of the latent subcircuit at these dormant time points. The gain in efficiency depends upon the amount of latency present in the circuit; circuits with much latency execute very efficiently. While those with little latency will show limited gains, the technique will never degrade the efficiency. Our method is based on an actual equivalent circuit that can be constructed with a minimal effort using a novel inte-

gration scheme, overdetermined polynomial integration or ODPM. In ODPM, a_{-1}, the coefficient multiplying Q_{n+1}, is held constant for any value of time step, h. This allows the same Jacobian to be used for multiple time steps. As presented, the Jacobian is constructed for the projected time step of a slowly moving section of the equations, and used to model the latent section at intermediate time steps. Another valid approach is to use the Jacobian from the previous time point until the latent section is ready to be simulated again.

One very important advantage of our model is that the internal node voltages of all dormant subcircuits are computed at each Newton iteration. These are monitored to assure that the dormant model is valid. Furthermore, truncation error is checked at each time point to assure that the results is acceptable. If a dormant subcircuit is discovered to have unacceptable truncation error, the simulation can be backed up a single global time step with assured accuracy. Other approaches cannot monitor the truncation error of dormant subcircuits. If a dormant subcircuit has unacceptable truncation error when it is finally simulated, the entire simulation must be backed up to the last time the dormant subcircuit was fully simulated. This "unzipping" of previous work can quickly override the gains of latency. Our approach defeats this nagging problem of time-domain latency simulation programs.

Two specific formulations for ODPM methods were derived and analyzed in this paper. It is possible for many other formulations to be derived. An ODPM method can be derived for any formulation that is normally kth order by holding a_{-1} constant and requiring the approach to be accurate to $(k-1)$th order.

The implicit subcircuit model has been implemented successfully in SUPPLE and has been shown to produce equivalent results to TISPICE. There are a few important implementation issues when applying the model to nonlinear devices. They are extensive and will be covered in later papers. The purpose of this paper is to introduce the concept and potential of ODPM's and the implicit subcircuit model.

References

[1] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Memo ERL-M520, Univ. Calif., Berkeley, Electronic Research Lab., May 1975.

[2] P. Yang and P. K. Chatterjee, "SPICE modeling for small geometry MOSFET circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-1, pp. 169-182, Oct. 1982.

[3] B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS—An MOS timing simulator," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 901-910, Dec. 1975.

[4] R. E. Bryant, "MOSSIM: A switch level simulator for MOS LSI," in *Proc. 18th Design Automation Conf.*, June 1981.

[5] C. J. Terman, "RSIM—A logic-level timing simulator," in *Proc. IEEE Int. Conf. Computer Design*, pp. 437-440, Nov. 1983.

[6] V. B. Rao and T. N. Trick, "Switch-level timing simulation of MOS VLSI circuits," in *Proc. Int. Symp. Circuits and Systems*, pp. 229-232, June 1985.

[7] J. Barby, J. Vlach, and K. Singhal, "Optimized polynomial splines for FET models," in *Proc. IEEE Symp. Circuits and Systems*, pp. 1159-1162, May 1984.

[8] T. Shima, H. Yamada, and R. L. M. Dang, "Table look-up MOSFET modeling system using a 2-D device simulator and monotonic piecewise cubic interpolation," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 121, 126, Apr. 1983.

[9] W. M. Coughran, E. Grosse, and D. J. Rose, "Variation diminishing splines in simulation," *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 2, pp. 696-705, Apr. 1986.

[10] C. F. Chen, C-Y Lo, H. N. Naham, and P. Subramaniam, "The second generation MOTIS mixed-mode simulator," in *Proc. 21st Design Automation Conf.*, pp. 10-17, June 1984.

[11] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-1, pp. 131-145, July 1982.

[12] D. Dumlugol, H. J. DeMan, P. Stevens, and G. G. Schrotten, "Local relaxation algorithms for event driver simulation of MOS networks including assignable delay modeling," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 193-202, July 1983.

[13] R. A. Saleh, "Iterated timing analysis and SPICE1," Memo. UCB/ERL M84/2, Electronics Research Lab., Univ. Calif., Berkeley, Jan. 1984.

[14] A. R. Newton and A. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation," *IEEE Trans. Computer-Aided Design*, vol. 3, pp. 308-331, Oct. 1984.

[15] J. White and A. Sangiovanni-Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits.* Hingham, MA: Kluwer, 1986.

[16] J. T. Deutsch, "Algorithms and architecture for multiprocessor-based circuit simulation," Ph.D. dissertation, Univ. Calif., Berkeley, Memo. No. UCB/ERL M85/39, May 1985; *Electron. Design*, pp. 153-168, Sept. 1984.

[17] G. Bischoff and S. Greenberg, "CAYENNE: A parallel implementation of the circuit simulator SPICE," in *Proc. Int. Conf. CAD*, Nov. 1986.

[18] G. K. Jacob, A. R. Newton, and D. O. Pederson, "Direct-method circuit simulation using multiprocessors," in *Proc. Int. Symp. Circuits Systems*, May 1986.

[19] P. Cox, R. Burch, and B. Epler, "Circuit partitioning for parallel processing," in *Proc. Int. Conf. CAD*, Nov. 1986.

[20] P. Cox, R. Burch, D. Hocevar, and P. Yang, "SUPPLE: Simulator utilizing parallel processing and latency exploitation," in *Proc. Int. Conf. CAD*, Nov. 1987.

[21] A. Vladimirescu and D. O. Pederson, "Circuit simulation on vector processors," in *Proc. IEEE ICCC '82*, pp. 172-175, 1982.

[22] B. Greer, "Converting SPICE to vector code," *VLSI Systems Design*, vol. VII, pp. 30-35, Jan. 1986.

[23] F. Yamamoto and S. Takahashi, "Vectorized LU decomposition algorithms for large-scale circuit simulation," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 232-239, July 1985.

[24] P. Yang, I. N. Hajj, and T. N. Trick, "SLATE: A circuit simulation program with latency exploitation and node tearing," in *Proc. IEEE Int. Conf. Circuits Computers*, Oct. 1980.

[25] K. A. Sakallah and S. W. Director, "SAMSON2: An event driven VLSI circuit simulator," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 668-685, Oct. 1985.

[26] K. A. Sakallah, "Polynomial terminal equivalent circuits as dormant models in event driven circuit simulation," in *Proc. Int. Conf. CAD*, pp. 179-181, Nov. 1985.

[27] G. K. Gupta, R. Sacks-Davis, and R. E. Ticher, "A review of recent developments in solving ODEs," *Computing Surveys*, vol. 17, no. 1, pp. 5-47, Mar. 1985.

[28] K. R. Jackson and R. Sacks-Davis, "An alternative implementation of variable step-size multistep formulas for stiff ODEs," *ACM Trans. Math. Soft.*, vol. 6, no. 3, pp. 295-318, 1980.

[29] R. Sacks-Davis, "Fixed leading coefficient implementation of SD-formulas for stiff ODE," *ACM Trans. Math. Soft.*, vol. 6, no. 4, pp. 540-562, 1980.

[30] O. Nevanlinna and W. Liniger, "Contractive methods for stiff differential equations," Part I, *BIT*, vol. 18, pp. 457-474, 1978; Part II, *BIT*, vol. 19, pp. 53-72, 1979.

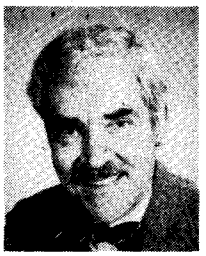
[31] W. Liniger, "The A-contractive second-order multistep formulas with variable steps," *SIAM J. Numer. Anal.*, vol. 20, pp. 1231-1238, 1983.

[32] A. Ruehli, P. Brennan, and W. Liniger, "Control of numerical stability and damping in oscillatory differential equations," in *Proc. IEEE Int. Conf. Circuits Computers*, G. Rabbat, ed. vol. 1, pp. 111-114, 1980.

[33] W. Liniger, F. Odeh, and A. Ruehli, "Integration methods for the solution of circuit equations," in *Circuit Analysis, Simulation and*

Design. Amsterdam, The Netherlands: Elsevier Science, 1986, pp. 235-279.
[34] C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations.* Englewood Cliffs, N.J.: Prentice-Hall, 1971.
[35] L. O. Chua and P-M Lin, *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques.* Englewood Cliffs, NJ: Prentice-Hall, 1975.
[36] R. E. Bank and D. J. Rose, "Global approximate Newton methods," *Numer. Math.*, vol. 37, pp. 279-295, 1981.
[37] P. Cox, R. Burch, and P. Yang, "A dormant subcircuit model for maximizing iteration latency," in *Proc. Int. Conf. CAD*, pp. 438-441, Nov. 1988.

*



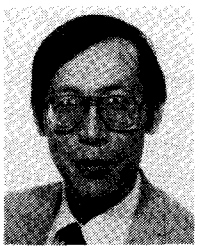
Paul F. Cox (M'87) received the Ph.D. degree in inorganic chemistry from the University of Texas.

In 1967, he joined the Central Research Laboratory, Texas Instruments, where he was involved in infrared spectroscopy, automated characterization system development, and real-time video image processing. In 1979, he became a Senior Member of the Technical Staff, VLSI Design Laboratory, Texas Instruments, where he is currently involved in the development of algorithms for computer-aided circuit design. His interests include parametric yield estimation, new circuit simulation algorithm development, and parallel processing.

*

Richard Burch received the B.S. degree in physics and the M.S.E.E. degree from the University of Oklahoma in 1984 and 1986, respectively. He is currently working towards the Ph.D. degree in electrical engineering at the University of Illinois at Urbana-Champaign, Urbana, IL.

In 1984, he joined Texas Instruments Semiconductor Process and Design Center as a member of the technical staff. His interests include research into circuit simulation, power and current estimation, and electromigration failure prediction.



Ping Yang (M'81-SM'86) received the B.S. degree in physics from the National Taiwan University, Taipei, Taiwan, in 1974, and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois, Urbana, in 1978 and 1980, respectively.

In 1980, he joined the Central Research Laboratory, Texas Instruments, Inc. He has coauthored two books, and presented or published more than 80 articles in international scientific and technical journals and conferences. His research interests include computer-aided design, circuits and systems, large-scale integrated circuits, and solid-state electronics.

Dr. Yang is a member of Phi Kappa Phi and the IEEE Circuits and Systems AdCom.

*

Dale E. Hocevar (S'79-M'82) received the B.S. degree in electrical engineering from the University of Tulsa, OK, and the M.S., and Ph.D. degrees from the University of Illinois at Urbana-Champaign, Urbana, IL, in 1980 and 1982, respectively.

From 1982 to 1983, he worked in the Research and Development Department, ARCO Oil, and Gas, Co., Plano, TX. In 1983, he joined the VLSI Design Laboratory, Texas Instruments, Inc., Dallas, TX, where he is presently a Senior Member of the technical staff. His current interests include large-scale circuit simulation, mixed-mode simulation, circuit optimization, statistical circuit design, and computer-aided design.

Dr. Hocevar is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Gamma.