Coupling Algorithms for Mixed-Level Circuit and Device Simulation

Kartikeya Mayaram, *Member, IEEE,* and Donald O. Pederson, *Fellow, IEEE*

***Abstract*—Mixed-level device and circuit simulation allows the use of the numerical/physical models (based upon solution of Poisson's equation and the current continuity equations) for critical devices in a circuit configuration. Effective coupling of device and circuit simulation capabilities is achieved by a proper choice of algorithms and architecture. Coupling algorithms for dc, transient, and small-signal ac analyses are presented and evaluated in the framework of CODECS, a mixed-level circuit and device simulator.**

I. Introduction

A MIXED-level device and circuit simulator provides a direct link between technology parameters and circuit performance and is therefore useful in predicting the effects that variations in technology and device designs have on circuit performance. It also provides an environment for evaluating the interactions between semiconductor devices and the circuits in which they are embedded. Conventional device-level simulation typically allows voltage or current boundary conditions and some parasitic capacitive or resistive elements to be specified for a device [1]. It cannot be used to evaluate the performance of the device under realistic dynamic boundary conditions imposed by a circuit.

Previous work [2]–[6] in mixed-level circuit and device simulation has focused on dc and transient analyses only. All of these simulators employ a similar full-Newton solution scheme, as in [4]–[6], or a block-relaxation algorithm, as in [2] and [3]. The use of other coupling algorithms has not been investigated. In this paper, CODECS[1] (coupled device and circuit simulator) [7] is used as a common framework for evaluating different coupling algorithms for dc, transient, and small-signal ac analyses. This study provides a basis for selecting algorithms that can be used to effectively couple existing circuit and device simulation capabilities.

Mixed-level device and circuit simulations have previously been used to study single-event upset in memory cells [5], [8]. Examples of applications of CODECS include the study of the delay of BiCMOS driver circuits [9]–[11], inductive turn off of power devices [12], evaluation of switch-induced error in MOS switched-capacitor circuits [13], and verification of analytical models for circuit simulation [14]. CODECS incorporates SPICE3 [15] for the circuit simulation capability, and a brief summary of the features is presented in Appendix I. Details can be found in [13]. The algorithms used to couple the device and circuit levels of simulation are described in Sections II, III, and IV and conclusions are presented in Section V.

Manuscript received January 15, 1991; revised August 7, 1991. This work was supported by the Semiconductor Research Corporation under Grant 82-11-008. This paper was recommended by Associate Editor D. Scharfetter.

K. Mayaram was with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA. He is now with AT&T Bell Laboratories, Allentown, PA 18104.

D. O. Pederson is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

IEEE Log Number 9107743.

[1]The source code for CODECS can be obtained from the Software Distribution Office, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

II. DC and Transient Analyses

Since dc and transient simulations are the most useful, emphasis has been placed on algorithms for dc and transient analyses in a mixed-level circuit and device-simulation environment. A dc operating point analysis is required before a transient run is initiated or for small-signal ac and pole–zero analyses. For this reason convergence under dc conditions is extremely important; hence, the algorithms used for dc analysis must exhibit good convergence properties. The transient analysis problem is better conditioned than the dc problem and solutions from the previous time points provide a good initial prediction for the solution at the present time point. It can be anticipated that an algorithm different from the one used in dc analysis may perform better.

This section investigates different ways to couple the device and circuit simulation capabilities for dc and transient analyses. The two-level Newton algorithm, being the most intuitive, is introduced first. In this algorithm one needs to compute the terminal conductances of numerical devices, and this problem is addressed next. The framework of the program is then developed and used as a basis for describing other algorithms. Implementation issues are also presented to illustrate the similarity between the different algorithms. An evaluation is then made on the convergence properties and run-time performance of the different algorithms using a variety of benchmark examples.

A. The Two-Level Newton Algorithm

The problem of mixed-level device and circuit simulation is best illustrated by an example of dc operating point analysis. Consider the simple circuit shown in Fig. 1. G

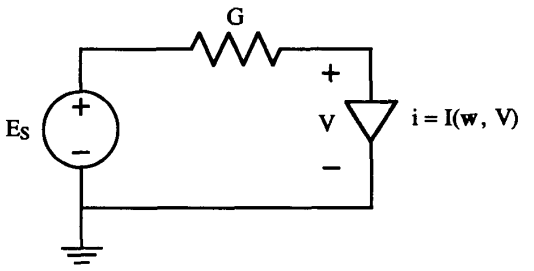


Fig. 1. Circuit used as an example.

is a linear conductance and E_s is a dc voltage source. The nonlinear device is a diode for which the characteristics are specified by a doping profile $N(x)$. The simulation problem can be stated as:

Given E_s, G and $N(x)$
Find V.

One solution for this problem is motivated by examining the problem from a circuit simulation point of view. In this case, analytical models are used for the nonlinear devices; the diode terminal characteristics are described by a closed-form expression, $i = I(V)$. When Newton's method is used to solve the nonlinear circuit equations, a linear circuit (the companion circuit [16]) is solved at each iteration until convergence is achieved. The companion circuit for the example under consideration is shown in Fig. 2. The linear conductance, G_{eq}, and the current source, I_{eq}, are obtained from the nonlinear characteristics as depicted in Fig. 3 and can be expressed as

$$G_{eq}^{k+1} = \left[\frac{\partial i}{\partial V}\right]_k \tag{1}$$

$$I_{eq}^{k+1} = i^k - \left[\frac{\partial i}{\partial V}\right]_k V^k \tag{2}$$

where k is the iteration number.

Once G_{eq} and I_{eq} are known, the circuit-level iteration can be performed. For numerical devices, the current–voltage characteristics are not known as closed-form expressions. Thus, G_{eq} and I_{eq} cannot be calculated by function evaluations and numerical techniques must be used. The partial differential equations (PDE's) describing a device have to be solved for each operating point before I_{eq} and G_{eq} can be calculated.

After space and time discretization the device-level equations result in a system of nonlinear algebraic equations [17]. These nonlinear equations are also solved by a Newton method. Once the equations have been solved for an applied bias V, the equivalent currents and conductances can be calculated as described in subsection II-B. The overall solution technique is a two-level Newton scheme wherein Newton's method is used at the device level and also at the circuit level. This is a special case of the multilevel Newton algorithm proposed in [18] for circuit simulation.

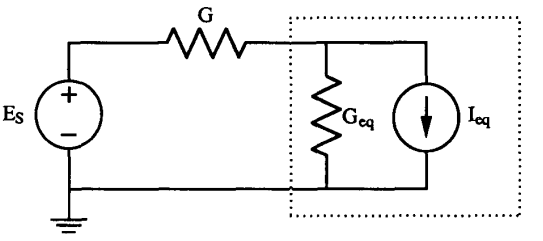


Fig. 2. Linearized companion circuit.

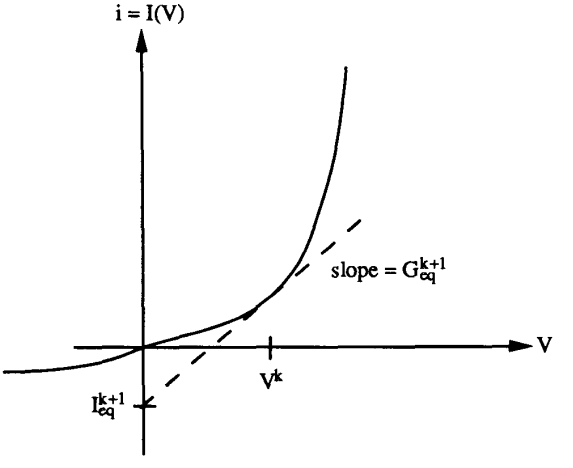


Fig. 3. Calculation of linearized conductance and current.

The flowchart of the two-level Newton algorithm is illustrated in Fig. 4. First, the contributions of all circuit elements that are represented by analytical models are entered into the circuit-level Jacobian matrix and the right-hand-side vector. Then the partial differential equations are solved for each numerical device, with the terminal voltages establishing the boundary conditions, until convergence is achieved at the device level. Once the solution at the device level has been obtained, G_{eq} and I_{eq} are calculated and assembled in the circuit-level equations. The linearized circuit-level equations are then solved and convergence is checked at the circuit level. If convergence is achieved, the solution has been obtained; otherwise the outer circuit-level loop is repeated.

B. Calculation of Conductances

This subsection describes the technique used for calculating the equivalent conductances for the two-level Newton scheme and is applicable to all types of numerical models. After space and time discretization the device-level equations can be represented as a set of nonlinear algebraic equations,

$$F(w(V), V) = 0 \tag{3}$$

where w is the vector of internal variables, i.e., the electrostatic potential, and the electron and hole concentrations at each spatial grid point. The dependence on the

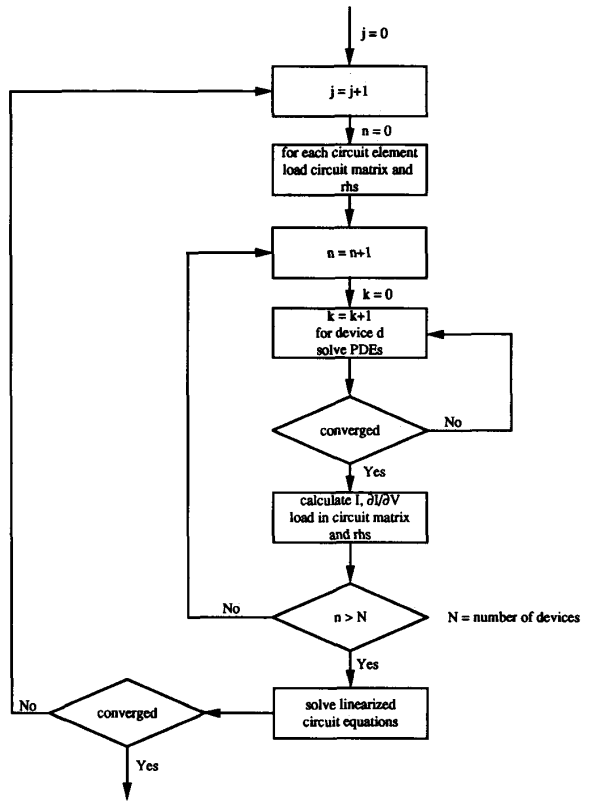


Fig. 4. Flowchart of the two-level Newton scheme.

boundary condition V is explicitly written in the above equation. CODECS presently implements only voltage boundary conditions; the continuation scheme of [19] could be used when the I–V characteristics are multivalued in V. For a two-terminal device, let $i = I(w, V)$ represent the terminal current as a function w and V; i is calculated by summing the current density components of the contact nodes. It should be noted that w is also an implicit function of V, since the value of w depends on the applied voltage. Equation (3) is solved for an applied voltage V_0 by Newton's method whereby

$$\Delta w = -J_w^{-1} F(w, V_0) \tag{4}$$

is solved at each iteration; $J_w = \partial F/\partial w$ is the Jacobian matrix of the device-level equations. J_w is decomposed into its LU factors at each iteration and Δw is obtained by forward and back substitutions. The iterations terminate when Δw satisfies the convergence tolerance and $|F(w, V_0)|$ is sufficiently small. At this stage w is the solution of $F(w, V_0) = 0$ and $I(w, V_0)$ can be calculated, since $w(V_0)$ is known. To calculate the linearized conductance $G_{eq} = \partial i/\partial V$, use is made of the chain rule, which gives

$$G_{eq} = \frac{\partial i}{\partial V} = \frac{\partial I}{\partial w}\frac{\partial w}{\partial V} + \frac{\partial I}{\partial V} \tag{5}$$

where $\partial I/\partial w$ and $\partial I/\partial V$ are obtained by symbolic differentiation of the function $I(w, V)$. The quantity $\partial w/\partial V$ is determined as in [20] or, equivalently, in the following manner. The derivative of (3) with respect to V is

$$J_w \frac{\partial w}{\partial V} + J_V = 0 \tag{6}$$

with $J_V = \partial F/\partial V$. From (6) one can solve for $\partial w/\partial V$ as

$$\frac{\partial w}{\partial V} = -J_w^{-1} J_V. \tag{7}$$

J_V has nonzero terms corresponding only to the contact nodes and can be easily assembled. Since J_w is available in its LU factors (calculated during the solution of (3) by use of (4)), only forward and back substitutions are required in calculating $\partial w/\partial V$, which is computationally inexpensive. Then G_{eq} can be calculated from (5). The conductance calculation and the computation of $\partial F/\partial V$ are described in Appendix II for a one-dimensional diode example.

C. The Modified Two-Level Newton Scheme

Newton's method requires a good initial guess to ensure convergence. A linear prediction step can be used to provide such a guess at the device level of simulation. The two-level Newton scheme with the linear prediction step is referred to as the modified two-level Newton algorithm. A first-order prediction is made by use of the forward-Euler scheme,

$$w^{k+1} = w^k + \left[\frac{\partial w}{\partial V}\right]_k \Delta V \tag{8}$$

where ΔV is the change in voltage from circuit iteration k to $k + 1$, and $\partial w/\partial V$ is calculated as in (7). As shown later, the modified two-level Newton scheme exhibits better convergence than the two-level Newton scheme.

D. Architecture of CODECS

It is clear from the previous subsections that a numerical device model for circuit simulation is similar to an analytical device model in several respects. Given the terminal voltages, the equivalent currents and conductances have to be calculated and used in the circuit-level equations. For any analytical model this task involves function evaluations, whereas for a numerical device the three PDE's have to be solved. The interface to a circuit simulator can be identical for the two types of models, as shown in Fig. 5, where the task of model evaluation is illustrated. The interface to the circuit simulator is through routines for model evaluation and for loading the equivalent currents and conductances in the circuit-level Jacobian and right-hand-side vector.

The overall framework of CODECS is shown in Fig. 6. The circuit simulator is the controlling program. It supports analytical models for the circuit elements and also stores the vector of node voltages. These voltages are available to the model-evaluation subroutines that calculate the equivalent conductances and currents for a de-

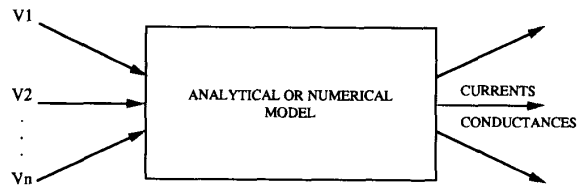


Fig. 5. The task of model evaluation. Given the terminal voltages, the terminal currents and conductances are calculated.

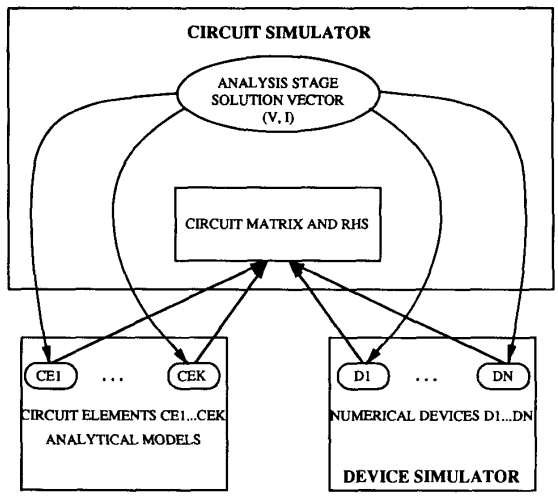


Fig. 6. Architecture of CODECS. Numerical devices are interfaced with the circuit simulator in a manner similar to that for analytical devices. The circuit node voltages establish the boundary conditions for the numerical devices. The device partial differential equations are solved by the device-level simulator of CODECS.

vice. The numerical devices are simulated by the device simulator of CODECS, and the interface to the circuit simulator is identical to that for analytical models. Device-level simulation is used to solve the PDE's for a numerical device for given terminal voltages. Then the terminal conductances and currents are calculated at the operating point and assembled in the circuit-level Jacobian matrix and right-hand-side vector.

E. The Full-Newton Algorithm

This algorithm for solving the mixed-level circuit and device simulation problem uses an alternative formulation of the problem. The device-level and circuit-level equations are combined and expressed as one system of equations. Newton's method is then applied to the complete system of equations. In contrast to the two-level Newton algorithm, where the device and circuit-level unknowns are solved separately in a decoupled manner, the complete set of unknowns is solved simultaneously. For the circuit of Fig. 1, the device-level equations are $\boldsymbol{F}(\boldsymbol{w}, V) = \boldsymbol{0}$ and these are combined with KCL at the circuit level to yield

$$\boldsymbol{F}(\boldsymbol{w}, V) = 0 \tag{9}$$

$$I(\boldsymbol{w}, V) + G(V - E_S) = 0. \tag{10}$$

Equations (9) and (10) are solved using Newton's method. The equations to be solved at each iteration are then

$$\boldsymbol{J}_w \Delta \boldsymbol{w} + \boldsymbol{J}_V \Delta V = -\boldsymbol{F}(\boldsymbol{w}, V) \tag{11}$$

$$\frac{\partial I}{\partial \boldsymbol{w}} \Delta \boldsymbol{w} + \frac{\partial I}{\partial V} \Delta V + G \Delta V = -I(\boldsymbol{w}, V) - G(V - E_S). \tag{12}$$

From (11), $\Delta \boldsymbol{w}$ can be expressed as

$$\Delta \boldsymbol{w} = \boldsymbol{J}_w^{-1}[-\boldsymbol{F}(\boldsymbol{w}, V) - \boldsymbol{J}_V \Delta V] \tag{13}$$

which can be rewritten as

$$\Delta \boldsymbol{w} = \Delta \hat{\boldsymbol{w}} - \boldsymbol{J}_w^{-1} \boldsymbol{J}_V \Delta V \tag{14}$$

where $\Delta \hat{\boldsymbol{w}} = \boldsymbol{J}_w^{-1}[-\boldsymbol{F}(\boldsymbol{w}, V)]$. Substituting $\Delta \boldsymbol{w}$ from (14) into (12), one obtains

$$\left[-\frac{\partial I}{\partial \boldsymbol{w}} \boldsymbol{J}_w^{-1} \boldsymbol{J}_V + \frac{\partial I}{\partial V} + G \right] \Delta V = -I(\boldsymbol{w}, V) - \frac{\partial I}{\partial \boldsymbol{w}} \Delta \hat{\boldsymbol{w}} - G(V - E_S). \tag{15}$$

This equation can be rewritten as

$$(G_{\text{eq}} + G) \Delta V = -I_T - G(V - E_S) \tag{16}$$

with $G_{\text{eq}} = -(\partial I/\partial \boldsymbol{w}) \boldsymbol{J}_w^{-1} \boldsymbol{J}_V + \partial I/\partial V$ and $I_T = I(\boldsymbol{w}, V) + (\partial I/\partial \boldsymbol{w})\, \Delta \hat{\boldsymbol{w}}$. Equation (16) is similar in form to that obtained with an analytical model for the diode of the earlier example, or by use of the two-level Newton algorithm. Thus the above technique can also be used to embed numerical models within the previously described framework. The full-Newton scheme can be implemented in two different ways.

1) Full LU-Decomposition Technique: $\boldsymbol{J}_w$ is decomposed into its LU factors and used to calculate $\Delta \hat{\boldsymbol{w}}$ and $\boldsymbol{J}_w^{-1} \boldsymbol{J}_V$ of (14) by forward and back substitutions. Then G_{eq} and I_T are computed. Equation (16) is solved whereby ΔV is obtained and $\Delta \boldsymbol{w}$ is calculated from (14), using the previously computed values of $\Delta \hat{\boldsymbol{w}}$ and $\boldsymbol{J}_w^{-1} \boldsymbol{J}_V$. The equations are solved to convergence. This technique is similar to the use of the block-LU decomposition with bordered-block-diagonal matrices in circuit simulation [21], [22].

2) Block-Iterative Technique: $\boldsymbol{J}_w$ is decomposed into its LU factors; $\Delta \hat{\boldsymbol{w}}$, G_{eq}, and I_T are calculated as above. ΔV is then obtained from (16), and $\Delta \boldsymbol{w}$ is assigned the value of $\Delta \hat{\boldsymbol{w}}$. This is equivalent to assuming $\Delta V = 0$ in (14) and ignoring the coupling term arising from ΔV. The equations are solved to convergence. A variant of this algorithm is used in MEDUSA [2].

F. Implementation Issues

The four algorithms described above have been implemented in the framework of CODECS shown in Fig. 6. The interface to the circuit simulator is through a model-evaluation subroutine which calculates and loads the device contributions in the circuit-level equations. The sub-

routines differ according to the algorithm used but the essential features are identical. First, the new terminal voltages are calculated and used to establish the boundary conditions for the device-level equations; then the device equations are solved. This is followed by calculation of the terminal currents and conductances, which are then loaded in the circuit Jacobian matrix and right-hand-side vector. The pseudo-C code for the four algorithms is shown below and illustrates the similarity between them. Furthermore, no algorithm has any significant advantage from an implementation point of view and all four techniques effectively decouple the device-level equations from the circuit-level equations, as seen from the pseudo code. The function *setBoundaryConditions* is used to establish the boundary conditions for the device; the function *biasSolution* solves the device equations to convergence or the iteration limit *iterLimit*, whichever is reached first. An *iterLimit* value of one allows calculation of $\Delta\hat{w}$ of (14). The function *updateSolution* is used to calculate $-J_w^{-1}J_V\,\Delta V$ and add it to the present solution vector. Therefore, it is used for the prediction step in the modified two-level Newton scheme (Eq. (8)) and in the full-LU decomposition algorithm for computing the solution at Newton iteration $k+1$, $w^{k+1} = \hat{w}^{k+1} - J_w^{-1}J_V\,\Delta V$, where $\hat{w}^{k+1} = w^k + \Delta\hat{w}^k$.

1) The Two-Level Newton Algorithm: For the two-level Newton scheme, at each operating point the new terminal voltages (boundary conditions) are imposed on the device and a solution is obtained for the new bias conditions:

```
setBoundaryConditions
 ( device, deltaV );
biasSolution
 ( device, iterLimit );
```

2) Modified Two-Level Newton Algorithm: As described earlier, the modified two-level Newton scheme makes use of a linear prediction step before solving the nonlinear device level equations. The pseudo-C code for this algorithm is given as

```
updateSolution ( device, deltaV );
biasSolution ( device, iterLimit );
```

3) The Full LU-Decomposition Algorithm: For the full LU-decomposition scheme the device simulator first determines the LU factors of J_w followed by a calculation of $\Delta\hat{w}$. The circuit node voltages are calculated by the circuit simulator. The quantity Δw can be obtained only after the circuit-level equations have been solved since ΔV is required for its calculation. The following approach is used. At the completion of the device-level solution, only $\hat{w}$ is calculated and stored. Before starting the next device-level iteration, w is calculated from $\hat{w}$ and ΔV. The new updated value of w is used for the next iteration. This sequence of operations allows a decoupling between the circuit and device simulators. At the first iteration of an operating point the terminal voltages are imposed on the device. However, in the subsequent iterations the subroutine *updateSolution* is used to establish the new boundary conditions, to calculate $-J_w^{-1}J_V\,\Delta V$; this gives the correct value of w to be used for the new iteration.

```
if ( CktNewtonIteration IS 1 ) {
     setBoundaryConditions
     ( device, deltaV );
} else {
     updateSolution ( device,
      deltaV );
}
biasSolution ( device, 1 );
```

4) The Block-Iterative Algorithm: The algorithm is similar to the two-level Newton scheme except that only one pass is made through the device-level equations for each circuit-level iteration and the calculation of conductances and currents is done in a different manner:

```
setBoundaryConditions ( device,
 deltaV );
biasSolution ( device, 1);
```

Four possible techniques to couple the device simulator to the circuit simulator have been described. These are the

1) modified two-level Newton algorithm (M2lev)
2) two-level Newton algorithm (2lev)
3) full LU decomposition technique (FullLU)
4) block-iterative technique (BlockIt).

These algorithms are now evaluated on the basis of their convergence and run-time performance.

G. DC Analysis Comparisons

The convergence properties of the four algorithms are examined by evaluating their performance on several benchmark circuits. A short summary of the circuits and the numerical models which are used is given in Table I; CODECS input listings are provided in [13].

In Table II the results for the dc operating point analysis of circuits[2] with one-dimensional numerical models for the bipolar transistor are given. The results are given as the number of circuit-level iterations followed by the total simulation time. A dash indicates that convergence was not achieved in 100 iterations.

As can be seen from Table II the modified two-level Newton scheme (M2lev) was always successful in finding an operating point, whereas the two-level Newton and full LU-decomposition schemes were only partially successful, and the block-iterative algorithm failed the test in all of these cases. Similar results were obtained with other circuits. Based on these experimental studies CODECS uses the modifed two-level Newton scheme for dc operating point analysis. The modified two-level Newton

[2]For the *Oscillator*, *VCO*, and *Astable* circuits, the dc operating point is the initial state of the circuit as obtained by the simulator under specified initial conditions. The oscillations are initiated in transient analysis by pulsing a current or voltage source in the circuit.

TABLE I
DESCRIPTION OF BENCHMARK CIRCUITS

Circuit	No. Nodes	No. Circuit Elements	No. Numerical Devices	Model Type	No. Grid Points
RTLinv	4	4	1 BJT	1D	61
Oscillator	5	8	1 BJT	1D	61
VCO	7	10	6 BJT	1D	61
Invchain	10	10	4 BJT	1D	61
Astable	6	8	2 BJT	1D	61
MECLgate	26	24	11 BJT	1D	61
Pass	6	7	1 MOS	2D	31 × 19
MOSinv	5	5	1 MOS	2D	31 × 19
ChargePump	7	7	1 MOS	2D	31 × 19

TABLE II
COMPARISON OF ITERATIONS AND RUN TIMES FOR DC OPERATING POINT ANALYSIS

Circuit	M2lev	2lev	FullLU	BlockIt
RTLinv	8/5.0 s	8/5.5 s	8/2.4 s	—
Oscillator	8/4.5 s	8/4.9 s	9/2.6 s	—
VCO	8/25 s	—	10/16 s	—
Invchain	9/22 s	—	—	—
Astable	9/11 s	—	—	—
MECLgate	51/81 s	51/94 s	—	—

TABLE III
COMPARISON OF NUMBER OF CIRCUIT ITERATIONS FOR TRANSIENT ANALYSIS

Circuit	M2lev	2lev	FullLU	BlockIt
Oscillator	16916	16916	18333	23836
VCO	5093	5109	5864	7028
Invchain	1563	1578	1716	2324
Astable	5930	6305	6369	9087
MECLgate	2450	2450	2609	3236*
Pass	236	236	295	338
MOSinv	287	313	336	533
ChargePump	1644	1661	1850	2661

scheme is computationally expensive compared with full-LU decomposition but is preferred for dc analysis since it is more robust and has worked well over a wide variety of examples.

H. Transient Analysis Comparisons

The transient simulation problem is better conditioned than the problem of simulating the dc operating point; hence, the algorithm that works best for simulation of the dc operating point may not be most suited for transient analysis. In this subsection the four algorithms are compared on the basis of their performance for transient mixed-level circuit and device simulations. The simulations are started with the dc operating point of the circuit being obtained by the modified two-level Newton algorithm. All simulations have been run with a latency check, which is described in the next subsection. The second-order backward-differentiation formula [23] is used for time discretization, and automatic time-step control is used based on local truncation error estimates. A starred entry indicates that the simulation was not completed successfully owing to a "time step too small" error, and the result is reported with the latency check turned off. In Table III, the number of circuit iterations are presented for transient analysis of the benchmark circuits.

The modified two-level Newton algorithm requires the smallest number of circuit-level iterations. The two-level Newton algorithm requires 10% more iterations in some examples, and the full-LU technique takes approximately 25% more iterations than the modified two-level Newton method. In all the examples algorithm BlockIt takes the largest number of circuit iterations and in the MECL-gate example the simulation could only be performed by turning off the latency check.

In Table IV are presented the number of time points that were accepted and rejected during the transient analysis. It is seen that the time points accepted and rejected are of the same order for the circuits using the above algorithms.

The simulation run times are presented in Table V. The full-LU decomposition scheme takes the smallest amount of time. The modified two-level Newton scheme on average is a factor of 1.7 slower than the full-LU decomposition scheme, and in some cases does even better than algorithm BlockIt. This might appear surprising at first because the modified two-level Newton scheme requires more CPU time for each circuit-level iteration. However, a smaller number of circuit-level iterations are required at each time point. Algorithm BlockIt has no apparent advantage; it requires more computational effort and does not work well with the latency check.

Based on the above experimental results CODECS makes use of the full-Newton algorithm for transient simulations. The modified two-level Newton algorithm is used only for dc analysis.

I. Latency Check in CODECS

At each time point the numerical devices converge in a different number of iterations depending on how their terminal voltages vary. If a device converges at a particular operating point and its terminal voltages do not change, then there is no need to evaluate the device. This form of latency, called iteration-domain latency, has been found

TABLE IV
COMPARISON OF TIME POINTS ACCEPTED AND REJECTED

Circuit	Number of Time Points Accepted				Number of Time Points Rejected			
	M2lev	2lev	FullLU	BlockIt	M2lev	2lev	FullLU	BlockIt
Oscillator	5274	5274	5274	5274	366	366	361	361
VCO	1099	1106	1125	1093	161	161	176	164
Invchain	401	404	401	410	17	18	17	20
Astable	1473	1583	1465	1554	198	234	181	219
MECLgate	619	619	619	619*	30	30	31	31*
Pass	82	82	82	82	8	8	8	8
MOSinv	95	104	95	104	4	8	4	8
ChargePump	497	497	493	492	74	74	73	75

TABLE V
COMPARISON OF TOTAL ANALYSIS TIME

Circuit	M2lev	2lev	FullLU	BlockIt
Oscillator	3126	3636	2352	3123
VCO	4085	5440	2911	3901
Invchain	890	965	514	806
Astable	2085	2538	1230	2031
MECLgate	3629	3931	2121	4577*
Pass	1786	1955	1059	1235
MOSinv	1626	2194	1155	1800
ChargePump	7172	8045	4039	5910

TABLE VI
COMPARISON OF TOTAL ANALYSIS TIME WITH AND WITHOUT LATENCY

Circuit	Latency Check	No Latency Check	Ratio
Oscillator	2352	2353	1.0
VCO	2911	4467	1.5
Invchain	514	908	1.8
Astable	1230	1713	1.4
MECLgate	2121	3893	1.8
Pass	1059	1160	1.1
MOSinv	1155	1239	1.0
ChargePump	4039	4356	1.1

to be useful in event-driven simulation [24]. The latency check in CODECS is based on a similar idea and is also similar to the bypass scheme of SPICE [25]. A numerical device is considered to be latent when all the following conditions are met for the full-LU algorithm.

(a) the device-level equations have converged;
(b) the terminal voltages meet the convergence criterion, i.e., they are within the error tolerances;
(c) the change of current is also below the tolerance for device currents.

The last two conditions can be described by the following equations:

$$|V_k - V_{k-1}| < \epsilon_r \,\mathrm{MAX}\,[|V_k|, |V_{k-1}|] + \epsilon_a \quad (17)$$

$$|I_k - \hat{I}_{k-1}| < \epsilon_r \,\mathrm{MAX}\,[|I_k|, |\hat{I}_{k-1}|] + \epsilon_a, \quad (18)$$

where $\hat{I}_k$ is the linearized terminal current that corresponds to the voltage V_k. A comparison of the use of the above latency scheme in CODECS is given in Table VI. The number of time points remains approximately the same; hence, only the analysis times are given in Table VI. An improvement in speed is achieved since the devices that have converged need not be reevaluated. The circuits *Oscillator*, *Pass*, *MOSinv*, and *ChargePump*, have only one numerical device; therefore, there is no improvement in performance. A latency check in the iteration domain is appropriate for circuits in which there are multiple numerical devices. On average 50% speedup is obtained for these test examples.

III. SMALL-SIGNAL AC ANALYSIS

Small-signal ac analysis is useful for analog circuit simulations. Since CODECS is intended to be a general-purpose coupled device and circuit simulator, it also provides a capability for small-signal ac analysis. Alternatively, one could run a transient simulation and extract the frequency-domain response using Fourier transform techniques. However, this approach is computationally expensive, and ac analysis provides a good way of obtaining the small-signal frequency-domain response.

The ac admittances for each device have to be computed and loaded in the linear circuit-level equations. The admittances are functions of frequency, and at a particular frequency, ω, the solution of the algebraic circuit-level equations gives the small-signal circuit node voltages and voltage source currents. For analytical device models the admittances are calculated at an operating point by function evaluations. For a numerical device the admittances can be calculated at the frequency ω by solving the small-signal device-level equations [26]. The solution of the device-level equations gives the small-signal ac values of the internal variables, the electrostatic potential, and the carrier concentrations at each spatial grid point. From this information the ac admittances for a numerical device can be calculated and then used in the circuit-level equations.

A. Calculation of AC Admittances for Numerical Devices

The small-signal ac terminal current, $\tilde{i}$, for a numerical device can be expressed as

$$\tilde{i}(\omega) = I[\tilde{w}(\omega), \tilde{V}, \omega], \quad (19)$$

where $\tilde{w}$ is the vector of small-signal values of the electrostatic potential and electron and hole concentrations,

$\tilde{V}$ is the applied small-signal voltage, and ω is the radian frequency. The explicit dependence on ω in (19) is through the displacement current component of the total current. The small-signal ac admittance is given by

$$Y(\omega) = \frac{\tilde{i}(\omega)}{\tilde{V}}. \tag{20}$$

If $\tilde{V}$ is taken to be unity,

$$Y(\omega) = \tilde{i}(\omega). \tag{21}$$

The small-signal ac current is given by the linear term of the Taylor series expansion of $i = I(w, V)$ at the operating point (w_0, V_0). Therefore,

$$Y(\omega) = \tilde{i}(\omega) = \frac{\partial I}{\partial w}\tilde{w} + \frac{\partial I}{\partial V}, \tag{22}$$

where $\partial I/\partial w$ and $\partial I/\partial V$ are evaluated at the dc operating point by use of symbolic differentiation, and $\tilde{w}$ is calculated as described in [26].

IV. Pole–Zero Analysis

Pole–zero analysis is used to determine the poles and zeros of transfer functions of the circuit linearized at an operating point. To obtain the poles and zeros the admittances of a device as a function of the complex frequency $s = \sigma + j\omega$ is computed. The technique used to calculate the admittance for a numerical device is an extension of the method used for small-signal ac analysis. Instead of using a frequency ω, the complex frequency s is used and the admittances are expressed as $Y(s)$. Given a value for s, $Y(s)$ can be calculated and used in the circuit-level equations. The circuit-level transfer function can then be computed and its poles and zeros can be determined. For numerical devices, $Y(s)$ is computed starting from the basic device equations. The unknowns are assumed to be of the form

$$w = w_0 + \tilde{w}e^{st}. \tag{23}$$

Using a Taylor series expansion at an operating point and retaining the linear terms, one can assemble the device-level equations in the form

$$[\boldsymbol{J}_w + \boldsymbol{D}]\tilde{\boldsymbol{w}} = \boldsymbol{B}, \tag{23}$$

where $\boldsymbol{J}_w$ is the dc Jacobian matrix of the device-level equations; $\boldsymbol{D}$ is a diagonal matrix with entries 0 corresponding to Poisson's equation, $-s$ corresponding to the electron current-continuity equation, and s corresponding to the hole current-continuity equation; $\tilde{\boldsymbol{w}}$ is the vector of small-signal values of the electrostatic potential and electron and hole concentrations; and $\boldsymbol{B}$ is the right-hand-side vector that accounts for the boundary conditions.

The above equations are solved for $\tilde{w}(s)$ by a direct solution method and $Y(s)$ is computed by

$$Y(s) = \frac{\partial I}{\partial w}\tilde{w}(s) + \frac{\partial I}{\partial V}, \tag{25}$$

where $\partial I/\partial w$ and $\partial I/\partial V$ are computed by symbolic differentiation. $Y(s)$ is then used in the circuit-level equations and the poles and zeros of the transfer function of the linearized circuit can be obtained by a root-finding algorithm of the type described in [27]. Spatial discretization of the numerically simulated device could add many poles and zeros, and this approach may not work well if the number of poles and zeros is very large.

V. Conclusions

A general framework for mixed-level circuit and device simulation has been described and used in the development of the simulation program CODECS. Various algorithms to couple the device and circuit simulators for dc and transient analyses have been implemented in CODECS. These algorithms are evaluated based on their convergence properties and run-time performance. This study provides guidelines for choosing a particular coupling algorithm. A modified two-level Newton algorithm is used for dc analysis whereas a full-block-LU decomposition algorithm is used for transient analysis. This combination of algorithms provides reasonable convergence and run-time performance. A simple latency scheme provides a 50% speedup. Coupling for small-signal ac and pole–zero analyses have also been described.

The techniques to couple the circuit and device simulators are general and can be used with other simulators as well.

Appendix I

CODECS incorporates SPICE3, a general-purpose circuit simulation program written in the C programming language, for the circuit-simulation capability and for analytical models of semiconductor devices. A device simulation capability has been developed that supports both one- and two-dimensional numerical models. Coupling of the device simulator to SPICE3 has been achieved in such a way that the core of the circuit-simulation program has been not modified; therefore, the coupling techniques can be easily adapted to other circuit- and device-simulation programs.

Nonlinear dc and transient, small-signal ac, and pole/zero analyses can be performed on circuits containing one- and two-dimensional numerical models for diodes and bipolar transistors and two-dimensional numerical models for MOSFET's. The numerical device models in CODECS include physical effects such as bandgap narrowing, Shockley–Read–Hall and Auger recombinations, concentration- and field-dependent mobilities, concentration–dependent lifetimes, and avalanche generation.

Appendix II

In the two-level Newton algorithms the conductances for numerical devices are calculated as described here. Conductance calculations involve computing $\boldsymbol{J}_V = \partial \boldsymbol{F}/\partial V$ at an operating point and the pseudo-C code is given be-

low for a diode example.

```
computeDiodeConductance ( pDevice, gd )
/* pDevice is the device pointer, gd is the calculated conductance */
{
 zeroRhsVector ( pDevice );
 /* store contribution of boundary nodes in rhs */
 assembleBoundaryRhsTerms ( pDevice );
 /* find the solution for the new rhs vector */
 /* device Jacobian already available in a factored form */
 solve ( pDevice->matrix );
 computeConductance ( pDevice );
}
```

The function assembleBoundaryRhsTerms needs further explanation. The right-hand-side vector corresponds to the terms $\partial F/\partial V$, and a one-dimensional diode example is used to illustrate the calculation of $\partial F/\partial V$. The diode is discretized in space using $L + 1$ grid points. For an ohmic contact at node $L + 1$, the potential is given by

$$\psi_{L+1} = \psi_{0,L+1} + V, \tag{A1}$$

where $\psi_{0,L+1}$ is the equilibrium potential and V is the applied voltage. Only the equations at node L have a direct dependence on V, through the dependence of ψ_{L+1} on V. Therefore, only the entries corresponding to node L are nonzero in the right-hand-side vector. The Poisson and current continuity equations at grid node L are given by

$$F_\psi = \frac{\psi_{L+1} - \psi_L}{h_L} - \frac{\psi_L - \psi_{L-1}}{h_{L-1}} + (N_L + p_L - n_L)\frac{h_L + h_{L-1}}{2} = 0 \tag{A2}$$

$$F_n = J_{n,L+1/2} - J_{n,L-1/2} + \left[-\frac{\partial p}{\partial t} + (G - R)\right]_L \frac{h_L + h_{L-1}}{2} \tag{A3}$$

$$F_p = J_{p,L+1/2} - J_{p,L-1/2} + \left[\frac{\partial n}{\partial t} - (G - R)\right]_L \frac{h_L + h_{L-1}}{2}. \tag{A4}$$

From these equations one obtains

$$\frac{\partial F_\psi}{\partial V} = \frac{\partial F_\psi}{\partial \psi_{L+1}} = \frac{1}{h_L} \tag{A5}$$

$$\frac{\partial F_n}{\partial V} = \frac{\partial F_n}{\partial \psi_{L+1}} = \frac{\partial J_{n,L+1/2}}{\partial \psi_{L+1}} \tag{A6}$$

$$\frac{\partial F_p}{\partial V} = \frac{\partial F_p}{\partial \psi_{L+1}} = \frac{\partial J_{p,L+1/2}}{\partial \psi_{L+1}}. \tag{A7}$$

These derivative terms are used to assemble the right-hand-side vector. A similar calculation is also used to assemble the right-hand-side vector for calculating the small-signal ac admittances [13].

Acknowledgment

The authors thank Prof. R. Dutton, Prof. P. Gray, Prof. C. Hu, and Dr. R. Guerrieri for several discussions. Thanks are also due to Dr. T. Quarles for help with SPICE3.

References

[1] M. R. Pinto, C. S. Rafferty, H. R. Yeager, and R. W. Dutton, "PISCES-IIB supplementary report," Stanford University, Stanford, CA, 1985.

[2] W. L. Engl, R. Laur, and H. K. Dirks, "MEDUSA—A simulator for modular circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-1, pp. 85-93, Apr. 1982.

[3] H. K. Dirks and K. Eickhoff, "Numerical models and table models for MOS circuit analysis," in *Proc. NASECODE IV* (Dublin, Ireland), June 1985, pp. 13-24.

[4] M. S. Mock, "Time-dependent simulation of coupled devices," in *Proc. NASECODE II* (Dublin, Ireland), June 1981, pp. 113-131.

[5] J. G. Rollins and J. Choma, "Mixed-mode PISCES-SPICE coupled circuit and device solver," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 862-867, Aug. 1988.

[6] J. R. F. McMacken and S. G. Chamberlain, "CHORD: A modulator semiconductor device simulation tool incorporating external network models," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 826-836, Aug. 1989.

[7] K. Mayaram and D. O. Pederson, "CODECS: A mixed-level device and circuit simulator," in *Proc. IEEE Int. Conf. Computer-Aided Design* (Santa, Clara, CA), Nov. 1988, pp. 112-115.

[8] J. S. Fu, C. L. Axness, and H. T. Weaver, "Memory SEU simulations using 2D transport calculations," *IEEE Electron. Device Lett.*, vol. EDL-6, pp. 422-424, Aug. 1985.

[9] G. P. Rosseel, R. W. Dutton, K. Mayaram, and D. O. Pederson, "Bipolar scaling for BiCMOS circuits," in *Dig. Tech. Papers 1988 Symp. VLSI Circuits* (Tokyo, Japan), Aug. 1988, pp. 67-68.

[10] G. P. Rosseel, R. W. Dutton, K. Mayaram, and D. O. Pederson, "Delay analysis for BiCMOS drivers," in *Proc. 1988 BCTM* (Minneapolis, MN) Sept. 1988, pp. 220-222.

[11] G. P. Rosseel and R. W. Dutton, "Scaling rules for bipolar transistors in BiCMOS circuits," in *IEDM Dig. Tech. Papers*, Dec. 1989, pp. 795-798.

[12] K. Mayaram, B. Tien, C. Hu, and D. O. Pederson, "Simulation and modeling for soft recovery of p-i-n rectifiers," in *IEDM Dig. Tech. Papers*, Dec. 1988, pp. 622-625.

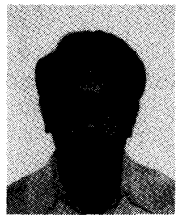
[13] K. Mayaram, "CODECS: A mixed-level circuit and device simulator," Memo No. UCB/ERL M88/77, Electronics Research Laboratory, University of California, Berkeley, Nov. 1988.

[14] H. J. Park, P. K. Ko, and C. Hu, "A charge conserving non-quasi-static (NQS) model for SPICE transient analysis," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 629-642, May 1991.

[15] T. Quarles, "The SPICE3 Implementation Guide," Memo No. UCB/ERL M89/44, Electronics Research Laboratory, University of California, Berkeley, Apr. 1989.

[16] L. O. Chua and P. M. Lin, *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques*. Englewood Cliffs, NJ: Prentice Hall, 1975.

[17] R. E. Bank, *et al.*, "Transient simulation of silicon devices and circuits," *IEEE Trans. Electron Devices*, vol. ED-32, pp. 1992-2006, Oct. 1985.
[18] N. B. Guy Rabbat, A. L. Sangiovanni-Vincentelli, and H. Y. Hsieh, "A multilevel Newton algorithm with macromodeling and latency for the analysis of large-scale nonlinear circuits in the time domain," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 733-740, Sept. 1979.
[19] W. M. Coughran, Jr., M. R. Pinto, and R. K. Smith, "Computation of steady-state CMOS latchup characteristics," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 307-323, Feb. 1988.
[20] Z. Yu, M. Vanzi, and R. W. Dutton, "An extension to Newton's method in device simulators—On an efficient algorithm to evaluate small-signal parameters and to predict initial guess," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 41-45, Jan. 1987.
[21] A. Vladimirescu, "LSI circuit simulation on vector computers," Memo No. UCB/ERL M82/75, Electronics Research Laboratory, University of California, Berkeley, Oct. 1982.
[22] P. Yang, "An investigation of ordering, tearing and latency algorithms for the time-domain simulation of large circuits," Rep. R-891, Coordinated Science Lab., Univ. of Illinois, Urbana, Aug. 1980.
[23] R. K. Brayton, F. G. Gustavson, and G. D. Hachtel, "A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas," *Proc. IEEE.*, vol. 60, pp. 98-108, Jan. 1972.
[24] R. Saleh, "Nonlinear relaxation algorithms for circuit simulation," Memo. No. UCB/ERL M87/21, Electronics Research Laboratory, University of California, Berkeley, Apr. 1987.
[25] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Memo No. ERL-M520, Electronics Research Laboratory, University of California, Berkeley, May 1975.
[26] S. E. Laux, "Techniques for small-signal analysis of semiconductor devices," *IEEE Trans. Electron Devices*, vol. ED-32, pp. 2028-2037, Oct. 1985.
[27] W. J. McCalla, *Computer-Aided Circuit Simulation Techniques*. Boston, MA: Kluwer Academic, 1987.



Kartikeya Mayaram (S'82-M'88) received the B.E. (Hons.) degree in electrical engineering from the Birla Institute of Technology and Science, Pilani, India, in 1981, the M.S. degree in electrical engineering from the State University of New York, Stony Brook, in 1982 and the Ph.D. degree in electrical engineering from the University of California, Berkeley, in 1988.

In 1988, he joined Texas Instruments, Dallas, TX, as a member of the Technical Staff. At present, he is a Member of the Technical Staff at AT&T Bell Laboratories, Allentown, PA.

Donald O. Pederson (S'49-A'51-M'56-F'64) received the B.S. degree from North Dakota State University in 1948 and the M.S. and Ph.D. degrees from Stanford University in 1949 and 1951. He was awarded an honorary Doctor of Applied Science by Katholieke Universiteit, Leuven, Belgium, in 1979. In 1991, he received the Berkeley Citation.

From 1951 to 1953 he was with the Electronics Research Laboratory, Stanford University, and from 1953 to 1955 with the Bell Telephone Laboratories, Inc. In 1955 he joined the University of California, Berkeley. He is now Professor Emeritus in the Department of Electrical Engineering and Computer Sciences, still engaged in teaching and research in the computer-aided design of microelectronic circuits. From 1988 to 1991, he was the E. L. and H. H. Buttner Professor of Electrical Engineering. From 1983 to 1985 he was Chairman of the Department. From 1960 to 1964 he was Director of the Electronics Research Laboratory.

Dr. Pederson was appointed a Distinguished Fulbright Lecturer to Ireland in 1988. He is a Fellow of the American Academy of Arts and Sciences (1991) and was a 1964 Guggenheim Fellow. He was the recipient of the 1969 IEEE Education Medal, of the IEEE Centennial Medal (1984), and, in 1985, of the IEEE Solid-State Circuit Council's Outstanding Development Award. In 1974, he was elected to the National Academy of Engineering, and in 1982 to the National Academy of Sciences.

Dr. Pederson is a member of the Board of Directors of Tektronix, Inc., of Valid Logic Systems, Inc., and of Varian Associates, Inc.