Circuit Embedded Device Simulation for Heterogenous Circuitry

Introduction

Simulation of advanced device structures using tools device simulation in conjunction with circuit components are presented. Two simulation approaches are described: a general mixed-mode simulator and a new linear mode implementation based on extended boundary conditions in the device simulator. Both methods allow for circuit simulation with the inclusion of detailed device physics. A comparative analysis for each method is presented along with results of a parallel computing implementation for the general mixed-mode simulator.

General Mixed-Mode Simulation

Mixed circuit/device simulation allows for the characterization of advanced devices including physical effects not available in standard circuit simulations [1]. For example, effects such as short channel effects in MOSFET's, high frequency effects in MESFET's, and temperature effects in high power devices can be included.

Unlike most commercial mixed-mode simulators that use a tightly integrated approach, this work adopts a loosely coupled algorithm. As a result, any device simulator may be added into the mixed-mode environment with the inclusion of a few additional routines and minor changes to the device simulator. In addition, each numerical device in the circuit may be simulated with an independent (or even different) simulator; allowing from parallization. Hence, the user is able to tailor the solutions by choosing the optimum simulator(s) for the devices under consideration. The main advantage of the loosely coupled algorithm is the modularity. The primary disadvantage, as will be discussed, is the increased computational cost.

Loosely Coupled Algorithm for General Mixed-Mode

The mixed-mode interface provides the necessary communications between SPICE and a numerical device simulator. Figure 1 shows how the two simulators are configured to talk with each other [6][7]. Similarly to an analytic model (i.e. BJT or MOSFET), a numerical device is seen as just another model in SPICE. This model communicates to the numerical device simulator through an interface routine. Like other analytic models, the numerical device provides terminal currents (and conductances) for given node voltages. The difference is that the latter finds the solution numerically.

This interface utilizes a two level Newton scheme to solve a circuit that includes numerical devices. On the outer level the circuit iterations are used to determine the node voltages. For each one of these iterations, the device simulator solves for the terminal currents and conductance matrix using another Newton iteration loop within the device simulator (the inner level). The advantage of this method is the modularity of the code.

Computational Costs

The two level Newton method for mixed circuit/device simulation incurs a significant computational cost. Mayaram had investigated that cost for a variety of algorithms including the two level Newton [3]. The most common of the alternate algorithms is the full Newton which had been implemented and examined by Rollins [4].

In the full Newton algorithm, the circuit matrix and device matrix are combined into a single matrix and all variables are solved simultaneously.

The two level Newton algorithm has the following advantages:

(1) It has better convergence behavior for DC analysis if node voltages are unknown. The full Newton requires all circuit node voltages to be specified within a certain percentage; otherwise, it is prone to not converge.

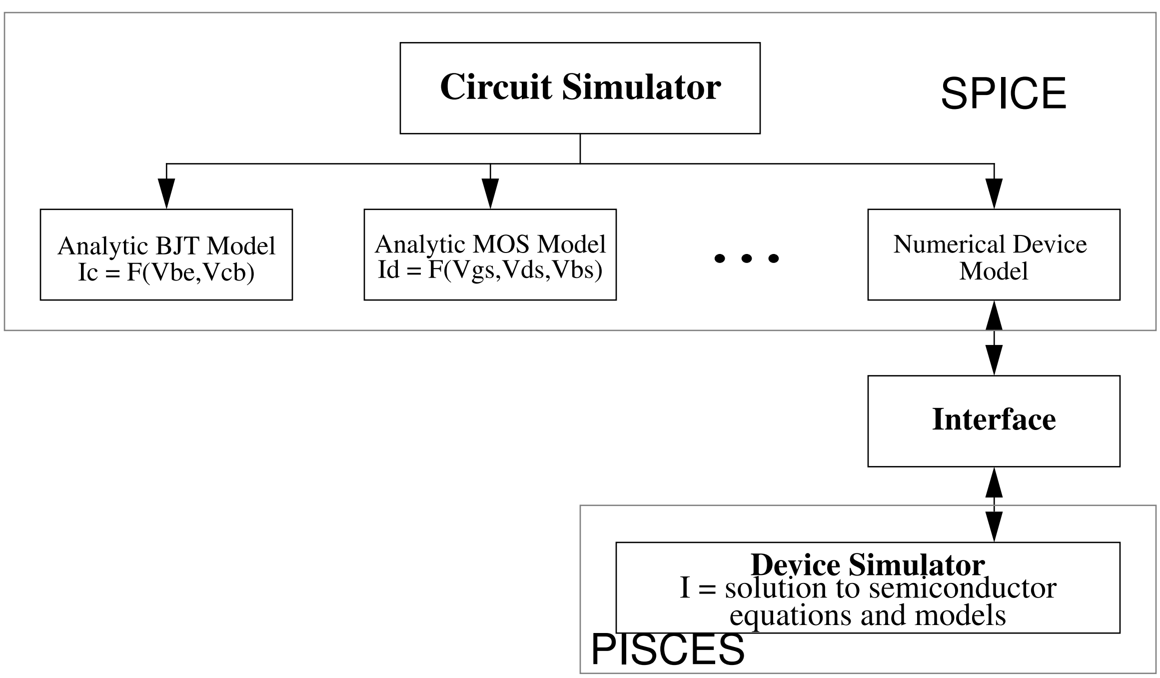


Figure 1: Block diagram of interfacing a numerical device simulator (such as PISCES) into SPICE. It becomes another circuit simulation block in a similar manner to an analytic model.

(2) The modularity in two level Newton allows many device simulators to be used simultaneously. For example, PISCES may be used for standard devices in the circuit whereas an alternate device simulator may be used for novel devices.

(3) It is easier to implement a parallel version of the two level Newton algorithm. Each numerical device simulation can be relegated to a node of a parallel machine whereas the matrix for the full Newton algorithm has to be partitioned to each node. (Parallelization is discussed in more detail in the following section.)

(4) By utilizing SPICE as the shell to loosely coupled the mixed-mode simulator, any improvements/ modifications in SPICE automatically benefit the mixed-mode simulation.

Likewise, there are a few disadvantages to the two level Newton algorithm:

(1) Given a good estimate of all the circuit node voltages, the full Newton algorithm converges more quickly than the two level Newton.

(2) Since transient analysis involves small voltage changes at each time step, making the problem well behaved, the full Newton method converges more quickly for this case. Mayaram determined a factor of 1.7 times as fast [3].

Although not fully documented, a two level Newton approach may be more computationally efficient than a full Newton for large problems. Suppose there are 10 numerical devices with 2500 nodes each in a unit cell. In the full Newton method an equation system of 3 x 10 x 2500 (3 semiconductor variables x 10 devices x 2500 nodes) has to be solved for each circuit iteration. On the other hand for a two level case, 10 matrixes of 3 x 2500 are solved per circuit iteration. Given the size of the matrices, the two level is expected to be faster. Preliminary work at IBM with a five stage ring oscillator has demonstrated this result. Previous work only solved circuits with devices containing a few hundred nodes, which tend to obscure the scaling factors [3].

Parallel Execution

In order to reduce the overall CPU time per circuit iteration, the numerical devices (i.e. the lower level Newton iterations) can be solved on different nodes on a network of machines. Because each device simulation is an independent unit, parallelization is relatively trivial.

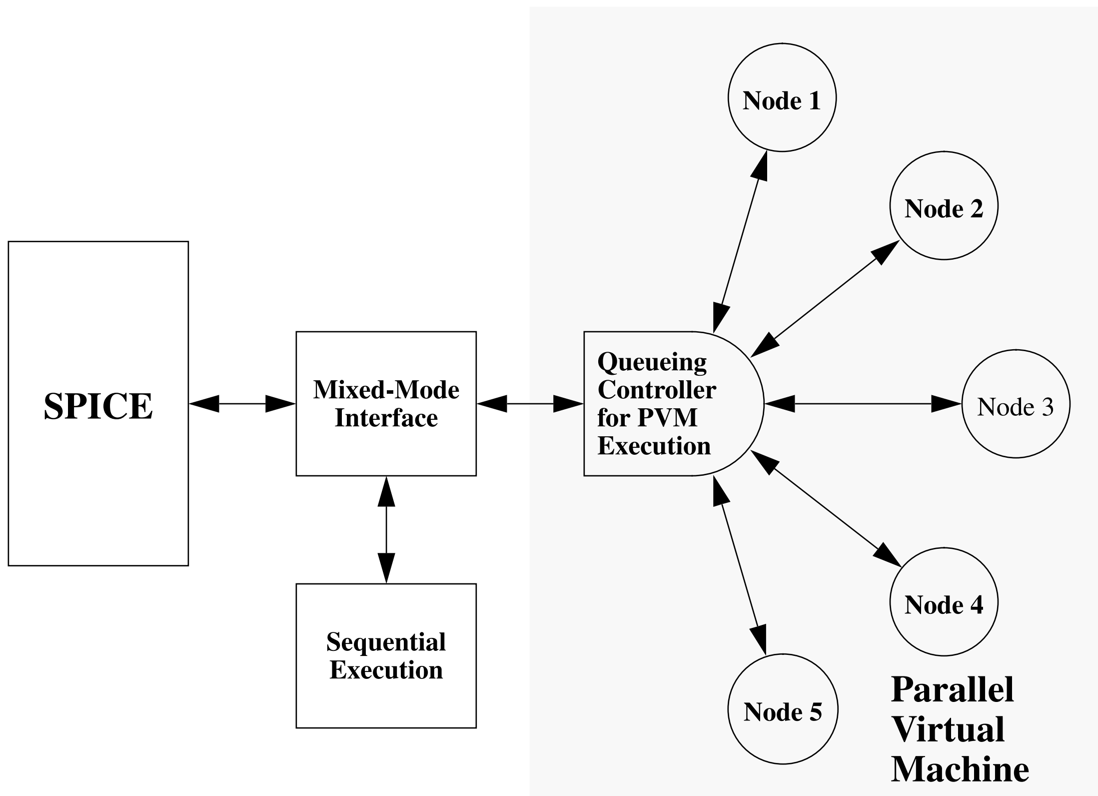


Figure 2: Parallel execution for mixed-mode simulations.

The mixed-mode interface contains three mechanisms by which the parallel simulations are executed. The most basic of these algorithms utilizes standard Unix sockets. Using UNIX libraries, each required device simulation (inner Newton level) is executed on one of the remote hosts previously specified by the user. A more elegant mechanism involves PVM (Parallel Virtual Machine) which allows a heterogenous network of computers to act like a single parallel machine [5]. Hence, parallel execution becomes a viable option in a typical industrial work environment. If a parallel machine is available, MPI (Message Passing Interface) provides a third option. The advantage of this method is the reduced overhead of not dealing with multiple operating systems [12]. Figure 2 shows a block diagram of how PVM is utilized. The other methods follow a similar approach.

The mixed-mode interface spawns a queueing process on a node of the virtual machine. This process ascertains the number of available nodes that can be used for a device simulation. Whenever the interface program needs solutions from a number of numerical devices, it sends a request to the queueing program which then executes the device simulations on one of the available nodes. Once the device simulations are complete, the queueing program sends back the solutions to the interface. If there are more desired solutions than nodes, the queueing program will try to balance the load as best as possible.

Linear Circuit Boundary Conditions in Device Simulation

The previous section has described a general mixed-mode approach where many devices are need to capture essential behavior and where computation time can be a major issue (hence, the need for the use of parallelism). Some simple yet important circuits consist of only a non-linear device with surrounding linear circuit components. Mixed-mode is one of the options for solving such problems, yet a practical alternative approach is to use more extensive circuit boundary conditions directly applied to the device simulator. This approach provides advantages in the computational time but does not incur the disadvantages of a tightly coupled mixed-mode simulator because only one device simulator is used and only linear components are available. Modularity is not an issue in this type of application.

Calculations of Circuit Boundary Conditions

Most device simulators allow finite external resistances to be placed on the contacts such that:

$$G(V_{app} - V_{con}) - I(\psi, n, p) = 0 \qquad \text{(Eq 1)}$$

where G is the conductance, V_{app} is the applied voltage, V_{con} is the contact voltage, and I is the current which is a function of the semiconductor variables ψ, n, and p where ψ is the potential, n is the electron concentration, and p is the hole concentration.

For a linear circuit surrounding a device, the previous equation can be expressed in matrix format. The voltages and currents become vectors and G becomes a matrix. Both G and V_{app} are constants.

In order to determine this matrix equation, consider Figure 3 where a linear circuit surrounds a numerical device. Substituting ideal current sources for the numerical device connections, modified nodal analysis (MNA) may be applied to the system with the assumption that the device current are unknowns:

$$G_c V_c = I_c + \overline{I_d} \qquad \text{(Eq 2)}$$

G_c is the circuit MNA conductance matrix; V_c is the unknown circuit node voltages and I_c is the known circuit currents; and $\overline{I_d}$ is the unknown device currents. $\overline{I_d}$ is zero for nodes to which the

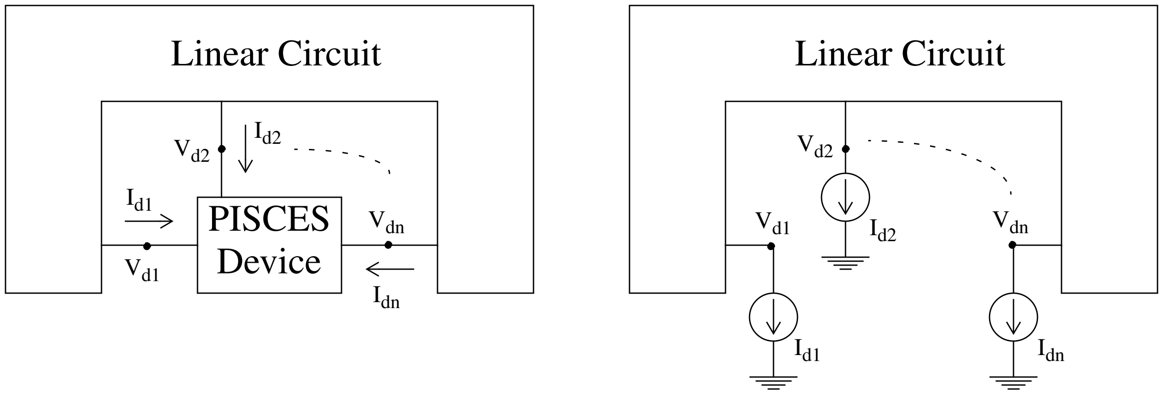


Figure 3: (a) Linear circuit surrounding a numerical device. (b) For the purpose of finding the boundary conditions, the numerical device is assumed to consist of ideal current sources of unknown values.

device is not connected. Hence, for a three contact numerical device, this vector only has three entries.

Within this system lays the boundary conditions for the numerical device as given in Equation 1. Solving Equation 3 and re-arranging yields Equation 4.

$$V_c + G^{-1}\overline{I_d} = G_c^{-1} I_c \qquad \text{(Eq 3)}$$

$$V_d + R_d \ I_d = E_d \qquad \text{(Eq 4)}$$

E_d can be found by LU decomposing G_c and then using backward substitution. Essentially, the linear circuit is solved for $\overline{I_d} = 0$. E_d is then just a subset of that solution for the circuit nodes connected to the numerical device.

Since $\overline{I_d}$ is symbolic, this derivation forces the computation of G_c^{-1} which is computationally inefficient. A better approach is to calculate R_d by perturbing the circuit solution for $\overline{I_d} = 0$ at the circuit nodes to which the device is connected. This process is represented in Equation 5:

$$r_{d_{ij}} = \frac{dv_{d_{ij}}}{di_{d_{ij}}} = \frac{\Delta v_{d_{ij}}}{\Delta i_{d_{ij}}} \qquad \text{(Eq 5)}$$

Hence, no inverse is needed and a simple LU decomposition of G_c with a few backward substitutions yields Equation 4. To clarify the notation, Equation 1 and Equation 4 are compared to find that V_d=V_{con}, E_d=V_{app}, G=R_d^{-1}, and I_d=I(ψ,n,p).

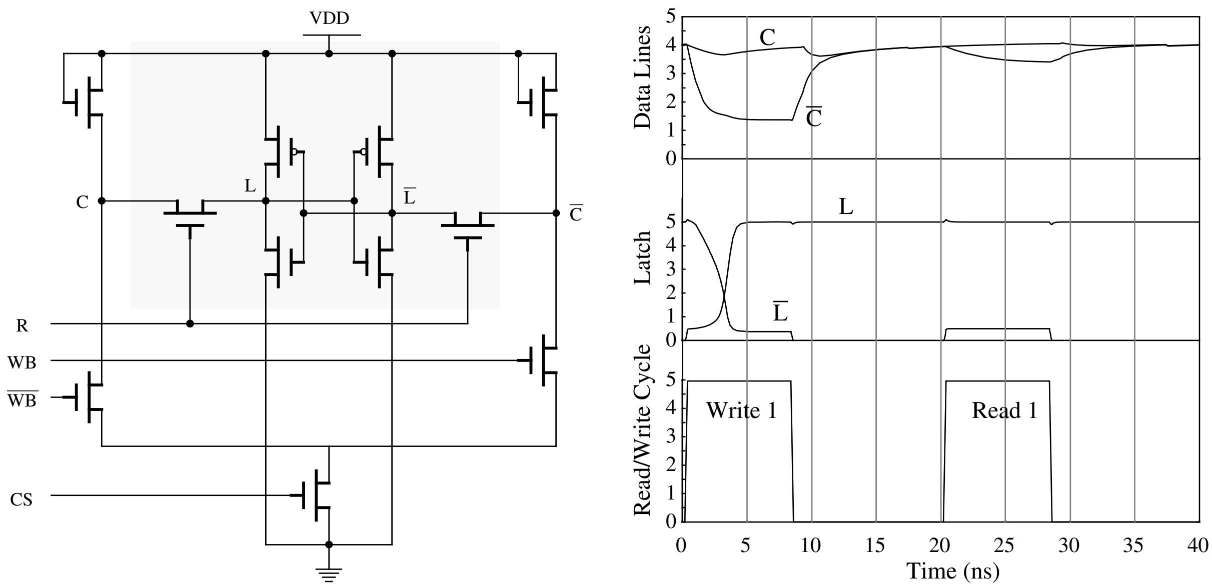
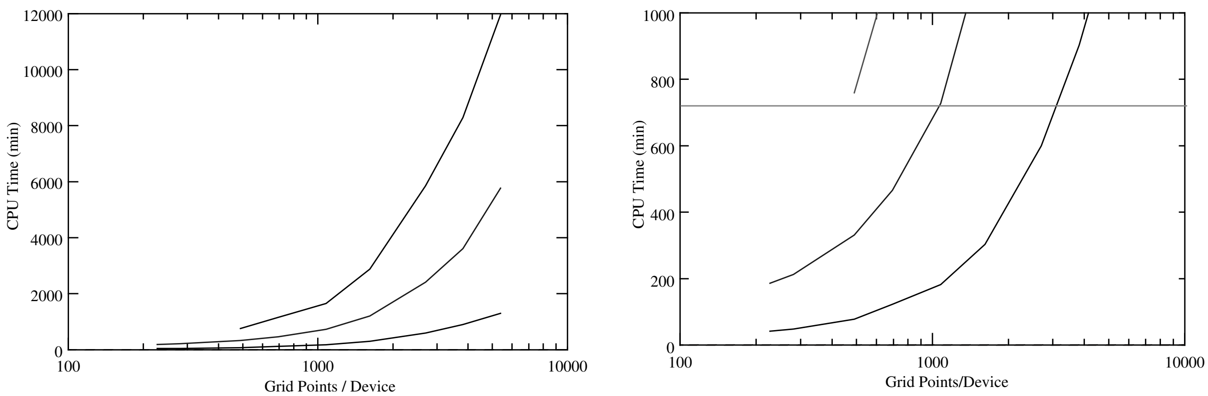


Figure 4: Circuit diagram and a read/write cycle for a SRAM cell and its surrounding control circuitry.

Performance Evaluation of Two Approaches

An SRAM example was used to analyze computational times for the general mixed-mode simulation. Figure 4 contains a circuit diagram for the simulated circuit and the response being measured. The six transistor SRAM cell is simulated with nine sets of CMOS numerical devices containing hundreds to thousands of nodes per device. In addition, three circuit configurations are simulated. One circuit has only the pass transistor as a numerical device, another has half the cell (3 transistors) as numerical devices, and the final configuration has all six transistors as numerical devices. The circuit was simulated for a read/write cycle of a “one” and a “zero” so that each half of the cell is exercised in a similar manner for cases of asymmetry. The non-numerical device transistors are modeled to approximate the numerical device characteristics using the standard SPICE MOS2 compact models.

The simulations were executed on a single node of an IBM SP1 machine. The CPU time in minutes is plotted in Figure 5a and Figure 5b. The first graph shows the total time for all the simulation runs and the second replots the data relative to the line representing 12 hours (an overnight run). The fastest time is around 45 minutes for a single 120 node device used as a pass transistor. For the 6 numerical transistor circuit with minimal node devices (less than 300 nodes), convergence became a problem and no data is available.



(a) (b)

Figure 5: CPU Time for simulating a read/write cycle of an SRAM cell. The curves from top to bottom are for 6 numerical transistors, 3 numerical transistors, and 1 numerical transistor respectively. (a) Total CPU times. (b) Times taking less than overnight.

Nodes	Elapsed Time	%CPU	%Idle	Improvement
1	51734s	90.9%	0.3%	1.00
2	27696s	90.7%	5.6%	1.87
3	20081s	90.5%	10.3%	2.58
4	18850s	90.3%	27.8%	2.74
5	17260s	89.8%	35.9%	3.00
6	12415s	88.3%	24.7%	4.17

Table 1: Resource statistics for parallel mixed-mode simulation

From a practical standpoint, a device of 500-1000 nodes is reasonable for modeling a MOS transistor in two dimensions. Examining the graphs, one finds that to simulate the entire SRAM cell using numerical transistors takes a little longer than overnight which may be too long. The parallelization of the general mixed-mode can reduce the CPU time significantly.

Table 1 contains a listing of the resource usage for a parallel execution of a 6 transistor SRAM cell with 500 nodes per device. The single node simulation is provided strictly for reference. The first column is the total elapsed time required for those results (i.e. how long the user has to wait for a solution.) The second column contains the percentage of the elapsed time spent on CPU usage. The average CPU time (summed over all device simulations for a particular configuration) is 45000s.

An important consideration is efficiency in parallel execution (i.e. effective usage of available resources). That information is summed up in the percent idle time and the overall improvement in wall-clock time. The percent idle is the time the nodes are idle (not doing any processing) divided by the total time for the simulation results. The improvement factor is the amount of improvement gained by using a parallel simulation. For an N node parallel simulation, the maximum expected improvement is N.

A couple of observations can be made from the tabulated results. First, the best performance is obtained for 2 and 3 node simulations. Less incremental improvement is obtained for 4 and 5 node simulations and mediocre improvement is obtained for a 6 node simulation.

These performance levels are a result of the parallelization scheme. In parallel execution, the entire device simulation is performed on a single node. Once that simulation is finished, another is executed on the same node if needed. For 2, 3, and 6 nodes simulations, the 6 numerical devices in the SRAM cell can be evenly distributed over all the nodes. On the other hand, for the 4 node simulations, 4 numerical devices are executed followed by two more. Assuming each device simulation takes approximately the same, 2 nodes are idle while 2 are processing. Hence, very little advantage is gained over the 3 node parallelization. A similar condition occurs for a 5 node parallel execution.

The question arise as to the larger idle time for the 6 node parallelization scheme over the 2 and 3 node scheme. This disparity is because in each circuit iteration, it is not always necessary to simulate all the numerical devices. This is part of the computational savings aspect of the code. Many times, only one part of the circuit is changing and hence, only those devices need to be iterated upon. As a result, all 6 nodes are not being used in every single iteration.

Another issue with parallelization involves the computation time required for one device relative to another. Consider the simulation of a cross-coupled inverter upon which strikes an alpha particle. Using the alpha particle capabilities of Fielday [10][11], simulations of upsets in cross coupled inverters can be investigated from a joint circuit/device perspective. Figure 6 shows the cross coupled pair along with the location of the strike. This simulation required one 3D mesh of 9500 nodes and three 2D mesh of 1900 nodes. The 3D device is necessary because the alpha particle strike is a 3D phenomena. The total computation time for this simulation was overnight

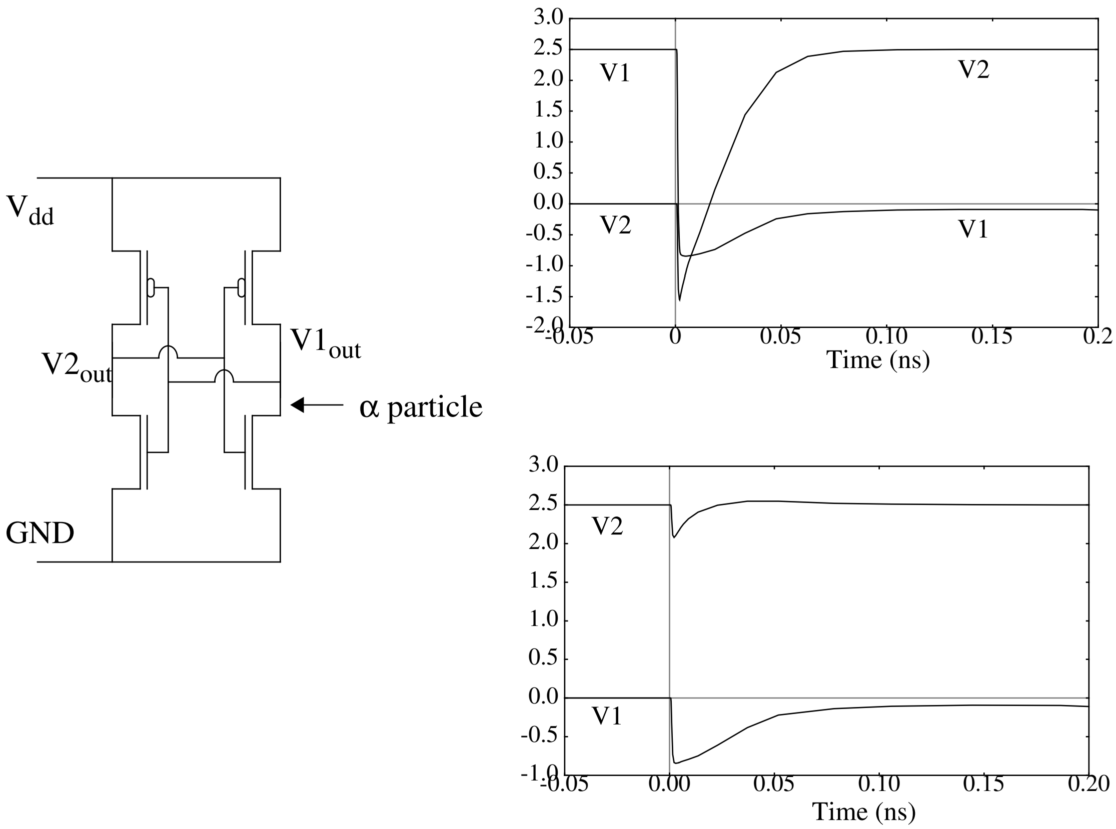


Figure 6: Simulation of an alpha particle strike on a cross coupled inverter. (a) Circuit diagram. (b) Simulation with V1 initial high. (c) Simulation with V2 initially high.

on two IBM 560 workstations using PVM. One node handled the three 2D devices whereas the other node handled the single 3D device. The time to solve the 3D device was the computational limit and the three 2D devices had little effect. Hence, the parallelization of the general mixed-mode becomes problem specific and one has to make intelligent choices in terms of how many nodes to relegate to the problem.

Given the computational burden of the general mixed-mode simulator the extended circuit boundary conditions to a device simulator can lead to a much faster execution when a single device is being analyzed. These boundary conditions are useful in problems involving transmission lines, parasitic analysis, and substrate modeling. In each of these examples only linear components surround the numerical device. Hence, a general mixed-mode simulation is not necessary.

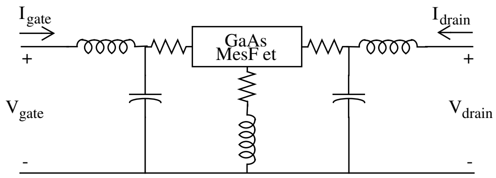
Simulation Type	General MM	Linear BC	Ratio
DC analysis of BJT inverter	438s	136s	3.22
Transient analysis of BJT inverter	2477s	1542s	1.61
AC analysis of BJT inverter	127s	87.7s	1.45
AC analysis of MESFET	10456s	8861s	1.18

Table 2: Comparison of general mixed-mode and linear circuit boundary conditions for three types of analysis.

To evaluate the computation cost of linear circuit boundary conditions versus that of general mixed-mode, three types of analysis are tested: DC, AC, and transient analysis. For a comparative circuit, a 200 node BJT inverter is used. For the DC simulation only the three bias resistors are important: collector resistor, bases resistor, and base-collector feedback resistor. The last of those could not be included in a typical device simulator without a linear circuit boundary condition for the feedback nature. The supply bias is swept from 3 to 5 volts by 1 volt increments and the input is swept from 0 to 5 volts by 0.1 volt increments. For the AC and transient analysis, additional capacitors and inductors are added to simulate interconnect parasitics. For the AC case, the inverter is biased at its 1/2 switching state and hence, acts like an amplifier. The frequency at the input is swept from 10kHz to 10GHz. For the transient case, there is a 200ns simulation during which a 100ns pulse is applied at the input. Both simulators use automatic time step control.

Table 2 shows a comparison of the computational times for the three BJT problems in addition to a MESFET to be discussed shortly. From the data one sees that DC analysis requires significantly more CPU time than either AC or transient. The reason for the excess CPU time during DC analysis is the extra circuit iteration in general mixed-mode. Even for a simple DC bias point, 3 circuit iterations are typically required; hence, the 3 fold increase in simulation time.

The transient analysis basically matches Mayaram's result of 1.7x in his comparison of loosely couple and tightly coupled mixed-mode [3]. The time transient comparison is highly dependant upon the time step control algorithm. If the time step is chosen too stringently, more CPU time is required. If it is chosen too loosely, the solution time is faster but the answer contains more truncation error effects.



$$Y_{11} = \frac{I_{gate}}{V_{gate}} \qquad Y_{12} = \frac{I_{gate}}{V_{drain}}$$

$$Y_{21} = \frac{I_{drain}}{V_{gate}} \qquad Y_{22} = \frac{I_{drain}}{V_{drain}}$$

$$y_{ij} = \frac{Y_{ij}}{Y_o}$$

$$\begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} = \begin{bmatrix} 1+y_{11} & y_{12} \\ y_{21} & 1+y_{22} \end{bmatrix}^{-1} \begin{bmatrix} 1-y_{11} & y_{12} \\ y_{21} & 1-y_{22} \end{bmatrix}$$

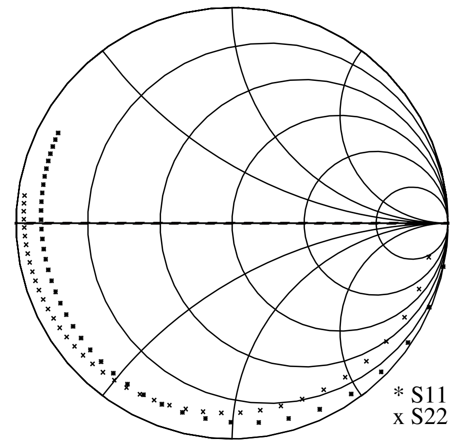


Figure 7: Determination of S-parameters on a two port network involving a non-linear MESFET with parasitics.

The general mixed-mode AC analysis compared favorably with the circuit boundary conditions approach. The reason is that AC analysis is a linear computation for the circuit. However, there still are two reasons for the increased CPU in general mixed-mode. First, the DC solution in general mixed-mode takes longer and hence, there is the set-up time for the AC analysis. Second for each frequency the general mixed-mode requires a DC operating point solution for the device simulator at the lower Newton level. Finding this operating point solution at each frequency does require a slice of CPU time in term of one Newton iteration even though the correct solution is read from a file. With circuit boundary conditions, the operating point solution is resident in memory.

The final computational comparison involves an industrial problem to calculate the S-parameters for a MESFET used in RF circuits. The packaging and interconnects are modeled with linear components. In order to determine the two port S-parameters (Figure 7) a small AC signal of 1 to 40 gigahertz is applied on each port. From the AC analysis, the Y-parameters are easily found. A simple transformation yields the S-parameters as plotted in the figure [13].

In this example, the computational time for AC analysis with circuit boundary conditions is only slightly better than that of general mixed-mode. The reason is related to the computational time required for the high frequency simulation of the MESFET. The small signal device simulation requires significantly more computations than any of the DC solutions. Since both methods require the same complete small signal solution at the operating point and that AC analysis is the computationally burdensome portion, both methods are comparable.

Conclusions

Two methods of circuit embedded device simulation were described. The first involves a modular general mixed-mode simulator so that any device simulator may be used. The computational cost of such a method can be expensive and hence, a parallel implementation is presented and analyzed. For certain problems, the modularity is not an issue and extended circuit boundary conditions can be applied to a numerical device. This method is also computationally analyzed for three modes of analyses: DC, AC, and transient. Both methods allow advanced device physics in a circuit simulation for improved analysis and design of circuit modules.

Acknowledgments

The authors would like to thank Steve Furkay and Jeff Johnson from IBM Microelectronics in Burlington, VT for the alpha particle problem and Matsushita Electric Industrial Co. in Japan for the MESFET example.

References

[1] Yu, Zhiping, Robert W. Dutton, and Hui Wang. "A Modularized, Mixed IC Device/ Circuit Simulation System." *Proceedings of the Synthesis and Simulation Meeting and International Exchange*. Kobe, Japan. April 1992.

[2] McCalla, William J. Fundamentals of Computer-Aided Circuit Simulation. Boston: Kluwer Academic Publishers, 1993.

[3] Mayaram, Kartikeya and Donald O. Pederson. "Coupling Algorithms for Mixed-Level Circuit and Device Simulation." *IEEE Transactions on Computer Aided Design*. Vol. II, No 8. August 1992. pp 1003-1012.

[4] Rollins, Gregory J. and John Choma. "Mixed-Mode PISCES-SPICE Coupled Circuit and Device Solver." *IEEE Transactions on Computer Aided Design*. Vol. 7, No. 8, August 1988. pp 862-867.

[5] Geist, Al, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM 3 User's Guide and Reference Manual*. Oak Ridge: Oak Ridge National Laboratory, 1993.

[6] Quarles, Thomas L. "Adding Devices to SPICE3." Memorandum No. UCB/ERL M89/45. Berkeley: University of California, 1989.

[7] Quarles, Thomas L. "SPICE3 Implementation Guide." Memorandum No. UCB/ERL M89/44. Berkeley: University of California, 1989.

[8] Laux, S. E. "Techniques for small-signal analysis of semiconductor devices." *IEEE Transactions on Electronic Devices*, Vol. ED-32, No. 10, pp. 22028-2037, 1985.

[9] Bank, R., W. M. Coughran, W. Fichtner, E. H. Grosse, D. J. Rose, and R. K. Smith, "Transient simulation of silicon devices and circuits." *IEEE Transactions on Electron Devices*, Oct. 1985. pp. 1992-2007.

[10] Furkay, Steve S., Jeff B. Johnson, Giovanni Fiorenza, and Ron Logan. *Fielday II User's Manual* Version No. 7.5.0. IBM Microelectronics, 1995.

[11] Johnson, Jeff B., Steve S. Furkay, and Daniel C. Cole. *Fielday II Theoretical Manual* Version No. 7.1.0. IBM Microelectronics, 1995.

[12] Gropp, William, Ewing Lusk, and Anthony Skjellum. Using MPI. Portable Parallel Programming with the Message-Passing Interface. Cambridge: The MIT Press, 1995.

[13] Wu, Ke-Chih, Zhiping Yu, Lydia So, Robert Dutton, and Junko Sato-Iwanaga. "Robust and Efficient ac Analysis of High-speed Devices." *Proceedings of International Electron Devices Meeting.* San Francisco, CA. December 1992.