An Overview of Algorithms in Gnucap

Albert T. Davis
Idaho State University
Pocatello, Idaho
Email: aldavis@ieee.org

Abstract— **This paper will present an overview of the algorithms in Gnucap. Gnucap is a mixed-signal circuit simulator.**

Algorithms to be presented include event driven analog simulation, the use of queues to accelerate simulation of large circuits, implicit mixed-mode simulation, where the simulator automates the interface between analog and digital portions of the circuit.

These algorithms provide equivalent accuracy to Spice with significant speedup for some classes of circuits, including large mostly passive circuits with a few active devices, and large mixed-mode circuits with latency.

An overview of work in progress will also be given. This includes cached model evaluation, which will exploit hierarchy and duplication in the circuit, and true multi-rate simulation.

I. Analog Simulation

The analog simulation is based on the LU decomposition, as in SPICE and many others. In mixed mode simulation, some parts of the circuit do not naturally fit this model. Some of the variables needed for this method may not exist, or may exist only in a different form. To address this issue, we will begin with this common form then extend it to fit the cases where only parts of the matrix change and only parts of the matrix exist. It will be used as a framework for the other methods, which will adapt dynamically.

II. Logic simulation

In developing the logic simulation algorithms, there are several goals to be considered.

1) Logic is considered to be an abstraction of a subset of the analog domain. The digital signals are abstractions of the analog signals: voltage and resistance.
2) The form chosen, both the internal data structures and the netlist description, must fit in a primarily analog simulator.
3) The logic algorithms employed use the same techniques as traditional logic simulators, in hopes that for purely logic circuits, the "mixed" simulation will not result in a significant penalty in time or space. This goal was not quite met. Since the analog model also exists, there is a significant cost in space, slightly worse than a fully analog simulator.

The algorithms for logic simulation are similar to those used in most logic simulators. They are clearly inadequate when race conditions exist and when the input voltage does not fit the logic model. The problems are used as indicators to switch to analog mode.

These abstractions work when all signals are proper, that is when there are no race or spike conditions. In race conditions signals arrive at a gate at different times that are not sufficiently different to be considered independently. The second transition may arrive after the first by a time that is less than the propagation delay of the gate.

III. Mixed simulation

The essence of mixed-mode simulation is that two or different types of simulation can be applied in the same circuit. This section discusses the data structures and decision algorithms to combine logic and circuit simulation.

A. Data Structures

The value at any node can be either continuous (voltage) or discrete (logic state). In addition, other information could also be useful at each node.

Both analog and digital values can exist at any node. A node can be either analog with digital derived from it, or digital with an analog approximation derived from it.

The admittance (Jacobean) matrix exists conceptually for all parts of the circuit, assuming that it may be all analog. For parts of the circuit that are simulated as digital, the corresponding parts of the matrix are not used and may be omitted. The matrix is allocated by subcircuit blocks, with the allocation for the hopefully digital blocks deferred until they are needed. Thus LU decomposition can be turned on and off for blocks of the matrix dynamically.

B. Choice of methods

Given that both analog and digital modes exist it is necessary to choose which mode to apply where. Usually the user does specify by asking for digital elements, but is often wrong and must be corrected. Analog elements will necessarily need analog simulation. Logic elements apparently need logic simulation, but this is not always true. Race conditions and poorly shaped signals can make logic simulation misleading. It is possible that a logic device was misused deliberately by the user. Two examples of this are making an oscillator out of two gates, and using a gate as an amplifier. The decision of which mode to use is made for each logic element, at run time, based on a set of rules.

The rules for deciding which mode to use are based on some assumptions. Both analog and digital information are available at any node, but only the information that is needed will actually be calculated. The conversion will be made on request. If the result of a conversion is suspect, it will be tagged for analog simulation. If any input or combination has

0-7803-7972-1/03/$17.00 c 2003 IEEE

a questionable state, the block will be simulated as analog, resulting in analog outputs. A logic block with analog inputs will be simulated as digital only if the input logic states can easily be determined from the voltages. A logic block with digital inputs from a different logic family will be simulated as if its inputs were analog. The decisions are based on a simple information that requires a minimum of storage: voltage, slope, and transition count since a bad transition.

C. Conversions

Voltage signals are transformed to logic signals by thresholding, roughly as in SAMSON[6][7]. For improper signals this simple conversion leads to illegal logic signal transitions. Slow transitions, illegal transitions and voltages outside the proper range for the type of circuit indicate to simulate that gate as analog. Once a conversion has been rejected, the signal will still be tested for validity as a digital signal. The signal will be accepted as digital after a number of clean transitions

Logic mode means that the logic function for the block is evaluated directly. Circuit mode means that the subcircuit representing the block is evaluated.

The logic to circuit conversion used here a ramped source with resistance, similar to that used in SAMSON[6].

IV. Incremental Matrix Solution

Large circuits usually use subcircuits, so we will take advantage of the ordering that results when global reordering is not done. A modular network forms a matrix in *bordered block diagonal* form. This form permits separate solution of the blocks (subnetworks), possibly in parallel. Possibly some can share storage, although that is beyond the scope of this paper. The network is derived from a composition of subnetworks. Each block represents a subcircuit, and the border represents the connections between them. Within a block, a submatrix has either a bump and spike or bordered block diagonal form, depending on whether it is made of subcircuits.

This work uses an inverted form of Crout's algorithm[1]. It calculates L and U directly without storing any intermediate results. This property enables single elements to be recalculated as needed.

Ordinarily, these operations are performed in place, so no memory beyond that already used to store the original matrix is needed. We chose not to perform factoring in place because by retaining the original A and b, partial updates and partial solutions are possible. Since the original matrix is maintained, only those elements that change need to be rebuilt. The matrix can be changed by adding the difference between the old and new values, and flagging those elements as changed.

The simulator may run in either a full mode or an incremental mode. The first iteration of an analysis is always run in full mode. Subsequent iterations, including the first iteration of a subsequent time step, are usually run in incremental mode.

Model evaluation and matrix loading are done in separate passes. When a model is evaluated, it is queued for loading. After all models that need to be evaluated have been evaluated, the load queue is processed to load the matrix.

V. Queues as a better bypass

In Spice, devices that have converged to what appears to be a final value may be bypassed. This still requires scanning every element on every iteration to check.

In Gnucap, the traditional bypass is replaced by a queue. On any evaluation, if a device is considered to be not converged, it is queued for evaluation on the next pass. If it is questionable, it is put in another queue which will be checked again on the next pass, where it will either be bypassed or evaluated. Constant components are always bypassed, except for the first pass. Logic gates are bypassed unless activated by an event.

When the evaluation of a device is bypassed, there can be no changes in the parameters loaded into the matrix, so it is not loaded. Combining this with the matrix incremental update, often large sections of the matrix decomposition can also be bypassed.

VI. Conclusions

The method has been shown to be effective in reducing simulation time for large linear subnetworks such as those resulting from RLGC models of transmission lines such as found in interconnect in PC boards and MCM's. It makes little difference for small nonlinear subnets. Although it did result in significant improvements in the LU phase of simulation, it does not hide inefficiencies elsewhere in the system, such as those resulting from a poor ordering.

It is reasonable to question whether improvements to the LU phase are significant, because even with traditional algorithms, the usual analog simulation is dominated by model evaluation, which would tend to mask any improvements. When most of the circuit is linear, this is no longer the case and LU improvements become significant. In mixed-signal simulation, this partial solution method allows whole blocks to be easily switched in and out as the simulation progresses. It also enables the use of queues as a more efficient alternative to bypass.

References

[1] Albert T. Davis. A vector approach to sparse nodal admittance matrices. In *30th Midwest Symposium on Circuits and Systems*, August 1987.

[2] H. deMan, G. Arnout, and P. Reynaert. Mixed-mode circuit simulation techniques and their implementation in DIANA. In P. Antognetti, D. O. Pederson, and H. de Man, editors, *Computer Design Aids for VLSI Circuits*. Martinus Nijhoff Publishers, Dordrecht, The Netherlands, 1980.

[3] James Ellis Kleckner. Advanced mixed-mode simulation techniques. Ph.d. thesis, University of California, Berkeley, 1984.

[4] A. R. Newton. Techniques for the simulation of large-scale integrated circuits. *IEEE Transactions on Circuits and Systems*, CAS-26(9):741–749, September 1979.

[5] David Overhauser, Ibrahim Hajj, and Yi-Fan Hsu. Automatic mixed-mode timing simulation. In *Proc. IEEE ICCAD*, pages 84–87, 1989.

[6] Karem A. Sakallah. Mixed simulation of electronic integrated circuits. Research Report CMUCAD-83-10, Carnegie-Mellon University, May 1983.

[7] Karem A. Sakallah and Stephen W. Director. SAMSON2: An event driven VLSI circuit simulator. *IEEE Transactions on Computer-Aided Design*, CAD-4(4):668–684, October 1985.

[8] Resve A. Saleh and A. Richard Newton. *Mixed-Mode Simulation*. Kluwer Academic Publishers, Boston, 1990.