anaLOG: A Functional Simulator for VLSI Neural Systems

by

John Paul Lazzaro

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science

Computer Science Department

Technical Report Number 5229:TR:86

California Institute of Technology

Pasadena, California

1986

This work supported by the System Development Foundation

Acknowledgements

Carver Mead, my thesis advisor, deserves special praise for his trusting and generous support of the anaLOG project. Working with David Gillespie, my research partner, has been a rewarding experience. My undergraduate advisor, Paul Mueller, and my mother, Marie Lazzaro, are continuing sources of guidance in my life. Significant contributions to the anaLOG project were made by Mark Bell, Jerry Burch, Lyn Dupré, Professor Joel Franklin, Andy Fyfe, Glenn Gribble, Scott Hemphill, Calvin Jackson, Dick Lyon, Mary Ann Maher, John Platt, and the students of the CS182 class at Caltech, 1985–1986. This thesis is dedicated to Nancy Lee Henderson.

This work supported by the System Development Foundation.

Table of Contents

Chapter 1
ana + LOG = anaLOG

1.1 Introduction

The emergence of a very-large-scale-integration (VLSI) design style for analog systems holds unique simulation challenges. In digital VLSI design, analog circuit simulators are used for the electrical verification of small circuit sections with high accuracy. The discrete nature of these systems permits the use of abstract switching models for the analysis of macroscopic logic and timing behavior. This is indeed fortunate, as a circuit simulation of a VLSI system using a traditional analog simulator is computationally impractical.

Analog VLSI design requires simulation tools that concentrate on the functional performance of an analog system, as opposed to a fully accurate electrical characterization, to verify correct behavior more efficiently. Such a simulator requires models that capture the functional essence of analog circuit primitives in a computationally simple manner. In addition, the structure of an analog design needs to be exploited by creating abstractions of circuit building blocks, rather than simulating circuits with their primitive description. This approach ultimately results in a hierarchical description of a large analog system, which can be simulated at different levels of abstraction, trading off accuracy and efficiency. The higher-level descriptions may be correct for only certain regions of operation, with range checking to indicate improper usage. Finally, the numerical methods used to simulate the description must reflect the unique requirements of functional simulation. By using these techniques, a simulation tool tailored to the verification of analog VLSI systems may be developed for a workstation environment.

The analog VLSI design style is directed toward system designers, as opposed to integrated circuit engineers. A successful analog VLSI designer must develop an intuition for parallel analog computation, as a digital VLSI system designer needs to develop skills in digital logic and circuit design. The user interface of a traditional circuit simulator, however, is directed toward an experienced electrical engineer, being batch mode (with textual input and output), unforgiving to the novice user, and isolated from other VLSI design tools. Clearly, the user interface of a analog VLSI simulation tool needs to be made more sympathetic and educational to a perspective designer from a nonengineering background.

A better simulation user interface portrays the metaphor of an electrical engineering lab bench. Using a bit-mapped display and a pointing device, the user can draw circuits in a schematic editing environment. As the circuit is being drawn, however, it is also being simulated, because a circuit under construction on a lab bench is always in operation. Symbolic multimeters and oscilloscope probes are available for the display of the ongoing simulation. Schematic voltage and current sources provide system input, with user control of the excitation, and graphical indication of source overload. The functional simulator, rather than standing alone, shares the lab bench with tools for documentation, integrated circuit mask design and verification, and other simulators.

LOG, an interactive graphical environment capable of hosting tools for simulation, documentation, and integrated circuit layout, has been developed by David Gillespie at Caltech. This thesis describes an analog functional simulator, Ana, which includes functional models for a standard cell library of analog computation building blocks designed by Carver Mead. Ana, LOG, the standard cell library, and a graphical layout editor form the basis of a design system used in the analog integrated circuits course at Caltech. The remainder of this chapter is an introduction to the LOG environment, and to the Ana simulator.

1.2 The LOG Environment

The LOG operating system provides a natural framework in which to manipulate systems describable as a list of interconnected objects with a static number of ports. The system includes a schematic editor, using a bit-mapped graphics screen and a pointing device, and an object-oriented interface mechanism for external programs. This mechanism allows full integration of tools for simulation, design, and documentation into LOG without source-code access; it functions as a software bus for a set of interrelated programs.

The user creates systems in LOG by graphically interconnecting objects, called *gates.* Gates are not dynamically definable; they are selected from a predefined catalog. Their definition can include an association with a tool; a gate thus can possess *type.* Each port of a gate, called a *pin,* also can be typed. When the user creates an instance of a gate, a *node* for each pin is created and is added to a global node list. The node assumes the type of the associated pin. The user edits the data structure by connecting pins together with *wires*; redundant nodes are eliminated from the node list during editing. The pins of a gate always are cognizant of their respective nodes. Type checking by LOG ensures nodes of different types can never be merged; type casting is available for exceptions.

The LOG interface mechanism for external programs is based on message passing. A tool has an interest in gates and nodes of its type. LOG sends messages to a tool concerning the definition, creation, manipulation, and deletion of these structures. LOG also sends general messages to a tool, such as "send debugging information about your tool." A tool is given access to the entire LOG data structure while responding to a message, with the understanding that only tool-specific sections may be modified. LOG also provides a procedure library to support data structure manipulation and the user interface.

Simulators are tools that predict the transient behavior of a system. Whereas most tools share only the LOG data structures, simulators share two dimensions of time. Because simulation and editing should appear to be simultaneous to the user, each simulator shares *CPU time* with other simulators and LOG. Multiple simulators may be used together to predict the behavior of a mixed type system; LOG manages *simulation time* as a global resource.

A simulator can respond quickly to most messages, allowing LOG to maintain a responsive graphical editing environment. Computationally intensive messages specify a maximum response time constraint; a simulator polices itself. LOG adjusts this *time-slice* according to user input activity, allowing transparent switches between a high-performance simulation environment and a crisp editing environment. Timed messages have protocols concerning the noncompletion of the message.

The management of simulation time is more complex. LOG begins with simulation time reset to zero; the user may also reset the system. During reset, a message sent to each simulator requests initialization of the system state. LOG and the user can request reset, but only simulators can advance simulation time and state. LOG regulates the simulation through several messages.

The *pass* message asks a simulator to compute an advance of simulation time and state for its system. The data structure may not have remained constant since the last pass message, but all pertinent changes have been reported to the simulator. A simulator cannot veto a data structure change or reset simulated time; it must contend with a dynamic system. A simulator reports whether time and state were successfully advanced as it returns from the pass message; an unsuccessful pass either implies that more CPU time is needed for simulation, or that the present data structure is not suitable for simulation.

A simulator advances the state of its own system; simulation time is shared by all simulators and must be managed globally. LOG allows all simulators to attempt advancing time; the smallest increase is accepted by LOG, while simulators bidding larger steps are requested to simulate up to the accepted time step. The *tstep* message informs a simulator of the status of its bid. After a uniform time advance, the simulators update internal system state in lockstep.

1.3 An Overview of Ana

A central aspect of an ideal analog VLSI simulation environment is the simultaneous editing and simulation of a system. The LOG operating system provides an excellent framework for this environment, offering a schematic editor with data structures appropriate for simulators, a communication system that informs tools of relevant system changes in an abstract manner, support for multitasking and multiple simulators, and the global management of a graphical user interface. The design of an analog simulation tool portraying the lab-bench metaphor in LOG raises several challenging issues.

In LOG's multitasking protocol, a tool must control its maximum response time to a simulation message. Constructing the simulation algorithm as a collection of distinct, computationally simple tasks is a method of response-time regulation. The message response can be considered a finite-state machine, executing each task in proper sequence until the allotted time has been exhausted, then saving the present position for a future message receipt.

A LOG simulator must contend with preserving the state of an incrementally changing system. In Ana, each pin of a gate has a small parasitic capacitance. When a gate is created, the parasitic capacitors are initialized. During the course of a simulation, these capacitors contain the complete state of the gate. When the data structure is changed, Ana uses this redundant information to evolve faithfully the state of the modified system from the original, presenting an illusion of continuous simulation. System state also can be reset to preset values by the user.

Ana attempts simulation only if every pin of each gate of a system is connected to some other pin, excepting measurement gates. A system of this nature is defined as *complete.* As a result, minor changes can be made transparently, but simulation stops during major editing, preserving system state and avoiding numerical instability.

The lab-bench metaphor demands robust simulation, as nature always finds the fixed point! The "soft" circuit modeling style of Ana promotes robustness. All excitation sources have output impedance and overload limiting. A capacitive path fully couples all pins of a gate to each other and to ground, and all active outputs posses DC impedance. Power and ground, the sole "hard" signal sources, are syntactically checked for shorts. Soft modeling allows a system to absorb abrupt changes gracefully.

The mathematical modeling style of Ana also contributes to its stable characteristics. All functions used in gate description are continuous with fully defined first derivatives throughout their range. The numerical methods in Ana use symbolic derivates, not finite differences. The avoidance of gross roundoff errors, and the use of functions with defined derivates, both increase stability.

An adaptive implementation of backward Euler integration is used in Ana. A full description and analysis of the method is given in Chapter 2 and Chapter 3. The implementation offers robust performance, with medium accuracy and computational simplicity. A noniterative equation solver, special methods for evolving the state of a dynamic data structure, and adaptive relaxation of accuracy during difficulty also aid convergence.

The responsiveness and flexibility of the LOG user interface must also extend to simulation tools. Ana can simulate a screen-sized system with sufficient speed for an interactive session. A hierarchical modeling system drives this performance; the library contains concise models for many common circuits. The simple, adaptive integration method also contributes to efficiency.

Equally important to user productivity is the leverage of the Ana user interface. Visual simulation feedback allows the quick correction of schematic entry errors; model parameters and system state can be changed with a single keystroke. Schematic multimeters can be added to display results; the simulator easily interfaces to oscilloscope and plotting tools for graphical output. The pointing device can become a status "probe" to scan the system quickly, and schematic wires can "glow" in various colors to show state. Many gates have warning displays to relate abnormal conditions; sources have "switches" to vary excitation. All these features can be activated without stopping the simulation.

Chapter 2
The Mathematics Of Ana

2.1 Introduction

In the LOG data structure, a system is described as a list of nodes $N_1, \ldots, N_n$ and a list of gates $G_1, \ldots, G_g$. A gate has an arbitrary number of pins, p. Each pin is connected to exactly one node. For a gate G_k with pins numbered $1, \ldots, l, \ldots, p$, the node of pin l is given by the function $Node(l)$. The current global simulation time is defined as t.

Ana views this system as a lumped electrical circuit. There is a voltage potential V_i between each node N_i and a common reference. The behavior of a gate G_k is defined as a set of functions $P_1^k, \ldots, P_l^k, \ldots, P_p^k$, where

$$P_l^k \equiv P_l^k\left(V_{Node(1)}, \ldots, V_{Node(p)}, \frac{dV_{Node(1)}}{dt}, \ldots, \frac{dV_{Node(p)}}{dt}, t\right)$$

is the current out of $Node(l)$ contributed by pin l of gate G_k.

By Kirchoff's current law, the sum of all the current out of a node must equal zero. The system of differential equations

$$\hat{F}_i\left(V_1, \ldots, V_n, \frac{dV_1}{dt}, \ldots, \frac{dV_n}{dt}, t\right) \equiv \sum_{\substack{all\ G_k \\ Node(l)=i}} P_l^k = 0 \qquad 1 \leq i \leq n$$

fully describes the original lumped-circuit system. Ana numerically solves these equations to obtain simulation results.

2.2 Solving the System of Differential Equations

Ana first converts this system of differential equations into a system of difference equations. The system variables $V_1, \ldots, V_n$ are thus transformed from continuous to discrete; V_i is now defined at a specific time t_z and is notated $V_i^{t_z}$. Ana requires its numerical methods to be highly robust, with medium accuracy and good efficiency. It is shown in Chapter 3 that the backward Euler method [1] meets these requirements. This method converts the differential equation

$$\frac{dV}{dt} = f(V, t)$$

into the difference equation

$$\frac{V^{t_z+\Delta t} - V^{t_z}}{\Delta t} = f\left(V^{t_z+\Delta t}, t_z + \Delta t\right),$$

where the timestep Δt is the distance between each discrete time value.

Using this method, the system of differential equations

$$\hat{F}_i \left(V_1, \ldots, V_n, \frac{dV_1}{dt}, \ldots, \frac{dV_n}{dt}, t \right) = 0 \qquad 1 \leq i \leq n$$

is transformed into the system of difference equations

$$\begin{aligned} &\hat{F}_i \Big(V_1^{t_z+\Delta t}, \ldots, V_n^{t_z+\Delta t}, \\ &\quad \frac{V_1^{t_z+\Delta t} - V_1^{t_z}}{\Delta t}, \ldots, \frac{V_n^{t_z+\Delta t} - V_n^{t_z}}{\Delta t}, t_z + \Delta t \Big) = 0 \qquad 1 \leq i \leq n. \end{aligned}$$

It is simple and convenient to derive a new system $F \equiv \hat{F}$, where

$$F_i \left(V_1^{t_z}, \ldots, V_n^{t_z}, V_1^{t_z+\Delta t}, \ldots, V_n^{t_z+\Delta t}, \Delta t \ , t_z + \Delta t \right) = 0 \qquad 1 \leq i \leq n.$$

Ana solves this system of difference equations to simulate the electrical system. When the simulation begins, $t_z = t_o$, and $V_1^{t_o}, \ldots, V_n^{t_o}$ are the initial conditions of the former differential equation. Algorithms for setting initial conditions correctly, and for choosing a suitable Δt throughout a simulation, are shown in Section 2.5.

2.3 Solving the System of Nonlinear Difference Equations

At time t_z of a simulation, the present state $V_1^{t_z}, \ldots, V_n^{t_z}$ is known. After an appropriate Δt is chosen, $V_1^{t_z+\Delta t}, \ldots, V_n^{t_z+\Delta t}$ are the only remaining unknown variables. F becomes the system of nonlinear algebraic equations

$$F_i \left(v_1, \ldots, v_n \right) \equiv F_i \left(V_1^{t_z+\Delta t}, \ldots, V_n^{t_z+\Delta t} \right) = 0 \qquad 1 \leq i \leq n.$$

The Newton–Raphson method [2] can be used to transform this nonlinear algebraic system into a linear one. Given an approximate solution $v_1^w, \ldots, v_n^w$, the method produces a better approximation $v_1^{w+1}, \ldots, v_n^{w+1}$, where $v_i^{w+1} = v_i^w + \delta_i$. It requires the existence of a good initial guess of the solution $v_1^0, \ldots, v_n^0$; Ana chooses $v_i^0 = V_i^{t_z}$. By expanding the system $F\left(v_1^w + \delta_1, \ldots, v_n^w + \delta_n\right)$ by Taylor's theorem and keeping only first-order terms, a system of of linear algebraic equations

$$F_i \left(v_1^w, \ldots, v_n^w \right) + \delta_1 \left. \frac{\partial F_i}{\partial v_1} \right|_w + \ldots + \delta_n \left. \frac{\partial F_i}{\partial v_n} \right|_w = 0 \qquad 1 \leq i \leq n$$

is produced. Solving this system yields $\delta_1, \ldots, \delta_n$, and ultimately the improved approximation $v_1^{w+1}, \ldots, v_n^{w+1}$. This process is iterated until a sufficiently close approximation is achieved; later sections include an analysis of the convergence properties of this method, and the establishment of a termination criterion.

2.4 Solving the System of Linear Equations

The $n \times n$ linear equation system produced by the Newton–Raphson method is usually sparse, because typical electrical circuits are sparsely connected. Sparse equation solution methods are more efficient than general techniques, but they are more complex to implement. In addition, sparse methods are iterative in nature; a robust implementation requires additional complexity. The current prototype of Ana uses a general noniterative solution method, the method of division by leading coefficients [2].

Using the notation $a_{ik} \equiv \left.\frac{\partial F_i}{\partial v_k}\right|_w, b_i \equiv -F_i\left(v_1^w, \ldots, v_n^w\right)$ the system of linear equations F can be rewritten as

$$a_{i1}\delta_1 + \ldots + a_{in}\delta_n = b_i \qquad 1 \leq i \leq n.$$

To solve for $\delta_1, \ldots, \delta_n$, each nonzero leading coefficient a_{i1} is first normalized. The equation with the former a_{i1} of largest absolute magnitude, called the *pivot equation*, is then subtracted from all others, reducing the $n \times n$ system to a $(n-1) \times (n-1)$ system. This procedure is repeated until a single equation in one unknown remains. All δ_i can then be obtained by direct substitution. A division by zero during calculation indicates a singular system.

2.5 Completing the Algorithm

The previous presentation leaves many details unspecified. The selection of a proper Δt for the backward Euler method, the iteration termination criterion for the Newton–Raphson method, and the correct response to a singular system by the linear equation solver are all unaddressed. In Ana, an adaptive control system is used to resolve these issues and complete the simulation algorithm. In the following discussion, all symbols in calligraphic type are user-accessible simulation parameters.

A singular system of linear equations has no solution. A nonsingular system of linear equations may not be numerically solvable, as a result of accumulated roundoff error during calculation. If a system of equations cannot be solved, the Boolean function *singular* is true.

The termination criterion for the Newton–Raphson method compares two successive approximate solutions of the nonlinear system. A relative comparison must be augmented by an absolute comparison to accommodate solutions of small magnitude without roundoff error. Ana considers a system solved at the end of iteration w, if for every i, either the relative condition

$$\left|\frac{v_i^{w-1} - v_i^w}{v_i^{w-1}}\right| \leq \mathcal{R}$$

or the absolute condition

$$|v_i^w| \leq \mathcal{A}$$

is met, where $\mathcal{R}$ is the relative accuracy and $\mathcal{A}$ is the absolute accuracy. If this condition is met, the Boolean function *converged* is true. If the initial guess of the solution $v_1^0, \ldots, v_n^0$ is not sufficiently close to the actual solution, Newton–Raphson may diverge. M, the maximum number of iterations, is a system *variable* under adaptive control. If the algorithm does not converge after M iterations, it probably never converges. If this condition holds, the Boolean function *toomany* is true.

Once a solution has been obtained, the error of the backward Euler method can be checked. The larger the voltage change for a timestep, the larger the approximation error. Ana accepts a solution $V_1^{t_z+\Delta t}, \ldots, V_n^{t_z+\Delta t}$ if for every i,

$$|V_i^{t_z+\Delta t} - V_i^{t_z}| \leq \mathcal{C},$$

where $\mathcal{C}$ is the maximum permissible voltage change for a timestep. If this condition is false, but

$$\Delta t \leq \mathcal{S},$$

where $\mathcal{S}$ is the smallest reasonable timestep, Ana still accepts the solution. This condition sacrifices accuracy for a robust simulation during difficult transitions; in practice, it is almost always invoked immediately after a data-structure change. If a solution is not accepted, the Boolean function *toobig* is true.

The choice of an improper timestep may result in a solution $V_1^{t_z+\Delta t}, \ldots, V_n^{t_z+\Delta t}$ that is physically impossible. Ana accepts a solution only if for all i,

$$\mathcal{L} \leq V_i^{t_z+\Delta t} \leq \mathcal{H}$$

where $\mathcal{L}$ and $\mathcal{H}$ are the lowest and highest plausible voltages respectively. If a solution is rejected, the Boolean function *overflow* is true.

An appropriately small Δt ensures an accurate simulation. A Δt smaller than necessary is inadvisable, as it needlessly increases computational cost. $\mathcal{Q}$ is the range of efficient solutions. If for all i

$$|V_i^{t_z+\Delta t} - V_i^{t_z}| \leq \frac{\mathcal{C}}{\mathcal{Q}},$$

the present Δt is inefficient, and in future simulation should be increased. The Boolean function *toosmall* is true if the current timestep is suboptimal.

The adaptive algorithm uses the previous Boolean definitions to control the simulation and update the variables Δt and M. At the end of every iteration w, Boolean conditions for accepting the present result and halting the simulation attempt are checked. If these conditions are both false, another iteration is performed.

If the Boolean condition *singular* or *toomany* or *overflow* is true, the simulation attempted has failed, due to an equation singularity, a nonconverging system, or a physically impossible result. The timestep Δt is reduced by the timestep divisor $\mathcal{D}$. The simulation is halted, and a new attempt to simulate the system begins.

If the condition *converged* `and not` *toobig* is true, Newton–Raphson has converged with a solution of desired accuracy, and the simulation result is accepted. If *toosmall* is true, Δt is unnecessarily small, and is increased by a factor $\mathcal{D}$. If Δt grows arbitrarily large, the simulator becomes unresponsive to sudden excitation. To prevent this condition, Δt is bounded by the maximum timestep $\mathcal{T}$. Convergence was achieved in w iterations; if the next simulation attempt does not converge in $\mathcal{I} \times w$ iterations, where $\mathcal{I}$ is the iteration factor, an improper Δt was chosen. Thus M is assigned the value $\mathcal{I} \times w$, and a new simulation attempt begins.

In many cases, Ana is used to find the DC fixed point of a system, and the simulation accuracy is irrelevant. In this case, the same control system is used, but *toobig* always returns false, and *toosmall* is always returns true.

Chapter 3
Properties of the Mathematics of Ana

3.1 Introduction

When choosing a numerical method for a particular application, robustness, computational complexity, and accuracy are three major considerations. The performance requirements of Ana, detailed in Chapter 1, clearly address these three issues. A robust simulation algorithm is essential to portray the lab-bench metaphor. In addition, computationally simple numerical methods are necessary for interactive response. Finally, since Ana is a functional simulator, highly accurate results are not needed. In this chapter, properties of the numerical methods used in Ana are compared with these desired characteristics.

3.2 Evaluating the Stability of the Backward Euler Method

The behavior of a linear resistor R and capacitor C connected in parallel can be described by the differential equation

$$\frac{dV}{dt} = -\frac{1}{RC}V \equiv -\lambda V$$

where V is the voltage across the components. The analytical solution of this equation is $V(t) = V_o \exp\left(-\lambda\left(t - t_o\right)\right)$, where t_o and V_o are initial conditions.

To analyze the stability of a numerical method for solving differential equations [1], it is interesting to compare its result for this problem with the correct solution $V(t)$. Recall that the backward Euler method converts the differential equation

$$\frac{dV}{dt} = f\left(V, t\right)$$

into the difference equation

$$\frac{V^{t_z+\Delta t} - V^{t_z}}{\Delta t} = f\left(V^{t_z+\Delta t}, t_z + \Delta t\right).$$

Applying this method to the test problem yields, after algebraic manipulation, the ratio

$$\frac{V^{t_z+\Delta t}}{V^{t_z}} = \frac{1}{1+\lambda\Delta t} \equiv X_n(\lambda\Delta t).$$

In contrast, the analytic solution to this test problem produces the ratio

$$\frac{V^{t_z+\Delta t}}{V^{t_z}} = \frac{\exp(-\lambda(t+\Delta t))}{\exp(-\lambda t)} = \exp(-\lambda\Delta t) \equiv X_a(\lambda\Delta t).$$

Note that $X_a(0) = X_n(0)$ and $X_a(\infty) = X_n(\infty)$. In addition, the error function $E(\lambda\Delta t) = |X_a(\lambda\Delta t) - X_n(\lambda\Delta t)|$ is bounded for all $0 < \lambda\Delta < \infty$. A numerical method is called *A-stable* if its solution to the test problem has these desirable properties. For an A-stable method, the choice of a timestep Δt affects only the accuracy, not the stability, of the test-problem solution.

This test problem is relevant to circuit simulation for several reasons. A typical analog system may have many RC subcircuits. At a specific time in the simulation, however, only a few of these subcircuits significantly affect system performance. An A-stable method allows an appropriate Δt to be chosen to track the significant subcircuits; a method that is not A-stable may require that a smaller timestep be chosen simply to ensure robust operation. In addition, note that any nonlinear differential equation

$$\frac{dV}{dt} = f(V, t)$$

may be approximated by the Taylor expansion

$$\frac{dV}{dt} = \frac{\partial f}{\partial V}(V - V_o) + \frac{\partial f}{\partial t}(t - t_o) + f(V_o, t_o).$$

Over a small interval $(t_o, t_o + \Delta t)$, $\frac{\partial f}{\partial V}$ can be approximated by Δt, yielding

$$\frac{dV}{dt} = \Delta t(V - V_o) + F(V_o, t_o).$$

The term $F(V_o, t_o)$ rarely affects stability. Therefore, the behavior of a numerical method on the test problem can indicate its stability when solving arbitrary nonlinear differential equations.

3.3 Accuracy and Complexity of the Backward Euler Method

The definition of the backward Euler method may be rewritten as [1]

$$V^{t_z+\Delta t} - V^{t_z} - \Delta t f\left(V^{t_z+\Delta t}, t_z + \Delta t\right) = 0.$$

If the numerical method is exact, the function

$$D(t_z, \Delta t) \equiv V(t_z + \Delta t) - V(t_z) - \Delta t f(V(t_z + \Delta t), t_z + \Delta t)$$

is equal to zero, where $V(t)$ is the true solution of the differential equation. The function $D(t_z, \Delta t)$, called the *local truncation error*, is a measure of the error of a numerical method. Replacing $V(t_z + \Delta t)$ in the definition of $D(t_z, \Delta t)$ by the Taylor series expansion

$$V(t_z + \Delta t) = V(t_z) + \Delta t\frac{dV(t_z)}{dt} + \frac{(\Delta t)^2}{2!}\frac{d^2V(t_z)}{dt^2} + \cdots$$

yields the expression

$$D(t_z, \Delta t) = \frac{(\Delta t)^2}{2!} \frac{d^2 V(t_z)}{dt^2} + \cdots.$$

Therefore, the local truncation error of the backward Euler method is $O((\Delta t)^2)$. The order of the local truncation error indicates the size of Δt necessary to achieve a certain accuracy, and thus is an indication of the computational complexity of the numerical method. Most simple numerical methods of higher order than is the backward Euler method are not A-stable, and therefore are unsuitable for use in Ana.

3.4 The Condition for Convergence of the Newton–Raphson Method

Given an approximate solution v^w of the nonlinear algebraic equation $f(v) = 0$, the Newton–Raphson method produces a better approximation

$$v^{w+1} = v^w - \frac{f(v^w)}{f'(v^w)} \equiv \phi(v^w)$$

as described Chapter 2. This iterative process continues until a sufficiently accurate solution is obtained, as measured by the error function $E(w) \equiv |v^{w+1} - v^w|$. For robust operation, each successive iteration must produce a smaller $E(w)$. The condition for convergence is shown below [2].

For the correct solution v^c, the equation $v^c = \phi(v^c)$ clearly holds, if $f'(v^c) \neq 0$. Subtracting this equation from the iteration equation $v^{w+1} = \phi(v^w)$ gives the expression

$$v^c - v^{w+1} = \phi(v^c) - \phi(v^w).$$

By the mean value theorem, the right side of the equation can be rewritten as

$$\phi(v^c) - \phi(v^w) = (v^c - v^w)\phi'(\xi^w) \qquad v^w \leq \xi^w \leq v.$$

Substitution yields the expression

$$v^c - v^{w+1} = (v^c - v^w)\phi'(\xi^w).$$

The condition of convergence for iteration $n-1$ is obtained by multiplying together instances of this equation for $0 \leq w \leq n-1$, which produces, after algebraic manipulation, the expression

$$v^c - v^n = (v^c - v^0)\phi'(\xi^0)\phi'(\xi^1)\cdots\phi'(\xi^{n-1}).$$

Note if $|\phi'(v)| < 1$ throughout the range (v^0, v^c), this equation can be rewritten as

$$|v^c - v^n| = |v^c - v^0| m^n,$$

where $m < 1$. In this event, the error $|v^c - v^n| \to 0$ as $n \to \infty$, and the process converges.

This condition can be easily expressed in terms of the initial function $f(v)$. Recalling the definition of ϕ,

$$\phi'(v) = \frac{d}{dv}\left(v - \frac{f(v)}{f'(v)}\right) = \frac{f(v)f''(v)}{(f'(v))^2}.$$

Thus, for the convergence condition $|\phi'(v)| < 1$ to be true in the neighborhood of the initial guess v^0, the inequality

$$|f(v^0)f''(v^0)| < |f'(v^0)|^2$$

must hold. Note that, if v^0 is sufficiently close to the correct solution v^c, $f(v^0) \to 0$ and the inequality is valid, if $f'(v^c) \neq 0$. Thus, the practical condition for convergence is the presence of a good initial guess v^0. This analysis is invalid if f is discontinuous, because the mean value theorem does not hold over the entire range of the function.

3.5 The Rate of Convergence of the Newton–Raphson Method

The rate of convergence of an iterative algorithm is a measure of its computational cost as a function of solution accuracy. This measure is easily derived for the Newton–Raphson method [3]. If the solution of the nonlinear algebraic equation $f(v) = 0$ is v^c, an approximate solution v^w can be written as $v^c + \epsilon_w$. The iteration equation

$$v^{w+1} = v^w - \frac{f(v^w)}{f'(v^w)}$$

can therefore be rewritten as

$$\epsilon_{w+1} = \epsilon_w - \frac{f(v^c + \epsilon_w)}{f'(v^c + \epsilon_w)}.$$

The Taylor series expansions of f and f' around v^c are

$$f(v^c + \epsilon_w) = \epsilon_w f'(v^c) + \frac{\epsilon_w^2}{2} f''(v^c) + \cdots$$

$$f'(v^c + \epsilon_w) = f'(v^c) + \epsilon_w f''(v^c) + \cdots.$$

Using these expansions, the expression

$$\epsilon_{w+1} = \epsilon_w - \frac{\epsilon_w f'(v^c) + \frac{\epsilon_w^2}{2} f''(v^c) + \cdots}{f'(v^c) + \epsilon_w f''(v^c) + \cdots}$$

is obtained, which after simplification yields

$$\epsilon_{w+1} = \frac{\epsilon_w f^{'}(v^c) + \epsilon_w^2 f^{''}(v^c) + \cdots - \epsilon_w f^{'}(v^c) - \frac{\epsilon_w^2}{2} f^{''}(v^c) - \cdots}{f^{'}(v^c) + \epsilon_w f^{''}(v^c) + \cdots}.$$

Truncating higher-order terms gives the final approximation

$$\epsilon_{w+1} \approx \frac{\epsilon_w^2 f^{''}(v^c)}{2f^{'}(v^c)}.$$

The error in the approximate solution v^w decreases quadratically with each iteration, and thus the rate of convergence for Newton–Raphson method is *second-order*. Methods that are of higher order than is Newton–Raphson typically have complex iteration procedures that offset their faster rate of convergence.

3.6 Properties of the Method of Division by Leading Coefficients

The method of division by leading coefficients is used in Ana to solve systems of linear algebraic equations. The stability and accuracy of this method are limited only by the inexact nature of a binary real number representation. For typical Ana systems of moderate size, these limitations are not significant.

The computational complexity of the method is derived easily. Recall from Chapter 2 that the method reduces an $m \times m$ system

$$a_{i1}\delta_1 + \ldots + a_{im}\delta_m = b_i \qquad 1 \leq i \leq m.$$

to an $(m-1) \times (m-1)$ system by dividing each each equation by its leading coefficient a_{i1}, and then subtracting one equation from the others. This reduction requires m^2 divisions and $m(m-1)$ subtractions. The reduction of a $n \times n$ system of equations to a single equation thus requires

$$\sum_{m=1}^{n-1} m^2 = \frac{n(n+1)(2n+1)}{6} = O(n^3)$$

divisions and

$$\sum_{m=1}^{n-1} m(m-1) = \frac{n(n-1)(2n-1)}{6} - \frac{n(n-1)}{2} = O(n^3)$$

subtractions. Completing the solution by direct substitution requires

$$\sum_{m=1}^{n} m = \frac{n(n+1)}{2} = O(n^2)$$

multiplications and

$$\sum_{m=1}^{n-1} m = \frac{n(n-1)}{2} = O(n^2)$$

subtractions. Thus, the method is $O(n^3)$ in both multiplications and subtractions. For sparse systems, there are more efficient methods for solving systems of linear equations. For the prototype of Ana, the simplicity of the method of leading coefficients is desirable; however, the final production version of Ana will use a more efficient sparse method.

Chapter 4
Simulating a Changing System

4.1 Introduction

During the simulation of a system of gates $G_1, \ldots, G_g$ and nodes $N_1, \ldots, N_n$, the user may at any time alter the data structure, producing a new system with gates $G'_1, \ldots, G'_{g'}$ and nodes $N'_1, \ldots, N'_{n'}$. When the data structure is changed, the state of the new system must be evolved from the old, maintaining the illusion of a continuous simulation. The techniques Ana uses to evolve system state are the focus of this chapter.

The state of a system is fully expressed by the voltages $V_1, \ldots, V_n$ associated with nodes $N_1, \ldots, N_n$. To aid the evolution of the system state during changes, this information is redundantly present in the list of gates $G_1, \ldots, G_g$. For a gate G_k with pins numbered $1, \ldots, l, \ldots, p$, each pin l is associated with a voltage v_l. The voltages $v_1, \ldots, v_g$ are used as the state variables for all memory elements contained in the gate G_k, because under normal circumstances, $V_{Node(l)} = v_l$. After a data-structure change, however, this may not be true. Re-establishing the condition

$$V_{Node(l)} = v_l \qquad \forall l,\ \forall G'_k$$

while maintaining the illusion of a continuous simulation is the main task of the state-evolution algorithm.

LOG reports data-structure changes on an atomic level, as messages describing the alteration of a single gate, node, or parameter. Parameter changes do not affect the structure of the system state; simulation can continue immediately after their receipt. Node and gate operations, however, fundamentally change the system. Most screen-editing operations decompose into multiple node and gate messages, all sent before LOG resumes simulation messages.

4.2 Evolving the System State

The algorithm for state evolution consists of two sections. The first section is the response of Ana to each atomic transformation reported by LOG. The second section operates after all atomic changes have made, and re-establishes the condition

$$V_{Node(l)} = v_l \qquad \forall l,\ \forall G'_k$$

for the new system. The first section is necessary to exchange information from the old system to the new before the latter disappears.

The first section of the algorithm is a collection of state-transfer rules for atoms. When a new node is created, its state V_i is undefined. When a node N_i is split into two new nodes N_j and N_k, V_j and V_j retain the value V_i. When two nodes N_j and N_k are combined into a single node N_i, V_i arbitrarily takes on the value V_j or V_k, preferring a defined state. The transfer rules for gates are similar. When a new gate G_k is created, the state of its pins $v_1, \ldots, v_p$ is undefined. When a copy of a gate is made, the copy keeps the state of the original gate.

The second section of the algorithm initializes all undefined nodes and pins. First, all undefined nodes N_i are initialized to the defined state v_l of an arbitrary pin associated with the node. Then, all undefined pin voltages v_l are initialized to the state V_i of the node associated with the pin, if defined. Finally, all remaining undefined nodes and pins are set to the reference voltage.

The condition $V_{Node(l)} = v_l$ now holds in all situations, except the connection of two defined pins to a single node. In this latter case, the simulation still is permitted to begin. Each gate connected to the node looks to its own v_l for state information, and influences the new $V_i^{t_z+\Delta t}$ for the node, effecting a weighted average of the dissimilar v_l's. After the first timestep of simulation is completed, $V_{Node(l)} = v_l$ again holds for the entire system.

This algorithm presents an intuitive evolution of state to the user. When a gate is created and added to a circuit, it assumes the state of the appropriate nodes. When a gate is removed from a circuit, it maintains its state. Subsystems can be copied with state intact. After two pins with different states are connected to each other, the simulation displays the resultant "fighting." At the start of a simulation session, the nodes of a system must "charge up" to a value, which is similar to applying power to a circuit at a lab bench.

4.3 Criterion for Simulating a System

In circuit theory, it is easy to create systems that are overconstrained and therefore have no solution. When using circuit theory to analyze a physical system, overconstrained systems indicate unrealistic component modeling. Future sections detail the soft modeling style of Ana, which makes the creation of an overconstrained system virtually impossible. Although the simulation of virtually any system is possible with Ana, there are valid reasons not to simulate certain systems.

Generally speaking, there are two types of circuit editing. Major editing involves the schematic capture of a large circuit section, and is primarily a data-entry task. Incremental editing involves the small alteration of an existing system, often in response to simulation results. Simulation during major editing is undesirable. A system under construction does not have a meaningful state, and the heuristic algorithm for evolving state may produce senseless results. When system construction is finished, the system state may be random, forcing the user to reset Ana to begin useful simulation. Simulation during incremental editing, however, is essential to the Ana user interface.

Ana distinguishes incremental editing from major editing by examining the data structure. Kirchoff's current law states that the sum of all the current into a node must be zero. Nodes associated with only one pin force the current through the pin to be zero, which is normally an uncharacteristic behavior of a useful circuit. Ana defines a system in which every node is connected to at least two nonmeasurement pins as a *complete* system, on which major editing is not occurring. Ana simulates only complete systems. This definition works well in practice, except for the occasional finished system that is not "complete!" The addition of a small parasitic capacitor to the offending node solves the problem.

Chapter 5
The Modeling Style of Ana

5.1 Introduction

As specified in Chapter 1, component models in an analog VLSI simulator must be robust, efficient, and functional. To meet these performance requirements, gate models in Ana follow a set of guidelines for robust simulation, and use a library of functions suited for the efficient description of MOS systems. Chapter 6 details the function library; the modeling style is the focus of this chapter.

The modeling guidelines address both the mathematical form and the behavioral function of a gate description. Mathematically, the preferred method of partial derivative evaluation for the Newton–Raphson method is given, and continuous techniques are shown for describing discontinuous gate behavior. The functional guidelines dictate parasitic elements in all gates, to prevent overconstrained systems and physically impossible simulation results.

5.2 The Symbolic Evaluation of Partial Derivatives

As discussed previously, the behavior of a gate G_k with pins $1, \ldots, l, \ldots, p$ is defined as a set of functions $P_1^k, \ldots, P_l^k, \ldots, P_p^k$, where

$$P_l^k \equiv P_l^k \left(V_{Node(1)}, \ldots, V_{Node(p)}, \frac{dV_{Node(1)}}{dt}, \ldots, \frac{dV_{Node(p)}}{dt}, t \right)$$

is the current out of $Node(l)$ contributed by pin l. The functions can be redefined with respect to the gate coordinate system, yielding the simpler expression

$$P_l^k \equiv P_l^k \left(V_1, \ldots, V_p, \frac{dV_1}{dt}, \ldots, \frac{dV_p}{dt}, t \right).$$

By applying the backward Euler method, these functions can be rewritten in finite-difference form. For the solution at a specific time $t_z + \Delta t$, the functions become $P_l^k \equiv P_l^k(v_1, \ldots, v_p)$, where $v_l \equiv V_l^{t_z+\Delta t}$. To solve a set of nonlinear algebraic equations that include these functions, the Newton–Raphson method requires the evaluation of the set of partial derivatives

$$\frac{\partial P_l^k}{\partial v_1}, \ldots, \frac{\partial P_l^k}{\partial v_n}$$

as well as the function P_l^k.

In the Ana library, the computation of partial derivatives is done by the numerical evaluation of the symbolic derivatives of P_l^k, rather than by employing a finite difference method. The symbolic method is more accurate than is that of finite differences, resulting in quicker convergence and simulation stability. When computing a function P_l^k with its full set of symbolic derivatives, redundant evaluation can be eliminated through clever factoring, lessening the efficiency advantage of the finite difference method.

5.3 Modeling Decisions with Fermi Functions

The function P_l^k should be continuous with respect to all variables v_l, for several reasons. A good initial guess of $v_1^0, \ldots, v_p^0$ is required for convergence of the Newton–Raphson method; around a discontinuity, such a guess is impossible. In addition, to use the symbolic method of partial-derivative evaluation, the first derivative of P_l^k with respect to all pin potentials must be fully defined. Electrical components, however, often exhibit sharp transitions. A set of switching functions with fully defined derivatives is used in the Ana gate library to describe decisions.

The Fermi function has a fully defined first derivative and exhibits the desired strong switching behavior. It is defined as

$$F_e(v) \equiv \frac{1}{1 + \exp(v/v_f)},$$

for the domain of all real v. The range of the function is bounded by $[0, 1]$. For $v/v_f \ll 0$, $F_e \to 1$, whereas for $v/v_f \gg 0$, $F_e \to 0$. The switching slope at zero is inversely proportional to $4v_f$. The switching point can be shifted to $\hat{v}$ by the substitution $v \equiv v' - \hat{v}$, and the function $\mathcal{R}F_e(v)$ has the expanded range $[0, \mathcal{R}]$.

Fermi functions can be easily used to make more complex decisions. The switching polarity can be reversed with the function $\overline{F_e}(v) \equiv 1 - F_e(v)$. Given two decisions $F_e(v_1)$ and $F_e(v_2)$, the binary `and` decision is $F_e(v_1)\,F_e(v_2)$. With the `and` and `not` functions available, any Boolean logic expression can be synthesized. The maximum and minimum functions can also be produced. The expressions

$$\max(v_1, v_2) \equiv v_1 F_e(v_2 - v_1) + v_2 \overline{F_e}(v_2 - v_1)$$

and

$$\min(v_1, v_2) \equiv v_2 F_e(v_2 - v_1) + v_1 \overline{F_e}(v_2 - v_1)$$

display the technique.

5.4 Behavioral Modeling for a Robust Simulation

Several precautions must be taken in the modeling of components to ensure a stable and sensible simulation. Inadequate modeling of parasitic capacitances can remove a transient behavior that is necessary to achieve steady state simulation conditions. The use of ideal voltage and current sources can result both in unsolvable, overconstrained systems and in underconstrained systems that exhibit physically impossible behaviors. The Ana gate library uses systematic techniques to avoid these problems.

To ensure the proper transient behavior of a gate G_k with pins $1, \ldots, l, \ldots, p$, there must be a capacitive path between each pin l and the other $p-1$ pins. In addition, there must be a capacitive path between each pin and the reference node. If the physical component does not exhibit parasitic behavior, the capacitance values chosen should be small enough to have negligible effect on simulation accuracy. This interconnection technique ensures that a gate with only one driven pin has a transient path to steady-state operation.

Several parasitic elements are added to voltage and current sources in Ana to prevent simulation difficulties. Linear resistors are added to independent voltage sources to model output impedance. Fermi functions are used to limit independent current sources; for a current source I_o connected across a voltage v, the expression

$$i = I_o F_e \left(\hat{v} - v\right)$$

limits the output current i when the output voltage is below $\hat{v}$. When parasitic elements cause a qualitative change in independent source behavior, graphical notification of limiting appears on the schematic gate symbol. All MOS devices are provided with output impedance; the functional form is shown in Chapter 6.

Chapter 6
The MOS Transistor

6.1 Introduction

Note that this chapter concerns the NFET4 and PFET4 models, which are now obsolete. The preferred models (NFET5, PFET5, PFET6) are based on the Vittoz model, and have much better properties than the models described here.

The electrical characteristics of the MOS transistor are well known; Mead and Maher [4] have developed an accurate device model from first principles. The MOSFET model in Ana is an engineering approximation of a device physics model, which efficiently captures the first-order behavior of the transistor. It is expressed as a combination of simpler functions, which also are useful for manual circuit analysis. These functions are used to describe more complex gates in the Ana library, as shown in later chapters.

In Ana, the MOS transistor is modeled as a nonlinear, voltage-controlled current source. The device has three terminals—named source, drain, and gate—with voltage potentials V_s, V_d, and V_g relative to the reference node. Differential voltages are notated with multiple subscripts. The gate is an isolated control terminal, whereas the source and drain are the current source terminals. The current I_{ds} is a function of all three potentials. The analysis in this chapter assumes an N-channel device; for a P-channel transistor, reverse the signs of all doubly subscripted variables in the presentation.

The behavior of the device is symmetric with respect to V_{ds}. If $V_{ds} > 0$, current flows from drain to source, and $|I_{ds}| = f(V_{gs}, V_{ds})$. Conversely, for $V_{ds} < 0$, current flows from source to drain, and $|I_{ds}| = f(V_{gd}, V_{ds})$. Note that, if V_{ds} equals zero, no current flows through the transistor. The behavioral symmetry of the MOS transistor is reflected in the full mathematical symmetry of the Ana model; in the following discussion, features are defined for forward current flow, then extended to the reverse-direction flow.

6.2 A Simplified Nonlinear Current Source

As noted, when $V_{ds} > 0$, current flows from drain to source, and $|I_{ds}| = f(V_{gs}, V_{ds})$. As a first approximation to I_{ds} for forward current flow, a function $I_c(V_{gs})$ can be defined. The function has two regions. For $0 \leq V_{gs} \leq V_t$, where V_t is the threshold voltage of the transistor, I_c is exponentially dependent on V_{gs}. This *subthreshold region* has the functional form

$$I_o \exp(V_{gs}/V_o),$$

where I_o and V_o are transistor parameters. For $V_{gs} \geq V_t$, I_c has a square dependence on V_{gs}. This *space-charge limited region* continues throughout the normal operating range of the device.

By applying current continuity at $V_{gs} = V_t$, and using Fermi functions, the expression

$$I_c\left(V_{gs}\right) \equiv I_o \exp\left(V_{gs}/V_o\right) F_e\left(V_{gs} - V_t\right) + aV_{gs}^2\, \overline{F_e}\left(V_{gs} - V_t\right)$$

can be defined, where $a \equiv (I_o \exp(V_t/V_o))/V_t^2$. Because of the symmetry of the device, $I_c(V_{gd})$ is an approximation of the reverse current magnitude $|I_{ds}| = f(V_{gd}, V_{ds})$ when $V_{ds} < 0$.

6.3 Output Impedance Extension

The MOS transistor has a finite output impedance. The Ana MOS model approximates this behavior as a linear current dependence on V_{ds} throughout the model range, as proposed by Early. For forward current flow, the expression

$$I_f \equiv I_c(V_{gs})G(V_{ds})$$

can be defined, where

$$G(V_{ds}) \equiv \left(1 + \frac{V_{ds}}{V_e}\right)$$

and V_e, the Early voltage, is a transistor parameter. Likewise, in the reverse direction, where $V_{ds} < 0$, the current magnitude can be expressed as

$$I_r \equiv I_c(V_{gd})G(V_{sd}).$$

By using Fermi functions, the current magnitude can be approximated over the entire range of V_{ds} as

$$I_t \equiv I_f\, \overline{F_e}\left(V_{ds}\right) + I_r\, F_e\left(V_{ds}\right).$$

Note that the transition of I_t around V_{ds} is not smooth. The next section of the chapter addresses this issue.

6.4 Completing the Model

The behavior of an MOS transistor around $V_{ds} = 0$ is dependent on the region of operation. In the subthreshold region, full current is delivered for $|V_{ds}| \geq V_o$. For $|V_{ds}| < V_o$, current magnitude decreases, reaching zero at $V_{ds} = 0$. This behavior is approximated by the expression

$$I_{ds} = I_t \tanh\left(\frac{3V_{ds}}{V_o}\right).$$

The hyperbolic tangent argument ensures that $I_{ds} \approx I_t$ when $V_{ds} \geq V_o$. Also note $\mathrm{sgn}(I_{ds}) = \mathrm{sgn}(V_{ds})$, correctly modeling bidirectional device behavior.

In the space-charge limited region, current derating begins at $V_{ds} = V_{gs} - V_t$ in the forward direction, and at $V_{sd} = V_{gd} - V_t$ in the reverse direction. Using Fermi functions, the previous equation can be modified to approximate this behavior, producing

$$I_{ds} = I_t \tanh\left(\frac{3V_{ds}}{V_o + V_f + V_r}\right),$$

where

$$V_f = V_{gs}\,\overline{F}_e\,(V_{gs} - V_t)\,\overline{F}_e\,(V_{ds})$$

and

$$V_r = V_{gd}\,\overline{F}_e(V_{gd} - V_t)\,F_e\,(V_{ds}).$$

This expression is the complete steady-state MOS transistor model in the Ana library. In accordance with the modeling guidelines in Chapter 5, the linear capacitances C_{gb}, C_{db}, C_{sb}, C_{gs}, and C_{gd} also are included, where b is the reference node. Key features of the model are graphed on the next four pages. These graphs are simulation results from Ana.

Chapter 7
The Transconductance Amplifier

7.1 Introduction

Most analog VLSI circuits can be simulated in Ana by using the MOS transistor model developed in Chapter 6, in conjunction with resistor and capacitor gates. Simulation is more efficient in several ways, however, if atomic models are created for commonly used circuits. In an atomic model, the description of each transistor can be simplified to reflect its role in the circuit. The model may be accurate for only certain regions of operation, with range checking to indicate improper usage. Nodes that are not accessible to the user can be eliminated, and replaced by a closed-form approximation of subcircuit behavior.

The technique of circuit modeling is well illustrated by the transconductance amplifier gate in Ana. The circuit schematic and gate symbol are shown on the next page. The circuit has one active current output, I_{out}, which is a function of the four user-accessible potentials V_1, V_2, V_{set}, and V_{out}, and of the two internal potentials V and V_m. All six transistors typically operate in the subthreshold range. The output current is a differential amplification of the potentials V_1 and V_2, scaled by a current controlled by V_{set}.

An atomic modeling strategy is easily applied to the transconductance amplifier. The internal nodes corresponding to V and V_m are eliminated in the Ana gate model. Gate behavior is accurate only for values of V_{set} that ensure subthreshold operation of all transistors; the user is informed of incorrect usage by a warning indicator that appears on the symbol. The output impedance of all six circuit transistors are lumped into a single term, and many other simplifications are made. These techniques are described in detail in this chapter.

7.2 Basic Amplification Function

The amplification output function $I_{out}(V_1, V_2, I_{set})$ can be easily derived, while eliminating the nodes corresponding to the potentials V and V_m from the gate description. As noted on the schematic diagram, $I_{out} = I_1 - I_2$ and $I_{set} = I_1 + I_2$. Dividing I_{out} by I_{set}, and assuming Q_2 and Q_3 are in subthreshold, yields the expression

$$I_{out} = I_{set}\frac{I_1 - I_2}{I_1 + I_2} = I_{set}\frac{\exp(V_1 - V) - \exp(V_2 - V)}{\exp(V_1 - V) + \exp(V_2 - V)},$$

where all potentials are scaled by the inverse of V_o. A factor of $\exp(-V)$ can be eliminated from the fraction, resulting in the expression

$$I_{out} = I_{set}\frac{\exp(V_1) - \exp(V_2)}{\exp(V_1) + \exp(V_2)}.$$

After algebraic manipulation, the simplified expression

$$I_{out} = I_{set} \tanh\left(\frac{V_1 - V_2}{2V_o}\right) \equiv I_{amp}$$

results, where all potentials are rescaled to correct units. As earlier stated, the circuit functions as an amplifier of the difference $V_1 - V_2$ over a domain of approximately $4V_o$ volts. The range of the function is scaled by I_{set}, which is controlled by V_{set}. In the following sections, several important additional characteristics of the circuit are modeled, leading to the formulation of the complete function I_{out}.

7.3 Computing the Scaling Current

In Section 7.2, the term I_{set} was not defined. If transistor Q_1 always is fully on, this definition is simply

$$I_{set} = I_c\left(V_{set}\right),$$

where I_c is taken from the MOS transistor model definition. However, the V_{ds} of Q_1 may not always be above V_o, which is the necessary condition for the transistor to be on. Recall that $I_{set} = I_1 + I_2$. The dominant term in this sum is the current corresponding to the voltage $\max(V_1, V_2)$. The voltage V is therefore approximately equal to $\max(V_1, V_2) - V_{set}$. If $\max(V_1, V_2) - V_{set} < V_o$, the current $I_c(V_{set})$ is linearly derated, as a result of the linear characteristic of the hyperbolic tangent function. The expression

$$I_{set} = I_c\left(\min\left(V_{set}, \max\left(V_1, V_2\right)\right)\right)$$

is a good approximation of this behavior. The min and max functions are implemented with Fermi functions, as shown in Chapter 5.

7.4 Computing the Output Impedance

The output transistors Q_5 and Q_3 both have a finite output impedance, which should be reflected in the amplifier model. Using the output impedance definition G of the MOS transistor and Fermi functions, the expression

$$I_{amp}\left(F_e(V_1 - V_2)G(V_{out}) + \overline{F}_e(V_1 - V_2)G(V_{dd} - V_{out})\right)$$

supplies the necessary impedance.

When a transconductance amplifier is fabricated, the output current function is not always an ideal hyperbolic tangent. One type of error introduced in manufacturing results in $\tanh(\infty) \neq -\tanh(-\infty)$. Two simulation parameters λ_p and λ_n can be added to the previous expression to simulate this effect, resulting in the new expression

$$I_{amp}\left(F_e(V_1 - V_2)G(V_{out})\lambda_n + \overline{F}_e(V_1 - V_2)G(V_{dd} - V_{out})\lambda_p\right) \equiv I_{tot}.$$

Note that λ_p and λ_n both default to unity, for an ideal amplifier model.

7.5 Output Voltage Constraints

If V_{out} is too large or too small, the circuit does not function as an amplifier. If V_{out} is larger than V_{dd}, the current through transistor Q_5 changes direction, drawing current from the output. Note that, if $V_1 > V_2$, V_g of Q_5 is at a potential $V_{dd} - V_{set}$, and significant reverse current flows into Q_5 for $V_{out} > V_{dd}$. However, if $V_1 < V_2$, V_g of Q_5 is at a potential V_{dd}, and noticeable reverse current flows from Q_5 only if $V_{out} > V_{dd} + V_{set}$ The addition of the term

$$I_{top} \equiv -\left(F_e(V_1 - V_2)I_c(V_{out} - V_{dd}) + \overline{F}_e(V_1 - V_2)I_c(V_{out} + V_{set} - V_{dd})\right)$$

to I_{amp} models this effect.

If V_{out} is too small, the differential transistor pair Q_2 and Q_3 ceases to operate correctly. If $V_2 > V_1$, $V = V_2 - V_{set}$ during normal operation. If $V_{out} \leq V_2 - V_{set}$, V is forced to a lower potential to maintain the proper magnitude of current through Q_3. If $V_{out} \leq V_1 - V_{set}$, however, Q_2 and thus Q_5 conduct. A large output current results, and the amplifier ceases to function as desired.

If $V_2 < V_1$, similar breakdown results. In this case, $V = V_1 - V_{set}$ during normal operation. If $V_{out} \leq V_1 - V_{set}$, V remains at $V = V_1 - V_{set}$, and the current through transistor Q_3 changes direction. If $V_{out} \leq V_2 - V_{set}$, a significant reverse current flows through Q_3 to the output, and the circuit does not operate correctly. For both $V_2 > V_1$ and $V_2 < V_1$, the effect of low V_{out} can be modeled by the addition of the term

$$I_{bot} = I_c\left(\min\left(V_1, V_2\right) - V_{out}\right)$$

to I_{amp}.

7.6 Completing the Model

Using the previous definitions, the circuit is modeled completely by the expression

$$I_{out} = I_{tot} + I_{top} + I_{bot}.$$

The model is accurate only when $V_{set} < V_t$, ensuring subthreshold operation of all transistors for normal V_{out}. To meet the transient modeling requirements, parasitic capacitances are added between each pair of nodes, and between each node and ground. Graphs of key model features are shown at the end of the chapter. These graphs are simulation results from Ana.

When the circuit is fabricated, the hyperbolic tangent amplifier characteristic is not always centered about $V_1 - V_2 = 0$. To allow simulation of this error, the simulation parameter δ_v is defined. In all the model equations previously presented,

$V_1 \to V_1 + \frac{1}{2}\delta_v$, and $V_2 \to V_2 - \frac{1}{2}\delta_v$. The default value of δ_v is zero, providing an ideal amplifier model.

7.7 The Wide-Range Transconductance Amplifier

In many system applications, the restrictive I_{bot} behavior of the transconductance amplifier cannot be tolerated. In these applications, a variant of the circuit, called the *wide-range transconductance amplifier*, is used. The circuit schematic and gate symbol are shown on the next page. Except for a less restrictive I_{bot} function, the circuit behaves in the same way as the standard transconductance amplifier.

Referring to the schematic, the behavior of transistor Q_8 for low V_{out} is symmetric to the behavior of transistor Q_9 for high V_{out}. Thus, the wide-range amplifier function I_{bot} can be deduced from the function I_{top} to be

$$I_{bot} \equiv -\left(F_e(V_1 - V_2)I_c(-V_{out}) + \overline{F_e}(V_1 - V_2)I_c(-V_{out} - V_{set})\right).$$

In all other respects, the standard and wide-range transconductance amplifiers are identically modeled in Ana.

Chapter 8
Rectification Circuits

8.1 Introduction

The circuits described in Chapter 7 transform a differential voltage into a bidirectional current, preserving the sign of the input signal. In applications such as power estimation, demodulation, and absolute-value computation, a unidirectional output signal is desired regardless of the input polarity. The *half-wave rectification operator* achieves this result by ignoring input signals of one sign, whereas the *full-wave rectification operator* produces a single polarity output by processing both positive and negative inputs identically. MOS circuits that implement these functions were designed by Massimo Sivilotti at Caltech, and are shown on the next page.

The atomic models for both rectification circuits are described in this chapter. Both circuits have one active output I_{out}, which is a function of the voltages V_1, V_2, V_{set}, and V_{out}. The descriptions of I_{out} contain expressions originally defined for the transconductance amplifier, illustrating the hierarchical nature of the Ana modeling style.

8.2 Basic Rectification Function

The basic descriptions of both rectifiers are derived from the transconductance amplifier function $I_{amp}(V_1, V_2) = I_{set} \tanh\left(\frac{V_1 - V_2}{2V_o}\right)$, shown in Chapter 7. For the half-wave rectifier, note that I_{mir} never can be a negative current. Thus, if $V_1 \geq V_2$, $V_{mir} \rightarrow V_{dd}$ to ensure $I_{mir} = 0$. As a result, $I_{out} = 0$, and input signals of positive polarity are rejected. When $V_2 \geq V_1$, the current mirror formed by Q_1 and Q_2 operates normally, and I_{out} is the negation of the amplifier output current. In this manner, only input signals of negative polarity are passed to the output. The expression

$$I^h_{amp}(V_1, V_2) \equiv -I_{amp}(V_1, V_2)\, \overline{F}_e(V_1 - V_2)$$

captures this behavior. From the schematic, the basic function of the full-wave amplifier is $I^h_{amp}(V_1, V_2) + I^h_{amp}(V_2, V_1)$, or, after simplification,

$$I^f_{amp}(V_1, V_2) \equiv I_{amp}(V_1, V_2)\left(F_e(V_1 - V_2) - \overline{F}_e(V_1 - V_2)\right).$$

As desired, this expression is essentially the absolute value of the transconductance amplifier basic function.

The output impedance of both circuits is simply the impedance of the active output transistor, which can be expressed by the MOS transistor function G. The two previous equations can be rewritten to include this impedance. It is convenient to include the transconductance amplifier fabrication parameters λ_n and λ_p in this restatement, resulting in the new definitions

$$I^h_{tot} \equiv -\lambda_p G(V_{dd} - V_{out}) I^h_{amp}(V_1, V_2)$$

and

$$I^f_{tot} \equiv G(V_{dd} - V_{out}) I_{amp}(V_1, V_2) \left(\lambda_n F_e(V_1 - V_2) - \lambda_p \overline{F}_e(V_1 - V_2) \right).$$

8.3 Output Voltage Constraints

The current mirror formed by Q_1 and Q_2 prevents the transconductance amplifier subfunctions I_{bot} and I_{top} from affecting rectification behavior, if V_{set} is set to ensure subthreshold amplifier operation. Thus, in the half-wave rectifier, the only component influenced by V_{out} is transistor Q_2, the current of which can reverse direction if V_{out} is sufficiently large. This transistor performs a similar function to the output transistor Q_5 in the transconductance amplifier. Following this previous analysis, the expression

$$I^h_{top} \equiv -\left(F_e(V_1 - V_2) I_c(V_{out} + V_{set} - V_{dd}) + \overline{F}_e(V_1 - V_2) I_c(V_{out} - V_{dd}) \right)$$

is obtained. This term is added to I^h_{tot} to model the half-wave rectifier completely.

From the schematic, the expression I^f_{top} for the full-wave rectifier is simply $I^h_{top}(V_1, V_2) + I^h_{top}(V_2, V_1)$. For a functional model, this expression can be approximated as

$$I^f_{top} \equiv -I_c(V_{out} - V_{dd} + V_{set}).$$

This term is added to I^f_{tot} to model the full-wave rectifier completely.

8.4 Completing the Model

The steady-state models for the half-wave and full-wave rectifiers are complete. These models are accurate for $V_{set} \le V_t$, ensuring subthreshold operation of all transistors for normal V_{out}. As in the transconductance amplifier, the simulation parameter δ_v is defined to model fabrication variations, and in all previous equations $V_1 \rightarrow V_1 + \frac{1}{2}\delta_v$ and $V_2 \rightarrow V_2 - \frac{1}{2}\delta_v$. Parasitic capacitances are added between each pair of nodes, and between each node and the reference node, to meet transient modeling requirements. Graphs of the simulation results from Ana are shown on the next page.

Chapter 9
The Horizontal Resistor Circuit

9.1 Introduction

Note this chapter, and the corresponding Analog gate, do not describe the horizontal resistor circuit in "Analog VLSI and Neural Systems," but an earlier, inferior circuit that was discarded after the invention of the circuit described in the book.

The polysilicon resistors available in a standard MOS fabrication process are not suitable for widespread use in analog VLSI systems. High-value resistors consume too much chip area, whereas low-value resistors dissipate an unacceptable amount of power. The horizontal resistor circuit, shown on the next page, provides a nonlinear variable resistance that can replace polysilicon resistors in many applications.

Referring to the schematic, the resistor terminals are the nodes associated with potentials V_{in} and V_{out}. The potentials V_{low} and V_{high} are complementary signals that control the resistance value. In the Ana circuit model, the two active currents I_{in} and I_{out} are defined as a function of all four potentials. V_{low} and V_{high} are normally set to ensure that all transistors operate in the subthreshold region.

9.2 Basic Resistance Function

For the circuit to operate as intended, the control voltages V_{low} and V_{high} must be set such that $I_{high} = I_{low} \equiv I_{set}$. Given this condition, by conservation of current, $I_{in} = I_{out} \equiv I$. Transistors Q_3 and Q_6 carry an equal current I_1, and transistors Q_4 and Q_5 carry an equal current I_2, as a result of circuit symmetry. By Kirchoff's current law, $I = I_2 - I_1$, and $I_{set} = I_2 + I_1$. Dividing I by I_{set}, and assuming all transistors operate in the subthreshold region, yields the expression

$$I = I_{set}\frac{I_2 - I_1}{I_2 + I_1} = I_{set}\frac{\exp(V_{in} - V_l) - \exp(V_{out} - V_l)}{\exp(V_{in} - V_l) + \exp(V_{out} - V_l)},$$

where all potentials are scaled by the inverse of V_o. By eliminating a factor of $\exp(-V_l)$ from the fraction, the previous equation can be rewritten as

$$I = I_{set}\frac{\exp(V_{in}) - \exp(V_{out})}{\exp(V_{in}) + \exp(V_{out})}.$$

After algebraic manipulation, the simplified expression

$$I = I_{set}\tanh\left(\frac{V_{in} - V_{out}}{2V_o}\right)$$

is obtained, where all potentials are rescaled to correct units. Notice that the circuit functions as a linear resistance over a range of approximately $4V_o$ volts. The slope of the linear region, and thus the resistance value, is determined by I_{set}, which is controlled by the complementary potentials V_{low} and V_{high}. In the following sections,

this expression is expanded to model circuit behavior if $I_{high} \neq I_{low}$, and if V_{in} and V_{out} are both too low or too high for correct operation of Q_1 or Q_2.

9.3 Definition of Control Currents

In this section, complete expressions for the control currents I_{low} and I_{high} are derived. First consider I_{low}. If transistor Q_2 is on and is operating in the subthreshold region, $I_{low} = I_c(V_{low})$. The potentials V_{in} and V_{out}, however, also can affect this current. Note that I_{low} is primarily carried by the N-channel transistor (Q_4 or Q_6) with the highest gate voltage. Thus, the potential V_l has the value $\max(V_{in}, V_{out}) - V_{low}$. If this potential is less than V_o, the current through transistor Q_2 is derated. The expression

$$I_c(V_{low})F_e\Big(V_{low} - \max(V_{in}, V_{out})\Big)$$

approximates this behavior.

The analysis of I_{high} is similar. If transistor Q_1 is on and is operating in the subthreshold region, $I_{low} = I_c(V_{dd} - V_{high})$. This current also can be affected by the potentials V_{in} and V_{out}. Note that I_{high} is primarily carried by the P-channel transistor (Q_3 or Q_5) with the lowest gate voltage. Thus, the potential V_h has the value $\min(V_{in}, V_{out}) - (V_{dd} - V_{high})$. If this potential exceeds $V_{dd} - V_o$, the current through transistor Q_1 is derated. The expression

$$I_c(V_{dd} - V_{high})F_e\Big(\min(V_{in}, V_{out}) - (V_{dd} - V_{high})\Big)$$

approximates this behavior.

As in the transconductance amplifier, when the horizontal resistor circuit is fabricated, the nonlinear resistance function can display the error $\tanh(\infty) \neq -\tanh(-\infty)$. The parameters λ_p and λ_n can be added to the previous equations to model this behavior, resulting in the final expressions

$$I_{low} \equiv \lambda_n I_c(V_{low})F_e\Big(V_{low} - \max(V_{in}, V_{out})\Big)$$

and

$$I_{high} \equiv \lambda_p I_c(V_{dd} - V_{high})F_e\Big(\min(V_{in}, V_{out}) - (V_{dd} - V_{high})\Big).$$

If $V_{low} > V_t$ or $V_{dd} - V_{high} > V_t$, the circuit is not in the subthreshold region of operation, and the model is not longer accurate. If either condition occurs, a warning indicator appears on the schematic symbol.

9.4 Completing the Model

The results of Sections 9.2 and 9.3 can be combined with Fermi functions to describe the currents I_{in} and I_{out}. If $I_{res} \equiv \tanh\left(\frac{V_{in}-V_{out}}{2V_o}\right)$, the definitions

$$I_{in} \equiv \left(\overline{F}_e(V_{in} - V_{out})I_{low} + F_e(V_{in} - V_{out})I_{high}\right) I_{res}$$

and

$$I_{out} \equiv \left(\overline{F}_e(V_{in} - V_{out})I_{high} + F_e(V_{in} - V_{out})I_{low}\right) I_{res}$$

are exact for $I_{high} = I_{low}$. If $I_{high} \neq I_{low}$, the expressions are a suitable approximation for a functional model.

To complete the steady-state model, the appropriate output impedance is added to each expression using the MOS transistor function G, producing the final definitions

$$I_{in} \equiv \left(\overline{F}_e(V_{in} - V_{out})G(V_{in})I_{low} + F_e(V_{in} - V_{out})G(V_{dd} - V_{in})I_{high}\right) I_{res}$$

and

$$I_{out} \equiv \left(\overline{F}_e(V_{in} - V_{out})G(V_{dd} - V_{out})I_{high} + F_e(V_{in} - V_{out})G(V_{out})I_{low}\right) I_{res}.$$

As in previous circuit models, the simulation parameter δ_v is defined to model fabrication variations, and distributed between the potentials V_{in} and V_{out}. Parasitic capacitances are added between each pair of nodes, and between each node and the reference node, to meet transient model requirements. Simulation results from Ana are shown on the next page.

Chapter 10
The Ganglion Circuit

10.1 Introduction

In biological systems, information often is encoded as the average value of a pulse train of standard height. The ganglion circuit, shown on the next page, imitates this neuronal behavior, generating a pulse stream $V_{out}(t)$ that has an average potential proportional to the input current I_{in}. The pulses switch between V_{dd} and the reference node, with a time constant set by the control voltage V_{leak}. The Ana model defines the active currents I_{in} and I_{out} as functions of the potentials V_{in}, V_{out} and V_{leak}.

10.2 Principles of Circuit Operation

Referring to the schematic, the transistors Q_3, Q_4, Q_5, and Q_6 function as a pair of inverters. These transistors can be functionally modeled as a noninverting buffer between V_{in} and V_{out}, which sets $V_{out} = V_{dd}$ if V_{in} is greater than the buffer threshold V_{th}, and sets V_{out} equal to the reference voltage if $V_{in} < V_{th}$.

It is convenient to analyze qualitative circuit behavior from the initial condition $V_{in} = V_{th} - \Delta V_1$. This condition implies V_{out} is at the reference voltage, and transistor Q_2 is off. The capacitor C_1 charges at a rate determined by I_{in} until $V_{in} = V_{th}$. At this instant the buffer switches, forcing the new conditions $V_{out} = V_{dd}$ and $V_{in} = V_{th} + \Delta V_2$. As a result, transistor Q_2 is on, and C_1 discharges at a rate determined by the current $I_{leak} - I_{in}$, until $V_{in} = V_{th}$. At this point, the buffer again switches, returning V_{out} to the reference voltage and setting $V_{in} = V_{th} - \Delta V_1$.

The quantities ΔV_1 and ΔV_2 can be determined easily, using conservation of charge in the system during buffer switching. The instant before the buffer switches to $V_{out} = V_{dd}$, capacitors C_1 and C_2 are connected in parallel between V_{in} and the reference node, and the total charge in the system is $Q_- = (C_1 + C_2)V_{th}$. After the buffer switches, C_1 and C_2 form a capacitive divider between V_{dd} and the reference node, and the total charge in the system is $Q_+ = C_1(V_{th} + \Delta V_2) - C_2(V_{dd} - V_{th} - \Delta V_2)$. Equating Q_+ and Q_- yields, after algebraic manipulation, the expression

$$\Delta V_2 = \left(\frac{C_2}{C_1 + C_2}\right) V_{dd}.$$

The same technique can be used to determine ΔV_1. The instant before the buffer output switches from V_{dd} to the reference voltage, the capacitive voltage divider topology is in place, and the total charge in the system is $Q_+ = C_1 V_{th} - C_2(V_{dd} - V_{th})$.

After the buffer switches, C_1 and C_2 form the parallel topology, and $Q_- = (C_1 + C_2)(V_{th} - \Delta V_1)$. Equating total charge and solving for ΔV_1 shows that

$$\Delta V_1 = \left(\frac{C_2}{C_1 + C_2}\right) V_{dd} = \Delta V_2,$$

as symmetry would suggest.

The pulse duration period, τ_d, when $V_{out} = V_{dd}$, and the pulse refractory period, τ_r, when V_{out} equals the reference voltage, can both be determined easily. Applying the capacitor definition for C_1 when V_{out} is high gives the expression

$$I_{cap} = C_1 \frac{dV_{in}}{dt} = C_1 \frac{-\Delta V_2}{\tau_d} = I_{in} - I_{leak}.$$

Solving for the pulse duration yields

$$\tau_d = \left(\frac{C_1 C_2}{C_1 + C_2}\right) \left(\frac{V_{dd}}{I_{leak} - I_{in}}\right).$$

This technique also can be applied when V_{out} is low. The capacitor definition for C_1 is

$$I_{cap} = C_1 \frac{dV_{in}}{dt} = C_1 \frac{\Delta V_1}{\tau_r} = I_{in}.$$

Algebraic manipulation yields the final expression

$$\tau_r = \left(\frac{C_1 C_2}{C_1 + C_2}\right) \left(\frac{V_{dd}}{I_{in}}\right).$$

As expected, as $I_{in} \rightarrow 0$, $\tau_r \rightarrow \infty$, and the average value of V_{out} is the reference voltage. Also, as $I_{in} \rightarrow \infty$, $\tau_d \rightarrow 0$ and $\tau_r \rightarrow 0$, and the average value of V_{out} is V_{dd}.

10.3 The Ana Model

The Ana ganglion model is not an abstraction of the switching dynamics of the circuit. Rather, the state capacitors C_1 and C_2 remain in the model. Efficient descriptions are substituted for the noninverting buffer $Q_3 \ldots Q_6$ and the leakage circuit Q_1 and Q_2.

The noninverting buffer can be modeled as a nonlinear, voltage-controlled voltage source with an output resistance R_s. This source can be described using a Fermi function, as

$$I_{sw} = \frac{1}{R_s}\Big(V_{dd} F_e \left(V_{th} - V_{in}\right) - V_{out}\Big).$$

I_{out} is the sum of I_{sw} and the current contributed by C_2.

If $V_{in} < V_{th}$, the current I_{leak} is set by Q_1 to be $I_c(V_{leak})$. If $V_{in} > V_{th}$, however, the noninverting buffer output is high, Q_2 is off, and $I_{leak} = 0$. The expression

$$I_{leak} = G(V_{in})I_c(V_{leak})F_e\left(V_{th} - V_{in}\right)$$

approximates this behavior, including the transistor output impedance. The current I_{in} is the sum of I_{leak} and the current contributed by C_1 and C_2.

With the addition of a parasitic capacitor between each pair of nodes, and between each node and the reference node, the ganglion model is complete. The model is correct only if Q_1 and Q_2 are operating in the subthreshold region; if $V_{leak} > V_t$, a warning indicator appears on the gate symbol. Graphs of Ana simulation results appear on the following pages.

Chapter 11
Topics for Further Research

The evolution of Ana from the prototype described in this thesis to a production tool is in many ways a straightforward process. Algorithm refinement and code optimization can improve program efficiency, and the user interface can be enhanced by fully integrating Ana into the LOG environment. An expansion of the gate library, however, presents several difficulties.

Creating the complete software definition of an Ana gate is a large task; the horizontal resistor gate, for example, is defined by approximately 1200 lines of Pascal source code, which required 50 hours of programmer time to generate. Recall that for a gate G^k with p pins, the Newton–Raphson method requires, for each pin l, the evaluation of the set of partial derivatives

$$\frac{\partial P_l^k}{\partial v_1}, \ldots, \frac{\partial P_l^k}{\partial v_p}$$

in addition to the behavioral function P_l^k. The generation and efficient coding of these symbolic derivatives are tedious and time-consuming manual tasks.

A LOG tool that transforms a set of functional descriptions $P_1^k \ldots P_p^k$ into an efficient software definition is clearly needed. Standard techniques in compiler design and symbolic mathematical manipulation are applicable to the design of a *simulation assembler*. This tool simplifies the conversion of a functional description into a software definition; the generation of the functions $P_k^1 \ldots P_k^p$ from a circuit schematic remains a manual operation.

A *simulation compiler* is a tool that generates a closed-form behavioral model of an analog circuit. Using this tool, a hierarchical model of a large analog system can be created easily. Circuit theory provides a compilation algorithm that generates exact descriptions of linear circuits. Unfortunately, there is no known method for generating an exact, closed-form description of an arbitrary nonlinear circuit.

Circuits designed in the analog VLSI style developed at Caltech, however, have several interesting semantic properties. Usually, the MOS transistor and the MOS capacitor are the only circuit primitives. Typically, transistors are used either in the subthreshold region or as components in simple logic gates. The MOS capacitor can be considered as a linear element without changing its qualitative behavior in most circuits.

In addition, an exact, completely closed-form description of a nonlinear circuit is not necessary in this application. Approximate descriptions, similar to the circuit models presented in this thesis, are sufficient for functional simulation. Eliminating most of the intermediate variables in a circuit model is an acceptable alternative to generating a closed-form expression.

Under these weakened source and object constraints, simulation compilation is a tractable research topic. This research may be related to fields as diverse as linear and nonlinear systems theory, deterministic and stochastic estimation theory, electronic circuit design, symbolic mathematical manipulation algorithms, and production-rule programming.

Bibliography

[1]Gupta, Gobal K., Ron Sacks-Davis, and Peter E. Tischer, *A Review of Recent Developments in Solving ODEs,* ACM Computing Surveys, Vol 17, No 1, pp. 4–47.

[2]Scarborough, James B., *Numerical Mathematical Analysis,* Baltimore, Maryland: John Hopkins University Press, 1958, pp. 192–211, 525–527.

[3]Action, Forman S., *Numerical Methods That Work,* New York, New York: Harper & Row, 1970, pp. 54–55.

[4]Mead, Carver A. and Mary Ann C. Maher, "A Charge-Controlled Model for Submicron MOS," *Proceedings of the Colorado Microelectronics Conference,* 1986.