Analog Integrated Circuits and Signal Processing 3, 163-180 (1993)

Sframe: An Efficient System for Detailed DC Simulation of Bipolar Analog Integrated Circuits Using Continuation Methods

ROBERT MELVILLE, SHAHRIAR MOINIAN AND PETER FELDMANN
AT&T Bell Laboratories, Murray Hill, NJ
LAYNE WATSON
Departments of Computer Science and Mathematics, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061-0106

Abstract. This work describes a simulation package for detailed studies of biasing networks for bipolar transistors. A sophisticated transistor model is introduced which captures many second-order effects, but which causes convergence difficulties for many existing methods used for computing an operating point. Artificial parameter numerical continuation techniques are introduced, then, as a robust and efficient means of solving bias networks employing our model. Sensitivity studies and natural parameter continuation studies based on the computed operating point (or points) are also discussed.

1. Introduction

Design analysis of high-performance analog integrated circuits requires detailed and accurate simulation of the dc behavior of the chip. Such analysis, which become an even more integral part of the design for advanced bipolar transistor technologies, include: computation of the dc operating point (or points) of the circuit; sensitivity studies of one or more outputs to one or more circuit parameters; design simulations at the extremes, dictated by variations in the fabrication process, and the electrical and environmental conditions in which the circuit will be operating, such as power supply and temperature variations; analyses and optimization of yield or performance in the face of statistical variation of process parameters. Of course, such analyses are only as good as the underlying device models!

In this paper we describe an experimental system called Sframe which is being incorporated into the design for manufacturability initiative at the Reading Works of AT&T Bell Laboratories. Our system is able to perform detailed and accurate dc analyses of integrated circuits containing several hundred transistors to be fabricated in a relatively complex junction isolated complementary technology.

The work of the fourth author was supported in part by Department of Energy grant DE-FG05-88ER25068, National Science Foundation grant CTS-8913198, and Air Force Office of Scientific Research grant 89--0497.

Highlights of our system include:

- Robust computation of the operating point of a circuit using an efficient continuation method; moreover, the method is able to detect multiple operating points.

  Generally speaking, continuation methods for operating point computation have a reputation in the simulation community for being too slow to be practical for any but the smallest of circuits. One of the conclusions of our work is that, when properly implemented, continuation techniques based on modern homotopy algorithms for operating point computation exhibit unsurpassed robustness with reasonable cost.
- A state-of-the-art four-terminal bipolar transistor dc model which treats various second-order effects not considered in simpler models. This model has been appropriately modified for use with continuation methods.
- An "incremental" facility which allows the operating point of a circuit to be updated quickly after a relatively small change to one or more simulation parameters. This facility is especially useful for exploration of a "design space" during statistical optimization.
- Parameter studies using continuation methods which can identify qualitatively different operating modes of the circuit. The numerical codes used to perform these studies are able to cope with turning points and folds in the solution manifold, which indicate more

than one solution for parameter values in a certain interval. Such analysis provide insight into both the quantitative and qualitative behavior of the design. A facility for *continuation to a target point* allows a designer to calculate the exact setting for a circuit parameter which causes an output variable to equal a desired value. This is also useful in statistical design.

- Analytically correct dc sensitivity analyses of a user-defined performance function to temperature, any device model parameter or any circuit parameter. Both direct and adjoint techniques are supported. These methods are superior to the perturbation technique, often used in simulators to estimate senstivities, which are too slow and inaccurate for any but the simplest kinds of sensitivity analyses.
- A novel software architecture in which the user's circuit is described by defining appropriate classes in C++. Sframe is designed in a highly modular fashion, and different numerical codes can be installed quickly through narrow, well defined interfaces. Moreover, the design of Sframe takes advantage of so-called "automatic differentiation" techniques which allow derivatives of model expressions to be computed accurately in a fashion which is transparent to the user. Such derivatives are needed for continuation and sensitivity studies.

Section 2 describes the use of *numerical continuation* methods in our program. A distinction between "artificial parameter" and "natural parameter" methods is drawn. The use of artificial parameter continuation for computation of operating points has been described elsewhere [1–3] so is reviewed here only briefly. Section 2.1 describes the various continuation options provided. Section 2.2 discusses *incremental* operating point computation, in which the operating point of a circuit is to be updated after a relatively small change to one or more circuit parameters.

In Section 3, we motivate the need for a highly accurate and detailed *transistor model*, and show how a continuation parameter is incorporated into the model for robust and efficient operating point computation. Section 4 discusses *automatic differentiation* to device model equations, and shows how Sframe takes advantage of this technique to provide almost any conceivable sensitivity information in a convenient fashion. In Section 5, we describe our experience with writing a simulation program in the C++ language, also using C++ as the input or "netlist" language. Finally, in Section 6, performance data on several designs are presented. All of our examples are taken from current industrial designs, and some of them are large by analog circuit standards (e.g., several hundred transistors).

2. Continuation Methods in Simulation

Continuation (homotopy) methods [4–6] provide both a theoretical and implementation basis for dc analysis of nonlinear networks. Consider the formulation of the operating point equations using Kirchhoff's laws. In the so-called *modified nodal* formulation [7], one introduces a voltage unknown for each node in the circuit, and an additional unknown for the current through each voltage source, then writes an equation which expresses Kirchhoff's current law at each node and Kirchhoff's voltage law across each voltage source. This gives n equations in n unknown voltages and currents.

The standard form for such equations is

$$F(x, a) = 0 \tag{1}$$

where, for the fixed vector of parameters a, $F(-, a)$ is a mapping from R^n into R^n, the set of real n-vectors, and x is a vector partitioned $x = (i; v)$ for current and voltage unknowns. The m-vector a represents circuit parameters. These equations can be highly nonlinear and standard Newton-Raphson iteration [8] typically exhibits only local convergence. Therefore, we are motivated to consider more robust and globally convergent procedures for operating point computation. Continuation theory considers an equation

$$H(x, \mu, a) = 0 \tag{2}$$

where x and a are as in (1) and the p-vector μ represents one or more *continuation parameters*, so that $H(-, a)$ in (2) is a mapping from R^{n+p} into R^n; i.e., there are more unknowns than equations. In other words, the system of equations is underdetermined. Thus, a "solution" to (2) is no longer a single point, but rather a curve or surface in R^{n+p}. In the remainder of the paper, we will restrict ourselves to the case $p = 1$, and assume that the parameter vector a in R^m is fixed. In the sequel, unless necessary, a will not be written explicitly.

In the continuation paradigm, one designs a function H such that a solution x_0 to the equation $H(x, \mu_0) = 0$ is already known or easily obtained for some fixed value μ_0; i.e., $H(x_0, \mu_0) = 0$. If, in addition, H is designed so that $H(x, \mu) = F(x)$ identically in x when $\mu = \mu_1$, then a solution to $H(x^*, \mu_1) = 0$ provides a solution to $F(x^*) = 0$. Examples of such a construction will be given later.

Assuming that such a solution exists, i.e., $H(x^*, \mu_1) = 0$, supporting theory [9–11] shows that in most cases, under reasonable assumptions about the smoothness of H and the choice of a, the points (x_0, μ_0) and (x^*, μ_1) are connected by a path in $(n + 1)$-dimensional space. With a fixed, we can compute x^* by "tracking" this path in $(n + 1)$-dimensional (x, μ) space. To take a simple example, suppose that μ represents the ambient temperature of a circuit. For $\mu_0 = 25$°C, a solution to $H(x, \mu_0) = 0$ represents an operating point of the circuit at room temperature, where μ has the dimension of degrees Centigrade. As μ is varied from 25°C to an elevated temperature, say 50°C, the solution to $H(x, \mu) = 0$ tracks the state of the circuit at each temperature.

Packaged numerical codes are available to accomplish this "curve tracking," i.e., to generate a set of points (x, μ) which satisfy $H(x, \mu) = 0$ for μ in the interval $[\mu_0, \mu_1]$ and a fixed a. The user supplies an initial point (x_0, μ_0, a), then the curve tracking algorithm takes over. It predicts a local direction vector "along the curve" by evaluating the *Jacobian matrix* of H with respect to x and μ. Iterative application of a predictor-corrector scheme allows the algorithm to track the curve until $\mu = \mu_1$. Sophisticated packages, such as HOMPACK [11–12] or PITCON [13], dynamically adjust their step length to adapt to changes in the curvature of the path.

In order to use such packages, the user must supply a numerically accurate Jacobian matrix. This matrix is of the form

$$\begin{bmatrix} \dfrac{\partial H}{\partial x} & \dfrac{\partial H}{\partial \mu} \end{bmatrix} \tag{3}$$

evaluated at a point (x, μ) along the solution path. Our computational experience with dc analyses of bipolar networks indicates the finite-difference approximations to this Jacobian matrix are inefficient and unreliable.

The notion of "sweeping" a parameter is intuitive, but can be misleading. Consider the symmetric flip-flop shown in figure 1. Suppose we treat the supply voltage (V_{cc}) as the continuation parameter, μ, and "sweep" μ from 0 to 6 V. Figure 2a shows the complete solution to $H(x, \mu) = 0$ for this example, in which x is the dc state vector of the circuit. At a critical value of μ (about 0.7 V) the operating point equations exhibit a *bifurcation* [14]. The three branches to the right of the critical point represent the two stable states of the flip-flop along with the metastable state. The bifurcation

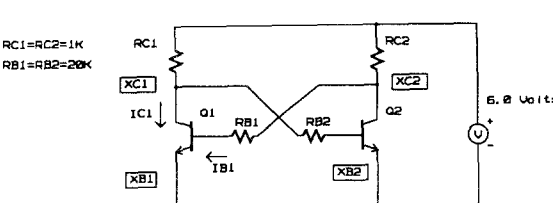


Fig. 1. Symmetric flip-flop.

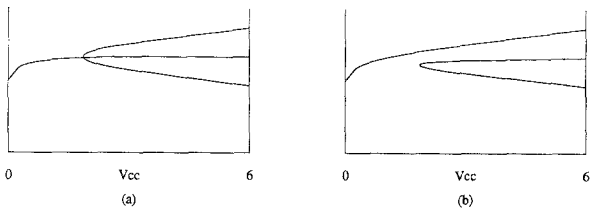


Fig. 2. Bifurcation diagrams for symmetric flip-flop.

diagram of figure 2a is valid only if the circuit is exactly balanced; if there is any asymmetry in the circuit, then the bifurcation diagram becomes the *unfolded* diagram of figure 2b; the bifurcation is gone, and only one solution is accessible from the start state x_0. Such an unfolding can be accomplished by suitable choice of the parameter vector a mentioned above. For example, suppose a encodes the values of the resistors and the scales of the transistors in the circuit. Any physical realization of the circuit will incur some imbalance in these values which can be modeled by appropriate slight perturbations in the a vector.

Another possibility, quite common in analog circuits, is a *turning point*. Consider the circuit of figure 3 taken from Ref. 15, in which the value of the input voltage is the continuation parameter μ. The output is taken as the current through this source, and is shown plotted against the source voltage. Note that for μ in a certain interval, the circuit exhibits more than one solution. At the end points of this interval, the solution manifold turns back on itself. This discussion is meant to show that the notion of "sweeping" might be a bit more complicated than it first appears. Turning points and (less often) bifurcations which are not unfolded *do* come up in practical analog circuit designs!

2.1. Artificial Parameter Continuation for Operating Point Computation

In the above examples, the continuation parameter has a natural circuit interpretation—voltage, temperature, etc. The V_{cc} continuation of figure 2 can be interpreted

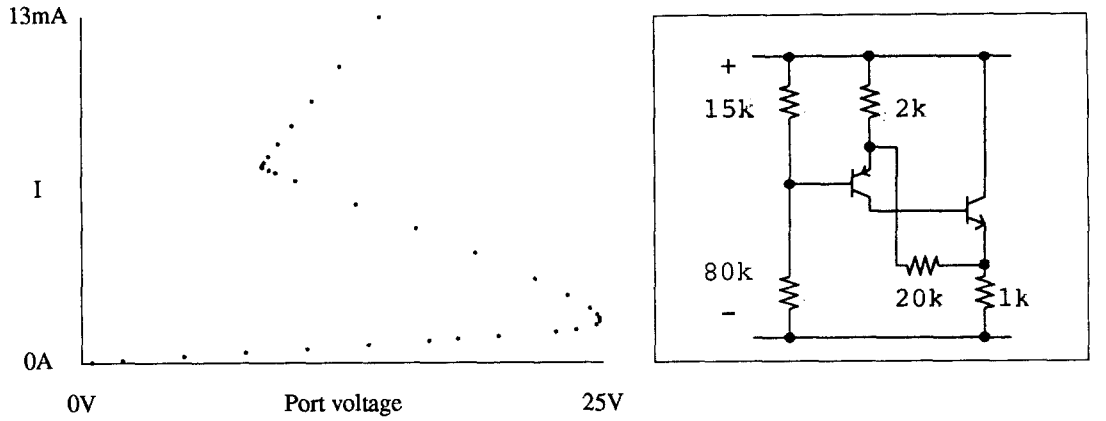


Fig. 3. Negative resistance circuit.

as an operating point computation starting from the trivial point of zero supply voltage and ending when the supply is "fully on." Because of the bifurcation, a numerical method used to track the solution manifold may falter at the point of the bifurcation, or (more likely) continue on through this point to the *metastable* state of the flip-flop. Neither of these situations is desirable. Instead, Sframe uses the notion of *artificial parameter* continuation to find an operating point. In this technique a parameter which need not have an obvious circuit interpretation is introduced into one or more nonlinear element models. Artificial parameter methods [10–11] generate smooth, bifurcation free paths which can be traversed quickly to the desired operating point.

As in the V_{cc} continuation above, the computation of an operating point when the artificial parameter is set to zero is trivial; moreover, when the artificial parameter reaches a value of one, the circuit has been retured to its original state. Consider, for example, the standard Ebers-Moll transistor model [36] of figure 4. A continuation parameter λ has been introduced which multiplies the current gains of the transistor. When λ equals zero, the model degenerates into a pair of back-to-back diodes. The transistor model actually used in Sframe is much more complicated than figure 4, however the continuation parameter is introduced into the complex model in much the same way; the detailed construction is described in Section 3.

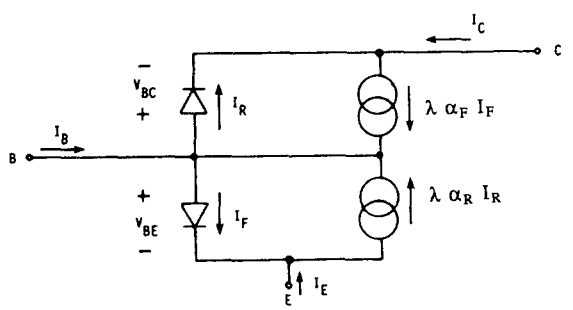


Fig. 4. Ebers-Moll model with continuation parameter.

Suppose we wish to find the operating point of a bipolar network containing transistors, diodes, resistors, and independent sources. Imagine all the transistors with the continuation parameter introduced as in figure 4. Now, consider the circuit when λ is set to zero. This so-called *start system* has a unique operating point, and it is easy to solve. In fact, it can be shown that the operating point equations are a diffeomorphism when λ is zero [2]. Thus, norm-reducing Newton methods [8, 16] work quite well, typically solving the circuit in a reasonable number of iterations (less than 30 for all the examples presented in Section 6). After solving the start system, use a continuation procedure to advance λ to 1; at this point, the transistor models are back to their original state. Points along the continuation path, for values of λ less than one, do not have much meaning to a designer, since they represent states of a circuit with a modified transistor model. Hence the term "artificial" parameter. As a notational convention, we use λ for such an artificial parameter, rather than μ.

A problem with this construction arises if two transistors are connected in a "cascode" configuration, with the collector of one transistor connected to the collector of another. When λ is set to zero, the transistors become simply a pair of diodes; in the cascode configuration, two diodes are connected anode to anode, which results in a node which is effectively disconnected from the rest of the circuit. This problem is

fixed by adding a "leakage" circuit from each node to ground, which is removed during the continuation process.

The leakage circuit consists of a conductance in series with a fixed voltage source. Let the conductance have a value $(1 - \lambda)$ Gscale, in which Gscale is a global constant. Thus, numerical singularities due to unconnected nodes are eliminated. Moreover, if the voltage source of each leakage connection is given a *random value*, then supporting theory [10] shows that bifurcations (as in figure 2) are eliminated with probability one. The presence of the random sources introduces asymmetry into the system to unfold any bifurcations.

Notice that we have designed a homotopy specific to operating point equations, rather than a general construction which can be applied to any system of nonlinear equations. One such general construction is the following homotopy [11]:

$$H_b(x, \lambda) = (1 - \lambda)(x - b) + \lambda F(x) \quad (4)$$

where F is an arbitrary system of nonlinear equations, and b is an n-vector which fixes a starting state for the construction. This construction *will* work for operating point computations, but is slower by a factor of 10 than the "gain" homotopy described above; perhaps the reputation of continuation methods for being "too slow" is due to the use of such general constructions. The "gain" homotopy takes advantage of the fast convergence of a damped-Newton scheme on a reduced problem where convergence is assured, then employs homotopy to get the final answer. Fortunately, HOMPACK allows a user to design such problem-specific homotopies, as well as providing (4) as a default.

The examples in the final section show that operating points can be obtained with an effort between two to five times that required by other, less robust, methods. Users seem quite willing to pay this extra computer cost in exchange for the robustness and analysis capabilities of continuation methods.

2.2. *Finding Multiple Operating Points*

Artificial parameter continuation methods provide an elegant aproach to identifying multiple dc operating points. Consider a circuit, such as the flip-flop above, which has three distinct operating points when $\lambda = 1$. Because of the way in which the continuation parameter has been introduced into the circuit, there is certainly a unique solution at $\lambda = 0$. Moreover, the random voltage sources unfold any bifurcations, so the picture looks like figure 2b (although the continuation parameter is no longer the supply voltage). The continuation process is tracking one branch of the solution set starting from the (unique) solution at $\lambda = 0$, and stops when it gets to $\lambda = 1$. There is no reason to stop here—instead, continue to follow the path; in some cases, it will exhibit a turning point and connect up with the branch containing the other two solutions as indicated in figure 5. In such case, we say that the multiple operating points have been "lambda threaded." Of course, this will require values of λ greater than 1.

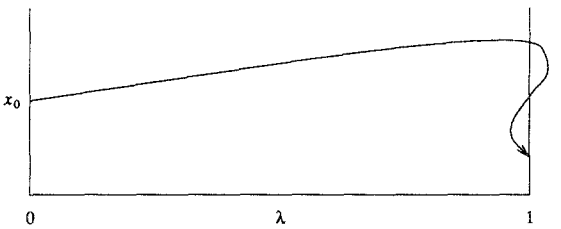


Fig. 5. Threading multiple solutions with continuation path.

The idea of letting the continuation path proceed past $\lambda = 1$ has appeared in the literature in various unrelated publications [17–19]. In particular, Diener [19] gives sufficient conditions on the function being studied to insure that *all* solutions will be traversed, however, Diener's condition is quite strong and there seems little hope of establishing it for circuit equations even on a restricted class of circuits. Some authors have presented algorithms guaranteed to find all solutions of a circuit [20], but these methods are based on multidimensional analogs of bisection, which can be rather slow in higher dimensions, or restrict modeling equations to be polynomials [21], which is undesirable in our application. Our program incorporates some specific features which "encourage" the continuation path to traverse multiple solutions and has successfully found multiple dc solutions of circuits containing hundreds of unknowns.

First, the transistor gains are not actually multiplied by λ, but by a function gain(λ) which stays in the range [0, 1], even for $\lambda > 1$. Second, note that the leakage circuit described above exhibits *negative* conductance for $\lambda > 1$. The presence of such negative conductance can generate unrealistically large currents in the network. For $\lambda > 1$, Sframe introduces a nonlinear negative resistance in the leakage circuitry. The *I-V* characteristic of this negative conductance has a saturating characteristic which keeps the voltages within reasonable bounds.

Both the gain function and the leakage conductance are designed to be periodic in λ with period 2. Thus, at $\lambda = 2$, the circuit again has a unique solution equal to the solution at $\lambda = 0$. This provides a convenient stopping criterion—if multiple operating points have not been detected by the time λ reaches 2, then the procedure can be stopped, since the behavior of the continuation will repeat for $\lambda > 2$. Figure 6 shows the continuation path threading the three operating points of a Brokaw reference [22]. The drawings plots the output voltage against λ, and has an expanded horizontal scale to emphasize the behavior of λ near 1.

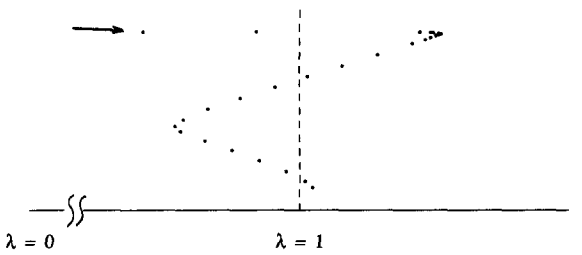


Fig. 6. Threading three solutions of Brokaw circuit.

2.3. Incremental Computations

Suppose that an operating point has been computed for a circuit, but the designer would like the operating point of a perturbed circuit—maybe at a different temperature, different supply voltage, etc. Call these two circuits A and B, with known operating point x_A for the A circuit. Again, one may draw a distinction between a natural parameter approach to this problem, and an artificial parameter approach. To be specific, suppose the operating point equations are parameterized in temperature T and the value of a single independent voltage sources, V.

Let the (known) point x_A satisfy

$$F(x_A, T_0, V_0) = 0, \tag{5}$$

and suppose an operating point x_B is desired which satisfies $F(x_B, T_1, V_1) = 0$. An obvious continuation method to find x_B would be to perform two separate continuations on T and V separately. Say, first perform a continuation on T to get a point x_1 such that $F(x_1, T_1, V_0) = 0$, then compute a path from x_1 to x_B such that $F(x_B, T_1, V_1) = 0$. This is rather slow. A better scheme is to use the single continuation

$$F(x, (1 - \mu)T_0 + \mu T_1, (1 - \mu)V_0 + \mu V_1) = 0 \tag{6}$$

and perform continuation on μ in the interval [0, 1]. Both schemes assume that F can be evaluated at arbitrary values of T and V, and has continuous derivatives with respect to these parameters. This might not be convenient if, for example, a device model is characterized at only a fixed set of temperatures. Moreover, like all natural parameter continuations, there is the possibility of a numerical singularity somewhere along the path. The use of an artificial parameter avoids both of these obstacles. Consider the homotopy

$$H_b(x, \lambda) = (1 - \lambda)F(x; T_0, V_0) + \lambda F(x; T_1, V_1) + \sigma(\lambda)(x - b) \tag{7}$$

in which b is an n-vector as in (4) and σ is a smooth function defined on the interval [0, 1] and equal to zero at the endpoints of this interval (e.g., $\sigma(\lambda) = K \sin(\lambda\pi/2)$ in which K is a scaling factor). According to the theory presented in [10], the introduction of the vector b practically insures smoothness of the solution to (7). Clearly, $H_b(x_A, 0) = 0$ for the starting point because $\sigma(0) = 0$, and the endpoint x_A is an operating point of the A circuit. This scheme does *not* require derivatives with respect to either circuit parameter V or T, and avoids numerical singularities with the random $\sigma(\lambda)(x - a)$ term.

There is, however, a disadvantage to the artificial parameter scheme. For values of λ strictly between zero and one, a solution to $H_b(x, \lambda) = 0$ is the operating point of some "mixture" of the A circuit, the B circuit, and the randomization. Hence, such intermediate points are not the solutions of a real circuit. In contrast, if a natural parameter continuation is used to move from the A circuit to the B circuit, then intermediate points along the continuation path are simply more sample values. These samples can be used to advantage in a statistical design scenario.

Continuation methods are implemented in Sframe with two public-domain packages: HOMPACK [11] and PITCON [13]. In general, an artificial parameter homotopy, with HOMPACK to perform the continuation, is much faster than a natural parameter continuation performed by PITCON. This is especially true for larger differences between the A and B circuits. HOMPACK is designed for speed rather than accuracy, except at the endpoint of the continuation. Thus, it is able to move quickly along the path to the solution to the B circuit. Timing results for some examples of incremental computation are presented in the last section.

In summary, there is no "best" method; the choice between an artificial parameter approach or a natural parameter approach depends on the context in which the results will be used.

2.4. Combining Continuation Operations

The artificial and natural parameter continuation facilities of Sframe may be usefully combined. As an example, we consider an all NPN voltage reference of novel design documented in [23]. The value of a particular resistance, r17, is crucial to the operation of the reference. For r17 = 20k, the circuit has a unique operating point. For r17 = 3.2k, the circuit has three operating points, one of which is the desired state of the circuit. The diagram of the corresponding unfolded bifurcation is shown in figure 7.

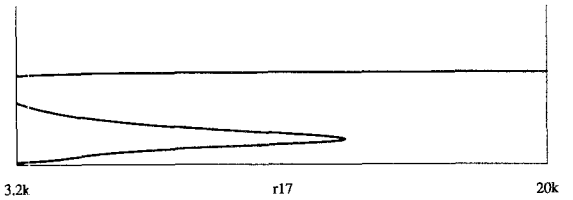


Fig. 7. Bifurcation diagram for voltage reference.

Appropriate start-up circuitry is included to insure that the circuit will settle into the desired state. However, if r17 is made too large, then the circuit undergoes a qualitative change of behavior, and has only one operating point, with or without the start-up mechanism. In this regime of operation, the performance of the circuit is compromised. Thus, the designer is led to ask for the "critical" value of r17, below which the circuit will have three states, one of which is the desired state. An exact answer to this question can be provided by the continuation operations available in Sframe. First, fix r17 at 3.2k and use the lambda-threading device to identify the three operating points for this value of the resistor, then initialize the circuit to the *metastable* state. Note that the turning point of figure 7 is accessible from the metastable state. Now, perform a (natural) parameter continuation study on the value of r17, trying to drive the value of the resistor back up to 20k. This will not be possible—rather, the continuation process will encounter a turning point at the critical value of r17—exactly the value requested by the designer. The PITCON code provides a facility to compute the exact location of such turning points.

2.5. Continuation to a Target Point

Suppose that a particular circuit parameter, say the value of an independent voltage source, is treated as a continuation parameter μ with initial value μ_0. In general, all other node voltages and branch currents in the circuit will depend on μ, although the relationship between such a designated output and μ is not necessarily a functional one!

As illustrated by figure 3, for certain values of μ, there may be more than one value of the output quantity corresponding to a particular value of μ. A useful extenstion of the continuation method is to "work backwards" for such a value. As a classic example, consider the design of a Widlar current source [24]. This circuit uses an emitter degeneration resistor Re to source or sink a very low load current with reasonable resistor values (thus saving chip area). However, the relationship between load current and the value of Re is nonlinear and some trial-and-error may be necesasry to set Re. The initial value of Re is chosen so that the resulting load current is less than the desired value. Then, target continuation is performed on the resistor value until the load current equals the desired value.

3. Bipolar Transistor Model

The bipolar transistor model implemented in Sframe is an advanced dc version of the extended Gummel-Poon bipolar model [25–26]. It models the electrical characteristics of bipolar transistors fabricated using AT&T's CBIC (complementary bipolar integrated circuits) junction isolated technology [27]. There are different versions of CBIC technologies supporting both vertical *npn* and *pnp* transistors, with characteristics that range from medium speed ($f_T \approx$ 800 MHz) and high function breakdown voltages to high-speed ($f_T \approx$ 12 GHz) small-geometry devices. Thus, a variety of effects, including those arising from junction isolation parasitics, must be included in the model. For example, for the high-voltage devices, due to higher intrinsic resistances for base and collector, both the basewidth modulation under different bias conditions and the collector resistance modulation and quasi-saturation have to be accounted for. Of course, for small-geometry devices, all lateral charge injections become important, due to the fact that charge injection from the emitter is not localized to the vertical base-emitter junction.

The circuit level model presented in this section characterizes both the primary transistor action and the most important parasitic phenomena in the CBIC transistor structures used in analog bipolar circuit designs. It should be mentioned that the model, in some simpler form, can be applied to other bipolar technologies as well. The complexities of such a highly nonlinear model

are a major source of convergence difficulty for circuit simulators which offer only conventional Newton-type procedures for the calculation of a dc operating point. Therefore, the model has been augmented to allow the use of robust numerical continuation methods.

This augmentation has been accomplished in such a way that the number of iterations during the artificial homotopy parameter study can be optimized. This is partially due to the fact that, for the initial value of the continuation parameter, the model results in operating point equations which are diffeomorphism [2] and therefore the more efficient norm reducing Newton method can be used with assured success to solve the start system.

3.1. Model Description Summary

Figure 8 represents a cross-sectional view of a typical *npn* transistor in a complementary bipolar technology. Superimposed on that figure are most of the possible transistor actions that can take place under different bias conditions in such a structure. QV is the primary transistor, the performance of which has to be optimized relative to the other transistors which act as parasitics. In a typical CBIC technology transistor QS2 would have both higher emitter and collector efficiency than QS1. Moreover, transistor QL which represents a lateral *npn* action from the emitter side-wall laterally to the collector contact, has a much smaller emitter/collector efficiency than QV. It is true that, for high frequency applications, QL becomes important due to its base width, which is much longer than that of QV, thus introducing an excess phase shift in the overall transistor ac response [28]. However, for dc applications QL can be left out with minimal loss in performance prediction accuracy. For similar reasons QS1 can also be left out. Therefore, the transistor model can be reduced to that shown in figure 9. In the reduced model, *RC* is the accumulated ohmic resistance of the collector contact, deep collector, and buried layer. RBX represents the sum of the contact and series external base ohmic resistances under the base contact diffusion, while RB and RBP signify modulated active base region resisances for QV and QS2 respectively.

Transistors QV and QS2 are modeled in a similar fashion, each having a full Gummel-Poon circuit-level topology. It is very important for SQ2 to have a full Gummel-Poon structure. This is due to the fact that in CBIC technology, QS2 is a lateral isolation transistsor with a relat i vely narrow base, hence, effects such as base width modulation (Early effect [29]) have to be modeled properly. This transistor will operate in cutoff mode when the structure is biased to activate QV in its normal mode. However, at higher charge injections, the voltage drop across RC can become high enough to forward bias QS2 and thus to turn it on. This usually happens when QV is in its quasi-saturation of saturation modes.

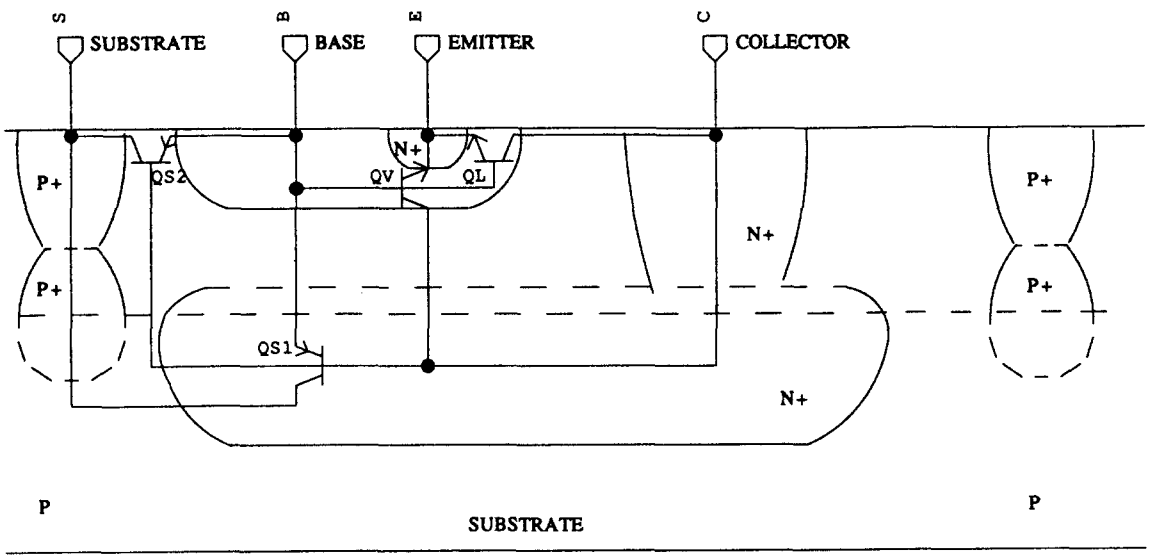


Fig. 8. Cross section of CBIC transistor.

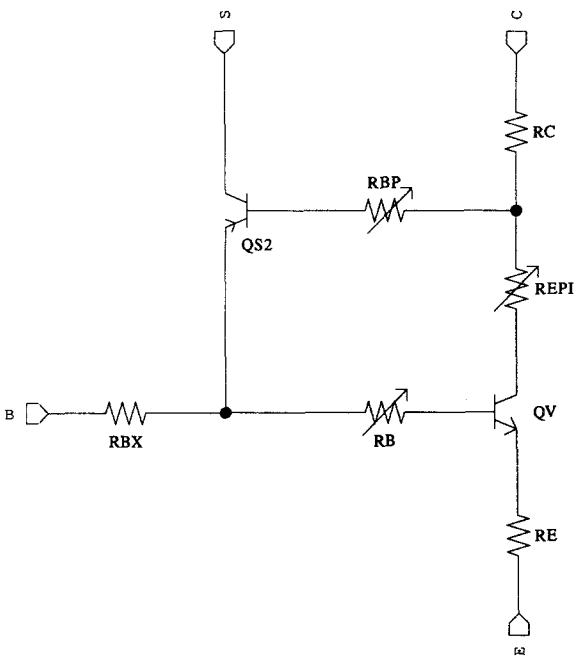


Fig. 9. Simplified transistor model.

3.2. Model Considerations for Transistors QV and QS2

In order to be used in conjunction with artificial paarameter continuation methods, the model incorporates an artificial parameter λ. As described in Section 2.1, the role of the continuation parameter λ is to modify the transistor model in a way that leads to an "easy" to solve circuit when $\lambda = 0$ while restoring the full model by the time $\lambda = 1$. In order to include the continuation parameter in the core Gummel-Poon model, its topology is modified as depicted in figure 10 for partial implementation of the *npn* transistor QV. The *pnp* transistor will have similar topology with opposite polarities for the currents and voltages.

In figure 10, $f(\lambda)$, $g(\lambda)$, and $h(\lambda)$ are suitable functions of the continuation parameter λ which best describe model behavior during parameter variation. We want λ to modulate the current gains of the transistors. Since, in general the short-circuit current gain imposes a linear relationship between base and collector currents, we choose $f(\lambda)$, $= \lambda$. Moreover, we chose $g(\lambda) = \beta_F(1 - \lambda)$ and $h(\lambda) = \beta_R(1 - \lambda)$ where β_F,

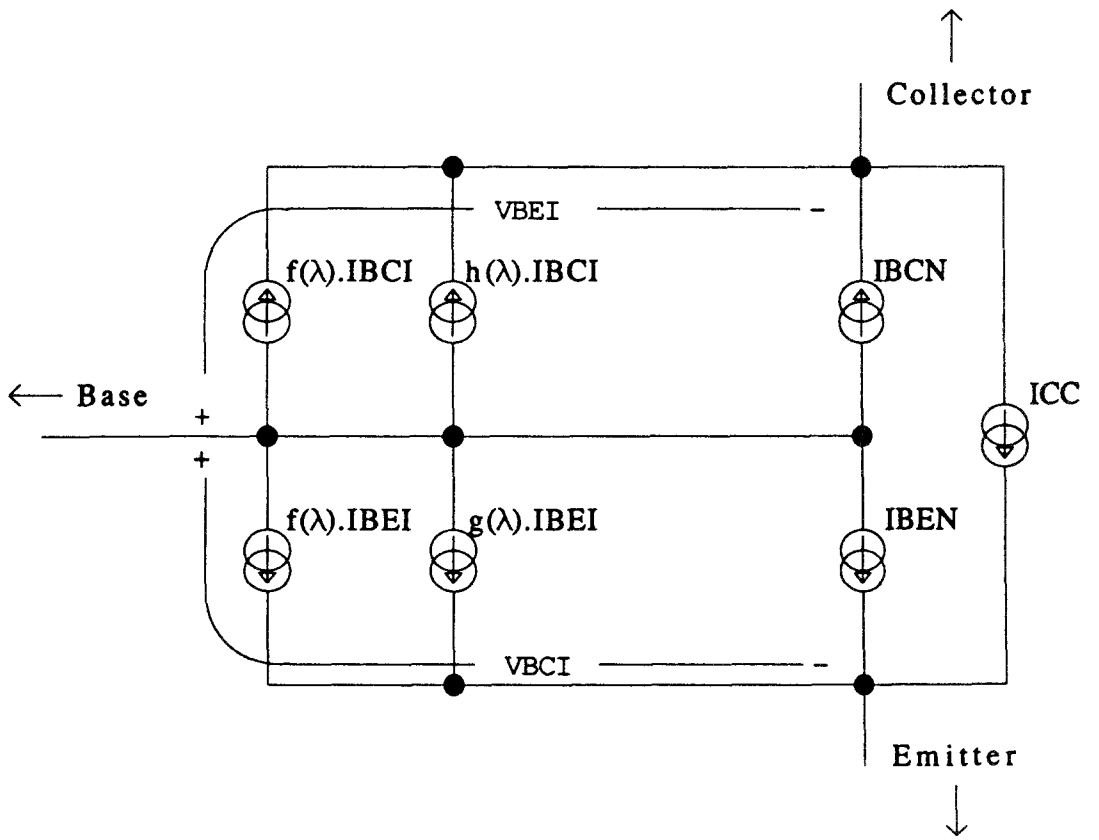


Fig. 10. Core of Gummel-Poon transistor with continuation parameter.

β_R are the current gains corresponding to the Gummel-Poon parameters BF and BR.

The following equations describe IBEI, the intrinsic base-emitter junction current; IBEN, the nonideal base-emitter current due to recombination in space-charge region; IBCI, the intrinsic base-collector junction current; and IBCN, the nonideal current due to base-collector space-charge recombination:

$$\text{IBEI} = \frac{\text{IS}}{\text{BF}} \left(e^{\text{VBE/NF } V_T} - 1\right) \tag{8}$$

$$\text{IBEN} = \text{ISE} \left(e^{\text{VBE/NE } V_T} - 1\right)$$

$$\text{IBCI} = \frac{\text{IS}}{\text{BR}} \left(e^{\text{VBC/NR } V_T} - 1\right)$$

$$\text{IBCN} = \text{ISC} \left(e^{\text{VBC/NC } V_T} - 1\right)$$

where parameters IS, BF, BR, ISE, ISC, NF, NR, NE, and NC are the Gummel-Poon model parameters [25].

Note that as the continuation parameter λ varies from 0 to 1, the functions $g(\lambda)$ and $h(\lambda)$ move from BF and BR to 0 respectively and $f(\lambda)$ from 0 to 1. Thus, for $\lambda = 1$, the model returns to the transitional Gummel-Poon topology. Consequently, the effective forward and reverse transistor short-circuit current gains BF and BR change from 0 to their final values BF and BR, as defined by the model parameter values. This results clearly from the ICC current source expression

$$\text{ICC} = \frac{\text{IS}}{\text{QB}} \left\{e^{\text{VBE/NF } V_T} - e^{\text{VBC/NR } V_T}\right\} \tag{9}$$

where QB is the normalized base charge as defined in the Gummel-Poon model. When the transistor is biased in active mode (VBE > 0, VBC < 0),

$$\text{ICC} \approx \frac{\text{IS}}{\text{QB}} e^{\text{VBE/NF } V_T} \tag{10}$$

Assuming

$$\text{IC} = \text{ICC} - f(\lambda)\,\text{IBCI} - h(\lambda)\,\text{IBCI} - \text{IBCN}$$

and

$$\text{IB} = f(\lambda)\,\text{IBEI} + g(\lambda)\,\text{IBEI} + \text{IBEN} + f(\lambda)\,\text{IBCI} + h(\lambda)\,\text{IBCI} + \text{IBCN}$$

then,

$$\beta = \left| \frac{\text{IC}}{\text{IB}} \right|_{\lambda=0} \approx \frac{(\text{IS/QB})\, e^{\text{VBE/NF } V_T}}{\text{IS}\left(e^{\text{VBE/NF } V_T} - 1\right) + \text{ISE}\left(e^{\text{VBE/NE } V_T} - 1\right)}$$

In normal active mode QB $\approx$ 1 and hence $\beta \leq 1$. Thus for $\lambda = 0$ such a transistor will have no current amplification capability and hence it will act as two back-to-back diodes.

3.3. *Resistor REPI*

To model the quasi-saturation effects in the epitaxial collector region in both *npn* and *pnp* transistors of the CBIC technology, the modified version of the model proposed by Kull et al. [26] has been utilized. This model, which accounts for collector resistance modulation due to injected minority carrier charge from a forward biased metalurgical junction into the collector, is highly nonlinear. In addition, due to the series connection to the collector of QV, which consists of a number of current generators, convergence problems may arise when the transistor is operating in quasi-saturation or high injection modes. For this reason a continuation parameter should be included to control the nonlinearity of the expression

$$\text{REPI} = \frac{\text{VEPI}\left[\frac{\text{RCO}}{\text{VT}}\right]\left[1 + \frac{\text{VBCO} - \text{VBCW}}{\text{VO}}\right]}{K1(\text{VBCO}) - K1(\text{VBCW}) \ln\left[\frac{1 - K1(\text{VBCO})}{1 + K1(\text{VBCW})}\right] + \left[\frac{\text{VBCO} - \text{VBCW}}{\text{VT}}\right]}$$

where

RCO = zero-bias active epitaxial collector resistance

VT = thermal voltage

VBCO = external base-collector voltage

VBCI = internal base-collector voltage

$K1(V) = \sqrt{1 + \gamma \exp(V/\text{VT})}$

$$\gamma = \left(2\frac{\text{ni}}{\text{NEPI}}\right)^2$$

$$\text{VO} = \frac{\text{WEPI vsp}}{\mu_{p\text{EPI}}}$$

WEPI = width of epitaxial collector

vsp = carrier saturation velocity

$\mu_{p\text{EPI}}$ = minority carrier hole mobility

To reduce the nonlinearity of the expression, the exponential terms of type $K1(V)$ modulated by γ is multiplied by a function of the continuation parameter:

$$y(\lambda) = \lambda\, e^{V(\lambda-1)/\mathrm{VT}} \tag{12}$$

which dampens the exponential in $K1(V)$ for small values of λ but has no effect when $\lambda = 1$. Thus,

$$K1(V) = \sqrt{1 + \gamma\,\lambda \exp(V(\lambda-1)/\mathrm{VT}) \exp(V/\mathrm{VT})} \tag{13}$$

3.4. Resistors RB and RBP

The two resistors RB and RBP which are associated with transistors QV and QS2 respectively (figure 10) are modulated base resistances and follow a relationship with respect to junction voltages of the core Gummel-Poon model as defined in [25]. Although, some form of continuation parameter can be defined for such base width modulations, since the currents through these resistors are not direct functions of exponentials, we have not found it necessary in this case.

4. Automatic Differentiation and Sensitivity Computations

We have shown in Section 2 that the use of homotopy methods for the solution of the nonlinear system of equations (1) requires the computation of the Jacobian matrix.

$$\frac{\partial H}{\partial(x,\ \mu)} = \left[\begin{array}{cc} \dfrac{\partial H}{\partial x} & \dfrac{\partial H}{\partial \mu} \end{array}\right] \tag{14}$$

Remember $H(x, \mu) = 0$ is a system of nonlinear equations $[h_1(x, \mu), \ldots, h_n(x, \mu)]^T = 0$ and the individual equations $h_i(x, \mu) = 0$, $i = 1, \ldots, n$, are the following form. (For simplicity, we ignore the equations corresponding to independent voltage sources.)

$$\sum_{j \in N_i} I_{ij}(V, \mu) = 0 \tag{15}$$

where N_i is the set of nodes adjacent to node i, I_{ij} denote currents of circuit branches ij and V is the n-vector of node voltages.

The currents I_{ij} belong typically to nonlinear semiconductor devices such as transistors, diodes, etc. The computation of the Jacobian matrix $\partial H/\partial(x, \mu)$ therefore requires the differentiation of the device currents expressions, with respect to the voltages applied to their nodes and with respect to the continuation parameters μ. As discussed above, the continuation parameter μ can be either a physical model parameter, such as the temperature, or an artificial parameter with no physical meaning such as λ described in the previous section. We already mentioned that according to our experience finite difference approximations to these derivatives are both inefficient and unreliable. Consequently, the use of homotopy-based methods for the solution of nonlinear circuit equations poses an additional burden on the device model routines. In addition to the values of device current derivatives with respect to the voltages applied to them $\partial I/\partial V$ required by most traditional circuit simulators, the routines must also calculate derivatives with respect to the continuation parameter $\partial I/\partial \mu$. In order to support both natural and artificial parameter continuation, device model routines must therefore be able to compute derivatives with respect to practically any of the model parameters.

Dc operating point computation is not the only application that needs the evaluation of device model expression derivatives. Once the operating point has been reached, partial derivatives of model expressions with respect to various model parameters are also necessary for sensitivity computation.

4.1. DC Sensitivity Computation

Efficient methods to compute dc circuit sensitivities are well known in theory [30]. Consider the original circuit equations $F(x, a) = 0$ which contain dependencies on some circuit parameters $a = [a_1, \ldots, a_s]^T$. Using a continuation method, circuit responses are obtained as the solution of the system $H(x, \mu_1, a) = 0$, at the final value of the continuation parameters $\mu = \mu_1$. Obviously the circuit responses x are dependent on the circuit parameters a, therefore, the sensitivities $\partial x/\partial a$ (assuming they exist) can be obtained by differentiating the circuit equations with respect to a

$$\frac{\partial}{\partial a} H(x(a), a, \mu_1) = \frac{\partial H}{\partial x}\frac{\partial x}{\partial a} + \frac{\partial H}{\partial a} = 0 \tag{16}$$

Assuming that $\partial H/\partial x$ in invertible, the sensitivities $\partial x/\partial a$ are obtained as the solution of the resulting linear system of equations

$$\frac{\partial H}{\partial x}\frac{\partial x}{\partial a} = -\frac{\partial H}{\partial a} \tag{17}$$

Moreover, the Jacobian matrix $\partial H/\partial x$ is already calculated and factored as a result of the preceding operating point calculation. Therefore is is possible to obtain sensitivities of all circuit responses x with respect to the circuit parameters at a cost of only one additional forward/backward substitution per parameter. This method is known as the *direct* method for sensitivity computation.

However, in many applications we are interested in the sensitivity of one or more functions of the circuit responses $f(x)$ with respect to a large number of parameters. Using the *direct* method described above we would have to perform a foward/backward substitution for each parameter a_i, $i = 1, \ldots, s$. By using the *adjoint* sensitivity computation method [30] all sensitivity values are produced simultaneously as the result of only one forward/backward substitution. The sensitivities we are interested in are expressed by

$$\frac{\partial f}{\partial a} = \sum_{i=0}^{n} \frac{\partial f}{\partial x_i} \frac{\partial x_i}{\partial a} = d^T \frac{\partial x}{\partial a} \tag{18}$$

where we use d to denote the n-vector $\nabla_x f$. Using (17),

$$\frac{\partial f}{\partial a} = -d^T \left[\frac{\partial H}{\partial x} \right]^{-1} \frac{\partial H}{\partial a} = -x_a^T \frac{\partial H}{\partial a} \tag{19}$$

Here x_a is the solution of the *adjoint system*

$$\left[\frac{\partial H}{\partial x} \right]^T x_a = \mathrm{d} \tag{20}$$

The factorization of the transposed Jacobian can be obtained from the available factorization of the Jacobian matrix. Therefore the solution of the adjoint system can be obtained at the cost of only one forward/backward substitution. Subsequently, each sensitivity value can be obtained at a cost of one vector inner product

$$\frac{\partial f}{\partial a} = x_a^T \frac{\partial H}{\partial a}$$

Despite the numerous applications of circuit sensitivities and the existence of efficient methods for their computation, sensitivities are rarely used outside a few embedded applications [31]. An important reason may be the fact that the matrix $\partial H/\partial a$ is required for sensitivity computation. As shown in the previous section, this matrix results from the derivatives of device currents with respect to the model parameters a. In Sframe the computation of the derivatives of device currents with respect to arbitrary model parameters is facilitated through automatic differentiation [32].

4.2. *Overview of Automatic Differentiation Techniques*

By computing derivatives of model expressions with respect to various parameters through automatic differentiation, the implementation of arbitrary parameter continuation and sensitivity computation can be done without adding a considerable burden on the device modeler. In addition, automatic differentiation permits the specification and computation of sensitivities of any function of circuit responses to any set of parameters in a particularly elegant and efficient way.

Automatic differentiation is a collection of software techniques which allow the computation of the derivatives of an expression, based on the computer code which implements the evaluation of that expression. Assume that, given an n-vector of independent parameters x, we want to evaluate on a computer an expression $f(x)$, and its gradient

$$\nabla_x f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right).$$

One could generate a program to compute $f(x)$ and then compute derivatives using the finite difference method. This procedure however is inexact, potentially numerically dangerous, and often inefficient.

Alternatively, one can differentiate symbolically the expression $f(x)$ with respect to all components x_i, $i = 1, \ldots, n$, and code the routines to evaluate the expressions $f(x)$ and $g_i(x) = \partial f/\partial x_i\,(x)$, $i = 1, \ldots, n$. Even if a computer algebra system is used to aid in the generation of symbolic derivative expressions, some hand editing is often needed to deal with condition ("if") statements and to eliminate common subexpressions.

In contrast, automatic differentiation techniques allow the computation of the derivative values $g_i(x)$ based on the computer code which evaluates $f(x)$. These techniques are efficient and provide numerical accuracy at least as good as that available from evaluating the symbolic derivative expressions (sometimes better). Moreover, because derivative values are calculated in a mechanical way from the given functional expressions, there is no danger of getting "out of synch" between a function and its derivative.

We provide here a brief summary of automatic differentiation techniques. In order to explain automatic differentiation, we assume the following simple model for the evaluation of expression $f(x)$ by a computer. Let y_i, $i = 1, \ldots, m$, be all intermediate results necessary for the computation of $f(x)$,

$$y_i = x_i, \qquad i = 1, \ldots, n$$

$$y_i = \phi_i(y_{k[1]}, \ldots, y_{k[n_i]}), \qquad i = n+1, \ldots, m,$$

$$k[j] < i, \; j = 1, \ldots, n_i$$

$$y_m = f(x)$$

where $\phi_i(y_{k[1]}, \ldots, y_{k[n_i]})$ are operations typically supported on a computer on the previous partial results,

such as addition, multiplication, sine, cosine, exponentiation, etc. Assume that we want to compute $\nabla_x f$ through automatic differentiation. There are two flavors of automatic differentiation techniques: the *forward* method [33] and the *reverse* method [34]. The *forward* method is as follows:

1. Initialize:
$$\nabla_x y_i = e_i, \quad i = 1, \ldots, n$$
where e_i is the i^{th} standard basis vector.
2. For each intermediate result y_i, $i = 1, \ldots, N$, we compute
$$\nabla_x y_i = \sum_{j=1}^{n_i} \frac{\partial \phi_i}{\partial y_{k[j]}} \nabla_x y_{k[j]}$$
For example,
$$c = a + b \rightarrow \nabla_x c = \nabla_x a + \nabla_x b$$
$$d = a\,b \rightarrow \nabla_x d = b\,\nabla_x a + a\,\nabla_x b$$
$$t = \sin(a) \rightarrow \nabla_x t = \cos(a)\,\nabla_x a$$
3. Finally, $\nabla_x f = \nabla_x y_m$.

Observe that the number of arithmetic operations for the computation of the gradient through forward automatic differentiation, increases proportionally to the number of parameters in x.

Using the *reverse* method we can compute the derivatives of $f(x)$ with respect to all the components of x at a maximum cost theoretically limited [34] to at most five times the number of operations necessary for the evaluation of $f(x)$ but significantly smaller in practice. The reverse method, however, requires the sequence of operations for the evaluation of $f(x)$ to be stored and replayed in reverse order for derivative computation. The algorithm is as follows:

1. Initialize:
$dy_m = 1, \quad dy_i = 0, \quad i = 1, \ldots, m-1$
2. Then for each y_i, $i = m, m-1, \ldots, n+1$:
for each $y_{i[j]}$, $j = 1, \ldots, n_i$,
$$dy_{i[j]} = dy_{i[j]} + \frac{\partial y_i}{\partial y_{i[j]}} dy_i$$
3. *When finished* dy_i, $i = 1, \ldots, n$, represent the desired partial derivatives $dy_i = \partial f / \partial x_i\,(x)$.

Note the interesting analogy between forward and backward automatic differentation on one side and the direct and adjoint sensitivity computation methods on the other side.

4.3. Implementation of Automatic Differentiation in Sframe

Automatic differentiations fits very well in Sframe's C++-based environment. Both the forward and the reverse technique can be implemented using the overloading feature of C++. The forward automatic differentiation method is implemented in the following way. A new class `gdouble` consisting of a value and a gradient vector is defined, the usual operators, +, −, *, ... and the transcendental functions sin, cos, log, ... are redefined for this class to handle gradient computations as described in the example above. All parameters, intermediate variables, and final results of the model code are declared as `gdoubles`, therefore, the evaluation of the model will produce the derivatives as well. The forward automatic differentiation method is more efficient when we are interested in derivatives with respect to a few number of parameters, and therefore particularly suitable in incremental simulation with natural parameter continuation.

The reverse method is implemented in a similar way. A new class `node` is defined to represent a node in an expression directed acyclic graph. The operators and the transcendental functions are redefined to insert the `nodes` in the expression graph. The evaluation of the model code results in the expression graph of the model outputs. Following, this graph can be interpreted as above to produce the results for the desired function and its gradient. This approach permits the efficient computation of derivatives with respect to multiple parameters simultaneously, and therefore is particularly suitable for sensitivity computation or continuation studies where many parameters are changed simultaneously.

5. The Architecture of Sframe

The name Sframe stands for "simulation framework" because the code is really a driver framework (in C++ [35]) for different numerical codes, typically written in FORTRAN. Sframe itself contains very little numerical code.

For any dc analysis, Sframe considers a function $G(x; \lambda, \mu)$, where G is a mapping from R^{n+2} into R^n. The n-vector x combines the current and voltage unknowns, λ is an artificial parameter used only for operating point computation, and μ is a natural continuation parameter. A procedure is provided to evaluate G given numerical values for these quantities, and return the result as an n-vector. In addition to evaluation of G itself, the various numerical solvers employed by Sframe need the Jacobian matrix of G at the point of evaluation. This is an $n \times (n+2)$ matrix which is quite sparse. A procedure is provided to compute this matrix, and deposit it into a storage area allocated during circuit setup. The matrix is not passed into the various solvers, rather the solvers request

manipulation of this matrix through calls to the matrix package.

A procedure solve is provided to solve $Jx = b$, where J is the most recently computed Jacobian matrix, and b is an arbitrary n-vector (of course, J above must be square; typically, a submatrix of the rectangular Jacobian matrix computed by Sframe is selected, which might be augmented with a single row or column border provided by a nonlinear numerical solver.)

After a call to solve, a call to re_solve will solve $Jx = b$ for a different value of b. The matrix algebra operations like solve are C++ routines which call a FORTRAN sparse matrix package. This arrangement allows for easy experimentation with different sparse solvers.

Three nonlinear solvers are supported in the present version of the program: a norm-reducing Newton code, HOMPACK and PITCON. The norm-reducing Newton code is used to solve the start system of the artificial parameter method used to get a dc operating point. In fact, for easier circuits, it is possible to start with $\lambda = 1$, and get the operating point using the norm-reducing code alone.

The present program uses a direct method for sparse matrix factorization, which works nicely for medium sized problems (say up to 3000 unknowns). In the present implementation, a symbolic factorization is computed only once at the beginning of a continuation study. A fresh numeric factorization may be computed at several points in the study depending on the conditioning of the Jacobian matrices encountered.

```
#include "mod.h"
#include "dev.h"
class Wilson : public Circuit
{
public:
  // declare models, nodes, devices
  GpMod npn; Gp4 q1 , q2 , q3;
  Node N2 , N3 , N1 , N99;
  Res rp , rl; Vsrc vcc;

  Wilson() // constructor
  : Circuit( "Wilson" )
  , npn(this, "npn", 1.0)
  , N2(this, 0, "Node-2")
  , N3(this, 0, "Node-3")
  , N1(this, 0, "Node-1")
  , N99(this, 0, "Node-99")
  , vcc(this, 0, "vcc", N99, GND, 15.0)
  , q1(this, 0, "q1", N1, N3, GND, GND, npn)
  , q2(this, 0, "q2", N3, N3, GND, GND, npn)
  , q3(this, 0, "q3", N2, N1, N3, GND, npn)
  , rp(this, 0, "rp", N99, N1, kilo(10.0))
  , rl(this, 0, "rl", N99, N2, kilo(5.0))
  {
    // assemble circuit in memory
    assemble();
  }
  void dcop()
  {
    // calculate operating point
    set_lambda( 0.0 );
    dnm_solve();
    hom_track();
    resId();
  }
};
```

5.1. Example Circuit Description

Here is an annotated description of the Sframe input for a Wilson current source [24]. This example shows an "application" circuit derived from the base-class Circuit which defines an operation dcop( ) using facilities provided in the base class.

6. Timings and Practical Results

Our first set of benchmark data concerns operating point computation. Data for five circuits are presented —all of these examples are actual in-house designs based on the AT&T CBIC bipolar technology. Note that lambda threading was used to identify all the operating points of the "vref" and "hybr" examples. The "#iterations" column reports the number of times a complete Jacobian matrix was computed and factored. This is broken down into two pieces—the iteration count for the damped-Newton solve of the start system, plus the iteration count of the homotopy curve tracking. Times are for a SPARCstation 2 running Sun UNIX. Some adjustment with numerical parameters is necessary for the various FORTRAN packages which are employed by Sframe. The results reported in table 1 were obtained using relatively "conservative" values which worked for every circuit in the suite.

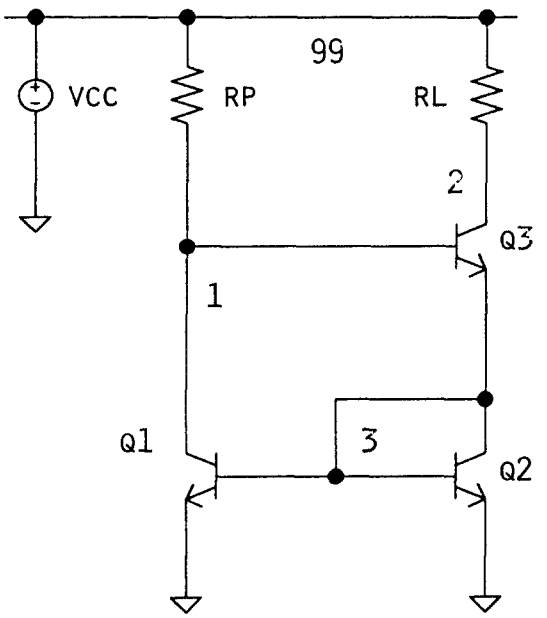


Table 1. Timing results for operating point computation.

Circuit	Comment	No. transistors	No. unknowns	No. iterations	No. time (min, s)
si7	Mass storage chip	241	1853	25+375	10m56s
bgatt	Band gap	12	124	19+64	14.4s
ups01a	Thermal shutdown	7	58	7+34	1.7s
vref	Brokaw reference (three solutions)	9	66	15+141	8.1s
reg	Regulator	49	432	27+523	2m54s
hybr	*npn* reference (three solutions)	9	14	10+143	1.1s

The next set of timing concern incremental exploration of the behavior of a circuit as temperature and supply voltages are varied. See table 2. The "reg" example from the previous benchmark suite is a good candidate for such a study. The circuit has two fixed voltage sources, which were exercised at the nominal setting, then ±20% of nominal. Temperature was varied over the standard commercial range of 0°C to 70°C. Thus, there are three quantities to be exercised through "nominal(N)," "high(H)," and "low(L)" for a total of 27 combinations. Each point in this space was reached from the nominal operating point using the construction of equation (7). The "arc-length" column reports an estimate of the length of the continuation path connecting the two operating points, which gives some measure of how "different" the two operating points are. This can be a useful diagnostic, since a large value of arc length may indicate that the circuit is not operating correctly at one of the extremes of the design space.

Table 2. Incremental exploration of a design space.

Temperature	Supply 1	Supply 2	No. iterations	Arc length
N	N	N	8	1.00
N	N	L	11	1.10
N	N	H	11	1.10
N	L	N	102	17.67
N	L	L	102	17.67
N	L	H	102	17.67
N	H	N	24	7.24
N	H	L	24	7.26
N	H	H	24	7.26
L	N	N	167	48.49
L	N	L	178	51.84
L	N	H	158	45.20
L	L	N	295	57.40
L	L	L	304	60.74
L	L	H	291	54.07
L	H	N	181	51.92
L	H	L	174	55.23
L	H	H	162	48.64
H	N	N	34	3.64
H	N	L	34	3.68
H	N	H	34	3.68
H	L	N	87	16.52
H	L	L	83	16.52
H	L	H	83	16.52
H	H	N	36	8.63
H	H	L	34	8.64
H	H	H	34	8.64

The sensitivity computation capability of sframe is illustrated through an analysis of a CBIC implementation of the "band-gap" reference shown in figure 11 [24]. The output reference voltage is required to be stable under variations in the process and operating conditions. Table 3 summarizes the values of reference voltage absolute sensitivity with respect to temperature T, band-gap resistor $R4$, a bias resistor $R3$, and the normalized sensitivity with respect to the reverse saturation current Is of the band-gap transistors.

Table 3.

d Vref/d T	−1.06	mV/K
d Vref/d R3	4.86e-5	mV/ohm
d Vref/d R4	−0.325	mV/ohm
(Is/Vref)d Vref/d Is	−0.214	

The sensitivity with respect to temperature is large, indicating the need for additional compensation circuitry. The analysis also reveals that the performance of the circuit is extremely sensitive to the value of $R4$, justifying the use of special techniques for its layout. At first glance the sensitivity to the saturation current is acceptable in view of the known process variations. However, the analysis assumed identical transistors in the band-gap generator section and did not take device mismatch into consideration. Sensitivities to saturation current mismatches among the various transistors can be computed without repeating the operting point computation. This is done by performing the sensitivity analysis using the individual saturation currents of the differently sized band-gap section transistors (BW5, BQ6_1, and BQ6_2) as independent parameters. Table 4 shows the results of this analysis.

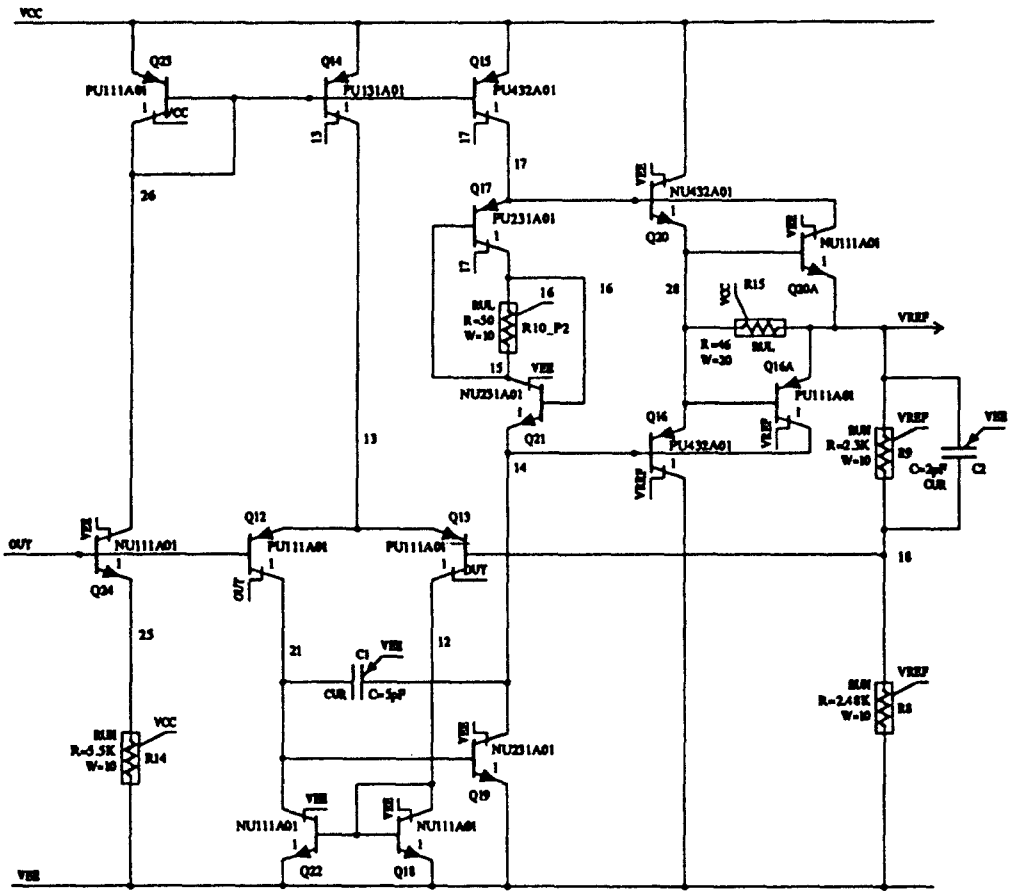


Fig. 11. Band-gap reference.

Table 4.

(Is/Vref)d Vref/d Δ Is(BQ5)	−1.52
(Is/Vref)d Vref/d Δ Is(BQ6_1)	0.88
(Is/Vref)d Vref/d Δ Is(BQ6_2)	0.88

These data indicate that sensitivities to device mismatches are one order of magnitude larger than the sensitivity with respect to the global saturation current. Again, special layout techniques need to be used to ensure excellent transistor matching.

The CPU time required for the sensitivity analysis is negligible in comparison to the operating point computation. Through the use of automatic differentiation, the computation of sensitivities with respect to any circuit or device parameter can be done without additional device modeling effort. Specifying complicated performance measures is also facilitated by automatic differentiation.

7. Conclusions

First, we observe that a wide variety of behaviors is evident in analog circuits. Phenomena such as turning points in solution curves or multiple operating points are not just academic considerations. Accurate quantitative results require a detailed model, but the use of such models increases the likelihood of convergence difficulties in a simulator. The continuation methods we have described solve the convergence problem with reasonable computing cost, as well as providing a handle on the issue of multiple solutions.

On the other hand, careful design and implementation of the numerical methods is necessary to achieve the desired level of robustness. In particular, derivatives must be smooth and numerically accurate. Almost all instances of convergence failure during the development of Sframe were traced to incorrect derivative computation; the adoption of automatic differentiation completely eliminated such problems.

The interface to device models which we have implemented allows the use of highly optimized code which circumvents the automatic differentiation mechanism for situations in which the highest possible speed is necessary—for example, if the simulator is in the inner loop of an optimization process.

Implementation of the project in C++, with C++ as the circuit description languages provided us with several advantages: we were freed from the task of designing an input language and associated parser; automatic differentiation and specification of performance measures for sensitivity calculation is facilitated by the operator overloading feature of C++; finally, the designer has the full power of a modern programming language available for the description of complex circuits, simulation tasks, and postprocessing of simulation results.

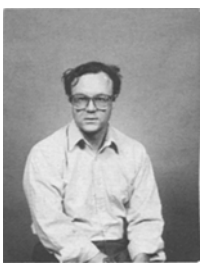
Acknowledgments

Herman Gummel was particularly helpful with the modeling effort. Tom Banwell supplied the interesting voltage reference circuit of figure 7. David Gay introduced us to automatic differentiation techniques. Werner Rheinboldt has given much useful advice on the details of PITCON. Several errors in the first draft were brought to our attention by the referees.

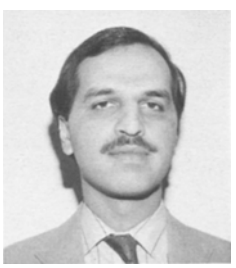
References

1. Lj. Trajković, R.C. Melville, and S.C. Fang, "Passivity and no-gain properties establish global convergence of a homotopy method for DC operating points," in *Proc. IEEE Int. Symp. on Circuits and Systems*, New Orleans, LA., 1990.
2. Lj. Trajković, R.C. Melville, and S.C. Fang, "Finding DC operating points of transistor circuits using homotopy methods," in *Proc. IEEE Int. Symp. on Circuits and Systems*, Singapore, 1991.
3. Lj. Trajković, R.C. Melville, and S.C. Fang, "Improving DC convergence in a circuit simulator using a homotopy method," in *IEEE Custom Integrated Circuits Conf.*, San Diego, CA, 1991.
4. C.B. Garcia and W.I. Zangwill, *Pathways to Solutions, Fixed Points, and Equilibria*, Englewood Cliffs, NJ: Prentice-Hall, pp. 1–23, 1981.
5. E.L. Allgower and K. Georg, *Numerical Continuation Methods: An Introduction*, Springer-Verlag, 1990.
6. S.L. Richter and R.A. DeCarlo, "Continuation methods: theory and applications," *IEEE Trans. Circuits Syst.*, Vol. CAS-30, pp. 347–352, 1983.
7. C.W. Ho, A.E. Ruehli, and P.A. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits Syst.*, Vol. CAS-22, pp. 504–509, 1975.
8. J.M. Ortega and W.C. Rheinboldt, *Iterative Solutions of Nonlinear Equations in Several Variables*, Academic Press: New York, pp. 161–165, 1969.
9. A. Sard, "The measure of the critical values of differential maps," *Bull. Amer. Math. Soc.*, Vol. 48, pp. 883–890, 1942.
10. S. Chow, J. Mallet-Paret, and J.A. Yorke, "Finding zeroes of maps: homotopy methods that are constructive with probability one," *Math. Comp.*, Vol. 32, No. 143, pp. 887–899, 1978.
11. L. Watson, S. Billups, and A. Morgan, "ALGORITHM 652 HOMPACK: a suite of codes for globally convergent homotopy algorithms," *ACM Trans. Math. Software*, Vol. 13, No. 3, pp. 281–310, 1987.
12. L.T. Watson, "Numerical linear algebra aspects of globally convergent homotopy methods," *SIAM Rev.*, Vol. 28, pp. 529–545, 1986.
13. W. Rheinboldt and J.V. Burkhardt, "A locally parameterized continuation process," *ACM TOMS*, Vol. 9, No. 2, pp. 215–235, 1983.
14. R. Seydel, *From Equilibrium to Chaos: Practical Bifurcation and Stability Analysis*, Elsevier: New York, 1988.
15. Lj. Trajković and A.N. Willson, Jr., "Behavior of nonlinear transistor one-ports: things are not always as simple as might be expected," in *30th Midwest Symp. Circuits and Systems*, Syracuse, New York, 1988.
16. R.E. Bank and D.J. Rose, "Global approximate Newton methods," *Numer. Math.*, Vol. 37, pp. 279–295, 1981.
17. L.O. Chua and A. Ushida, "A switching-parameter algorithm for finding multiple solutions of nonlinear resistive circuits," *Int. J. Circuit Theory Appl.*, Vol. 4, pp. 215–239, 1976.
18. F.H. Brannin, "Widely convergent methods for finding multiple solutions of simultaneous nonlinear equations," *IBM J. Res. Dev.*, pp. 504–522, Sept. 1972.
19. I. Diener, "On the global convergence of path-following methods to determine all solutions to a system of nonlinear equations," *Math. Program.*, Vol. 39, pp. 181–188, 1987.
20. J. Gan and Y.M. Song, "All DC solutions to nonlinear circuits," in *Proc. IEEE Int. Symp. Circuits and Systems*, New Orleans, LA., pp. 918–921, 1990.
21. G. Wettlaufer and W. Mathis, "Finding all DC-equilibrium points of nonlinear circuits," in *Proc. 32nd Midwest Symp. Circuits and Systems*, Urbana, IL, 1989.
22. A.P. Browkaw, "A simple three-terminal IC bandgap reference," *IEEE J. Solid-State Circuits*, Vol. SC-9, pp. 388–393, 1974.
23. T. Banwell, "An NPN voltage reference," *IEEE J. Solid-State Circuits*, Vol. 26, 1991.
24. P.R. Gray and R.G. Mayer, *Analysis and Design of Analog Integrated Circuits*, 2nd ed. Wiley: New York, 1984.
25. H.K. Gummel and H.C. Poon, "An integral charge control model of bipolar transistors," *BSTJ*, Vol. 49, pp. 827–852, 1970.
26. G.M. Kull, L.W. Nagel, S.W. Lee, P. Lloyd, E.J. Prendergast, and H. Dirks, "Unified circuit model for bipolar transistors including quasi-saturation effects," *IEEE Trans. Electron Dev.*, Vol. ED32, No. 6, pp. 1103–1113, 1985.
27. B.W. McNeil, "A high-frequency complementary-bipolar array for fast analog circuits," in *CICC 87 Digest of Technical Papers*, pp. 635–638, March 1987.
28. S. Moinian, M.A. Brooke, and J.C. Choma, "BITPAR: a process derived bipolar transistor parameterization," *IEEE J. Solid-State Circuits*, Vol. SC-21, No. 2, pp. 344–352, 1986.
29. I.M. Early, "Effects of space-charge layer widening in junction transistors," *Proc. IRE*, Vol. 40, pp. 1401–1406, 1952.

30. S.W. Director and R.A. Rohrer, "Automated network design—the frequency domain case," *IEEE Trans. Circuit Theory,* Vol. CT-16, pp. 330–337, 1969.
31. R.A. Rohrer, L. Nagel, R. Meyer, and L. Weber, "Computationally efficient electronic circuit noise calculations," *IEEE J. Solid-State Circuits*, Vol. SC-6, pp. 204–213, 1971.
32. A. Griewank, "On automatic differentiation," in *Mathematical Programming: Recent Developments and Applications* (M. Iri and K. Tanabe, eds.), KTK Scientific/Kluwer Academic Publishers, 1989.
33. L.B. Rall, "Automatic differentiation: techniques and applications," in *Lecture Notes in Computer Science*, No. 120, Springer, 1981.
34. B. Speelpenning, "Compiling fast partial derivatives of functions given by algorithms," Ph.D. Dissertation, Department of Computer Science, University of Illinois at Urbana Campaign, 1980.
35. B. Stroustrup, *The C++ Programming Language,* Addison-Wesley, 1986.
36. I. Getreu, *Modeling the Bipolar Transistor,* Tektronix, Inc., Beaverton, Oregon, pp. 9–23, 1976.



Robert Melville received a bachelor's degree in computer science from the University of Delaware in 1976 and a Ph.D. in computer science from Cornell University in 1981. He was a junior faculty member at The Johns Hopkins University before joining AT&T Bell Laboratories in 1985. His work at the labs concentrates on CAD tools for simulation and design of analog circuits, with particular emphasis on the qualitative analysis of nonlinear systems. He is a member of SIAM, and has served as a referee for *IEEE Transactions on Computer Aided Design*, the *SIAM Journal of Computing*, and numerous professional conferences.



Shahriar Moinian received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles in 1982, 1983, and 1987 respectively. From 1987 to 1988 he was with Science Applications International Corporation, working on advanced bipolar transistor modeling and processing. In 1988 he joined AT&T Bell Laboratories, Reading, PA, where he is presently a member of technical staff. His current interests include modeling, CAD, and simulation in the areas of complementary bipolar transistor technology. Dr. Moinian is a member of IEEE and ACM.

Peter Feldmann was born in Timişoara, Romania, in 1958. He received the B.Sc. Degree, summa cum laude, in computer engineering in 1983 and the M.Sc. degree in electrical engineering in 1987, both from the Technion, Israel, and the Ph.D. in 1991 from Carnegie Mellon, Pittsburgh, PA. From 1985 through 1987 he worked for Zoran Microelectronics in Haifa as VLSI design engineer on the design of digital signal processors. Currently, he is member of technical staff at Bell Laboratories, Murray Hill, NJ. His research interests include CAD for VLSI circuits, more specifically simulation and statistical circuit design.

Layton T. Watson was born in Vanderburg County, Indiana, on December 24, 1947. He received a B.A. (magna cum laude) in psychology and mathematics from the University of Evansville, Evansville, Indiana, in 1969 and a Ph.D. in mathematics from the University of Michigan, Ann Arbor, in 1974. He has worked for USNAD Crane, Sandia National Laboratories, and General Motors research Laboratories and has served on the faculties of the University of Michigan and Michigan State University, Blacksburg, where he is currently a professor of computer sicence and mathemataics. His current research interests include fluid dynamics, structural mechanics, homotopy algorithms, parallel computation, mathematical software, and image processing. Dr. Watson is a member of Phi Kappa Phi, Blue Key, Psi Chi, Kappa Mu Epsilon, Phi Beta Chi, ACM, IEEE, and SIAM. He is an associate editor of the *SIAM Journal on Optimization* and the *ORSA Journal on Computing.*