Rollins, J.G., Bendix, P. "Computer Software for Circuit Analysis and Design"
The Electrical Engineering Handbook
Ed. Richard C. Dorf
Boca Raton: CRC Press LLC, 2000

13[1]

Computer Software for Circuit Analysis and Design

J. Gregory Rollins
Technology Modeling Associates, Inc.

Peter Bendix
LSI Logic Corp.

13.1 Analog Circuit Simulation
Introduction • DC (Steady-State) Analysis • AC Analysis • Transient Analysis • Process and Device Simulation • Process Simulation • Device Simulation • Appendix
13.2 Parameter Extraction for Analog Circuit Simulation
Introduction • MOS DC Models • BSIM Extraction Strategy in Detail

13.1 Analog Circuit Simulation

J. Gregory Rollins

Introduction

Computer-aided simulation is a powerful aid during the design or analysis of electronic circuits and semiconductor devices. The first part of this chapter focuses on analog circuit simulation. The second part covers simulations of semiconductor processing and devices. While the main emphasis is on analog circuits, the same simulation techniques may, of course, be applied to digital circuits (which are, after all, composed of analog circuits). The main limitation will be the size of these circuits because the techniques presented here provide a very detailed analysis of the circuit in question and, therefore, would be too costly in terms of computer resources to analyze a large digital system.

The most widely known and used circuit simulation program is SPICE (simulation program with integrated circuit emphasis). This program was first written at the University of California at Berkeley by Laurence Nagel in 1975. Research in the area of circuit simulation is ongoing at many universities and industrial sites. Commercial versions of SPICE or related programs are available on a wide variety of computing platforms, from small personal computers to large mainframes. A list of some commercial simulator vendors can be found in the Appendix.

It is possible to simulate virtually any type of circuit using a program like SPICE. The programs have built-in elements for resistors, capacitors, inductors, dependent and independent voltage and current sources, diodes, MOSFETs, JFETs, BJTs, transmission lines, transformers, and even transformers with saturating cores in some versions. Found in commercial versions are libraries of standard components which have all necessary

[1]The material in this chapter was previously published by CRC Press in *The Circuits and Filters Handbook,* Wai-Kai Chen, Ed., 1995.

parameters prefitted to typical specifications. These libraries include items such as discrete transistors, op amps, phase-locked loops, voltage regulators, logic integrated circuits (ICs) and saturating transformer cores.

Computer-aided circuit simulation is now considered an essential step in the design of integrated circuits, because without simulation the number of "trial runs" necessary to produce a working IC would greatly increase the cost of the IC. Simulation provides other advantages, however:

- The ability to measure "inaccessible" voltages and currents. Because a mathematical model is used all voltages and currents are available. No loading problems are associated with placing a voltmeter or oscilloscope in the middle of the circuit, with measuring difficult one-shot wave forms, or probing a microscopic die.
- Mathematically ideal elements are available. Creating an ideal voltage or current source is trivial with a simulator, but impossible in the laboratory. In addition, all component values are exact and no parasitic elements exist.
- It is easy to change the values of components or the configuration of the circuit. Unsoldering leads or redesigning IC masks are unnecessary.

Unfortunately, computer-aided simulation has its own problems:

- Real circuits are distributed systems, not the "lumped element models" which are assumed by simulators. Real circuits, therefore, have resistive, capacitive, and inductive parasitic elements present besides the intended components. In high-speed circuits these parasitic elements are often the dominant performance-limiting elements in the circuit, and must be painstakingly modeled.
- Suitable predefined numerical models have not yet been developed for certain types of devices or electrical phenomena. The software user may be required, therefore, to create his or her own models out of other models which are available in the simulator. (An example is the solid-state thyristor which may be created from a NPN and PNP bipolar transistor.)
- The numerical methods used may place constraints on the form of the model equations used.

The following sections consider the three primary simulation modes: DC, AC, and transient analysis. In each section an overview is given of the numerical techniques used. Some examples are then given, followed by a brief discussion of common pitfalls.

DC (Steady-State) Analysis

DC analysis calculates the state of a circuit with fixed (non-time varying) inputs after an infinite period of time. DC analysis is useful to determine the operating point (Q-point) of a circuit, power consumption, regulation and output voltage of power supplies, transfer functions, noise margin and fanout in logic gates, and many other types of analysis. In addition DC analysis is used to find the starting point for AC and transient analysis. To perform the analysis the simulator performs the following steps:

1. All capacitors are removed from the circuit (replaced with opens).
2. All inductors are replaced with shorts.
3. Modified nodal analysis is used to construct the nonlinear circuit equations. This results in one equation for each circuit node plus one equation for each voltage source. Modified nodal analysis is used rather than standard nodal analysis because an ideal voltage source or inductance cannot be represented using normal nodal analysis. To represent the voltage sources, loop equations (one for each voltage source or inductor), are included as well as the standard node equations. The node voltages and voltage source currents, then, represent the quantities which are solved for. These form a vector $\mathbf{x}$. The circuit equations can also be represented as a vector $\mathbf{F}(\mathbf{x}) = \mathbf{0}$.
4. Because the equations are nonlinear, Newton's method (or a variant thereof) is then used to solve the equations.

Example 13.1. **Simulation Voltage Regulator:** We shall now consider simulation of the type 723 voltage regulator IC, shown in Fig. 13.1. We wish to simulate the IC and calculate the sensitivity of the output IV

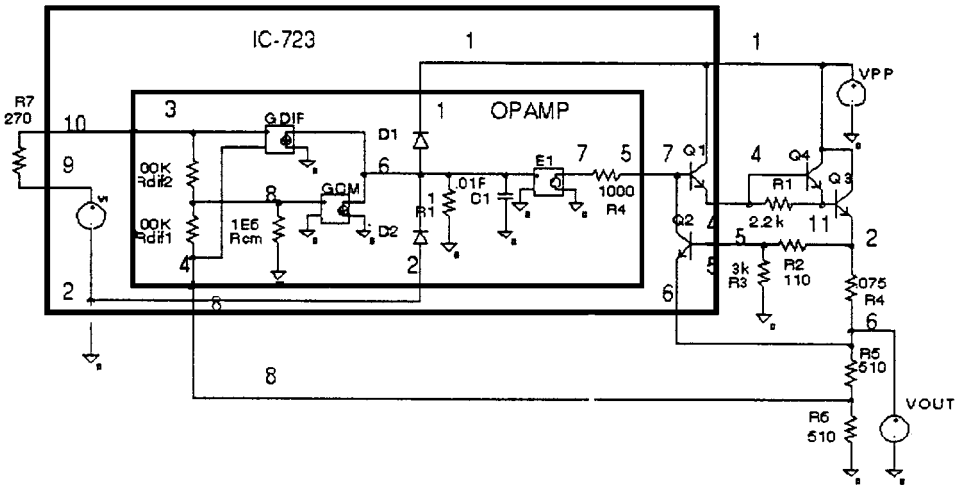

FIGURE 13.1 Regulator circuit to be used for DC analysis, created using PSPICE.

characteristic and verify that the output current follows a "fold-back" type characteristic under overload conditions.

The IC itself contains a voltage reference source and operational amplifier. Simple models for these elements are used here rather than representing them in their full form, using transistors, to illustrate model development. The use of simplified models can also greatly reduce the simulation effort. (For example, the simple op amp used here requires only eight nodes and ten components, yet realizes many advanced features.)

Note in Fig. 13.1 that the numbers next to the wires represent the circuit nodes. These numbers are used to describe the circuit to the simulator. In most SPICE-type simulators the nodes are represented by numbers, with the ground node being node zero. Referring to Fig. 13.2, the 723 regulator and its internal op amp are represented by subcircuits. Each subcircuit has its own set of nodes and components. Subcircuits are useful for encapsulating sections of a circuit or when a certain section needs to be used repeatedly (see next section).

The following properties are modeled in the op amp:

1. Common mode gain
2. Differential mode gain
3. Input impedance
4. Output impedance
5. Dominant pole
6. Output voltage clipping

The input terminals of the op amp connect to a "T" resistance network, which sets the common and differential mode input resistance. Therefore, the common mode resistance is RCM + RDIF = 1.1E6 and the differential mode resistance is RDIF1 + RDIF2 = 2.0E5.

Dependent current sources are used to create the main gain elements. Because these sources force current into a 1-Ω resistor, the voltage gain is Gm*R at low frequency. In the differential mode this gives (GDIF*R1 = 100). In the common mode this gives (GCM*R1*(RCM/(RDIF1 + RCM = 0.0909). The two diodes D1 and D2 implement clipping by preventing the voltage at node 6 from exceeding VCC or going below VEE. The diodes are made "ideal" by reducing the ideality factor n. Note that the diode current is $I_d = I_s[\exp(V_d/(nV_t)) - 1]$, where V_t is the thermal voltage (0.026 V). Thus, reducing n makes the diode turn on at a lower voltage.

A single pole is created by placing a capacitor (C1) in parallel with resistor R1. The pole frequency is therefore given by 1.0/(2*π*R1*C1). Finally, the output is driven by the voltage-controlled voltage source E1 (which has a voltage gain of unity), through the output resistor R4. The output resistance of the op amp is therefore equal to R4.

To observe the output voltage as a function of resistance, the regulator is loaded with a voltage source (VOUT) and the voltage source is swept from 0.05 to 6.0 V. A plot of output voltage vs. resistance can then be obtained

```
Regulator circuit.
* Complete circuit *
* Load source*
vout 6 0
* Power input *
vpp 1 0 11
x1 1 0 4 5 6 7 8 9 10 ic723
* Series Pass transistors *
q3 1 4 11 mq3
q4 1 11 2 mq4
r1 4 11 2.2k
r2 5 2 110
r3 5 0 3k
r4 2 6 0.075
r5 6 8 510
r6 8 0 510
r7 9 10 270
* Control cards *
.op
.model mq3 npn(is=1e-9 bf=30
+ br=5 ikf=50m)
.model mq4 npn(is=1e-6 bf=30
+ br=5 ikf=10)
.dc vout 1 5.5 .01
.plot dc i(vout)
.probe

.subckt ic723 1 2  4 5 6 7 8 9 10
* Type 723 voltage regulator *
x1 1 2 10 8 7 opamp
* Internal voltage reference *
vr 9 2 2.5
q1 3 7 4 mm
q2 7 5 6 mm
.model mm npn (is=1e-12 bf=100
+ br=5)
.ends ic723
* Ideal opamp with limiting
.subckt opamp 1   2   3   4   5
*              vcc vee +in -in out
rdif1 3 8 1e5
rdif2 4 8 1e5
rcm  8 0 1e6
* Common mode gain *
gcm 6 0 8 0 1e-1
* Differential mode gain *
gdif 6 0 4 3 100
r1 6 0 1
* Single pole response *
c1 6 0 .01
d1 6 1 ideal
d2 2 6 ideal
e1 7 0 6 0 1
rout 5 7 1e3
.model ideal d (is=1e-6 n=.01)
.ends opamp
```

FIGURE 13.2 SPICE input listing of regulator circuit shown in Fig. 13.1.

by plotting VOUT vs. VOUT/I(VOUT) (using PROBE in this case; see Fig. 13.3). Note that for this circuit, eventhough a current source would seem a more natural choice, a voltage source must be used as a load rather than a current source because the output characteristic curve is multivalued in current. If a current source were used it would not be possible to easily simulate the entire curve. Of course, many other interesting quantities can be plotted; for example, the power dissipated in the pass transistor can be approximated by plotting IC(Q3)*VC(Q3).

For these simulations PSPICE was used running on an IBM PC. The simulation took < 1 min of CPU time.

Pitfalls. Convergence problems are sometimes experienced if "difficult" bias conditions are created. An example of such a condition is if a diode is placed in the circuit backwards, resulting in a large forward bias voltage, SPICE will have trouble resolving the current. Another difficult case is if a current source were used instead of

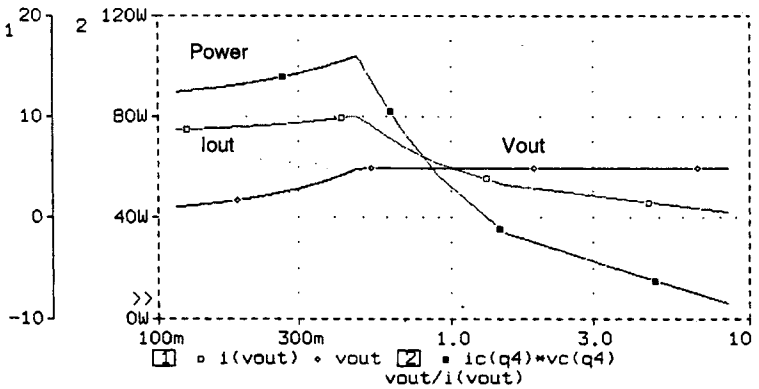

FIGURE 13.3 Output characteristics of regulator circuit using PSPICE.

a voltage to bias the output in the previous example. If the user then tried to increase the output current above 10 A, SPICE would not be able to converge because the regulator will not allow such a large current.

AC Analysis

Ac analysis uses phasor analysis to calculate the frequency response of a circuit. The analysis is useful for calculating the gain. 3 dB frequency input and output impedance, and noise of a circuit as a function of frequency, bias conditions, temperature, etc.

Numerical Method

1. A DC solution is performed to calculate the Q-point for the circuit.
2. A linearized circuit is constructed at the Q point. To do this, all nonlinear elements are replaced by their linearized equivalents. For example, a nonlinear current source $I = aV_1^2 + bV_2^3$ would be replaced by a linear voltage controlled current source $I = V_1(2aV_{1q}) + V_2(3bV_{2q}^2)$.
3. All inductors and capacitors are replaced by complex impedances, and conductances evaluated at the frequency of interest.
4. Nodal analysis is now used to reduce the circuit to a linear algebraic complex matrix. The AC node voltages may now be found by applying an excitation vector (which represents the independent voltage and current sources) and using Gaussian elimination (with complex arithmetic) to calculate the node voltages.

AC analysis does have limitations and the following types of nonlinear or large signal problems cannot be modeled:

1. Distortion due to nonlinearities such as clipping, etc.
2. Slew rate-limiting effects
3. Analog mixers
4. Oscillators

Noise analysis is performed by including noise sources in the models. Typical noise sources include thermal noise in resistors $I_n^2 = 4kT\,\Delta f/R$, and shot $I_n^2 = 2qI_d\,\Delta f$, and flicker noise in semiconductor devices. Here, T is temperature in Kelvins, k is Boltzmann's constant, and Δf is the bandwidth of the circuit. These noise sources are inserted as independent current sources, $\mathrm{In}_j(f)$ into the AC model. The resulting current due to the noise source is then calculated at a user-specified summation node(s) by multiplying by the gain function between the noise source and the summation node $A_{js}(f)$. This procedure is repeated for each noise source and then the contributions at the reference node are root mean squared (RMS) summed to give the total noise at the reference node. The equivalent input noise is then easily calculated from the transfer function between the circuit input and the reference node $A_{is}(f)$. The equation describing the input noise is therefore:

$$I_i = \frac{1}{A_{is}(f)} \sqrt{\sum_j \left[A_{js}(f)\,In_j(f)\right]^2}$$

Example 13.2. **Cascode Amplifier with Macro Models:** Here, we find the gain, bandwidth, input impedance, and output noise of a cascode amplifier. The circuit for the amplifier is shown in Fig. 13.5. The circuit is assumed to be fabricated in a monolithic IC process, so it will be necessary to consider some of the parasitics of the IC process. A cross-section of a typical IC bipolar transistor is shown in Fig. 13.4 along with some of the parasitic elements. These parasitic elements are easily included in the amplifier by creating a "macro model" for each transistor. The macro model is then implemented in SPICE form using subcircuits.

The input to the circuit is a voltage source (VIN), applied differentially to the amplifier. The output will be taken differentially across the collectors of the two upper transistors at nodes 2 and 3. The input impedance of the amplifier can be calculated as VIN/I(VIN) or because VIN = 1.0 just as 1/I(VIN). These quantities are shown plotted using PROBE in Fig. 13.6. It can be seen that the gain of the amplifier falls off at high frequency

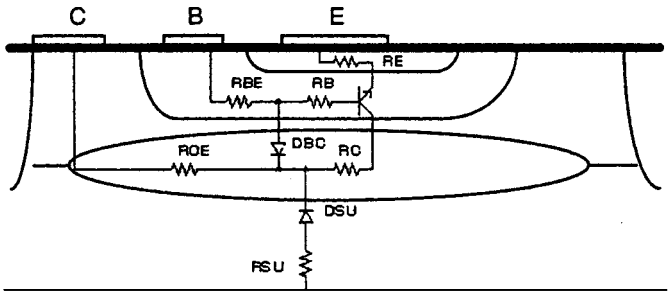

FIGURE 13.4 BJT cross-section with macro model elements.

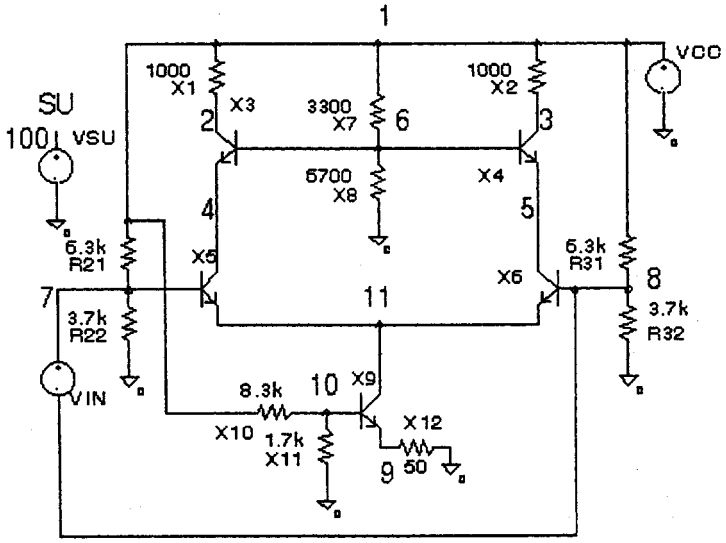

FIGURE 13.5 Cascode amplifier for AC analysis, created using PSPICE.

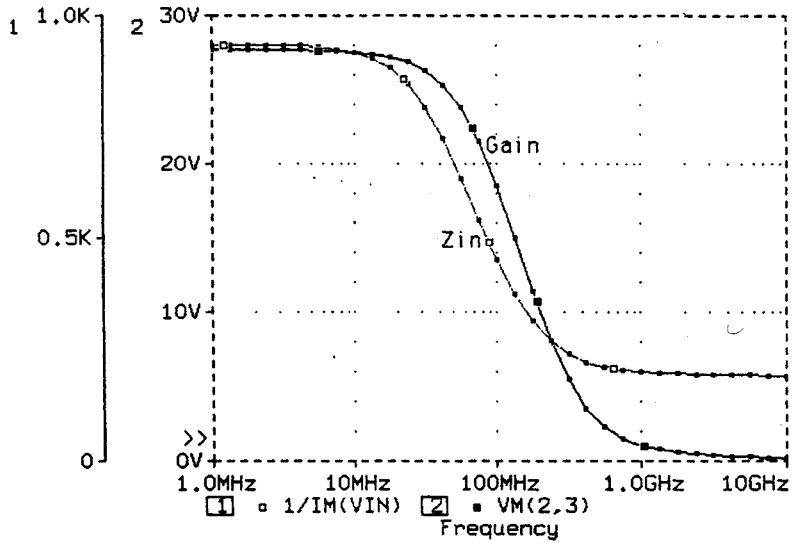

FIGURE 13.6 Gain and input impedance of cascode amplifier.

as expected. The input impedance also drops because parasitic capacitances shunt the input. This example took <1 min on an IBM PC.

Pitfalls. Many novice users will forget that AC analysis is a linear analysis. They will, for example, apply a 1-V signal to an amplifier with 5-V power supplies and a gain of 1000 and be surprised when SPICE tells them that the output voltage is 1000 V. Of course, the voltage generated in a simple amplifier must be less than the power supply voltage, but to examine such clipping effects, transient analysis must be used. Likewise, selection of a proper Q point is important. If the amplifier is biased in a saturated portion of its response and AC analysis is performed, the gain reported will be much smaller than the actual large signal gain.

Transient Analysis

Transient analysis is the most powerful analysis capability of a simulator because the transient response is so hard to calculate analytically. Transient analysis can be used for many types of analysis, such as switching speed, distortion, basic operation of certain circuits like switching power supplies. Transient analysis is also the most CPU intensive and can require 100 or 1000 times the CPU time as a DC or AC analysis.

Numerical Method

In a transient analysis time is discretized into intervals called time steps. Typically the time steps are of unequal length, with the smallest steps being taken during portions of the analysis when the circuit voltages and currents are changing most rapidly. The capacitors and inductors in the circuit are then replaced by voltage and current sources based on the following procedure.

The current in a capacitor is given by $I_c = CdV_c/dt$. The time derivative can be approximated by a difference equation:

$$I_c^k + I_c^{k-1} = 2C\frac{V_c^k - V_c^{k-1}}{t^k - t^{k-1}}$$

In this equation the superscript k represents the number of the time step. Here, k is the time step we are presently solving for and $(k-1)$ is the previous time step. This equation can be solved to give the capacitor current at the present time step.

$$I_c^k = V_c^k\left(2C/\Delta t\right) - V_c^{k-1}\left(2C/\Delta t\right) - I_c^{k-1}.$$

Here, $\Delta t = t^k - t^{k-1}$, or the length of the time step. As time steps are advanced, $V_c^{k-1} \rightarrow V_c^k$; $I_c^{k-1} \rightarrow I_c^k$. Note that the second two terms on the right hand side of the above equation are dependent only on the capacitor voltage and current from the previous time step, and are therefore fixed constants as far as the present step is concerned. The first term is effectively a conductance ($g = 2C/\Delta t$) multiplied by the capacitor voltage, and the second two terms could be represented by an independent current source. The entire transient model for the capacitor therefore consists of a conductance in parallel with two current sources (the numerical values of these are, of course, different at each time step). Once the capacitors and inductors have been replaced as indicated, the normal method of DC analysis is used. One complete DC analysis must be performed for each time point. This is the reason that transient analysis is so CPU intensive. The method outlined here is the trapezoidal time integration method and is used as the default in SPICE.

Example 13.3. **Phase-Locked Loop Circuit:** Figure 13.7 shows the phase-locked loop circuit. The phase detector and voltage-controlled oscillator are modeled in separate subcircuits. Examine the VCO subcircuit and note the PULSE-type current source ISTART connected across the capacitor. The source gives a current pulse 03.E-6 s wide at the start of the simulation to start the VCO running. To start a transient simulation SPICE first computes a DC operating point (to find the initial voltages V_c^{k-1} on the capacitors). As this DC point is a valid, although not necessarily stable, solution, an oscillator will remain at this point indefinitely unless some perturbation is applied to start the oscillations. Remember, this is an ideal mathematical model and no noise

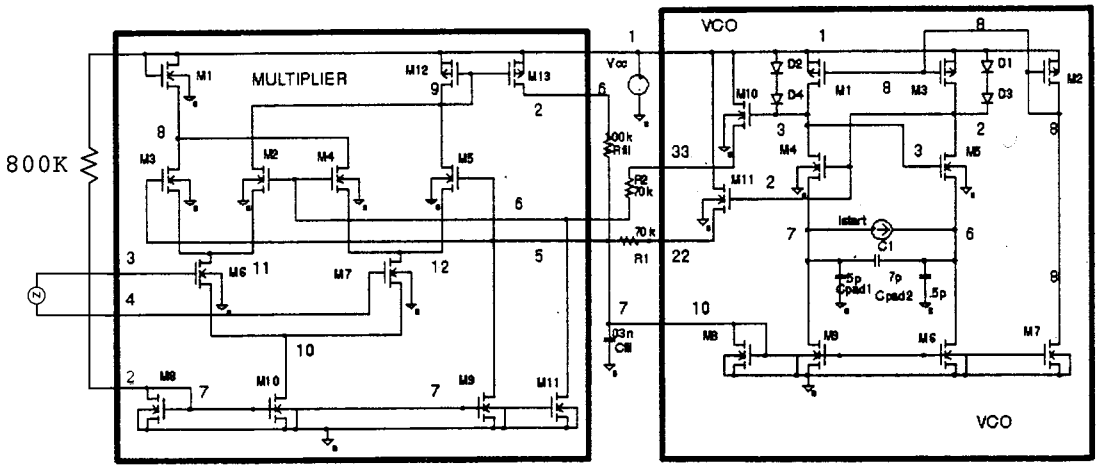

FIGURE 13.7 Phase-locked loop circuit for transient analysis, created with PSPICE.

sources or asymmetries exist that would start a real oscillator—it must be done manually. The capacitor C1 would have to be placed off-chip, and bond pad capacitance (CPAD1 and CPAD2) have been included at the capacitor nodes. Including the pad capacitances is very important if a small capacitor C1 is used for high-frequency operation.

In this example, the PLL is to be used as a FM detector circuit and the FM signal is applied to the input using a single frequency FM voltage source. The carrier frequency is 600 kHz and the modulation frequency is 60 kHz. Figure 13.8 shows the input voltage and the output voltage of the PLL at the VCO output and at the phase detector output. It can be seen that after a brief starting transient, the PLL locks onto the input signal

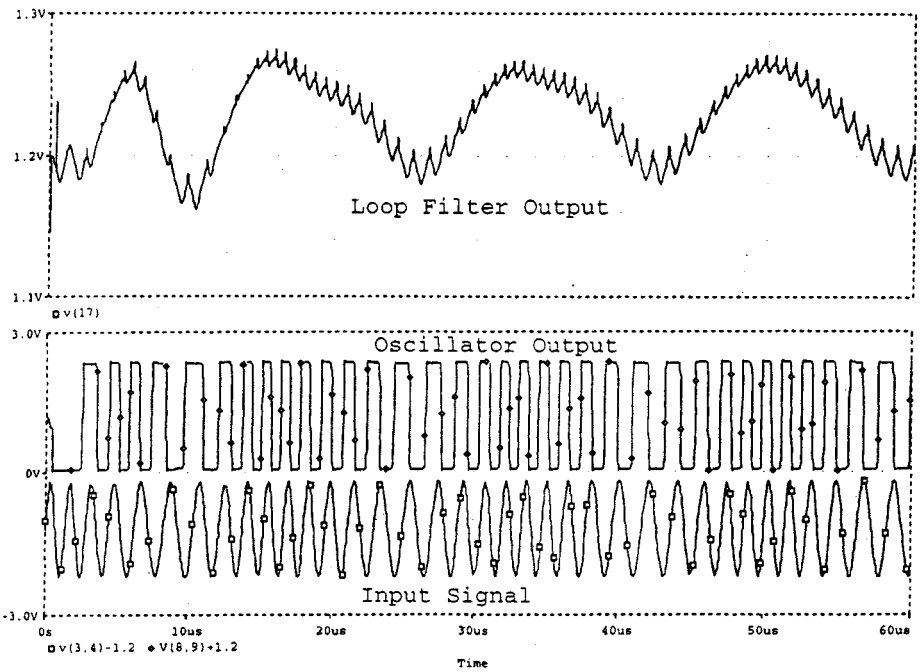

FIGURE 13.8 Transient analysis results of PLL circuit, created using PSPICE.

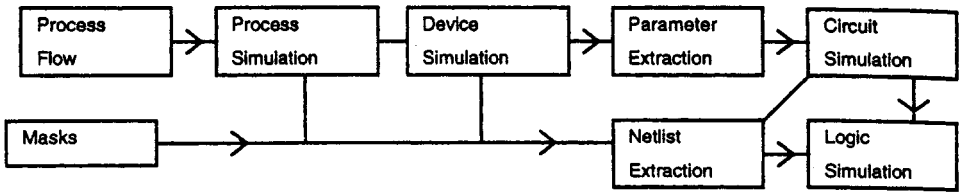

FIGURE 13.9 Data flow for complete process-device-circuit modeling.

and that the phase detector output has a strong 60-kHz component. This example took 251 s on a Sun SPARC-2 workstation (3046 time steps, with an average of 5 Newton iterations per time step).

Pitfalls. Occasionally SPICE will fail and give the message "Timestep too small in transient analysis", which means that the process of Newton iterations at certain time steps could not be made to converge. One of the most common causes of this is the specification of a capacitor with a value that is much too large, for example, specifying a 1-F capacitor instead of a 1 pF capacitor (an easy mistake to make by not adding the "p" in the value specification). Unfortunately, we usually have no way to tell which capacitor is at fault from the type of failure generated other than to manually search the input deck.

Other transient failures are caused by MOSFET models. Some models contain discontinuous capacitances (with respect to voltage) and others do not conserve charge. These models can vary from version to version so it is best to check the user's guide.

Process and Device Simulation

Process and devices simulation are the steps that precede analog circuit simulation in the overall simulation flow (see Fig. 13.9). The simulators are also different in that they are not measurement driven as are analog circuit simulators. The input to a process simulator is the sequence of process steps performed (times, temperatures, gas concentrations) as well as the mask dimensions. The output from the process simulator is a detailed description of the solid-state device (doping profiles, oxide thickness, junction depths, etc.). The input to the device simulator is the detailed description generated by the process simulator (or via measurement). The output of the device simulator is the electrical characteristics of the device (IV curves, capacitances, switching transient curves).

Process and device simulation are becoming increasingly important and widely used during the integrated circuit design process. A number of reasons exist for this:

- As device dimensions shrink, second-order effects can become dominant. Modeling of these effects is difficult using analytical models.
- Computers have greatly improved, allowing time-consuming calculations to be performed in a reasonable amount of time.
- Simulation allows access to impossible to measure physical characteristics.
- Analytic models are not available for certain devices, for example, thyristors, heterojunction devices and IGBTS.
- Analytic models have not been developed for certain physical phenomena, for example, single event upset, hot electron aging effects, latchup, and snap-back.
- Simulation runs can be used to replace split lot runs. As the cost to fabricate test devices increases, this advantage becomes more important.
- Simulation can be used to help device, process, and circuit designers understand how their devices and processes work.

Clearly, process and device simulation is a topic which can be and has been the topic of entire texts. The following sections attempt to provide an introduction to this type of simulation, give several examples showing what the simulations can accomplish, and provide references to additional sources of information.

Process Simulation

Integrated circuit processing involves a number of steps which are designed to deposit (deposition, ion implantation), remove (etching), redistribute (diffusion), or transform (oxidation) the material of which the IC is made. Most process simulation work has been in the areas of diffusion, oxidation, and ion implantation; however, programs are available that can simulate the exposure and development of photo-resist, the associated optical systems, as well as gas and liquid phase deposition and etch.

In the following section a very brief discussion of the governing equations used in SUPREM (from Stanford University, California) will be given along with the results of an example simulation showing the power of the simulator.

Diffusion

The main equation governing the movement of electrically charged impurities (acceptors in this case) in the crystal is the diffusion equation:

$$\frac{\partial C}{\partial t} = \nabla \cdot \left(D\nabla C - \frac{DqC_a}{kT} \mathbf{E} \right)$$

Here, C is the concentration (#/cm^3) of impurities, C_a is the number of electrically active impurities (#/cm^3), q is the electron charge, k is Boltzmann's constant, T is temperature in degrees Kelvin, D is the diffusion constant, and E is the built-in electric field. The built-in electric field E in (V/cm) can be found from:

$$\mathbf{E} = -\frac{kT}{q}\frac{1}{n}\nabla n$$

In this equation n is the electron concentration (#/cm^3), which in turn can be calculated from the number of electrically active impurities (C_a). The diffusion constant (D) is dependent on many factors. In silicon the following expression is commonly used:

$$D = F_{IV}\left(D_x + D_+ \frac{n_i}{n} + D_- \frac{n}{n_i} + D_= \left(\frac{n}{n_i} \right)^2 \right)$$

The four D components represent the different possible charge states for the impurity: (x) neutral, (+) positive, (–) negative, (=) doubly negatively charged. n_i is the intrinsic carrier concentration, which depends only on temperature. Each D component is in turn given by an expression of the type

$$D = A \exp\left(-\frac{B}{kT} \right)$$

Here, A and B are experimentally determined constants, different for each type of impurity (x, +, –, =). B is the activation energy for the process. This expression derives from the Maxwellian distribution of particle energies and will be seen many times in process simulation. It is easily seen that the diffusion process is strongly influenced by temperature. The term F_{IV} is an enhancement factor which is dependent on the concentration of interstitials and vacancies within the crystal lattice (an interstitial is an extra silicon atom which is not located on a regular lattice site; a vacancy is a missing silicon atom which results in an empty lattice site) $F_{IV} \propto C_I + C_v$. The concentration of vacancies, C_v, and interstitials, C_I, are in turn determined by their own diffusion equation:

$$\frac{\partial C_v}{\partial t} = +\nabla \cdot D_V \cdot \nabla C_V - R + G$$

In this equation D_V is another diffusion constant of the form $A \exp(-B/kT)$. R and G represent the recombination and generation of vacancies and interstitials. Note that an interstitial and a vacancy may recombine and in the process destroy each other, or an interstitial and a vacancy pair may be simultaneously generated by knocking a silicon atom off its lattice site. Recombination can occur anywhere in the device via a bulk recombination process $R = A(C_V C_1)\exp(-B/kT)$. Generation occurs where there is damage to the crystal structure, in particular at interfaces where oxide is being grown or in regions where ion implantation has occurred, as the high-energy ions can knock silicon atoms off their lattice sites.

Oxidation

Oxidation is a process whereby silicon reacts with oxygen (or with water) to form new silicon dioxide. Conservation of the oxidant requires the following equation:

$$\frac{dy}{dt} = \frac{\mathbf{F}}{N}$$

Here, F is the flux of oxidant (#/cm^2/s), N is the number of oxidant atoms required to make up a cubic centimeter of oxide, and dy/dt is the velocity with which the Si-SiO_2 interface moves into the silicon. In general the greater the concentration of oxidant (C_0), the faster the growth of the oxide and the greater the flux of oxidant needed at the Si-SiO_2 interface. Thus, $F = k_s C_0$. The flux of oxidant into the oxide from the gaseous environment is given by:

$$F = h(HP_{ox} - C_0)$$

Here H is a constant, P is the partial pressure of oxygen in the gas, and C_0 is the concentration of oxidant in the oxide at the surface and h is of the form $A \exp(-B/kT)$. Finally, the movement of the oxidant within the already existing oxide is governed by diffusion: $\mathbf{F} = D_0 \nabla C$. When all these equations are combined, it is found that (in the one-dimensional case) oxides grow linearly $dy/dt \propto t$ when the oxide is thin and the oxidant can move easily through the existing oxide. As the oxide grows thicker $dy/dt \propto \sqrt{t}$ because the movement of the oxidant through the existing oxide becomes the rate-limiting step.

Modeling two-dimensional oxidation is a challenging task. The newly created oxide must "flow" away from the interface where it is begin generated. This flow of oxide is similar to the flow of a very thick or viscous liquid and can be modeled by a creeping flow equation:

$$\nabla_2 V \propto \nabla P$$

$$\nabla \cdot V = 0$$

V is the velocity at which the oxide is moving and P is the hydrostatic pressure. The second equation results form the incompressibility of the oxide. The varying pressure P within the oxide leads to mechanical stress, and the oxidant diffusion constant D_0 and the oxide growth rate constant k_s are both dependent on this stress. The oxidant flow and the oxide flow are therefore coupled because the oxide flow depends on the rate at which oxide is generated at the interface and the rate at which the new oxide is generated depends on the availability of oxidant, which is controlled by the mechanical stress.

Ion Implantation

Ion implantation is normally modeled in one of two ways. The first involves tables of moments of the final distribution of the ions which are typically generated by experiment. These tables are dependent on the energy and the type of ion being implanted. The second method involves Monte-Carlo simulation of the implantation process. In Monte-Carlo simulation the trajectories of individual ions are followed as they interact with (bounce off) the silicon atoms in the lattice. The trajectories of the ions, and the recoiling Si atoms (which can strike more Si atoms) are followed until all come to rest within the lattice. Typically several thousand trajectories are

simulated (each will be different due to the random probabilities used in the Monte-Carlo method) to build up the final distribution of implanted ions.

Process simulation is always done in the transient mode using time steps as was done with transient circuit simulation. Because partial differential equations are involved, rather than ordinary differential equations, spatial discretization is needed as well. To numerically solve the problem, the differential equations are discretized on a grid. Either rectangular or triangular grids in one, two, or three dimensions are commonly used. This discretization process results in the conversion of the partial differential equations into a set of nonlinear algebraic equations. The nonlinear equations are then solved using a Newton method in a way very similar to the method used for the circuit equations in SPICE.

Example 13.4. **NMOS Transistor:** In this example the process steps used to fabricate a typical NMOS transistor will be simulated using SUPREM-4. These steps are

1. Grow initial oxide (30 min at 1000 K)
2. Deposit nitride layer (a nitride layer will prevent oxidation of the underlying silicon)
3. Etch holes in nitride layer
4. Implant P+ channel stop (boron dose = 5e12, energy = 50 keV)
5. Grow the field oxide (180 min at 1000 K wet O_2)
6. Remove all nitride
7. Perform P channel implant (boron dose = 1e11, energy = 40 keV)
8. Deposit and etch polysilicon for gate
9. Oxidize the polysilicon (30 min at 1000 K, dry O_2)
10. Implant the light doped drain (arsenic dose = 5e13 energy = 50 keV)
11. Deposit sidewall space oxide
12. Implant source and drain (arsenic, dose = 1e15, energy = 200 keV)
13. Deposit oxide layer and etch contact holes
14. Deposit and etch metal

The top 4 μm of the completed structure, as generated by SUPREM-4, is shown in Fig. 13.10. The actual simulation structure used is 200 μm deep to allow correct modeling of the diffusion of the vacancies and interstitials. The gate is at the center of the device. Notice how the edges of the gate have lifted up due to the diffusion of oxidant under the edges of the polysilicon (the polysilicon, as deposited in step 8, is flat). The dashed contours show the concentration of dopants in both the oxide and silicon layers. The short dashes

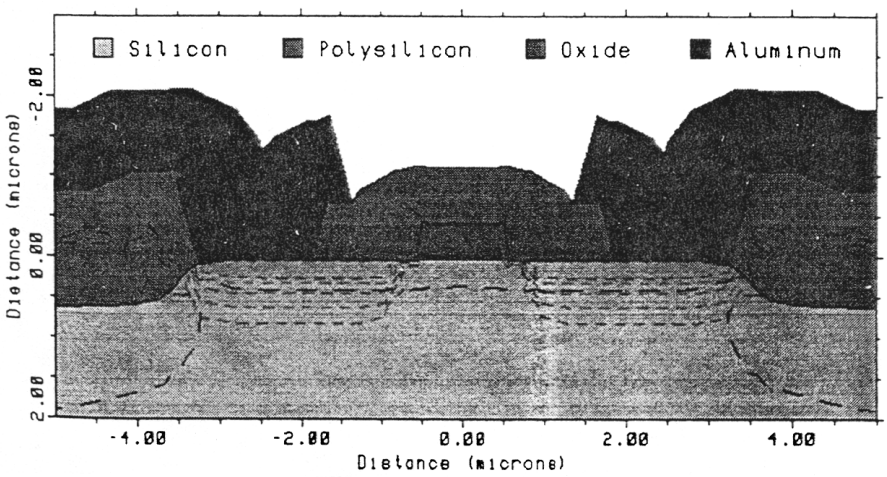

FIGURE 13.10 Complete NMOS transistor cross section generated by process simulation, created with TMA SUPREM-4.

indicate N-type material, while the longer dashes indicate P-type material. This entire simulation requires about 30 min on a Sun SPARC-2 workstation.

Device Simulation

Device simulation uses a different approach from that of conventional lumped circuit models to determine the electrical device characteristics. Whereas with analytic or empirical models all characteristics are determined by fitting a set of adjustable parameters to measured data, device simulators determine the electrical behavior by numerically solving the underlying set of differential equations. The first is the Poisson equation, which describes the electrostatic potential within the device

$$\nabla \cdot \varepsilon \cdot \nabla \Psi = q\left(N_a^- - N_d^+ - p + n - Q_f\right)$$

N_d and N_a are the concentration of donors and acceptors, i.e., the N- and P-type dopants. Q_f is the concentration of fixed charge due, for example, to traps or interface charge. The electron and hole concentrations are given by n and p, respectively, and Ψ is the electrostatic potential.

A set of continuity equations describes the conservation of electrons and holes:

$$\frac{\partial n}{\partial t} = \left(\frac{1}{q}\nabla \cdot \mathbf{J}_n - R + G\right)$$

$$\frac{\partial p}{\partial t} = \left(-\frac{1}{q}\nabla \cdot \mathbf{J}_p - R + G\right)$$

In these equations R and G describe the recombination and generation rates for the electrons and holes. The recombination process is influenced by factors such as the number of electrons and holes present as well as the doping and temperature. The generation rate is also dependent upon the carrier concentrations, but is most strongly influenced by the electric field, with increasing electric fields giving larger generation rates. Because this generation process is included, device simulators are capable of modeling the breakdown of devices at high voltage. $\mathbf{J}_n$ and $\mathbf{J}_p$ are the electron and hole current densities (in amperes per square centimeter). These current densities are given by another set of equations

$$\mathbf{J}_n = q\mu\left(-n\nabla\Psi + \frac{kT_n}{q}\nabla n\right)$$

$$\mathbf{J}_p = q\mu\left(-p\nabla\Psi - \frac{kT_p}{q}\nabla p\right)$$

In this equation k is Boltzmann's constant, μ is the carrier mobility, which is actually a complex function of the doping, n, p, electric field, temperature, and other factors. In silicon the electron mobility will range between 50 and 1000 and the hole mobility will normally be a factor of 2 smaller. In other semiconductors such as gallium arsenide the electron mobility can be as high as 5000. T_n and T_p are the electron and hole mean temperatures, which describe the average carrier energy. In many models these default to the device temperature (300 K). In the first term the current is proportional to the electric field ($\nabla\Psi$), and this term represents the drift of carriers with the electric field. In the second term the current is proportional to the gradient of the carrier concentration (∇n), so this term represents the diffusion of carriers from regions of high concentration to those of low concentration. The model is therefore called the drift-diffusion model.

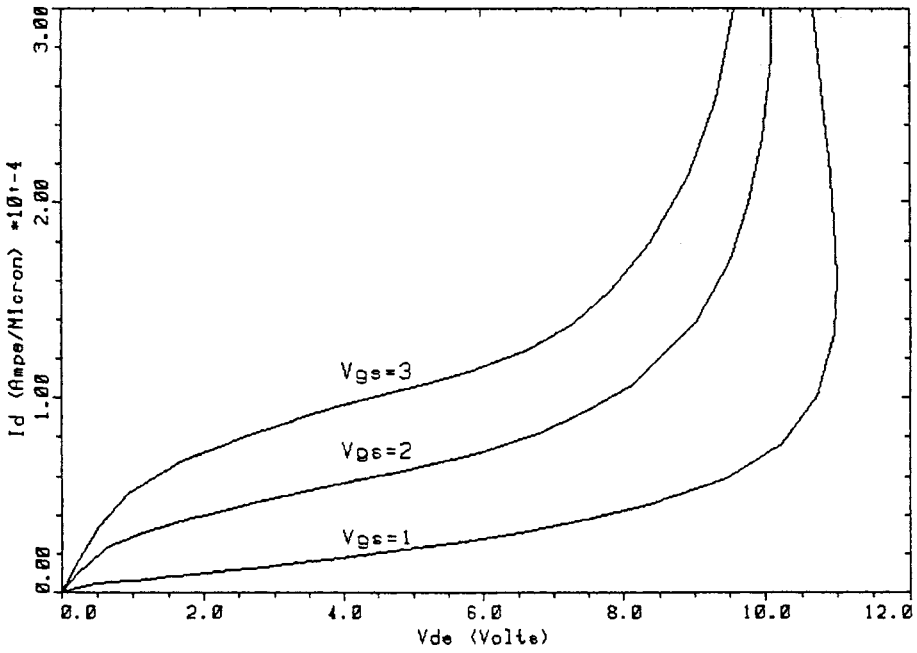

FIGURE 13.11 I_d vs. V_{ds} curves generated by device simulation, created with TMA MEDICI.

In devices in which self-heating effects are important, a lattice heat equation can also be solved to give the internal device temperature:

$$\sigma(T)\frac{\partial T}{\partial t} = H + \nabla \cdot \lambda(T) \cdot \nabla T$$

$$H = -(\mathbf{J}_n + \mathbf{J}_p) \cdot \nabla \Psi + H_R$$

where H is the heat generation term, which includes resistive (Joule) heating as well as recombination heating, H_u. The terms $\sigma(T)$, $\lambda(T)$ represent the specific heat and the thermal conductivity of the material (both temperature dependent). Inclusion of the heat equation is essential in many power device problems.

As with process simulation partial differential equations are involved, therefore, a spatial discretization is required. As with circuit simulation problems, various types of analysis are available:

- Steady state (DC), used to calculate characteristic curves of MOSFETs, BJTs diodes, etc.
- AC analysis, used to calculate capacitances, *Y*-parameters, small signal gains, and *S*-parameters.
- Transient analysis used for calculation of switching and large signal behavior, and special types of analysis such as radiation effects.

Example 13.5. **NMOS IV Curves:** The structure generated in the previous SUPREM-IV simulation is now passed into the device simulator and bias voltages are applied to the gate and drain. Models were included with account for Auger and Shockley Reed Hall recombination, doping and electric field-dependent mobility, and impact ionization. The set of drain characteristics obtained is shown in Fig. 13.11. Observe how the curves bend upward at high V_{ds} as the device breaks down. The V_g, = 1 curve has a negative slope at I_d = 1.5e-4A as the device enters snap-back. It is possible to model this type of behavior because impact ionization is included in the model.

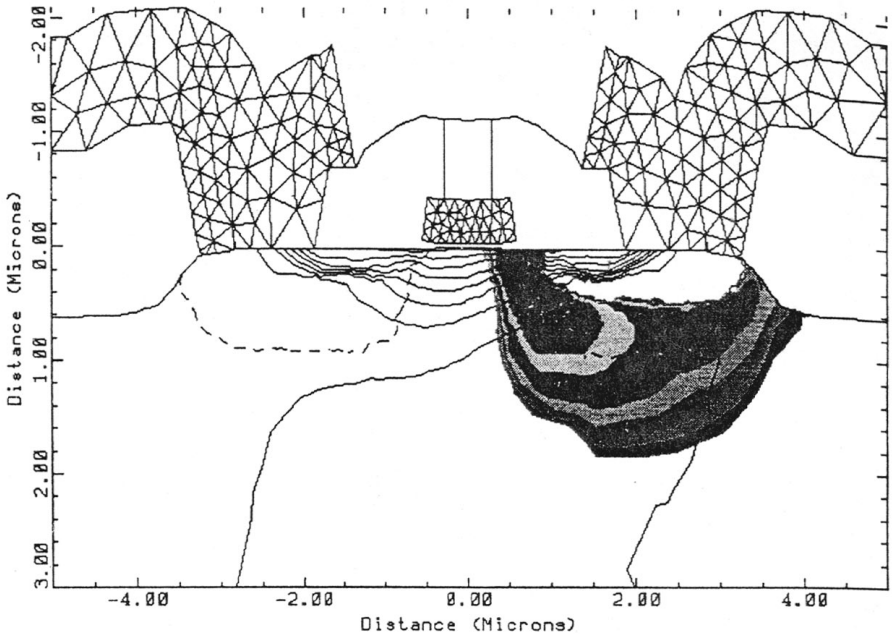

FIGURE 13.12 Internal behavior of MOSFET under bias, created with TMA MEDICI.

Figure 13.12 shows the internal behavior of the device with V_{gs} = 3 V and I_d = 3e-4A. The filled contours indicate impact ionization, with the highest rate being near the edge of the drain right beneath the gate. This is to be expected because this is the region in which the electric field is largest due to the drain depletion region. The dark lines indicate current flow from the source to the drain. Some current also flows from the drain to the substrate. This substrate current consists of holes generated by the impact ionization. The triangular grid used in the simulation can be seen in the source, drain, and gate electrodes. A similar grid was used in the oxide and silicon regions.

Appendix

Circuit Analysis Software

SPICE2, SPICE3: University of California, Berkeley
PSPICE: MicroSim Corporation, Irvine, CA (used in this chapter)
HSPICE: Meta Software, Campbell, CA
IsSPICE: Intusoft, San Pedro, CA
SPECTRE: Cadence Design Systems, San Jose, CA
SABRE: Analogy, Beaverton, OR

Process and Device Simulators

SUPREM-4, PISCES: Stanford University, Palo Alto, CA
MINIMOS: Technical University, Vienna, Austria
SUPREM-4, MEDICI, DAVINCI: Technology Modeling Associates, Palo Alto, CA (used in this chapter)
SEMICAD: Dawn Technologies, Sunnyvale, CA

Related Topics

3.2 Node and Mesh Analysis • 23.1 Processes • 27.1 Ideal and Practical Models

References

P. Antognetti and G. Massobrio, *Semiconductor Device Modeling with SPICE,* New York: McGraw-Hill, 1988.
P. W. Tuinenga, *SPICE, A Guide to Circuit Simulation and Analysis Using PSPICE,* Englewood Cliffs, N.J.: Prentice-Hall, 1988.
J. A. Connelly and P. Choi, *Macromodeling with SPICE,* Englewood Cliffs, N.J.: Prentice-Hall, 1992.
S. Selberherr, *Analysis and Simulation of Semiconductor Devices,* Berlin: Springer-Verlag, 1984.
R. Dutton and Z. Yu, *Technology CAD, Computer Simulation of IC Process and Devices,* Boston: Kluwer Academic, 1993.

13.2 Parameter Extraction for Analog Circuit Simulation

Peter Bendix

Introduction

Definition of Device Modeling

We use various terms such as device characterization, parameter extraction, optimization, and model fitting to address an important engineering task. In all of these, we start with a mathematical model that describes the transistor behavior. The model has a number of parameters which are varied or adjusted to match the IV (current-voltage) characteristics of a particular transistor or set of transistors. The act of determining the appropriate set of model parameters is what we call device modeling. We then use the model with the particular set of parameters that represent our transistors in a circuit simulator such as SPICE[1] to simulate how circuits with our kinds of transistors will behave. Usually the models are supplied by the circuit simulator we chose. Occasionally we may want to modify these models or construct our own models. In this case we need access to the circuit simulator model subroutines as well as the program that performs the device characterization.

Steps Involved in Device Characterization

Device characterization begins with a test chip. Without the proper test chip structures, proper device modeling cannot be done from measured data. A good test chip for MOS technology would include transistors of varying geometries, gate oxide capacitance structures, junction diode capacitance structures, and overlap capacitance structures. This would be a minimal test chip. Additional structures might include ring oscillators and other circuits for checking the AC performance of the models obtained. It is very important that the transistors be well designed and their geometries be chosen appropriate for the technology as well as the desired device model. Although a complete test chip description is beyond the scope of this book, be aware that even perfect device models cannot correct for a poor test chip.

Next we need data that represent the behavior of a transistor or set of transistors of different sizes. these data can come from direct measurement or they can be produced by a device simulator such as PISCES.[2]

It is also possible to use a combination of a process simulator like SUPREM-IV[3] coupled to a device simulator, to provide the simulated results. The benefits of using simulation over measurement are that no expensive measurement equipment or fabricated wafers are necessary. This can be very helpful when trying to predict the device characteristics of a new fabrication process before any wafers have been produced.

Once the measured (or simulated) data are available, parameter extraction software is used to find the best set of model parameter values to fit the data.

[1]SPICE is a circuit simulation program from the Department of Electrical Engineering and Computer Science at the University of California at Berkeley.

[2]PISCES is a process simulation program from the Department of Electrical Engineering at Stanford University, Stanford, CA.

[3]SUPREM-IV is a process simulation program from the Department of Electrical Engineering at Stanford University, Stanford, CA.

Least-Squares Curve Fitting (Analytical)

We begin this section by showing how to do least-squares curve fitting by analytical solutions, using a simple example to illustrate the method. We then mention least-squares curve fitting using numerical solutions in the next section. We can only find analytical solutions to simple problems. The more complex ones must rely on numerical techniques.

Assume a collection of measured data, $m_1, \ldots, m_n$. For simplicity, let these measured data values be functions of a single variable, v, which was varied from v_1 through v_n, measuring each m_i data point at each variable value v_i, i running from 1 to n. For example, the m_i data points might be drain current of an MOS transistor, and the v_i might be the corresponding values of gate voltage. Assume that we have a model for calculating simulated values of the measured data points, and let these simulated values be denoted by $s_1, \ldots, s_n$. We define the least-squares, root mean square (RMS) error as

$$\text{Error}_{\text{rms}} = \left[\frac{\sum_{i=1}^{n} \left\{ \text{weight}_i \left(s_i - m_i \right) \right\}^2}{\sum_{i=1}^{n} \left\{ \text{weight}_i m_i \right\}^2} \right]^{1/2} \tag{13.1}$$

where a weighting term is included for each data point. The goal is to have the simulated data match the measured data as closely as possible, which means we want to minimize the RMS error. Actually, what we have called the RMS error is really the relative RMS error, but the two terms are used synonymously. There is another way of expressing the error, called the absolute RMS error, defined as follows:

$$\text{Error}_{\text{rms}} = \left[\frac{\sum_{i=1}^{n} \left\{ \text{weight}_i \left(s_i - m_i \right) \right\}^2}{\sum_{i=1}^{n} \left\{ \text{weight}_i m_{\min} \right\}^2} \right]^{1/2} \tag{13.2}$$

where we have used the term $m_{\min}$ in the denominator to represent some minimum value of the measured data. The absolute RMS error is usually used when the measured values approach zero to avoid problems with small or zero denominators in (13.1). For everything that follows, we consider only the relative RMS error. The best result is obtained by combining the relative RMS formula with the absolute RMS formula by taking the maximum of the denominator from (13.1) or (13.2).

We have a simple expression for calculating the simulated data points, s_i, in terms of the input variable, v, and a number of model parameters, $p_1, \ldots, p_m$. That is,

$$s_i = f\left(v_i, p_1, \ldots, p_m \right) \tag{13.3}$$

where f is some function. Minimizing the RMS error function is equivalent to minimizing its square. Also, we can ignore the term in the denominator of (13.1) as concerns minimizing, because it is a normalization term. In this spirit, we can define a new error term,

$$\text{Error} = \left(\text{Error}_{\text{rms}} \right)^2 \left[\sum_{i=1}^{n} \left\{ \text{weight}_i m_i \right\}^2 \right] \tag{13.4}$$

and claim that minimizing Error is equivalent to minimizing Error_{rms}. To minimize Error, we set all partial derivatives of it with respect to each model parameter equal to zero; that is, write

$$\frac{\partial(\text{Error})}{\partial p_j} = 0, \qquad \text{for } j = 1, \ldots, m \tag{13.5}$$

Then solve the above equations for the value of p_j.

Least-Square Curve Fitting (Numerical)

For almost all practical applications we are forced to do least-squares curve fitting numerically, because the analytic solutions as previously discussed are not obtainable in closed form. What we are calling least-squares curve fitting is more generally known as nonlinear optimization. Many fine references on this topic are available. We refer the reader to [Gill et al., 1981] for details.

Extraction (as Opposed to Optimization)

The terms "extraction" and "optimization" are, unfortunately, used interchangeably in the semiconductor industry; however, strictly speaking, they are not the same. By optimization, we mean using generalized least-squares curve fitting methods such as the Levenberg-Marquardt algorithm [Gill et al., 1981] to find a set of model parameters. By extraction, we mean any technique that does not use general least-squares fitting methods. This is a somewhat loose interpretation of the term extraction. The main point is that we write the equations we want and then solve them by whatever approximations we choose, as long as these approximations allow us to get the extracted results in closed form. This is parameter extraction.

Extraction vs. Optimization

Extraction has the advantage of being much faster than optimization, but it is not always as accurate. It is also much harder to supply extraction routines for models that are being developed. Each time you make a change in the model, you must make suitable changes in the corresponding extraction routine. For optimization, however, no changes are necessary other than the change in the model itself, because least-squares curve fitting routines are completely general. Also, if anything goes wrong in the extraction algorithm (and no access to the source code is available), almost nothing can be done to correct the problem. With optimization, one can always change the range of data, weighting, upper and lower bounds, etc. A least-squares curve fitting program can be steered toward a correct solution.

Novices at device characterization find least-squares curve fitting somewhat frustrating because a certain amount of user intervention and intuition is necessary to obtain the correct results. These beginners prefer extraction methods because they do not have to do anything. However, after being burned by extraction routines that do not work, a more experienced user will usually prefer the flexibility, control, and accuracy that optimization provides.

Commercial software is available that provides both extraction and optimization together. The idea here is to first use extraction techniques to make reasonable initial guesses and then use these results as a starting point for optimization, because optimization can give very poor results if poor initial guesses for the parameters are used. Nothing is wrong with using extraction techniques to provide initial guesses for optimization, but for an experienced user this is rarely necessary, assuming that the least-squares curve fitting routine is robust (converges well) and the experienced user has some knowledge of the process under characterization. Software that relies heavily on extraction may do so because of the nonrobustness of its optimizer.

These comments apply when an experienced user is doing optimization locally, not globally. For global optimization (a technique we do not recommend), the above comparisons between extraction and optimization are not valid. The following section contains more detail about local vs. global optimization.

Strategies: General Discussion

The most naive way of using an optimization program would be to take all the measured data for all devices, put them into one big file, and fit to all these data with all model parameters simultaneously. Even for a very high quality, robust optimization program the chances of this method converging are slight. Even if the program does converge, it is almost certain that the values of the parameters will be very unphysical. This kind of approach is an extreme case of global optimization. We call any optimization technique that tries to fit with parameters to data outside their region of applicability a global approach. That is, if we try to fit to saturation

region data with linear region parameters such as threshold voltage, mobility, etc., we are using a global approach. In general, we advise avoiding global approaches, although in the strategies described later, sometimes the rules are bent a little.

Our recommended approach is to fit subsets of relevant parameters to corresponding subsets of relevant data in a way that makes physical sense. For example, in the MOS level 3 model, VT0 is defined as the threshold voltage of a long, wide transistor at zero back-bias. It does not make sense to use this parameter to fit to a short channel transistor, or to fit at nonzero back-bias values, or to fit to anywhere outside the linear region. In addition, subsets of parameters should be obtained in the proper order so that those obtained at a later step do not affect those obtained at earlier steps. That is, we would not obtain saturation region parameters before we have obtained linear region parameters because the values of the linear region parameters would influence the saturation region fits; we would have to go back and reoptimize on the saturation region parameters after obtaining the linear region parameters. Finally, never use optimization to obtain a parameter value when the parameter can be measured directly. For example, the MOS oxide thickness, TOX, is a model parameter, but we would never use optimization to find it. Always measure its value directly on a large oxide capacitor provided on the test chip. The recommended procedure for proper device characterization follows:

1. Have all the appropriate structures necessary on your test chip. Without this, the job cannot be performed properly.
2. Always measure whatever parameters are directly measurable. Never use otpimization for these.
3. Fit the subset of parameters to corresponding subsets of data, and do so in physically meaningful ways.
4. Fit parameters in the proper order so that those obtained later do not affect those obtained previously. If this is not possible, iteration may be necessary.

Naturally, a good strategy cannot be mounted if one is not intimately familiar with the model used. There is no substitute for learning as much about the model as possible. Without this knowledge, one must rely on strategies provided by software vendors, and these vary widely in quality.

Finally, no one can provide a completely general strategy applicable to all models and all process technologies. At some point the strategy must be tailored to suit the available technology and circuit performance requirements. This not only requires familiarity with the available device models, but also information from the circuit designers and process architects.

MOS DC Models

Available MOS Models

A number of MOS models have been provided over time with the original circuit simulation program, SPICE. In addition, some commercially available circuit simulation programs have introduced their own proprietary models, most notably HSPICE.[1] This section is concentrated on the standard MOS models provided by UC Berkeley's SPICE, not only because they have become the standard models used by all circuit simulation programs, but also because the proprietary models provided by commercial vendors are not well documented and no source code is available for these models to investigate them thoroughly.

MOS Levels 1, 2, and 3. Originally, SPICE came with three MOS models known as level 1, level 2, and level 3. The level 1 MOS model is a very crude first-order model that is rarely used. The level 2 and level 3 MOS models are extensions of the level 1 model and have been used extensively in the past and present [Vladimirescu and Liu, 1980]. These two models contain about 15 DC parameters each and are usually considered useful for digital circuit simulation down to 1 μm channel length technologies. They can fit the drain current for wide transistors of varying length with reasonable accuracy (about 5% RMS error), but have very little advanced fitting capability for analog application. They have only one parameter for fitting the subthreshold region, and no parameters for fitting the derivative of drain current with respect to drain voltage, G_{ds} (usually considered critical for analog applications). They also have no ability to vary the mobility degradation with back-bias, so the fits to I_{ds} in the saturation region at high back-bias are not very good. Finally, these models do not interpolate well over device

[1]HSPICE is a commercially available, SPICE-like circuit simulation program from Meta Software, Campbell, CA.

geometry; e.g., if a fit it made to a wide-long device and a wide-short device, and then one observes how the models track for lengths between these two extremes, they usually do not perform well. For narrow devices they can be quite poor as well. Level 3 has very little practical advantage over level 2, although the level 2 model is proclaimed to be more physically based, whereas the level 3 model is called semiempirical. If only one can be used, perhaps level 3 is slightly better because it runs somewhat faster and does not have quite such an annoying kink in the transition region from linear to saturation as does level 2.

Berkeley Short-Channel Igfet Model (BSIM). To overcome the many short-comings of level 2 and level 3, the BSIM and BSIM2 models were introduced. The most fundamental difference between these and the level 2 and 3 models is that BSIM and BSIM2 use a different approach to incorporating the geometry dependence [Ouster et al., 1988; Jeng et al., 1987]. In level 2 and 3 the geometry dependence is built directly into the model equations. In BSIM and BSIM2 each parameter (except for a very few) is written as a sum of three terms

$$\text{Parameter} \;-\; \text{Par}_0 + \frac{\text{Par}_L}{L_{\text{eff}}} + \frac{\text{Par}_W}{W_{\text{eff}}}, \tag{13.6}$$

where Par_0 is the zero-order term, Par_L accounts for the length dependence of the parameter, Par_W accounts for the width dependence, and L_{eff} and W_{eff} are the effective channel width and length, respectively. This approach has a large influence on the device characterization strategy, as discussed later. Because of this tripling of the number of parameters and for other reasons as well, the BSIM model has about 54 DC parameters and the BSIM2 model has over 100.

The original goal of the BSIM model was to fit better than the level 2 and 3 models for submicron channel lengths, over a wider range of geometries, in the subthreshold region, and for nonzero back-bias. Without question, BSIM can fit individual devices better than level 2 and level 3. It also fits the subthreshold region better and it fits better for nonzero back-biases. However, its greatest shortcoming is its inability to fit over a large geometry variation. This occurs because (13.6) is a truncated Taylor series in $1/L_{\text{eff}}$ and $1/W_{\text{eff}}$ terms, and in order to fit better over varying geometries, higher power terms in $1/L_{\text{eff}}$ and $1/W_{\text{eff}}$ are needed. In addition, no provision was put into the BSIM model for fitting G_{ds}, so its usefulness for analog applications is questionable. Many of the BSIM model parameters are unphysical, so it is very hard to understand the significance of these model parameters. This has profound implications for generating skew models (fast and slow models to represent the process corners) and for incorporating temperature dependence. Another flaw of the BSIM model is its wild behavior for certain values of the model parameters. If model parameters are not specified for level 2 or 3, they will default to values that will at least force the model to behave well. For BSIM, not specifying certain model parameters, setting them to zero, or various combinations of values can cause the model to become very ill-behaved.

BSIM2. The BSIM2 model was developed to address the shortcomings of the BSIM model. This was basically an extension of the BSIM model, removing certain parameters that had very little effect, fixing fundamental problems such as currents varying the wrong way as a function of certain parameters, adding more unphysical fitting parameters, and adding parameters to allow fitting G_{ds}. BSIM2 does fit better than BSIM, but with more than twice as many parameters as BSIM, it should. However, it does not address the crucial problem of fitting large geometry variations. Its major strengths over BSIM are fitting the subthreshold region better, and fitting G_{ds} better. Most of the other shortcomings of BSIM are also present in BSIM2, and the large number of parameters in BSIM2 makes it a real chore to use in device characterization.

BSIM3. Realizing the shortcomings of BSIM2, UC Berkeley recently introduced the BSIM3 model. This is an unfortunate choice of name because it implies BSIM3 is related to BSIM and BSIM2. In reality, BSIM3 is an entirely new model that in some sense is related more to level 2 and 3 than BSIM or BSIM2. The BSIM3 model abandons the length and width dependence approach of BSIM and BSIM2, preferring to go back to incorporating the geometry dependence directly into the model equations, as do level 2 and 3. In addition, BSIM3 is a more physically based model, with about 30 fitting parameters (the model has many more parameters, but

the majority of these can be left untouched for fitting), making it more manageable, and it has abundant parameters for fitting G_{ds}, making it a strong candidate for analog applications.

It is an evolving model, so perhaps it is unfair to criticize it at this early stage. Its greatest shortcoming is, again, the inability to fit well over a wide range of geometries. It is hoped that future modifications will address this problem. In all fairness, however, it is a large order to ask a model to be physically based, have not too many parameters, be well behaved for all default values of the parameters, fit well over temperature, fit G_{ds}, fit over a wide range of geometries, and still fit individual geometries as well as a model with over 100 parameters, such as BSIM2. Some of these features were compromised in developing BSIM3.

Proprietary Models. A number of other models are available from commercial circuit simulator vendors, the literature, etc. Some circuit simulators also offer the ability to add a researcher's own models. In general, we caution against using proprietary models, especially those which are supplied without source code and complete documentation. Without an intimate knowledge of the model equations, it is very difficult to develop a good device characterization strategy. Also, incorporating such models into device characterization software is almost impossible. To circumvent this problem, many characterization programs have the ability to call the entire circuit simulator as a subroutine in order to exercise the proprietary model subroutines. This can slow program execution by a factor of 20 or more, seriously impacting the time required to characterize a technology. Also, if proprietary models are used without source code, the circuit simulator results can never be checked against other circuit simulators. Therefore, we want to stress the importance of using standard models. If these do not meet the individual requirements, the next best approach is to incorporate a proprietary model whose source code one has access to. This requires being able to add the individual model not only to circuit simulators, but also to device characterization programs; it can become a very large task.

MOS Level 3 Extraction Strategy in Detail

The strategy discussed here is one that we consider to be a good one, in the spirit of our earlier comments. Note, however, that this is not the only possible strategy for the level 3 model. The idea here is to illustrate basic concepts so that this strategy can be refined to meet particular individual requirements.

In order to do a DC characterization, the minimum requirement is one each of the wide-long, wide-short, and narrow-long devices. We list the steps of the procedure and then discuss them in more detail.

STEP 1. Fit the wide-long device in the linear region at zero back-bias at V_{gs} values above the subthreshold region, with parameters VT0 (threshold voltage), U0 (mobility), and THETA (mobility degradation with V_{gs}).

STEP 2. Fit the wide-short device in the linear region at zero back-bias, at V_{gs} values above the subthreshold region, with parameters VT0, LD (length encroachment), and THETA. When finished with this step, replace VT0 and THETA with the values from step 1, but keep the value of LD.

STEP 3. Fit the narrow-long device in the linear region at zero back-bias, at V_{gs} values above the subthreshold region, with parameters VT0, DW (width encroachment), and THETA. When finished with this step, replace VT0 and THETA with the values from step 1, but keep the value of DW.

STEP 4. Fit the wide-short device in the linear region at zero back-bias, at V_{gs} values above the subthreshold region, with parameters RS and RD (source and drain series resistance).

STEP 5. Fit the wide-long device in the linear region at all back-biases, at V_{gs} values above the subthreshold region, with parameter NSUB (channel doping affects long channel variation of threshold voltage with back-bias).

STEP 6. Fit the wide-short device in the linear region at zero back-bias, at V_{gs} values above the subthreshold region, with parameter XJ (erroneously called the junction depth; affects short-channel variation of threshold voltage with back-bias).

STEP 7. Fit the narrow-long device in the linear region at zero back-bias, at V_{gs} values above the subthreshold region, with parameter DELTA (narrow channel correction to threshold voltage).

STEP 8. Fit the wide-short device in the saturation region at zero back-bias (or all back-biases) with parameters VMAX (velocity saturation), KAPPA (saturation region slope fitting parameter), and ETA (V_{ds} dependence of threshold voltage).

STEP 9. Fit the wide-short device in the subthreshold region at whatever back-bias and drain voltage is appropriate (usually zero back-bias and low V_{ds}) with parameter NES (subthreshold slope fitting parameter). One may need to fit with VT0 also and then VT0 is replaced after this step with the value of VT0 obtained from step 1.

This completes the DC characterization steps for the MOS level 3 model. One would then go on to do the junction and overlap capacitance terms (discussed later). Note that this model has no parameters for fitting over temperature, although temperature dependence is built into the model that the user cannot control.

In Step 1 VT0, U0, and THETA are defined in the model for a wide-long device at zero back-bias. They are zero-order fundamental parameters without any short or narrow channel corrections. We therefore fit them to a wide-long device. It is absolutely necessary that such a device be on the test chip. Without it, one cannot obtain these parameters properly. The subthreshold region must be avoided also because these parameters do not control the model behavior in subthreshold.

In Step 2 we use LD to fit the slope of the linear region curve, holding U0 fixed from step 1. We also fit with VT0 and THETA because without them the fitting will not work. However, we want only the value of LD that fits the slope, so we throw away VT0 and THETA, replacing them with the values from step 1.

Step 3 is the same as step 2, except that we are getting the width encroachment instead of the length.

In Step 1 the value of THETA that fits the high V_{gs} portion of the wide-long device linear region curve was found. Because the channel length of a long transistor is very large, the source and drain series resistances have almost no effect here, but for a short-channel device, the series resistance will also affect the high V_{gs} portion of the linear region curve. Therefore, in step 4 we fix THETA from step 1 and use RS and RD to fit the wide-short device in the linear region, high V_{gs} portion of the curve.

In Step 5 we fit with NSUB to get the variation of threshold voltage with back-bias. We will get better results if we restrict ourselves to lower values of V_{gs} (but still above subthreshold) because no mobility degradation adjustment exists with back-bias, and therefore the fit may not be very good at higher V_{gs} values for the nonzero back-bias curves.

Step 6 is just like step 5, except we are fitting the short-channel device. Some people think that the value of XJ should be the true junction depth. This is not true. The parameter XJ is loosely related to the junction depth, but XJ is really the short-channel correction to NSUB. Do not be surprised if XJ is not equal to the true junction depth.

Step 7 uses DELTA to make the narrow channel correction to the threshold voltage. This step is quite straightforward.

Step 8 is the only step that fits in the saturation region. The use of parameters VMAX and KAPPA is obvious, but one may question using ETA to fit in the saturation region. The parameter ETA adjusts the threshold voltage with respect to V_{ds}, and as such one could argue that ETA should be used to fit measurements of I_{ds} sweeping V_{gs} and stepping V_{ds} to high values. In doing so, one will corrupt the fit in the saturation region, and usually we want to fit the saturation region better at the expense of the linear region.

Step 9 uses NFS to fit the slope of the $\log(I_{ds})$ vs. V_{gs} curve. Often the value of VT0 obtained from step 1 will prevent one from obtaining a good fit in the subthreshold region. If this happens, try fitting with VT0 and NFS, but replacing the final value of VT0 with that from step 1 at the end, keeping only NFS from this final step.

The above steps illustrate the concepts of fitting relevant subsets of parameters to relevant subsets of data to obtain physical values of the parameters, as well as fitting parameters in the proper order so that those obtained in the later steps will affect those obtained in earlier steps minimally. Please refer to Figs. 13.13 and 13.14 for how the resulting fits typically appear (all graphs showing model fits are provided by the device modeling software package Aurora, from Technology Modeling Associates, Inc., Palo Alto, CA).

An experienced person may notice that we have neglected some parameters. For example, we did not use parameters KP and GAMMA. This means KP will be calculated from U0, and GAMMA will be calculated from NSUB. In a sense U0 and NSUB are more fundamental parameters than KP and GAMMA. For example, KP

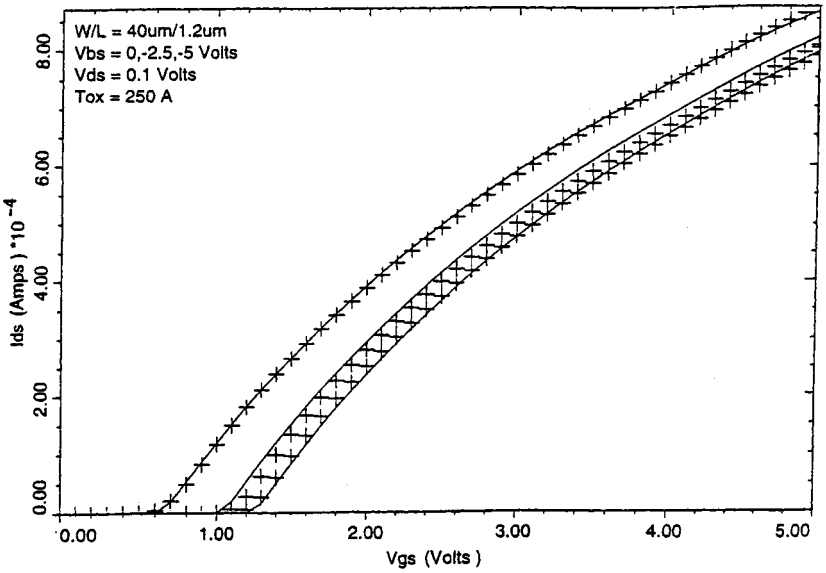

FIGURE 13.13 Typical MOS level 3 linear region measured and simulated plots at various V_{bs} values for a wide-short device.

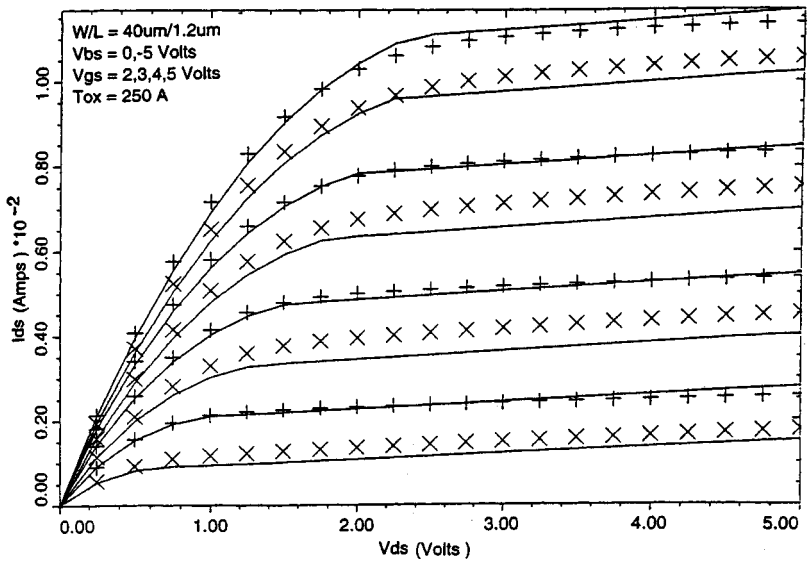

FIGURE 13.14 Typical MOS level 3 saturation region measured and simulated plots at various V_{gs} and V_{bs} values for a wide-short device.

depends on U0 and TOX; GAMMA depends on NSUB and TOX. If one is trying to obtain skew models, it is much more advantageous to analyze statistical distributions of parameters that depend on a single effect than those that depend on multiple effects. KP will depend on mobility and oxide thickness; U0 is therefore a more fundamental parameter. We also did not obtain parameter PHI, so it will be calculated from NSUB. The level 3 model is very insensitive to PHI, so using it for curve fitting is pointless. This illustrates the importance of being very familiar with the model equations. The kind of judgments described here cannot be made without such knowledge.

Test Chip Warnings. The following hints will greatly assist in properly performing device characterization.

1. Include a wide-long device; without this, the results will not be physically correct.
2. All MOS transistors with the same width should be drawn with their sources and drains identical. No difference should be seen in the number of source/drain contacts, contact spacing, source/drain contact overlap, poly gate to contact spacing, etc.
3. Draw devices in pairs. That is, if the wide-long device is W/L = 20/20, make the wide-short device the same width as the wide-long device; e.g., make the short device 20/1, not 19/1. If the narrow-long device is 2/20, make the narrow-short device of the same width; i.e., make is 2/1, not 3/1, and similarly for the lengths. (Make the wide-short and the narrow-short devices have the same length.)

BSIM Extraction Strategy in Detail

All MOS model strategies have basic features in common; namely, fit the linear region at zero back-bias to get the basic zero-order parameters, fit the linear region at nonzero back-bias, fit the saturation region at zero back-bias, fit the saturation region at nonzero back-bias, and then fit the subthreshold region. It is possible to extend the type of strategy we covered for level 3 to the BSIM model, but that is not the way BSIM was intended to be used.

The triplet sets of parameters for incorporating geometry dependence into the BSIM model, (13.6), allow an alternate strategy. We obtain sets of parameters without geometry dependence by fitting to individual devices without using the Par_L and Par_W terms. We do this for each device size individually. This produces sets of parameters relevant to each individual device. So, for device number 1 of width W(1) and length L(1) we would have a value for the parameter VFB which we will call VFB(1); for device number n of width W(n) and length L(n) we will have VFB(n). To get the Par_0, Par_L, and VFB_W we fit to the "data points" VFB(1), . . .,VFB(n) with parameters VFB_0, VFB_L, and VFB_W using (13.6) where L_{eff} and W_{eff} are different for each index, 1 through n.

Note that as L and W become very large, the parameters must approach Par_0. This suggests that we use the parameter values for the wide-long device as the Par_0 terms and only fit the other geometry sizes to get the Par_L and Par_W terms. For example, if we have obtained VFB(1) for our first device which is our wide-long device, we would set VFB_0 = VFB(1), and then fit to VFB(2), . . .,VBF(n) with parameters VFB_L and VFB_W, and similarly for all the other triplets of parameters. In order to use a general least-squares optimization program in this way the software must be capable of specifying parameters as targets, as well as measured data points.

We now list a basic strategy for the BSIM model:

STEP 1. Fit the wide-long device in the linear region at zero back-bias, at V_{gs} values above the subthreshold region, with parameters VFB (flatband voltage), MUZ (mobility), and U0 (mobility degradation), with DL (length encroachment) and DW (width encroachment) set to zero.

STEP 2. Fit the wide-short device in the linear region at zero back-bias, at V_{gs} values above the subthreshold region, with parameters VFB, U0, and DL.

STEP 3. Fit the narrow-long device in the linear region at zero back-bias, at V_{gs} values above the subthreshold region, with parameters VFB, U0, and DW.

STEP 4. Refit the wide-long device in the linear region at zero back-bias, at V_{gs} values above the subthreshold region, with parameters VFB, MUZ, and U0, now that DL and DW are known.

STEP 5. Fit the wide-short device in the linear region at zero back-bias, at V_{gs} values above the subthreshold region, with parameters VFB, RS, and RD. When finished, replace the value of VFB with the value found in step 4.

STEP 6. Fit the wide-long device in the linear region at all back-biases, at V_{gs} values above the subthreshold region, with parameters K1 (first-order body effect), K2 (second-order body effect), U0, and X2U0 (V_{bs} dependence of U0).

STEP 7. Fit the wide-long device in the saturation region at zero back-bias with parameters U0, ETA (V_{ds} dependence of threshold voltage), MUS (mobility in saturation), U1 (V_{ds} dependence of mobility), and X3MS (V_{ds} dependence of MUS).

STEP 8. Fit the wide-long device in the saturation region at all back-biases with parameter X2MS (V_{bs} dependence of MUS).

STEP 9. Fit the wide-long device in the subthreshold region at zero back-bias and low V_{ds} value with parameter N0; then fit the subthreshold region nonzero back-bias low V_{ds} data with parameter NB; and finally fit the subthreshold region data at higher V_{ds} values with parameter ND. Or, fit all the subthreshold data simultaneously with parameters N0, NB, and ND.

Repeat Steps 6 through 10 for all the other geometries, with the result of sets of geometry-independent parameters for each different size device. Then follow the procedure described previously for obtaining the geometry-dependent terms Par_0, Par_L, and Par_W.

In the above strategy we have omitted various parameters either because they have minimal effect or because they have the wrong effect and were modified in the BSIM2 model. Because of the higher complexity of the BSIM model over the level 3 model, many more strategies are possible than the one just listed. One may be able to find variations of the above strategy that suit the individual technology better. Whatever modifications are made, the general spirit of the above strategy probably will remain.

Some prefer to use a more global approach with BSIM, fitting to measured data with Par_L and Par_W terms directly. Although this is certainly possible, it is definitely not a recommended approach. It represents the worst form of blind curve fitting, with no regard for physical correctness or understanding. The BSIM model was originally developed with the idea of obtaining the model parameters via extraction as opposed to optimization. In fact, UC Berkeley provides software for obtaining BSIM parameters using extraction algorithms, with no optimization at all. As stated previously, this has the advantage of being relatively fast and easy. Unfortunately, it does not always work. One of the major drawbacks of the BSIM model is that certain values of the parameters can cause the model to produce negative values of G_{ds} in saturation. This is highly undesirable, not only from a modeling standpoint, but also because of the convergence problems it can cause in circuit simulators. If an extraction strategy is used that does not guarantee non-negative G_{ds}, very little can be done to fix the problem when G_{ds} becomes negative. Of course, the extraction algorithms can be modified, but this is difficult and time consuming. With optimization strategies, one can weight the fitting for G_{ds} more heavily and thus force the model to produce non-negative G_{ds}. We, therefore, do not favor extraction strategies for BSIM, or anything else. As with most things in life, minimal effort provides minimal rewards.

BSIM2 Extraction Strategy

We do not cover the BSIM2 strategy in complete detail because it is very similar to the BSIM strategy, except more parameters are involved. The major difference in the two models is the inclusion of extra terms in BSIM2 for fitting G_{ds} (refer to Fig. 13.15, which shows how badly BSIM typically fits $1/G_{ds}$ vs. V_{ds}). Basically, the BSIM2 strategy follows the BSIM strategy for the extraction of parameters not related to G_{ds}. Once these have been obtained, the last part of the strategy includes steps for fitting to G_{ds} with parameters that account for channel length modulation and hot electron effects. The way this proceeds in BSIM2 is to fit I_{ds} first, and then parameters MU2, MU3, and MU4 are used to fit to $1/G_{ds}$ vs. V_{ds} curves for families of V_{gs} and V_{bs}. This can be a very time consuming and frustrating experience, because fitting to $1/G_{ds}$ is quite difficult. Also, the equations describing how G_{ds} is modeled with MU2, MU3, and MU4 are very unphysical and the interplay between the parameters makes fitting awkward. The reader is referred to Fig. 13.16, which shows how BSIM2 typically fits $1/G_{ds}$ vs. V_{ds}. BSIM2 is certainly better than BSIM but it has its own problems fitting $1/G_{ds}$.

BSIM3 Comments

The BSIM3 model is very new and will undoubtedly change in the future [Huang et al., 1993]. We will not list a BSIM3 strategy here, but focus instead on the features of the model that make it appealing for analog modeling.

BSIM3 has terms for fitting G_{ds} that relate to channel length modulation, drain-induced barrier lowering, and hot electron effects. They are incorporated completely differently from the G_{ds} fitting parameters of BSIM2. In BSIM3 these parameters enter through a generalized Early voltage relation, with the drain current in saturation written as

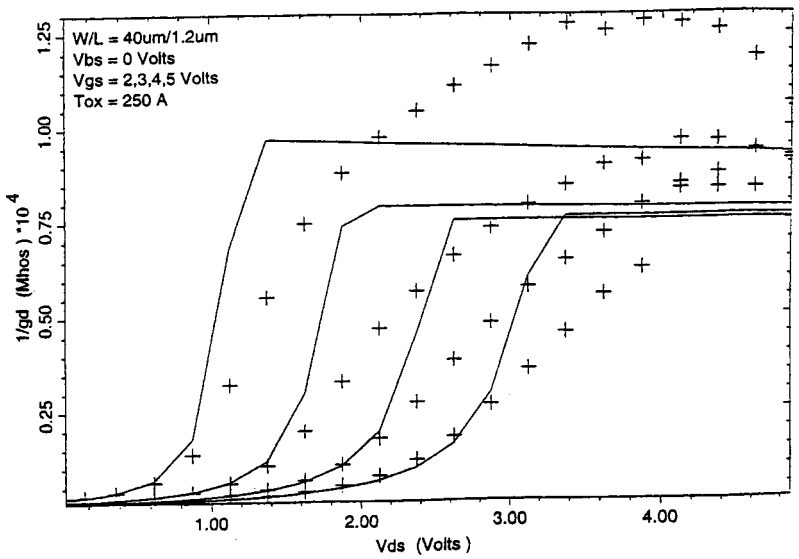

FIGURE 13.15 Typical BSIM $1/G_d$ vs. V_{ds} measured and simulated plots at various V_{gs} values for a wide-short device.

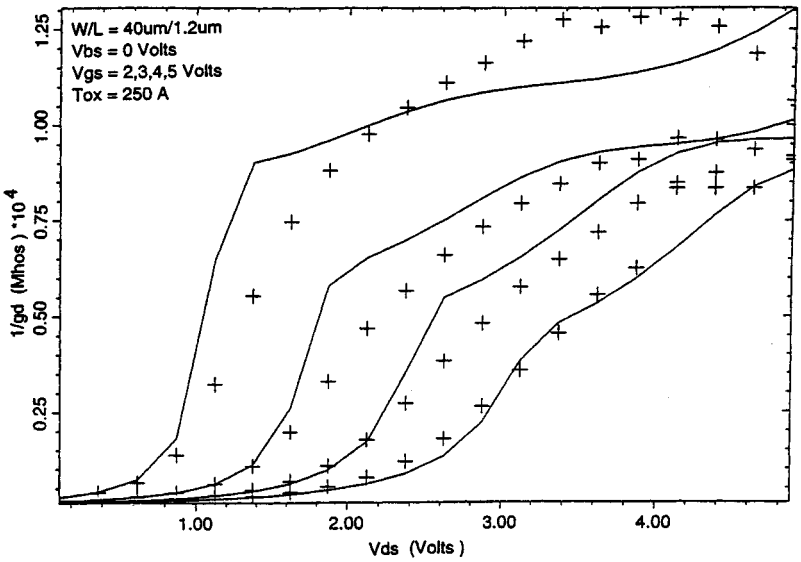

FIGURE 13.16 Typical BSIM2 $1/G_d$ vs. V_{ds} measured and simulated plots at various V_{gs} values for a wide-short device.

$$I_{ds} = I_{d\,\mathrm{sat}}\left[1 + \frac{\left(V_{ds} - V_{d\,\mathrm{sat}}\right)}{V_A}\right] \tag{13.7}$$

where V_A is a generalized Early voltage made up of three terms as

$$\frac{1}{V_A} = \frac{1}{V_{\mathrm{ACLM}}} + \frac{1}{V_{\mathrm{ADIBL}}} + \frac{1}{V_{\mathrm{AHCE}}} \tag{13.8}$$

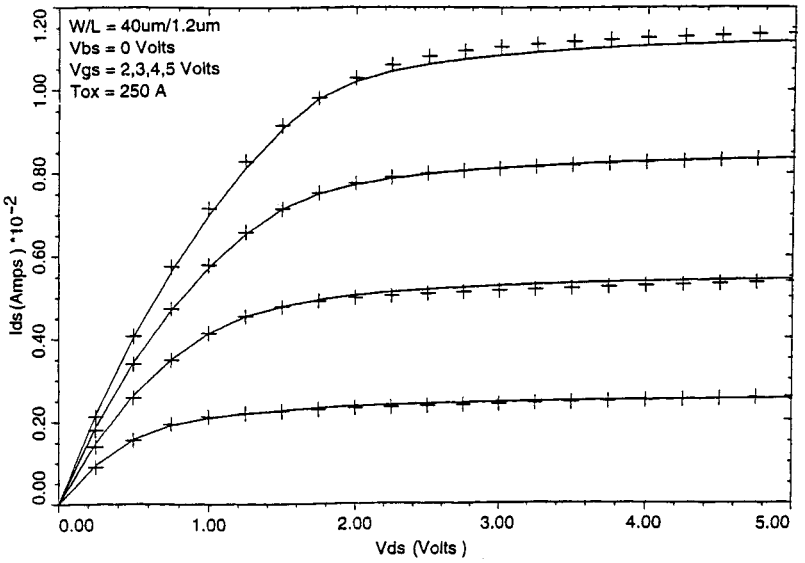

FIGURE 13.17 Typical BSIM3 saturation region measured and simulated plots at various V_{gs} values for a wide-short device.

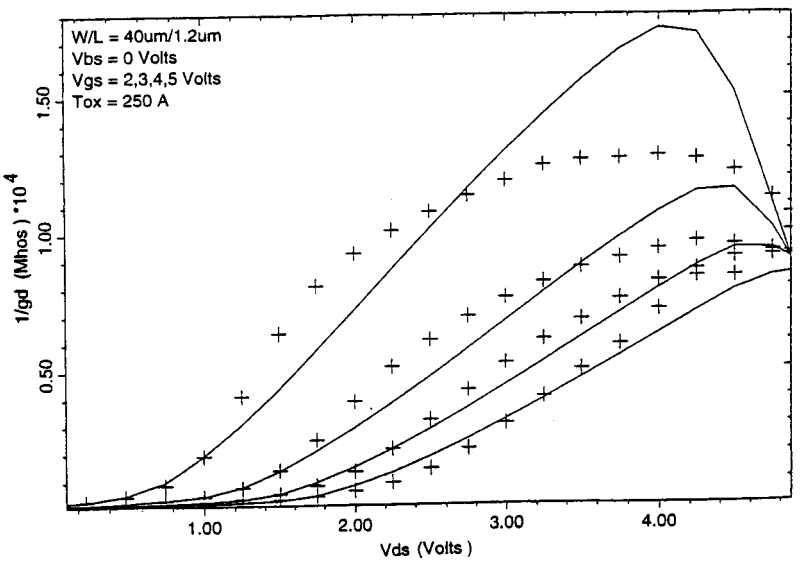

FIGURE 13.18 Typical BSIM3 $1/G_d$ vs. V_{ds} measured and simulated plots at various V_{gs} values for a wide-short device.

with the terms in (13.8) representing generalized Early voltages for channel length modulation (CLM), drain-induced barrier lowering (DIBL), and hot carrier effects (HCE). This formulation is more physically appealing than the one used in BSIM2, making it easier to fit $1/G_{ds}$ vs. V_{ds} curves with BSIM2. Figures 13.17 and 13.18 show how BSIM3 typically fits I_{ds} vs. V_{ds} and $1/G_{ds}$ vs. V_{ds}.

Most of the model parameters for BSIM3 have physical significance so they are obtained in the spirit of the parameters for the level 2 and 3 models. The incorporation of temperature dependence is also easier in BSIM3 because the parameters are more physical. All this, coupled with the fact that about 30 parameters exist for BSIM3 as compared to over 100 for BSIM2, makes BSIM3 a logical choice for analog design. However, BSIM3 is evolving, and shortcomings to the model may still exist that may be corrected in later revisions.

Which MOS Model To Use?

Many MOS models are available in circuit simulators, and the novice is bewildered as to which model is appropriate. No single answer exists, but some questions must be asked before making a choice:

1. What kind of technology am I characterizing?
2. How accurate a model do I need?
3. Do I want to understand the technology?
4. How important are the skew model files (fast and slow parameter files)?
5. How experienced am I? Do I have the expertise to handle a more complicated model?
6. How much time can I spend doing device characterization?
7. Do I need to use this model in more than one circuit simulator?
8. Is the subthreshold region important?
9. Is fitting G_{ds} important?

Let us approach each question with regard to the models available. If the technology is not submicron, perhaps a simpler model such as level 3 is capable of doing everything needed. If the technology is deep submicron, then use a more complicated model such as BSIM, BSIM2, or BSIM3. If high accuracy is required, then the best choice is BSIM3, mainly because it is more physical than all the other models and is capable of fitting better.

For a good physical understanding of the process being characterized. BSIM and BSIM2 are not good choices. These are the least physically based of all the models. The level 2 and 3 models have good physical interpretation for most of the parameters, although they are relatively simple models. BSIM3 is also more physically based, with many more parameters than level 2 or 3, so it is probably the best choice.

If meaningful skew models need to be generated, then BSIM and BSIM2 are very difficult to use, again, because of their unphysical parameter sets. Usually, the simplest physically based model is the best for skew model generation. A more complicated physically based model such as BSIM3 may also be difficult to use for skew model generation.

If the user is inexperienced, none of the BSIM models should be used until the user's expertise improves. Our advice is to practice using simpler models before tackling the harder ones.

If time is critical, the simpler models will definitely be much faster for use in characterization. The more complicated models require more measurements over wider ranges of voltages as well as wider ranges of geometries. This, coupled with the larger number of parameters, means they will take some time with which to work. The BSIM2 model will take longer than all the rest, especially if the G_{ds} fitting parameters are to be used.

The characterization results may need to be used in more than one circuit simulator. For example, if a foundry must supply models to various customers, they may be using different circuit simulators. In this case proprietary models applicable to a single circuit simulator should not be used. Also, circuit designers may want to check the circuit simulation results on more than one circuit simulator. It is better to use standard Berkeley models (level 2, level 3, BSIM, BSIM2, and BSIM3) in such cases.

If the subthreshold region is important, then level 2 or level 3 cannot be used, and probably not even BSIM; BSIM2 or BSIM3 must be used instead. These two models have enough parameters for fitting the subthreshold region.

If fitting G_{ds} is important, BSIM2 and BSIM3 are, again, the only choices. None of the other models have enough parameters for fitting G_{ds}.

Finally, if a very unusual technology is to be characterized, none of the standard models may be appropriate. In this case commercially available specialized models or the user's own models must be used. This will be a large task, so the goals must justify the effort.

Skew Parameter Files

This chapter discussed obtaining model parameters for a single wafer, usually one that has been chosen to represent a typical wafer for the technology being characterized. The parameter values obtained from this wafer correspond to a typical case. Circuit designers also want to simulate circuits with parameter values representing the extremes of process variation, the so-called fast and slow corners, or skew parameter files. These represent the best and worst case of the process variation over time.

Skew parameter values are obtained usually by tracking a few key parameters, measuring many wafers over a long period of time. The standard deviation of these key parameters is found and added to or subtracted from the typical parameter values to obtain the skew models. This method is extremely crude and will not normally produce a realistic skew model. It will almost always overestimate the process spread, because the various model parameters are not independent—they are correlated.

Obtaining realistic skew parameter values, taking into account all the subtle correlations between parameters, is more difficult. In fact, skew model generation is often more an art than a science. Many attempts have been made to utilize techniques from a branch of statistics called multivariate analysis [Dillon and Goldstein, 1984]. In this approach principal component or factor analysis is used to find parameters that are linear combinations of the original parameters. Only the first few of these new parameters will be kept; the others will be discarded because they have less significance. This new set will have fewer parameters than the original set and therefore will be more manageable in terms of finding their skews. The user sometimes must make many choices in the way the common factors are utilized, resulting in different users obtaining different results.

Unfortunately, a great deal of physical intuition is often required to use this approach effectively. To date, we have only seen it applied to the simpler MOS models such as level 3. It is not known if this is a viable approach for a much more complicated model such as BSIM2 [Power et al., 1993].

Related Topic

24.3 The Metal-Oxide Semiconductor Field-Effect Transistor (MOSFET)

References

W. R. Dillon and M. Goldstein, *Multivariate Analysis Methods and Applications,* New York: John Wiley & Sons, 1984.

P. E. Gill, W. Murray, and M. Wright, *Practical Optimization,* Orlando, Fla.: Academic Press, 1981.

J. S. Duster, J.-C. Jeng, P. K. Ko, and C. Hu, "User's Guide for BSIM2 Parameter Extraction Program and The SPICE3 with BSIM Implementation," Electronic Research Laboratory, Berkeley: University of California, 1988.

J.-H. Huang, Z. H. Liu, M.-C. Jeng, P. K. Ko, and C. Hu, "BSIM3 Manual," Berkeley: University of California, 1993.

M.-C. Jeng, P. M. Lee, M. M. Kuo, P. K. Ko, and C. Hu, "Theory, Algorithms, and User's Guide for BSIM and SCALP," Version 2.0, Electronic Research Laboratory, Berkeley: University of California, 1987.

J. A. Power, A. Mathewson, and W. A. Lane, "An Approach for Relating Model Parameter Variabilities to Process Fluctuations," *Proc. IEEE Int. Conf. Microelectronic Test Struct.,* vol. 6, Mar. 1993.

W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C,* Cambridge, U.K.: Cambridge University Press, 1988.

B. J. Sheu, D. L. Scharfetter, P. K. Ko, and M.-C. Jeng, "BSIM: Berkeley Short-Channel IGFET Model for MOS Transistors," *IEEE J. Solid-State Circuits,* vol. SC-22, no. 4, Aug. 1987.

A. Vladimirescu and S. Liu, "The Simulation of MOS Integrated Circuits Using SPICE2," memorandum no. UCB/ERL M80/7, Berkeley: University of California, 1980.

Further Information

Other recommended publications which are useful in device characterization are

L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," memorandum no. ERL-M520, Berkeley: University of California, 1975.

G. Massobrio and P. Antognetti, *Semiconductor Device Modeling with SPICE,* New York: McGraw-Hill, 1993.