Circuit Simulation and Modeling

This column was introduced in the January issue of this magazine, and since then we have received excellent responses and contributions. We believe this column will serve as a very useful forum for people to exchange information; however, in order to have the column appear regularly, we still need your strong support to communicate and participate actively. Please contact me:

Dr. Ping Yang
Semiconductor Process and Design Center
Texas Instruments, Inc.
MS 369, P.O. Box 655621
12840 Hillcrest Plaza
Dallas, TX 75265
(214) 995-7901

As mentioned in the January issue, this column includes the readers' comments and questions, as well as any other short material on simulation and modeling. In this issue, we are presenting the paper "SPICE2 Application Notes for Dependent Sources" by Bert Epler of Texas Instruments, Inc., who has been working on circuit simulation programs for more than 20 years. In this paper, he presents a collection of notes that show how dependent sources can be used effectively to model the transfer function of analog and digital circuitry. Both the flexibility and capability of SPICE can be greatly enhanced by utilizing those dependent sources effectively; moreover, a significant speedup in simulation time can also be achieved.

Ping Yang
Editor

SPICE2 Application Notes for Dependent Sources by Bert Epler

Abstract

This paper presents an assortment of SPICE2 application notes for dependent sources. These notes have been collected over the last 14 years and show how to better utilize the dependent sources. This paper explains how dependent sources can be used to model the transfer function of analog and digital circuitry. With this capability, full circuit simulation, such as phase-lock loops, A/D converters, and many other functions, can be performed. Speedups of 2× for simple inverters and 40× for two-input nand gates are possible by replacing these circuits with a single dependent source.

Introduction

SPICE stands for "Simulation Program for Integrated Circuit Emphasis" [1], [2]. It is a circuit analysis program that originated at the University of California, Berkeley (UCB).

The idea of macromodeling or black-box modeling has been around for a long time. Chua and Lin [3] describe in great detail how to black-box-model junction diodes and bipolar transistors. Boyle et al. [4] explained macromodeling of integrated circuit operational amplifiers. Early versions of SPICE had dependent sources, but only with a few different types of transfer functions. With the extensions made to the dependent sources in TISPICE [5], it will be shown how a wide variety of transfer functions can be used.

First, the four different types of SPICE-dependent sources are reviewed (see Fig. 1). These elements define a transfer function relating the input and output. The type of functions used to define this transfer function can be a linear, polynomial, look-up table, or user-defined FORTRAN subroutine (see Fig. 2). All of these types of dependent sources are currently available in the production version of TISPICE. The look-up table and user-defined FORTRAN subroutine are extensions made to UCB SPICE.

Next, examples are shown on how to use the dependent sources with each of these types of functions. The transfer function for both analog and digital circuitry may be modeled using this technique. This allows circuit simulation and logic simulation. The user can define the level of description for each section of the circuitry. A simple example might be to perform basic mathematical operations such as addition, subtraction, multiplication, division, differentiation, or integration.

Finally, more advanced analog applications are discussed by showing how to model things such as an operational amplifier with limiting action and slew rate, and lossy transmission lines. Digital applications cover the basic logic gates. These include inverters, and, or, nand, xor, and xnor gates. These gates may contain zero-delay or the propagation-delay time for low-to-high and high-to-low transitions. By using these as building blocks, a complete family of digital elements may be built.

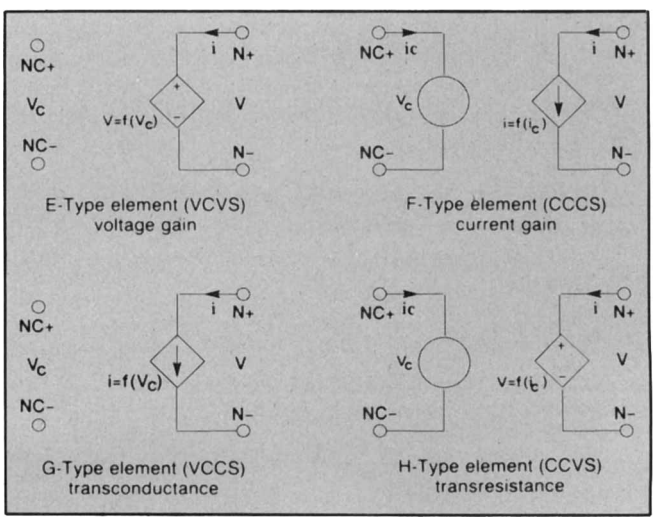

Fig. 1 SPICE-dependent sources.

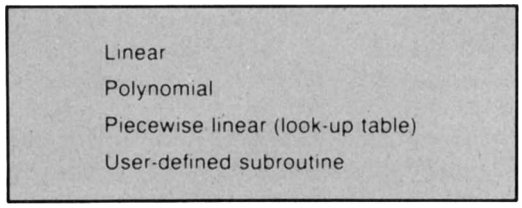

Fig. 2 Function types for dependent sources.

Dependent Sources

There are four different types of dependent sources available in SPICE2. These dependent sources are expressed by linear, polynomial, table look-up, and user-defined FORTRAN functions for dc, frequency, and transient simulations. The linear function is a subset of the polynomial function. The table look-up function is sometimes referred to as piecewise linear (PWL). Applications enhancing the modeling of circuit characteristics and extending the output capability will now be explained.

All possible voltage and current combinations of dependent sources are available in SPICE2 and are characterized by the following four equations:

$$I=F(V) \qquad V=F(V) \qquad I=F(I) \qquad V=F(I)$$

representing transconductance, voltage gain, current gain, and transresistance.

These G-, E-, F- and H-type elements correspond to voltage-controlled current source (VCCS), voltage-controlled voltage source (VCVS), current-controlled current source (CCCS), and current-controlled voltage source (CCVS), respectively. The current or voltage source is the dependent variable, while the voltage controlled or current controlled is the independent variable. The number of different independent variables sets the dimension and is arbitrary.

The controlling voltage may be the differential voltage between any two node voltages, whereas the controlling currents must be the current through an independent voltage source. Each controlling variable must be of the same type—either all voltages or all currents. If it is a current, it must be a current through an independent voltage source. If an independent source has not been used, then insert a zero-value independent voltage source at this point in the circuit. To convert this current to a voltage use a CCVS source.

The relationships between the independent and dependent variables are functions expressed by polynomials, piecewise linear tables, or user-defined FORTRAN subroutines. The polynomial function is currently available in Berkeley's version of SPICE2. The PWL and user-defined FORTRAN functions have been implemented in Texas Instrument's version of SPICE2.

Polynomial Functions

The polynomial functions are specified by a set of coefficients "P0, P1, . . . , Pn." The number of dimensions and the number of coefficients are arbitrary. The meaning of the coefficients depends upon the dimension of the polynomial, as shown in the examples described subsequently.

If the function is one-dimensional, that is, a function of one argument, the function value "FV" is determined by the following expression in "FA," the function argument:

```
F(V) = P0 + (P1*FA) + (P2*FA**2) + (P3*FA**3) + (P4*FA*4) + ...
```

If the polynomial is one-dimensional and exactly one coefficient is specified, SPICE2 assumes it to be "P1" (and P0= 0.0) in order to facilitate the input of linear controlled sources.

If the function is two-dimensional, with arguments FA and FB, the function value FV is determined by the following expression:

```
F(V) = P0 + (P1*FA)       + (P2*FB)       + (P3*FA**2)     + (P4*FA*FB)
          + (P5*FB**2)    + (P6*FA**3)    + (P7*FA**2*FB)
          + (P8*FA*FB**2) + (P9*FB**3)    + ...
```

If the function is three-dimensional, with arguments FA, FB, and FC, the function value FV is determined by the following expression:

```
F(V) = P0 + (P1*FA)       + (P2*FB)       + (P3*FC)       + (P4*FA**2)
          + (P5*FA*FB)    + (P6*FA*FC)    + (P7*FB**2)    + (P8*FB*FC)
          + (P9*FC**2)    + (P10*FA**3)   + (P11*FA**2*FB)
          + (P12*FA**2*FC) + (P13*FA*FB**2)
          + (P14*FA*FB*FC) + (P15*FA*FC**2)
          + (P16*FB**3)
          + ...
```

The general format for dependent sources is as follows:

```
GXXXXXX N+ N- <POLY(ND)>  NC1+ NC1- ... P0 <P1 ...>     {for VCCS}
GXXXXXX N+ N-             NC1+ NC1-      P1             {1 dimension}

EXXXXXX N+ N- <POLY(ND)>  NC1+ NC1- ... P0 <P1 ...>     {for VCVS}
EXXXXXX N+ N-             NC1+ NC1-      P1             {1 dimension}

FXXXXXX N+ N- <POLY(ND)>  VC1  VC2  ... P0 <P1 ...>     {for CCCS}
FXXXXXX N+ N-             VC1               P1          {1 dimension}

HXXXXXX N+ N- <POLY(ND)>  VC1  VC2  ... P0 <P1 ...>     {for CCVS}
HXXXXXX N+ N-             VC1               P1          {1 dimension}
```

Note that "POLY(ND)" has to be specified only if the dependent source is multidimensioned (one dimension is the default).

PWL Functions

The piecewise-linear function has a format similar to the polynomial previously described, except "POLY" is replaced by "PWL" and the polynomial coefficients are replaced with a model name and a scale factor. One-, two-, and three-dimensional PWLs are allowed. PWLs have the general form:

```
GXXXXXX N+ N- PWL(ND) NC1+ NC1- ... MNAME SCALE
                                                {for VCCS}

EXXXXXX N+ N- PWL(ND) NC1+ NC1- ... MNAME SCALE
                                                {for VCVS}

FXXXXXX N+ N- PWL(ND) VC1   VC2   ... MNAME SCALE
                                                {for CCCS}

HXXXXXX N+ N- PWL(ND) VC1   VC2   ... MNAME SCALE
                                                {for CCVS}
```

The model name is defined on a PWL ".MODEL" card with the following form:

```
                                    NO
.MODEL MNAME PWL(ND) SOURCE CARDS SYMMETRY <ODD  =INDEPENDENT#>
                                    EVEN=INDEPENDENT#

DATA DEPENDENT INDEPENDENT1 <INDEPENDENT2 INDEPENDENT3> ...
+ <   OR   >
INCR1 START STOP INCREMENT ...
INCR2 START STOP INCREMENT ...
INCR3 START STOP INCREMENT ...
DATA DEPENDENT ...
```

"MNAME" is the model name. "ND" is the number of dimensions and has a maximum value of three.

The source of dependent and independent values may be on cards or a data set. "SOURCE CARDS" indicates that the data appear on the model card. Only the card input format will be discussed here.

Symmetry around the origin may be odd or even. The independent number indicates which independent variable controls this symmetry.

The data consists of a series of dependent and independent data points. It has two different input forms. The first form starts with the keyword "DATA" followed by a string of data points. The dependent value is specified first, followed by its corresponding independent values. The user repeats this procedure until all data points are described. The data points may be in any order. The product of the number of different independent values must equal the number of dependent values. This requirement provides the proper number of table entries. Extrapolation is used outside the table. Zero-valued entries are recommended, instead of extrapolation, to provide more accurate data. The second form uses an increment format for each independent variable. The input for the first independent variable begins with the keyword "INCR1" followed by its start, stop, and increment values. The start value begins with the most negative value. If the increment value changes, then repeat the start, stop, and increment sequence until the last stop value equals the most positive value for the first independent variable. Repeat this input format for the second independent variable by using the keyword "INCR2" and for the third independent variable by using the keyword "INCR3." Next, the corresponding dependent values are specified following the keyword "DATA." These are ordered with the last independent variable changing the slowest.

User-Defined FORTRAN Subroutines

The user-defined function has a format similar to the polynomial, except "POLY" is replaced by "USER" and the polynomial coefficients are now coefficients passed to the FORTRAN subroutine. As many coefficients as necessary may be specified. UNAME is the name of the user-defined subroutine. User-defined sources have the general form:

```
GXXXXXX N+ N- USER(ND) NC1+ NC1- ... UNAME(P0 <P1 ...>)
                                                {for VCCS}

EXXXXXX N+ N- USER(ND) NC1+ NC1- ... UNAME(P0 <P1 ...>)
                                                {for VCVS}

FXXXXXX N+ N- USER(ND) VC1   VC2   ... UNAME(P0 <P1 ...>)
                                                {for CCCS}

HXXXXXX N+ N- USER(ND) VC1   VC2   ... UNAME(P0 <P1 ...>)
                                                {for CCVS}
```

The user-defined FORTRAN source has the following general form:

```
      SUBROUTINE UNAME(ICOM,VALUE1,VALUE2,COEF,V)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ICOM(1),VALUE2(1),COEF(1),V(1)
C...  Compute VALUE1 as a function of ICOM, COEF, and V
C...  Compute VALUE2 as a function of ICOM, COEF, and V
      RETURN
      END
```

The value of the function VALUE1 and its derivative, VALUE2, must be evaluated with respect to each independent variable. Inputs consist of the coefficients array COEF, the controlling voltage or current array V, and the SPICE2 label common array ICOM. ICOM(1) is the first word in the label common of the SPICE2 program. By knowing the correct offset into the ICOM array, any SPICE2 variable may be referenced. This structure has been set up so that the user-defined subroutines do not have to be changed when new SPICE2 versions are updated into production.

The procedure to use a user-defined subroutine in SPICE2 consists of a two-step job. The first step compiles the source and the second step links this object "on the fly" to SPICE2. These subroutines can be used in the dc, frequency, and transient analyses.

Analog and Digital Applications Using Dependent Sources

Dependent sources are used in two basic ways. One method simply models circuit characteristics by replacing the circuit description with an equivalent dependent source. This has the advantage of reducing the element count considerably and can be used as the first approximation. Another way uses dependent sources that are decoupled and isolated from the original circuit description. The purpose is to evaluate a function and then output that function just as any other output.

To ensure that the controlling nodes have a dc path to ground, additional resistances between the controlling nodes and ground may be required.

Signal names have been implemented into TISPICE and will be used in some examples instead of node numbers for easier signal tracing.

If the same circuit configuration will be used in several different places, it may be more efficient to define it in a subcircuit. Once this has been done, it can be called as many times as necessary. In TISPICE, the subcircuit values can be assigned names. These names can be defined in the subcircuit definition with a corresponding default value. As these subcircuits are called, the subcircuit values can be changed. In some examples, subcircuits will be used with values assigned by names.

Polynomial Applications

Let us begin with a few simple applications using the one-dimensional polynomial or linear functions and then progress to the more complex two- and three-dimensional polynomial functions.

A positive conductance (see Fig. 3) may be modeled with a VCCS having the controlling voltage across itself.

```
G+   N+   N−   N+   N−   1MILLIMHO
```

SPICE does not allow negative conductance values to be

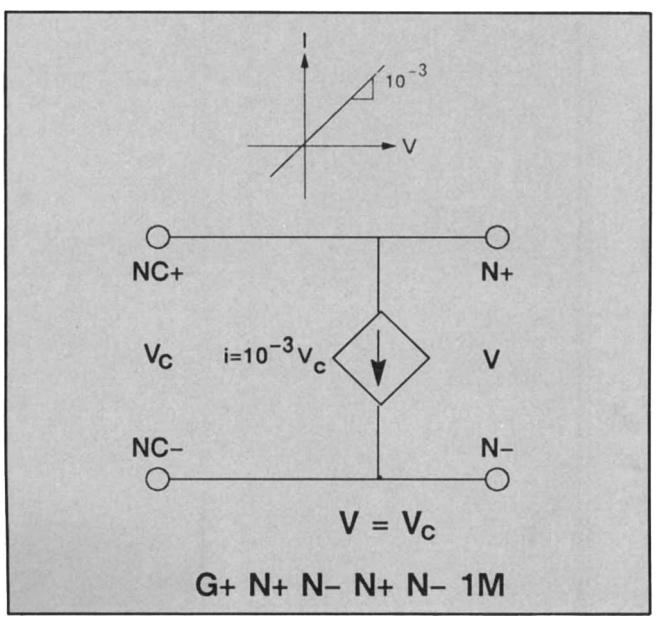

Fig. 3 Positive conductance.

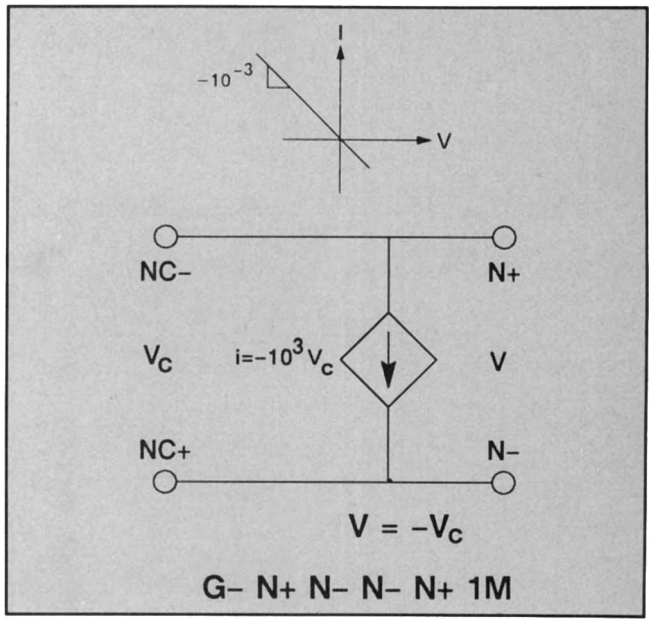

Fig. 4 Negative conductance.

input; a way around this is to simply reverse the controlling nodes (see Fig. 4).

```
G−   N+   N−   N−   N+   1MILLIMHO
```

A linear operational amplifier, with a voltage gain of 1000, uses a VCVS (see Fig. 5).

```
EOP   OUT+   OUT−   IN+   IN−   1000
```

Radiation exposure effects are simulated by using a single source to model the radiation level, gamma dot. This may be coupled to each individual bipolar transistor by using a linear VCCS across each base-collector junction. Now, by varying this one source, in a dc transfer curve analysis or in a transient analysis, the radiation is coupled to each transistor (see Fig. 6).

```
VLEVEL GAMMA 0 DC 1V
RLEVEL GAMMA 0 1
* BC JUNCTION OF Q1
G1 Q1_collector Q1_base GAMMA 0 1E−7
* BC JUNCTION OF Q2
G2 Q2_collector Q2_base GAMMA 0 1E−7
```

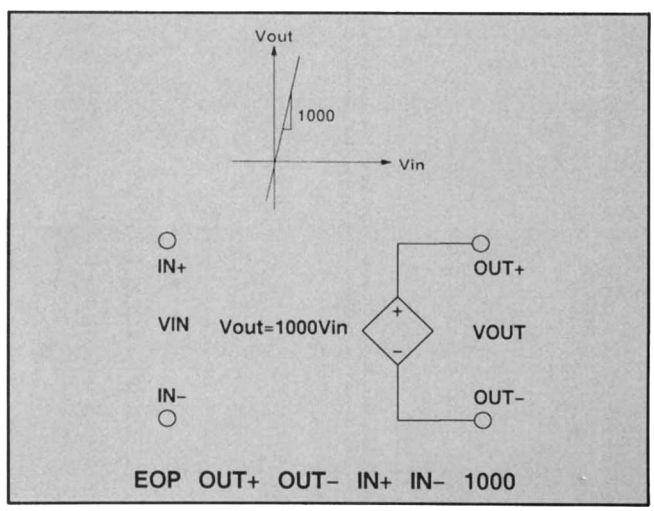

Fig. 5 Operational amplifier.

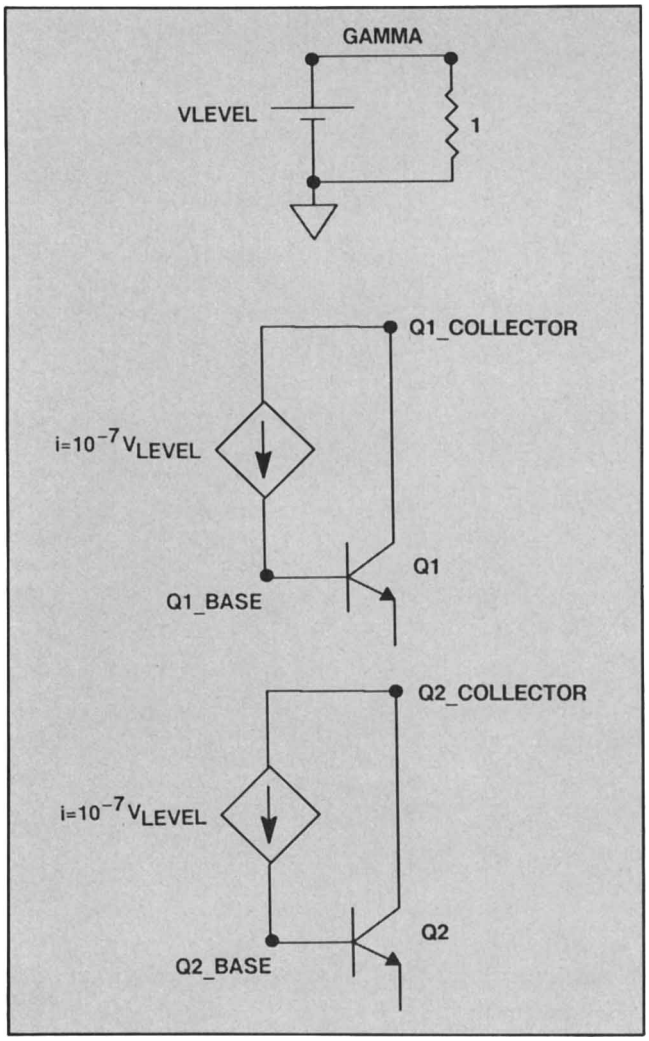

Fig. 6 Radiation coupling.

Basic mathematical operations such as addition, subtraction, multiplication, and division can be calculated by selecting the proper controlling voltage or current, and the polynomial coefficients.

To add two node voltages, use a two-dimensional polynomial VCVS and set the coefficients to P0=0, P1=1, P2=1. The following example adds the voltages at nodes 1 and 2.

```
EADD 3 0 POLY(2) 1 0 2 0      0 1 1
RADD 3 0 1
```

Any number of node voltages can be added in this manner by setting the dimension, controlling nodes, and coefficients properly (see Fig. 7).

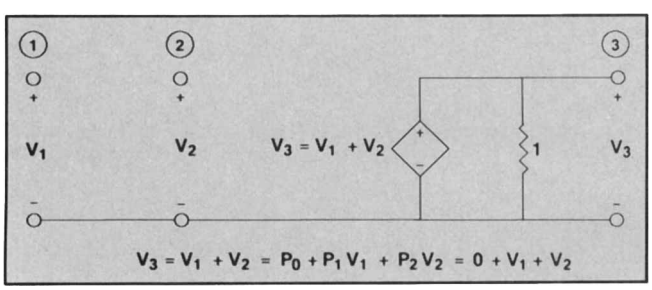

Fig. 7 Adding two voltages.

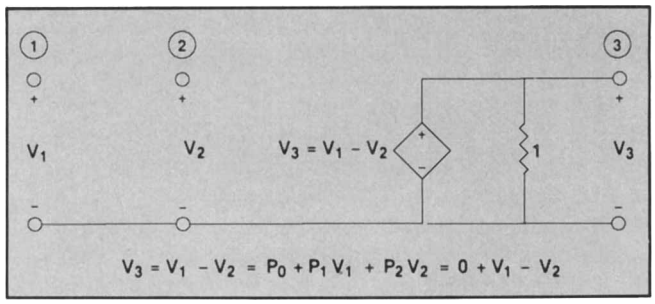

Fig. 8 Subtracting two voltages.

To subtract two node voltages, use a two-dimensional polynomial and set the coefficients to P0=0, P1=1, P2=−1 (see Fig. 8). The following example subtracts the voltages at nodes 1 and 2.

```
ESUB 3 0 POLY(2) 1 0 2 0      0 1 −1
RSUB 3 0 1
```

To multiply two node voltages, use a two-dimensional polynomial and set P0=0, P1=0, P2=0, P3=0, and P4=1 (see Fig. 9). The following example multiplies the voltage at nodes 1 and 2.

```
EMUL 3 0 POLY(2) 1 0 2 0      0 0 0 0 1
RMUL 3 0 1
```

A unique circuit, using two polynomial control sources in parallel, permits the division of two voltages or two currents [6] (see Fig. 10). Let us assume that we would like to divide two voltages at nodes 1 and 2, and place the answer at node 3. To VCCSs are connected in parallel, from node 3 to ground. The first VCCS is linear with node voltage 1.

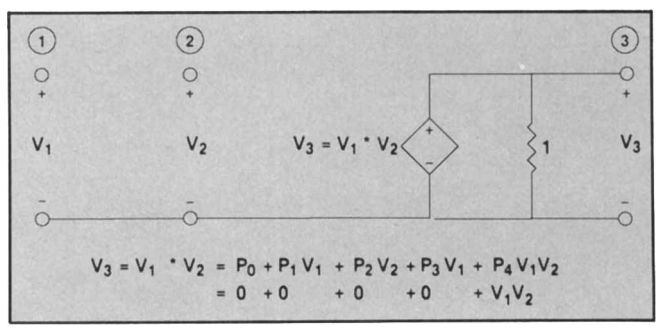

Fig. 9 Multiplying two voltages.

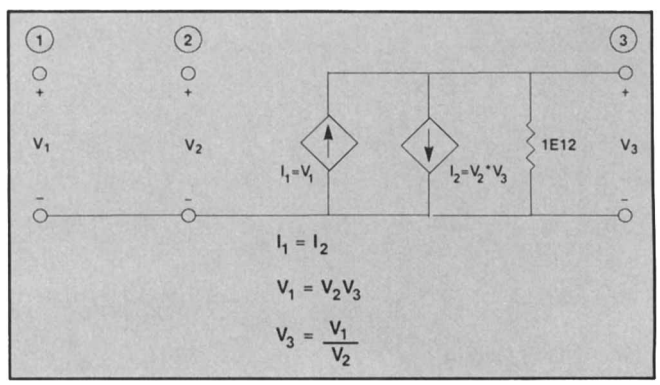

Fig. 10 Dividing two voltages.

The second VCCS is a two-dimensional polynomial with the first controlling voltage being the voltage at node 2. The second controlling voltage is the voltage at node 3. The current of the second VCCS is the product of the two controlling voltages. To do this, simply set P0=P1=P2=P3=0 and P4=1. The first VCCS has an equation I1=V1. The second VCCS has an equation I2=V3*V2. Note that nodes 3 and 0 for the current sources of G1 and G2 have been reversed, making I1=I2. The common voltage at node 3 is V3=V1/V2.

```
*DIVIDE V1/V2
G1 0 3 1 0 1
G2 3 0 POLY(2) 2 0 3 0      0 0 0 0 1
R  3 0 1E12
```

To differentiate (Fig. 11) or integrate (Fig. 12) a voltage, first couple this voltage to an isolated node that will not load the original circuit. Do this with a linear control source. Then let this linear control source drive a simple RC circuit to either differentiate or integrate the voltage. The following examples differentiate and integrate the voltage at node 1, with the resultant voltage at node 3. Note that the VCVS that couples the voltage to be differentiated has a voltage gain equal to the reciprocal of the RC time constant. The VCVS that couples the voltage to be integrated has a voltage gain equal to the RC time constant.

```
*DIFFERENTIATE WITH RC TIME CONSTANT OF 1PS
EDIFF 2 0 1 0 1E12
C       2 3 1PF
R       3 0 1OHM

*INTEGRATE WITH RC TIME CONSTANT OF 1 SECOND
EINT 2 0 1 0 1
R      2 3 1OHM
C      3 0 1ONEFARAD
```

Now, after knowing these basic ways to formulate and evaluate expressions, let us compute the equivalent capac-

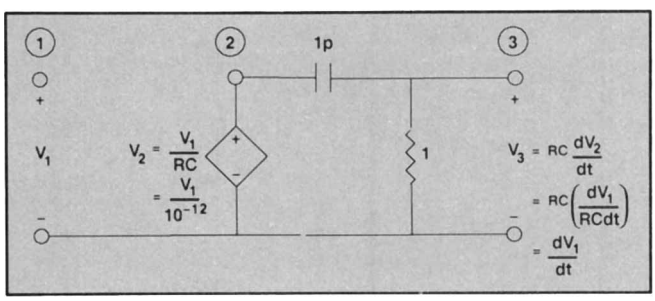

Fig. 11 Differentiating a voltage.

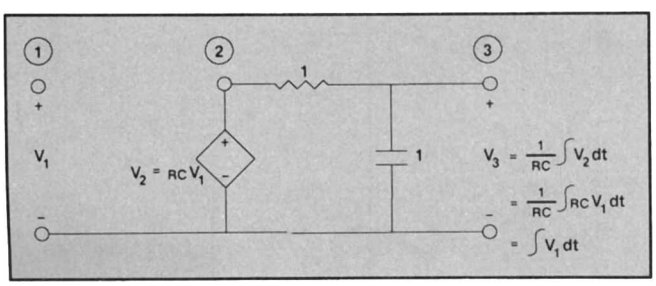

Fig. 12 Integrating a voltage.

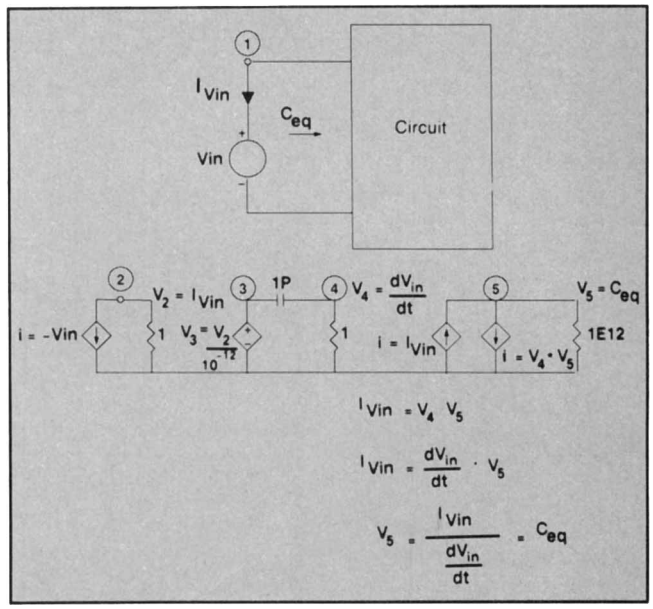

Fig. 13 Equivalent input capacitance.

itance at an input voltage (see Fig. 13). The equation is $C = I/(dv/dt)$, where C is the equivalent capacitance, I the current in the input voltage supply, and dv/dt the derivative of the input voltage with respect to time. First, convert the current in the input voltage supply to a voltage with a CCVS. The most common way of describing positive voltage supplies is to set the negative node (N−) to ground and specify a positive voltage value. Note that the polarity of the current through the independent supply is negative for this procedure. Next, couple this input voltage to a differentiating RC network by using a VCVS. Then divide these two voltages to form the effective capacitance.

```
* INPUT
VIN   1 0 PULSE(0V 10V 10NS 10NS 10NS 100NS 1000NS)
* FORM INPUT CURRENT REPRESENTED BY A VOLTAGE
HIN   2 0 VIN −1
RIN   2 0 1
* COUPLE INPUT VOLTAGE
EIN   3 0 1 0 1E12
* DIFFERENTIATE WITH RC TIME CONSTANT OF 1PS
C     3 4 1PF
R     4 0 1OHM
* DIVIDE I BY DV/DT
G1    0 5 2 0 1
G2    5 0 POLY(2) 4 0 5 0      0 0 0 0 1
RDIV 5 0 1E12
* PLOT ANSWER
.PLOT TR V(5)
```

The next example shows how to integrate the total instantaneous circuit power (see Fig. 14). First, convert the power supply current to a voltage using a linear CCVS with a −1 value to correct the polarity. Then multiply this by the supply voltage using the two-dimensional polynomial technique. Finally, integrate with the auxiliary RC network to form the final answer.

```
* INPUT
VDD 1 0 DC 5
* FORM INPUT CURRENT REPRESENTED BY A VOLTAGE
HIN   2 0 VDD −1
RIN   2 0 1
* MULTIPLY I*V
EMU 3 0 POLY(2) 2 0 1 0      0 0 0 0 1
```

```
* INTEGRATE WITH RC TIME CONSTANT OF 1 SECOND
R     3 4 1OHM
C     4 0 1ONEFARAD
* PLOT ANSWER
.PLOT TR V(4)
```

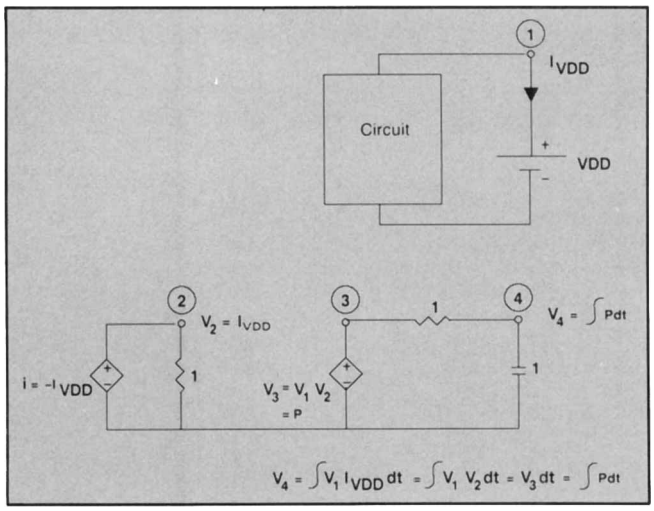

Fig. 14 Integrating the instantaneous power.

PWL Applications

Let us begin with a few simple applications using the one-dimensional PWL functions, and then progress to the more complex two- and three-dimensional PWL functions.

A simple application for a one-dimensional PWL is to generate a triangular waveform (see Fig. 15). While this could easily be created with the independent PWL source for transient analysis, the dc transfer curve analysis assumes that the input is linear. A Schmitt trigger circuit, for example, is a case in which one would like to output the full characteristic curves on a single plot.

```
* INPUT
VIN   IN 0
RIN   IN 0 1
* TRIANGULAR WAVEFORM USING PWL TABLE
ETRI IN2 0 PWL(1) IN 0 TRIANGULAR
* TRIANGULAR DATA
.MODEL TRIANGULAR PWL(1) SYMMETRY NO SOURCE CARDS DATA
      0,0 2.5,5 5,0
* DC TRANSFER CURVE
.DC VIN 0V 5V 1V
* PLOT FULL OUTPUT CURVES AS INPUT GOES UP AND THEN DOWN
.PLOT DC V(OUT)
```

Another example shows how the limiting action of an operational amplifier can be modeled by a one-dimensional PWL table (see Fig. 16). This also improves dc convergence because of the lower gain outside the active region.

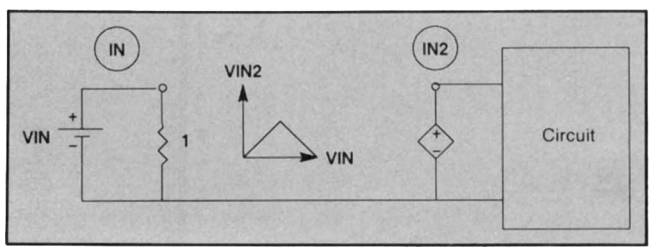

Fig. 15 Triangular input.

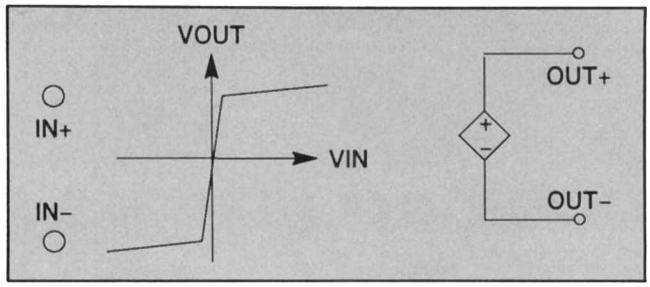

Fig. 16 Operational amplifier with limiting action.

```
* OPERATIONAL AMPLIFIER WITH LIMITING ACTION
ELIMIT OUT+ OUT− PWL(1) IN+ IN− OPAMP
* LIMITING ACTION IN PWL MODEL
.MODEL OPAMP PWL(1) SYMMETRY ODD 1 SOURCE CARDS DATA
+ 0,0   1000,1   1100,10
```

Now let us add the slew rate and frequency response to the operational amplifier (see Fig. 17). The same example that is used in [3] is listed below. This has a dc open-loop gain of 835,000, input resistance of 2 MΩ, output resistance of 75 Ω, peak output voltage of 15 V, slew rate of 0.5 V/μsec, and a dominant-pole frequency of 8.8 rad/sec.

```
*   CALL SUBCIRCUIT DEFINING OPERATIONAL AMPLIFIER
X   IN1   IN2   OUT   OPAMP
*   DEFINE OPAMP SUBCIRCUIT
.SUBCKT OPAMP IN1 IN2 OUT
X1   IN1     IN2     OUT1   OPIN
X2   OUT1   OUT2            OPRC
X3   OUT2   OUT             OPOUT
.END
*   DEFINE OPIN SUBCIRCUIT
.SUBCKT OPIN IN1 IN2 OUT
RIN   IN1    IN2   2MEG
G     OUT   0     PWL(1)   IN1   IN2   OPIN
C     OUT   0     .136UF
R     OUT   0     835K
.ENDS
.MODEL OPIN PWL(1) SYMMETRY ODD 1 SOURCE CARDS DATA
+0 0   68MA 68MV 68MA 100V
```

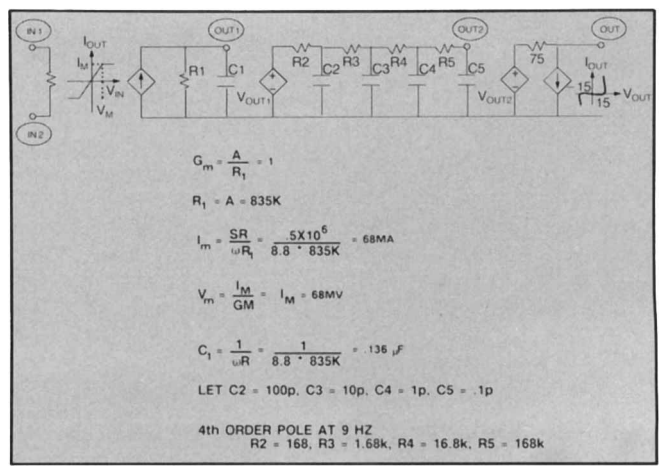

Fig. 17 Operational amplifier with slew rate and frequency response.

```
*  4-SECTION RC LADDER WITH POLE AT 9 MEGAHERTZ
.SUBCKT OPRC IN1 OUT
E    OUT1  0  IN1  0   1
R1   OUT1  OUT2        168
R2   OUT2  OUT3       1.68K
R3   OUT3  OUT4       16.8K
R4   OUT4  OUT        168K
C1   OUT2  0          100P
C2   OUT3  0           10P
C3   OUT4  0            1P
C4   OUT   0           .1P
R    OUT   0          1E12
.ENDS
*  LIMIT OUTPUT VOLTAGE TO 15V
.SUBCKT OPOUT IN OUT
E  OUT1 0 IN 0 1
ROUT OUT1 OUT 75OHMS
G      OUT 0 PWL(1) OUT 0 OPOUT
.ENDS
.MODEL OPOUT PWL(1) SYMMETRY ODD 1 SOURCE CARDS
+ INCR1 0V 15V 15V     15V 16V 1V
+ DATA  0A 0A            1A
```

A simple inverter can be modeled by a one-dimensional PWL table (see Fig. 18). This table will define the high- and low-voltage levels, the transition gain, and the input threshold voltage level. A simple zero-delay inverter follows:

```
X1 IN OUT INVPWL
.SUBCKT INVPWL IN OUT
E OUT 0 PWL(1) IN 0 INVPWL
RIN IN 0 1E12
ROUT OUT 0 1E12
.ENDS
.MODEL INVPWL PWL(1) SYMMETRY NO SOURCE CARDS DATA
+ 5V     0V
+ 5V    2.4V
+ 0V    2.6V
+ 0V     5V
```

A simple two-input nand gate can be modeled by a two-dimensional PWL table (see Fig. 19). This table will define the truth table for the nand gate. Obviously, all of the other standard two-input logic gates, such as and, or, nor, xor, and xnor, could be modeled in a similar way simply by changing the truth table.

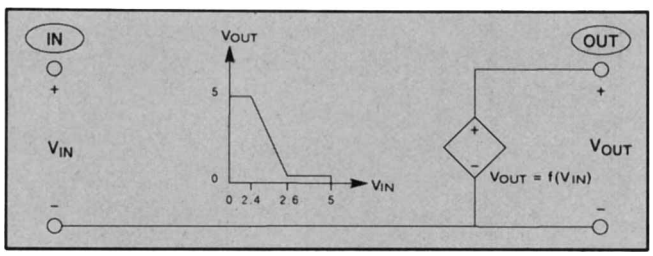

Fig. 18 PWL Inverter.

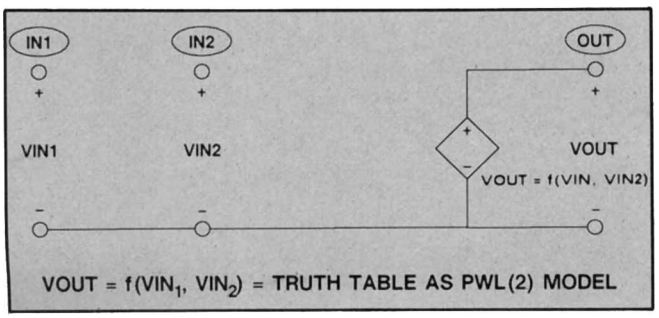

Fig. 19 Two-input PWL NAND.

```
X1 IN1 IN2 OUT NAND2PWL
.SUBCKT NAND2PWL IN1 IN2 OUT
E OUT 0 PWL(2) IN1 0 IN2 0 NAND2PWL
RIN1    IN1    0   1E12
RIN2    IN2    0   1E12
ROUT    OUT    0   1E12
.ENDS
.MODEL NAND2PWL PWL(2) SYMMETRY NO SOURCE CARDS DATA
+ 5V     0V     0V
+ 5V     0V   2.4V
+ 5V     0V   2.6V
+ 5V     0V     5V
+ 5V   2.4V     0V
+ 5V   2.4V   2.4V
+ 5V   2.4V   2.6V
+ 5V   2.4V     5V
+ 5V   2.6V     0V
+ 5V   2.6V   2.4V
+ 0V   2.6V   2.6V
+ 0V   2.6V     5V
+ 5V     5V     0V
+ 5V     5V   2.4V
+ 0V     5V   2.6V
+ 0V     5V     5V
```

Switches are very useful and can be used in a wide scope of applications (see Fig. 20). To model a switch, use a VCCS with one controlling voltage sensing the voltage across the

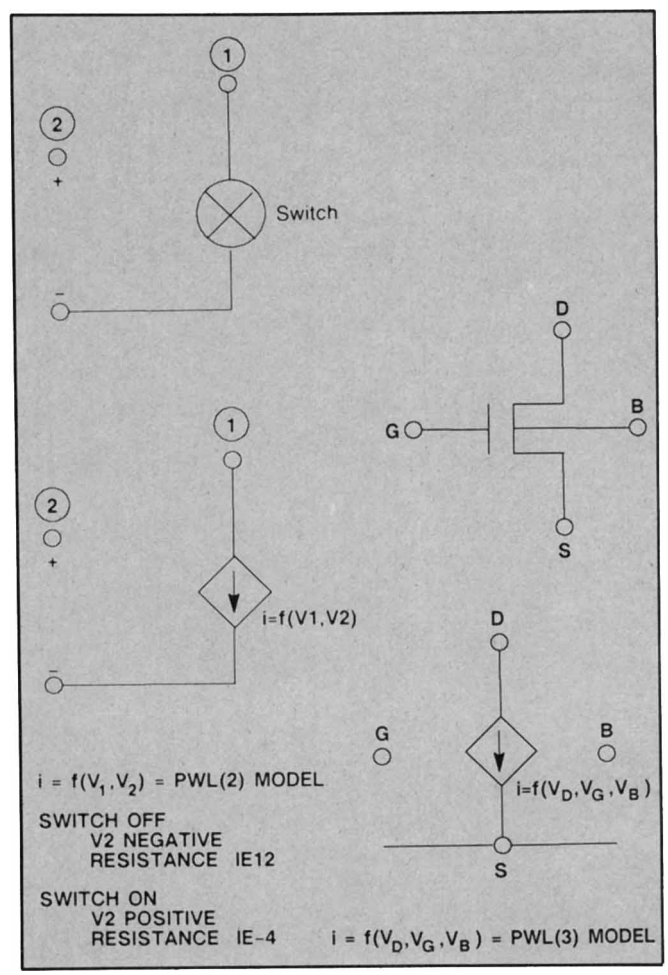

Fig. 20 PWL Switch.

switch itself. This determines the on/off resistance characteristics. The second controlling voltage actually sets the trip point of the switch. When it is positive, the switch is on; when it is negative, the switch is off.

```
* SWITCH
GSWITCH 1 0 PWL(2) 1 0 2 0 SWITCH
.MODEL SWITCH PWL(2) SYMMETRY ODD 1      SOURCE CARDS DATA
+       0A     0V   -1V
+   1E-10   100V   -1V
+       0A     0V    0V
+   2E-10   100V    0V
+       0A     0V    1V
+      1E6   100V    1V
```

Three-dimensional VCCS PWL tables for MOS characteristic data can be taken in the laboratory and used directly in SPICE2 (see Fig. 21). The three controlling voltages are drain-to-source, gate-to-source, and bulk-to-source. The scale factor on the VCCS card allows the same PWL model to be used for transistors with different sizes.

```
* PWL MOSFET
.SUBCKT MOSFET D G S B SCALE=1
G D S PWL(3) D S G S B S MOSFET (SCALE)
.ENDS
.MODEL MOSFET PWL(3) SYMMETRY ODD 1      SOURCE CARDS
+ INCR1    0 10   5      INCR2 0 10 5     INCR3 -10 0 5
+ DATA     0  0   0
+          0 1M 4M
+          0 2M 5M
+          0  0   0
+          0 2M 5M
+          0 3M 6M
+          0  0   0
+          0 3M 6M
+          0 4M 7M
```

User-Defined FORTRAN Subroutine Applications

A simple diode subroutine as a function of area A, saturation current IS, and VT has the following form:

```
      SUBROUTINE GDIODE(ICOM,VALUE1,VALUE2,COEF,V)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ICOM(1),VALUE2(1),COEF(1),V(1)
C ... Compute VALUE1 as a function of ICOM, COEF, and V
C ... Compute VALUE2 as a function of ICOM, COEF, and V
      A          = COEF(1)
      XIS        = COEF(2)
      VT         = COEF(3)
      EVD        = DEXP(V(1)/VT)
      VALUE1     = A*XIS*(EVD-1.0D0)
      VALUE2(1)  = A*XIS*EVD/VT
      RETURN
      END
```

The corresponding VCCS element card for A=.6, IS=1.E-14, and VT=0.026 is shown:

```
G1 1 0 USER(1) 1 0 GDIODE(.6,1.E-14,0.026)
```

The FORTRAN source for a two-input NAND gate is listed subsequently. It allows the low-level, high-level, and threshold-level voltages to be specified.

```
      SUBROUTINE NAND2(ICOM,VALUE1,VALUE2,COEF,V)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION COEF(3),V(2),VALUE2(2),ICOM(1)
C ... Compute VALUE1 as a function of ICOM, COEF, and V
C ... Compute VALUE2 as a function of ICOM, COEF, and V
      VLOW       = COEF(1)
      VHIGH      = COEF(2)
      VT         = COEF(3)
      VIN1       = V(1)
      VIN2       = V(2)
      IF(VIN1.LE.VT.AND.VIN2.LE.VT) VALUE1=VHIGH
      IF(VIN1.LE.VT.AND.VIN2.GE.VT) VALUE1=VHIGH
      IF(VIN1.GE.VT.AND.VIN2.LE.VT) VALUE1=VHIGH
      IF(VIN1.GE.VT.AND.VIN2.GE.VT) VALUE1=VLOW
      VALUE2(1) = 0.0D0
      VALUE2(2) = 0.0D0
      RETURN
      END
```

Calling a two-input NAND gate with a VCVS that uses a user-defined subroutine looks like this:

```
E1 OUT 0 USER(2) IN1 0 IN2 0 NAND2(0V,5V,1.5V)
```

Other applications, such as the lossy transmission line [7], have been used during dc and frequency analysis with the resistances as VCCS and capacitances as a function of frequency. TISPICE permits capacitances also to be defined by user-defined routines.

Summary

This paper has pointed out how to use the dependent sources effectively. The output capability has been enhanced by demonstrating ways these dependent sources can be described to add, subtract, multiply, divide, differentiate, and integrate voltages and currents. Several dependent source applications have been shown modeling both analog and digital circuit characteristics. The number of applications for these dependent sources is only limited by one's ingenuity.

References

[1] L. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," Memorandum ERL-M520, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA, 1975.

[2] E. Cohen, "Program Reference for SPICE2," Memorandum ERL-M592, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA, 1976.

[3] L. O. Chua and P-M. Lin, *Computer-Aided Analysis of Electronic Circuits*, Englewood Cliffs, NJ: Prentice-Hall, 1975.

[4] G. R. Boyle, B. M. Cohn, and D. O. Pederson, "Macromodeling of Integrated Circuit Operational Amplifiers," *IEEE J. Solid-State Circuits*, vol. SC-9, no. 6, Dec. 1974.

[5] *User's Guide to SPICE2: Simulation Program with Integrated Circuit Emphasis*, Design Automation Division, Texas Instruments, Dallas, TX.

[6] J. Wyatt, D. Mikulecky, and J. DeSimone, "Network Modelling of Reaction-Diffusion Systems and Their Numerical Solution Using SPICE," Department of Physiology, Medical College of Virginia, Richmond, VA.

[7] R. Hester, E. Lawrence, and B. Epler, "MOS Active Filter Realization with Lossy Transmission Lines," Technical Report, Texas Instruments, Dallas, TX.