CHAPTER 1

Introduction

Circuit simulation is a technique for checking and verifying the design of electrical and electronic circuits and systems prior to manufacturing and deployment. It is used across a wide spectrum of applications, ranging from integrated circuits and microelectronics to electrical power distribution networks and power electronics. Circuit simulation is a mature and established art and also remains an important area of research. This text covers the theoretical background for circuit simulation, as well as the numerical techniques that are at the core of modern circuit simulators. Circuit simulation combines a) mathematical modeling of the circuit *elements*, or *devices*, b) formulation of the circuit/network equations, and c) techniques for solution of these equations. We will focus mainly on the formulation and solution of the network equations and will not cover device modeling in any detail.

Compared to simulators that operate on a design description at higher levels of abstraction, such as logic or functional simulators, circuit simulators use a *detailed* (so-called *circuit level* or *transistor level*) description of the circuit and perform a relatively *accurate* simulation. Typically, such a simulation uses physical models of the circuit elements, solves the resulting differential and algebraic equations, and generates time-waveforms of node voltages and element currents. In general, a circuit simulator allows the use of any simulation primitive, provided it can be described with an appropriate *device model*. In practice, while some devices (e.g., resistors, capacitors) are two-terminal elements, others (e.g., transistors) have more than two terminals. However, a multiterminal element is usually modeled as a subcircuit consisting of only two-terminal elements. Thus, a common starting point for studying circuit simulation is to restrict the formulation to two-terminal elements. For integrated circuits, circuit simulators often work with an *extracted* circuit description, which gives better accuracy. Thus circuit capacitance, resistance, and inductance can be included in the analysis, be they prespecified discrete components or parasitic.

Early techniques for circuit simulation using computers were introduced in the 1950s and 1960s, and several limited-scope simulators were developed. The first general-purpose circuit simulator to experience widespread usage was SPICE, developed by L. W. Nagel at the University of California, Berkeley, in the

Circuit Simulation, by Farid N. Najm

early 1970s, under the supervision of Professor D. O. Pederson. According to Nagel (1975), SPICE evolved from an earlier simulator called CANCER that, in turn, "*emerged from a series of courses that were instructed by Professor R. A. Rohrer*" at Berkeley. The original CANCER program, described in Nagel and Rohrer (l971), was followed up by SPICE, described in Nagel and Pederson (l973), and then SPICE2, which is described in Nagel's 1975 thesis. The rest, as they say, is history, as SPICE has become the *de facto* standard circuit simulator. At the time of this writing, SPICE3f is the latest version of the program. Further information on SPICE and its history is available in Kundert (1995), in Vladimirescu (1994), and in Pederson (1984).

SPICE was developed specifically with integrated circuits in mind. Indeed, the acronym stands for *Simulation Program with Integrated Circuit Emphasis*. This, coinciding with the birth and growth of the integrated circuits industry in the 1970s, led to widespread adoption of the program. Furthermore, the availability of the SPICE code and documentation from Berkeley, for a nominal fee, spurred the development of similar circuit simulators elsewhere. Today, there are thousands of copies of circuit simulators in use across the industry, and there are many SPICE-like simulators in the market. Some semiconductor companies average over 1 million circuit simulation runs per week!

Circuit simulation issues to be covered in this book include a) device equations, b) equation formulation, and c) solution techniques. In this chapter, we will briefly introduce each of these issues and then present the overall structure of a circuit simulator.

1.1 DEVICE EQUATIONS

In this context, a *device* is any simulation *primitive*, or *element*, described by means of a current-voltage relationship. Thus, a resistor is described by the (Ohm's law) equation $v = Ri$. By convention, the positive (reference) direction for current is determined from the positive (reference) direction for voltage, as shown in Fig. 1.1, so that current flows in the device from the positive reference node to the negative node.

When the element equation contains no terms with powers of 2 or higher, the element is said to be a *linear element*. A network of linear elements is said to be a *linear circuit*. The resistor element equation is algebraic. On the other hand, a capacitor is described by $i = Cdv/dt$, which is a *first-order ordinary differential*

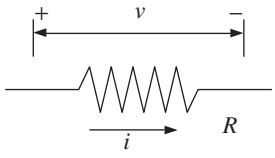

Figure 1.1: A resistor.

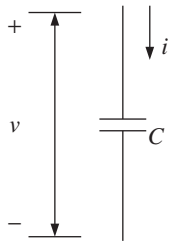

Figure 1.2: A capacitor.

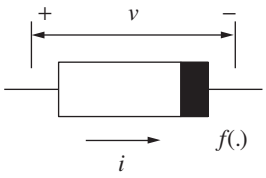

Figure 1.3: Nonlinear resistor.

equation. It is first order because it contains only first-order derivatives and it is ordinary because it contains no partial derivatives. Since there are no powers of 2 or more, this too is a linear equation. Here too, the reference direction for current is based on the reference direction for voltage, as in Fig. 1.2.

Wiring is typically modeled using lumped R, L, or C elements, so that metal interconnect is described by a system of *linear first-order differential equations*. Resistors and capacitors are examples of *two-terminal* linear devices. In general, a two-terminal device may be described by an i-v equation $i = f(v)$, where f can be any function $f : \mathbb{R} \to \mathbb{R}$. When f is a nonlinear function, the device is said to be *nonlinear* and is given the (nonlinear resistor) symbol shown in Fig. 1.3.

A pn-junction diode is an example of a commonly used two-terminal nonlinear device. Transistors, such as BJTs and MOSFETs are three-terminal nonlinear devices (four-terminal, if a detailed model is used that includes the body voltage).

1.2 EQUATION FORMULATION

The behavior of a circuit is captured by a set of equations that are formulated by combining the element equations and Kirchoff's Current and Voltage Laws (KCL and KVL). In general, this results in a set of simultaneous *nonlinear first-order differential equations*. For a purely *resistive*, *linear*, circuit, the equations are simply a system of simultaneous linear algebraic equations.

As an example, consider the linear circuit shown in Fig. 1.4. From KCL, we can write:

$$i = i_1 = i_2 \tag{1.1}$$

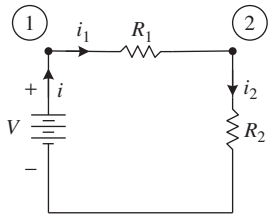

Figure 1.4: A simple linear circuit.

The element equations provide:

$$v_1 = V, \quad i_1 = \frac{v_1 - v_2}{R_1}, \quad i_2 = \frac{v_2}{R_2} \tag{1.2}$$

which, substituted into KCL ($i_1 = i_2$), gives:

$$\frac{1}{R_1}(v_1 - v_2) = \frac{1}{R_2}v_2 \tag{1.3}$$

KVL around the loop then provides:

$$V = R_1 i_1 + R_2 i_2 = (R_1 + R_2)i = (R_1 + R_2)\frac{v_2}{R_2} \tag{1.4}$$

where, in the last step, we have benefited from KCL ($i = i_2$) and the element equation $i_2 = v_2/R_2$, and this then leads to:

$$v_2 = \frac{R_2}{R_1 + R_2}V \tag{1.5}$$

With v_2 in hand then, using (1.3), we get the value of v_1, and the element equation $i_2 = v_2/R_2$ can then be used to solve for $i = i_1 = i_2$.

The above *ad hoc* approach of solving equations by substitution and similar operations does not scale well to large circuits. Instead, we need a *systematic* and *automatic* approach for formulating and solving the circuit equations. For now, we maintain our focus on the simple case of linear resistive circuits. There are two popular approaches for systematic equation formulation, *sparse tableau analysis* (STA) and *modified nodal analysis* (MNA). Sparse tableau analysis, described in Hachtel et al. (1971), involves the following steps:

1. Write KCL as $Ai = 0$, where A is a *reduced incidence matrix* that we will introduce later on, and i is a vector of all branch currents.
2. Write KVL as $u = A^T v$, where u is a vector of all branch voltages and v a vector of all nodal voltages to ground.
3. Write the element equations as $Zi + Yu = s$, where Z and Y are matrices and s is a vector.

The combination of these three sets of linear algebraic equations leads to the sparse tableau system:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & I & -A^T \\ Z & Y & 0 \end{bmatrix} \begin{bmatrix} i \\ u \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} \tag{1.6}$$

This formulation has some key features in that it can be applied to *any* circuit in a systematic fashion, the equations can be assembled *directly* from the input (circuit specification), as we will see later on, and the coefficients matrix is *very* sparse (has mostly zero elements), although it is larger in dimension than the MNA matrix. Modified nodal analysis, described in Ho et al. (1975), involves the following steps:

1. Write KCL as $Ai = 0$.
2. Use the element equations to eliminate as many current variables as possible from KCL, leading to equations in terms of mostly branch voltages.
3. Use KVL to replace all the branch voltages by nodal voltages to ground.
4. Append element equations of those elements whose current variables could not be eliminated as additional equations of the MNA system.

We will see the details of this process later on, and it leads to the MNA system:

$$\begin{bmatrix} Y & B \\ C & Z \end{bmatrix} \begin{bmatrix} v \\ i \end{bmatrix} = \begin{bmatrix} s_v \\ s_i \end{bmatrix} \tag{1.7}$$

As with STA, this formulation can be applied to *any* circuit in a systematic fashion and the equations can be assembled *directly* from the input (circuit specification). As well, the coefficient matrix is sparse, but often not as sparse as the STA matrix, although it is smaller in dimension. The MNA matrix can become singular during the numerical solution process and, therefore, requires careful pivoting.

With the larger matrix size, STA can take longer to formulate the equations than MNA, but it solves them faster; it is well suited for repeated use as in statistical analysis. Most modern circuit simulators use the MNA approach.

As an example of the MNA formulation, consider the linear resistive circuit in Fig. 1.5. We write KCL at every node and then eliminate the current variables using the element equations, as follows:

$$\text{KCL at node 2: } \frac{V - v_2}{R_1} = \frac{v_2 - v_3}{R_2} \quad \Rightarrow \quad \left(\frac{1}{R_1} + \frac{1}{R_2}\right) v_2 - \frac{1}{R_2} v_3 = \frac{V}{R_1}$$

$$\text{KCL at node 3: } \frac{v_2 - v_3}{R_2} = \frac{v_3}{R_3} \quad \Rightarrow \quad -\frac{1}{R_2} v_2 + \left(\frac{1}{R_2} + \frac{1}{R_3}\right) v_3 = 0$$

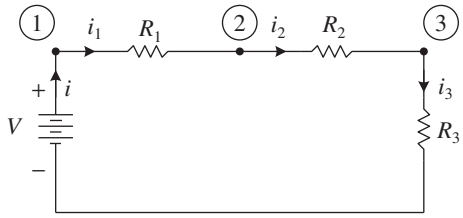

Figure 1.5: A linear circuit, used to demonstrate the MNA formulation.

This leads to the MNA matrix equation:

$$\begin{bmatrix} \left(\frac{1}{R_1}+\frac{1}{R_2}\right) & \frac{-1}{R_2} \\ \frac{-1}{R_2} & \left(\frac{1}{R_2}+\frac{1}{R_3}\right) \end{bmatrix} \begin{bmatrix} v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \frac{V}{R_1} \\ 0 \end{bmatrix} \tag{1.8}$$

Notice that the system matrix can be written as the sum of three matrices:

$$\begin{bmatrix} \left(\frac{1}{R_1}+\frac{1}{R_2}\right) & \frac{-1}{R_2} \\ \frac{-1}{R_2} & \left(\frac{1}{R_2}+\frac{1}{R_3}\right) \end{bmatrix} = \begin{bmatrix} \frac{1}{R_1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{R_2} & \frac{-1}{R_2} \\ \frac{-1}{R_2} & \frac{1}{R_2} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{R_3} \end{bmatrix} \tag{1.9}$$

each of which relates to a specific element. These contributions of the various elements are called *element stamps*, as we will see later on. In general, a resistor like R_2, which is not connected to ground, has the following element stamp:

$$\begin{array}{ccccccc} & & v^+ & & v^- & & \\ & & \vdots & & \vdots & & \\ n^+ & \cdots & +G_2 & \cdots & -G_2 & \cdots \\ & & \vdots & & \vdots & & \\ n^- & \cdots & -G_2 & \cdots & +G_2 & \cdots \\ & & \vdots & & \vdots & & \end{array}$$

where $G_2 = 1/R_2$. This and similar element stamps are used to directly build the required MNA matrix as the simulator is reading (parsing) the circuit description file.

1.3 SOLUTION TECHNIQUES

As seen in the above examples, such as in (1.8), solving linear resistive circuits reduces to solving the *linear system*:

$$Ax = b \tag{1.10}$$

This is a classical problem that is basic to many engineering disciplines and has a variety of solution techniques. Direct methods of solution include matrix inversion, Gaussian elimination, and LU factorization. Indirect methods (relaxation methods) of solution include Gauss-Jacobi and Gauss-Seidel, successive over-relaxation, etc. The most common method is LU factorization, which proceeds as follows:

1. Factor A as $A = LU$, where L is lower-diagonal and U is upper-diagonal.
2. Solve $Lz = b$ for z, by forward substitution.
3. Solve $Ux = z$ for x, by backward substitution.

We will see the details of this process later on and we will recognize that a most desirable property throughout all this is *matrix sparsity*.

1.3.1 Nonlinear Circuits

Solving nonlinear circuits is typically done using *Newton's method*. We will see that this means that we repeatedly, until convergence, perform the following two steps:

1. *Linearize* the circuit equations around a candidate solution point.
2. *Solve* the resulting linear circuit using LU factorization to discover a better solution point.

In this way, the MNA formulation for linear resistive circuits turns out to be *sufficient*, because the solution of a nonlinear circuit is reduced to repeated solutions of linearized versions of that circuit.

As an example of the process of linearization around a candidate solution point, consider a nonlinear resistor with the element equation $i = f(v)$, as depicted in Fig. 1.6. The equation of the tangent line at the point (v_0, i_0) is:

$$i = f'(v_0)\,[v - v_0] + i_0 = f'(v_0)v + I_{eq} \tag{1.11}$$

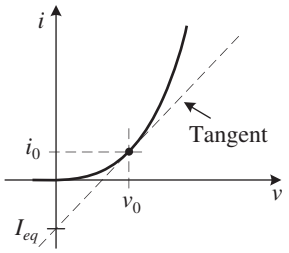

Figure 1.6: An i-v characteristic for a nonlinear resistor, showing linearization around a candidate solution point.

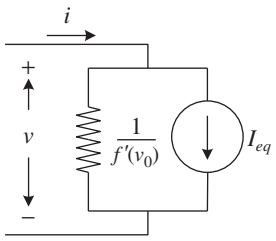

Figure 1.7: A linear circuit that has the same current–voltage characteristic as the tangent line in Fig. 1.6.

where $I_{eq} = i_0 - f'(v_0)v_0$. This equation is also the i-v characteristic of the sub-circuit shown in Fig. 1.7. This sub-circuit is called a *companion model*, of the nonlinear resistor. The element stamp of the companion model is then used to build the matrix of the linearized circuit, and the resulting linear system is solved. This process is repeated until the successive candidate solution points have converged to their final value.

1.3.2 Dynamic Circuits

All the preceding has been for resistive circuits and we have focused on the solution at a single point in time, a so-called `DC Analysis`. In general, circuits include dynamic (L, C) elements, and we are interested in the response over time, a so-called `Transient Analysis`. This is done by using a *finite difference approximation* of the derivative, such as:

$$i = C\frac{dv}{dt} \approx C\frac{(v(t + \Delta t) - v(t))}{\Delta t} \tag{1.12}$$

By replacing all derivatives by their finite difference approximations, the circuit equations effectively become *algebraic*, rather than differential, and possibly nonlinear. Given the solution at time t, i.e., $v(t)$ and $i(t)$, these equations are then solved for $v(t + \Delta t)$ and $i(t + \Delta t)$. Thus, by this operation of *time discretization*, the problem is reduced to solving a possibly nonlinear resistive network at every time-point, based on the use of another kind of *companion model* for the dynamic elements.

1.4 CIRCUIT SIMULATION FLOW

The flow-chart shown in Fig. 1.8 is useful to visualize the overall simulation flow inside a circuit simulator. The simulator repeatedly applies time discretization, element linearization, and matrix equation solution. In the following chapters, we will describe the many details, and pitfalls, of these various activities.

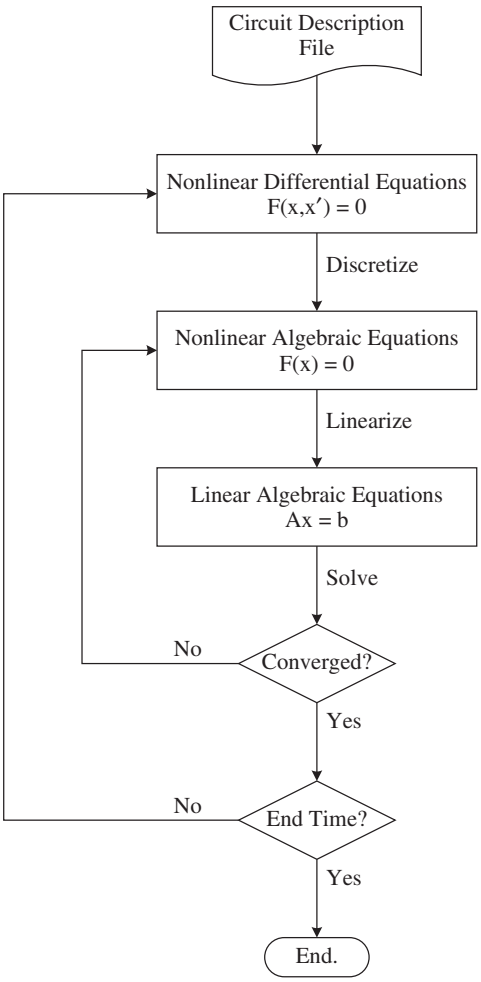

Figure 1.8: Overall circuit simulation flow.

1.4.1 Analysis Modes

Circuit simulators offer different *analysis modes*. Berkeley's *The Spice Page*, at the web site:

`http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE`

lists the following analysis modes for `SPICE3`:

- `DC Analysis`: Determines the DC operating point of the circuit with inductors shorted and capacitors opened. It is automatically performed prior to a `Transient Analysis`, to determine the initial conditions, and prior to an

`AC Small-Signal Analysis`, to determine the linearized, small-signal models for nonlinear devices. It can also be used to get the DC transfer curves by means of a `DC sweep`.

- `Transient Analysis`: Computes the output variables, voltages and currents, as functions of time over a user-specified time interval.
- `AC Small-Signal Analysis`: Finds the AC output variables as functions of frequency (transfer function), over a user-specified range of frequencies.
- `Pole-Zero Analysis`: Computes the poles and/or zeros in the small-signal AC transfer function. It is time-consuming for large circuits.
- `Small-Signal Distortion Analysis`: Computes steady-state harmonic and intermodulation products for small input signal magnitudes.
- `Sensitivity Analysis`: Finds the sensitivity of an output variable with respect to all circuit variables, including model parameters.
- `Noise Analysis`: Studies device-generated noise for the given circuit, providing the noise contributions of each device to the output voltage.

In addition, analysis at different temperatures is allowed; further details are available in Vladimirescu (1994). Other simulators offer additional capabilities, such as statistical analysis, switched capacitor circuit analysis, etc. However, perhaps the most common usage of circuit simulation involves running, first, a `DC Analysis`, followed by a `Transient Analysis`. Thus, our study will focus on only these two analysis modes.

Notes Additional background on circuit simulation is available in various texts, such as in Chua and Lin (1975), sections 2.1–2.5 and 3.1–3.5, in Vladimirescu (1994), in Vlach and Singhal (1994), sections 1.1–1.5, and in Pillage et al. (1995), chapter 1.

Problems

1.1. (Computer Project) Write a parser, in `C` or `C++`, that can read a circuit specification in terms of a simple "language" that we now describe. The language is quite limited and restrictive, and represents the bare minimum that will be needed for subsequent projects in this book. The parser should not be case-sensitive, and should interpret any contiguous sequence of spaces or tabs as equivalent to a single space. Every line of the input file should describe a single circuit element, and the description of every circuit element should be given wholly within a single input line. The order of lines in the input file is immaterial and any characters following a % in an input line should be considered as *comments* and ignored. Circuit node names should be non-negative integers, from the set $\{0, 1, 2, \ldots\}$, and the node name `0` should be reserved and used for the *ground* or *reference* node.

The accepted circuit elements and their specifications are given below. In this, the symbol <node.*>, where * can be any single alphanumeric character, denotes a node name. Specifically, <node.+> denotes the node that is the positive voltage reference point for the element and <node.-> denotes the negative reference node. The positive direction of current in any element is assumed to be from <node.+> to <node.->. The symbol <int> denotes a non-negative integer, and <value> denotes a non-negative real number. The <value> given for a circuit parameter, like resistance or capacitance, should be in the standard units: Volt, Ampere, Ohm, Farad, or Henry, but it should *not* include the corresponding unit. Finally, anything inside brackets, such as [G2] or [<value>] is an *optional* field.

- Voltage source: Only independent DC voltage sources are allowed, specified as:

```
V<int>   <node.+>   <node.->   <value>
```

- Current source: Only independent DC current sources are allowed, specified as:

```
I<int>   <node.+>   <node.->  <value>   [G2]
```

- Resistor: Only linear resistors are allowed, specified as:

```
R<int>   <node.+>   <node.->  <value>   [G2]
```

- Capacitor: Only linear capacitors are allowed, specified as:

```
C<int>   <node.+>   <node.->  <value>   [G2]
```

- Inductor: Only linear inductors are allowed, and they should be specified as:

```
L<int>   <node.+>   <node.->   <value>
```

- Diode: The diode model, and its parameter values, will not be part of the input description. Instead, the model will be built into any subsequent simulation code that you will write and only the terminals should be specified here. Optionally, a scale factor can also be included so as to allow the specification of diodes that are larger than minimum size. The specification is:

```
D<int>   <node.+>   <node.->   [<value>]
```

- BJT: Similar to the diode model, only the terminals and an optional scale factor are given. Let `QN` denote an npn device and `QP` denote pnp; the specification is:

```
QN<int>   <node.C>   <node.B>   <node.E>   [<value>]
QP<int>   <node.C>   <node.B>   <node.E>   [<value>]
```

  where the nodes represent the collector, base, and emitter terminals, respectively.
- MOSFET: Similar to the above, we give only the terminals and an optional scale factor, and the body terminal is to be ignored:

```
MN<int>   <node.D>   <node.G>   <node.S>   [<value>]
MP<int>   <node.D>   <node.G>   <node.S>   [<value>]
```

  where `MN` denotes an n-channel device and `MP` is p-channel, and the nodes represent the drain, gate, and source terminals, respectively.

The parser should create a data structure, as a linked list of records, where each record describes a separate circuit element, including its terminals and parameter values. Test your parser on circuits of your choice. An example is given in Fig. 2.35.