Chapter 1
Circuit Simulation

1.1 Historical Perspective

Circuit simulators, as we know them today, first began to appear in the late 1960's and early 70's. However, it was the explosive growth of the integrated circuit market in the 1970's that precipitated the rise of importance of circuit simulation. With integrated circuits, prototypes were expensive to build and difficult to troubleshoot. Circuit simulators were necessary to evaluate designs before they were fabricated. As designs became larger and more complicated, the need to use circuit simulators increased.

Two groups contributed significantly to the development of the modern circuit simulator. The ASTAP group at IBM developed many of the numerical method's used. And the SPICE group at the University of California at Berkeley developed and propagated the de facto standard simulator.

The simulation effort at Berkeley started as a class project of Prof. Ron Rohrer. That modest beginning resulted in a flurry of simulation programs being developed and culminated in the release of SPICE in 1972 and then SPICE2 in 1975. SPICE was written by Larry Nagel, then under the guidance of Prof. Don Pederson. SPICE became very important for three reasons. First, SPICE was designed to be used to simulate integrated circuits. Unlike the simulators that preceded it, SPICE had all the models one needed to simulate integrated circuits built into it. As such, it was easier to use than earlier simulators.

Second, the source code for SPICE was made available to anyone who wanted it at a nominal cost. And third, Berkeley graduates took SPICE with them as they went to work at electronics companies throughout the country.

In the late 70's and early 80's, most versions of SPICE were proprietary and only used in-house by the integrated circuit manufacturers. At the time, it seemed as if every large electronics company in the country had developed their own version of SPICE. This occurred because circuit simulation was vital to IC manufactures and at least initially, there were few commercial simulators available that were suitable for IC design. Also, the IC manufacturers viewed having a good version of SPICE as a strategic advantage that allowed them to get designs to market quickly and reliably.

This situation began to change in the late 80's and early 90's as the commercial simulators began to surpass the internally developed simulators in terms of capabilities and performance. When this happened, the strategic value of an internally developed simulator disappeared. Commercial simulators starting replacing internal simulators, starting with the smaller companies and working up. Today, only the largest companies are still developing their own proprietary simulators.

In the late 80's, Berkeley upgraded SPICE by releasing SPICE3, which had a new architecture that made it considerably easier to add new component models and was written in C. While SPICE3 was architecturally a big step forward from SPICE2, algorithmically it was largely the same.

At the same time, Berkeley also released a new type of circuit simulator called Spectre. Spectre used harmonic balance to directly compute the steady-state solution of nonlinear circuits in the frequency domain. It was targeted for use on microwave circuits. Spectre was picked up by Hewlett-Packard, where it became known as their Microwave Nonlinear Simulator, or MNS, and by Cadence, where the harmonic balance algorithms were replaced by transient analysis algorithms.

Cadence took a slightly different approach with Spectre than is typ-

ical. Rather than trying to increase the speed of the simulator by loosening the tolerances or employing faster, but less reliable, timing analysis algorithms, Cadence instead took a conservative approach. It started with the standard SPICE algorithms, discarded those that reduced reliability, such as bypass, and implemented each one from scratch with the goal of improving speed, as well as accuracy and reliability. By exploiting the 15 years of evolutionary improvements in simulation algorithms that had occurred since SPICE was written, Cadence was able to make Spectre 3-5 times faster than traditional versions of SPICE, while improving its accuracy and reliability.

It was during the process of developing Spectre than many of the issues that are discussed in this book were first encountered and explored. It is because Spectre was designed to address these issues that it plays a central part in this book. However, the book does not focus exclusively on Spectre. It discusses issues applicable to all simulators and so is useful for anyone that uses a circuit simulator.

A more comprehensive history of circuit simulation in general, and SPICE in particular, is available from Pederson [pederson84] and in Vladimirescu's *The SPICE Book* [vladimirescu94].

1.2 Algorithmic Perspective

The algorithms used in SPICE now define the traditional approach to circuit simulation. This approach is referred to as the *direct method* for simulating a circuit. With direct methods, the nonlinear ordinary differential equations that describe the circuit are first formulated and then converted to a system of difference equations by a multi-step integration method such as the trapezoidal rule. The nonlinear difference equations are solved using the Newton-Raphson algorithm, which generates a sequence of linear equations that are solved using sparse Gaussian elimination. Direct methods have proven to be the most reliable and general methods available.

In the late 70's and early 80's, attempts were made by several groups to develop alternate approaches that would provide better performance on the large digital circuits of the day. Two basic methods were explored, explicit integration methods and relaxation methods.

Using explicit integration methods, such as forward Euler, eliminates the need to actually solve the large system of differential equations that describe the circuit. Instead, the solution at a particular time step is extrapolated from the previous time point. It is assumed that there are no floating capacitors and there is at least one capacitor connecting every node in the circuit to ground. The extrapolation is performed by evaluating the circuit equations (not solving them) to determine the current into the grounded capacitors. The slope of the voltage waveforms are then computed directly from

$$\dot{v}(t) = C^{-1} i(t). \tag{1.1}$$

Explicit integration methods are very fast, especially if the grounded capacitors are linear and so are easily invertible. However, they have not gained wide acceptance because they are unstable and so generate results that blow-up if the circuit contains time constants that are shorter than the time step being used. This is a serious problem that makes explicit methods unstable on most circuits. Digital MOS circuits tend to have time constants that are all about the same size, so explicit methods can sometimes be used with great success.

Relaxation methods exploit latency in the circuit by breaking it into smaller pieces and solving each piece independently. If the signals in one or more pieces are latent, then it is not necessary to solve for them. The waveform relaxation methods take this idea one step further. Circuits are still partitioned into subcircuits, but the subcircuits are solved independently over an interval of time rather than for a single time point. This allows the simulator to exploit multi-rate behavior as well as latency. Multi-rate behavior is where the signals in subcircuits are changing, but where the signals in one are changing much more slowly than in another. In this case, the simulator is free to choose larger time steps in the subcircuit whose signals are changing slowly.

The drawback to relaxation methods is that because the entire circuit is not evaluated at once, it is sometimes necessary to make assumptions about signals before they have been computed, and if the assumptions turn out to be incorrect, the subcircuits that depend on those signals have to be reevaluated. Consider the circuit shown in Figure 1.1 and assume $\mathcal{N}_1$ is evaluated before $\mathcal{N}_2$. Further assume