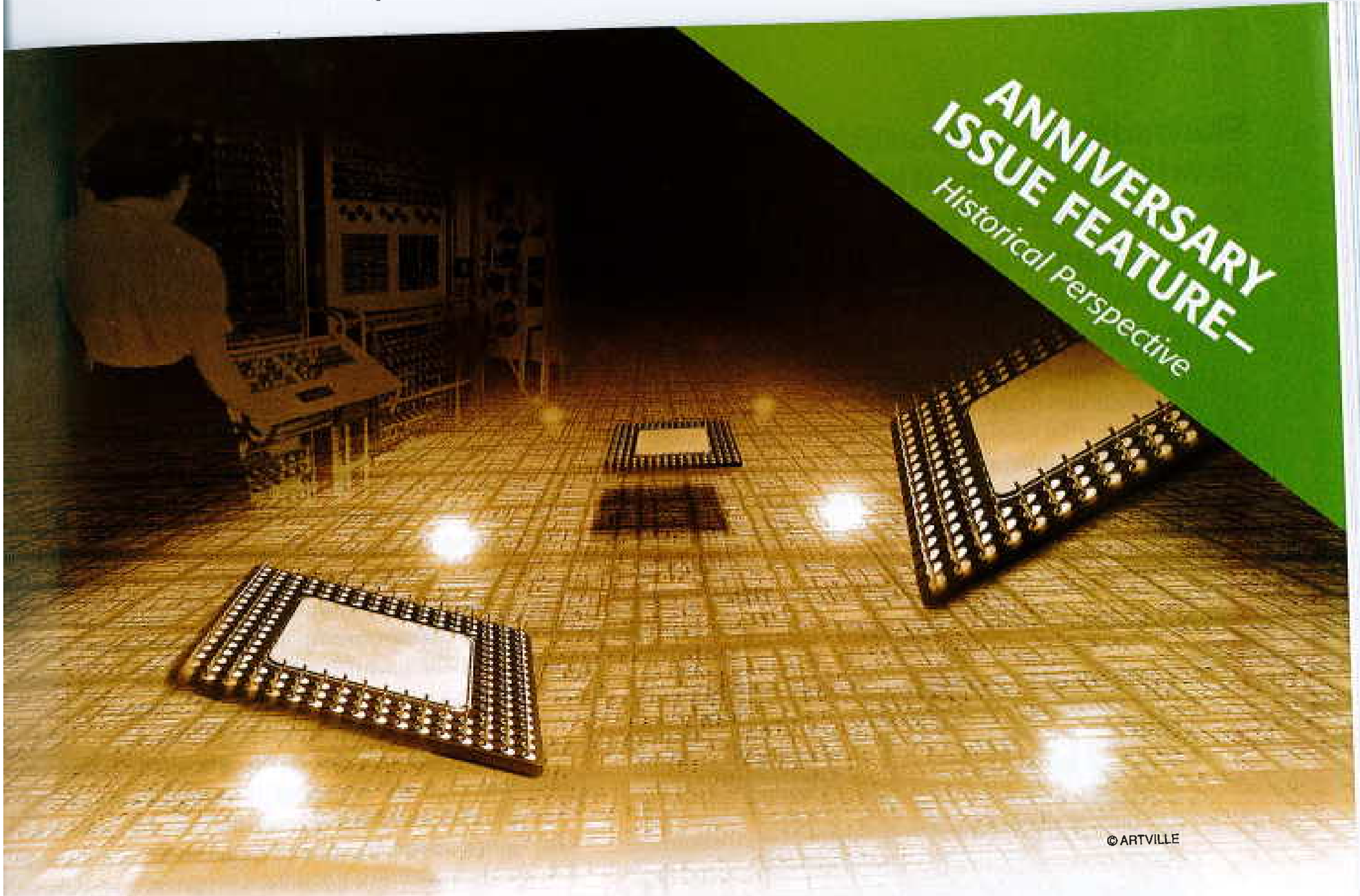


Circuit Simulation in the Dark Ages

Steve Maas

This is about history. Please try to contain your excitement.

Yes, we all went into technology because, to put it charitably, we weren't excited by history, literature, or any of the humanities, and the few of us who had some stirrings of interest in those areas weren't excited enough to make it our lives' work. Or at least, we weren't enamored of the options those fields presented, which rarely included gainful employment. As a result, we now enjoy a reputation among the literati as obtuse and illiterate geeks. I'm sure that gives them great comfort when their toilets flood their bathrooms and they have no idea how to shut off the water.

Still—and at the risk of being a traitor to my class—I enjoy reading novels and listening to music, although I confess that my musical tastes are a couple of hundred years behind the times. I also have a controlled fascination with history. And more to the point, I think it's important for technical people to know about history, especially the history of technology. Although it's arguably possible to do good research and technology

Steve Maas (smaas@awrcorp.com) is with AWR Corporation, El Segundo, California 90278, USA.

Digital Object Identifier 10.1109/MMM.2012.2189990
Date of publication: 7 May 2012

The earliest computers were developed for specific purposes and, while technically programmable, were not programmable in today's sense.

development work without any historical perspective, I've found that historical ignorance has some pretty serious downsides:

1) First, it's easy to reinvent the wheel. When I was the IEEE Transactions on Microwave Theory and Techniques editor, I saw a lot of this. Whenever I received a paper that just seemed too fundamental, I sent it to the oldest reviewer I could find. Invariably, that reviewer would return a reference to earlier, identical work. The problem was not simply a failure to do necessary background research; it was a failure to understand the simple fact that, in the evolution of any technology, fundamental work is completed early and rapidly, and subsequent work consists largely of ever finer fine-tuning. If you come up with something fundamental in a mature field, someone undoubtedly has thought of the same thing long ago. That's the way all technologies develop, not just electronic ones.
2) Second, historical ignorance regularly results in work that is destined to go nowhere, not because it has no value but because it just doesn't fit into the evolutionary pattern that any technology follows. At the risk of being a little controversial, I'd suggest microelectromechancial systems (MEMS) as an example. MEMS technology exists in direct opposition to a clear trend in which mechanical devices and controls are inexorably replaced by electronic ones, as the latter are faster, cheaper, smaller, more versatile, and more reliable. MEMS technology attempts to reverse that trend. Now, if we are to paddle upstream against that evolutionary current, there has to be an awfully good reason for doing so. I still haven't heard it. Indeed, we're still waiting for the MEMS "killer app"—that is, an application so compelling that it makes this technology essential.
3) Finally, it's important to understand how technologies evolve. That evolution can be surprising. Conventional wisdom holds that technologies undergo a kind of natural selection, in which useful technologies win out because of their merits and less satisfactory ones are discarded. In reality, however, economics, innate conservatism, human inertia, and simple luck are powerful forces influencing technological change. For example, today we use scattering (S) parameters for largely historical reasons. In the 1960s they made sense, as newly emerging microwave test systems measured traveling waves, and S parameters were formulated in terms of traveling-wave quantities. Today, however, circuit simulators immediately convert S parameters to admittance (Y) parameters before they can be used. From a purely technical standpoint, Y parameters might be more practical. But we've always used S parameters, and we're comfortable with them. We know what to look for when they are plotted on a Smith chart. So that's what we continue to do.

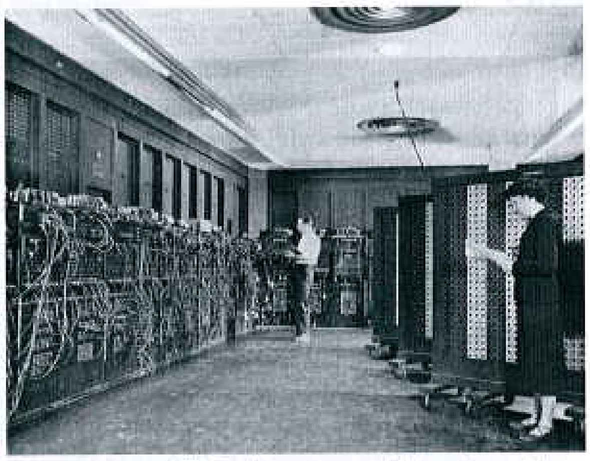


Figure 1. *Part of ENIAC, in a basement laboratory of the University of Pennsylvania's Moore School of Electrical Engineering. Some of the patch cords used to program the computer are visible on the left. Programming ENIAC required several weeks of planning, about a week to plug in all the patch cords, and days to weeks of debugging.*

I could carry this topic much further, and perhaps it should be the basis of a future article. But instead, I'd like to discuss the evolution of circuit technology, how it has fit a pattern, and what has motivated it. From that, perhaps we can understand where this technology is going. I think that's a lot more interesting, and quite a bit more valuable, than reciting a long, dull story of what happened and when. We'll still do some of that, of course, because it's good for you and may even make you feel academic.

Computers and Software

In discussing the history of any computer software technology, it seems that a good place to start is with the computers themselves. The earliest computers were developed for specific purposes and, while technically programmable, were not programmable in today's sense. ENIAC [1], normally credited with being the first real electronic digital computer, developed at the University of Pennsylvania's Moore School of Electrical Engineering, was intended for ballistics calculations and was programmed with switches and patch cables. Figure 1 shows a picture of part of ENIAC. Colossus [2], shown in Figure 2, was designed and built at Bletchley Park, the famous British center for code breaking established during the Second World War. It was designed to test code-breaking schemes, once the nature of the cipher was partially understood. It is likely that Colossus may actually have predated ENIAC, but its existence was kept secret until the 1970s. The inventors of ENIAC, in contrast, went to great lengths to develop the technology once wartime secrecy had ended.

Rapid advances in computer technology, especially the development of high-level programming languages, led to wider and more general use of such machines. In the 1960s, Fortran (the name is no longer spelled using all capital letters) and COBOL were the only widely used high-level computer languages. Students in the sciences learned Fortran, which included necessary functions for technical computation, while COBOL was designed for business applications and was somewhat less frequently taught to business students. Much of the early microwave computer-aided design (CAD) software was written in Fortran. Although most programmers have abandoned those languages, they are still in use, and you can still buy a Fortran or COBOL compiler. While most COBOL programming today is for the maintenance of old software, Fortran, because of its superior numerical libraries, is still preferred in many of the sciences when computational speed and numerical precision are critical.

Early computers using high-level languages were programmed with punched cards. Somewhat later, time-shared teletype terminals became an option as well. Before teletype terminals, the process of running a program was, by today's standards, almost intolerably slow and laborious. It was necessary to punch the cards with a keypunch machine (Figure 3), which always seemed to be located in the overheated and underventilated basement of the computer center, crowded with technogeek students whose hygiene was no better than it is today. The user would submit the "deck" of cards, wait a day or so, and receive output printed on fanfold paper saying that the program had a trivial error. Back to the keypunch and repeat. Eventually, however, commercial time-sharing terminals obviated much of this laborious process. Even with a 110 b/s modem connection and a clunking electromechanical teletype terminal, the improvement in design efficiency seemed like a gift from the gods. Finally, by the early 1980s, electronic terminals with fast modems and dedicated telephone lines allowed the engineer to have a terminal in his office, connected to a remote mainframe or a local "minicomputer" (which actually might have occupied an entire room).

It's easy to say that computers of the mid-20th century were slow and their memory expensive, but that raises the question, as compared to what? While today they seem insufferably primitive even when compared with a modern handheld calculator, in their era they provided an enormous advantage over the only alternative, calculation by hand. Computers spread throughout society in industrialized countries. They were quickly viewed as indispensable.

By the early 1980s, electronic terminals with fast modems and dedicated telephone lines allowed the engineer to have a terminal in his office, connected to a remote mainframe or a local "minicomputer" (which actually might have occupied an entire room).

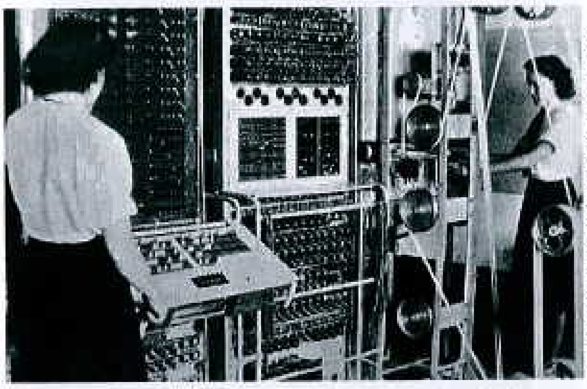


Figure 2. *One of the few existing photographs of Colossus. The coded message was read into the machine by paper tape, and the console at the foreground was used to enter an educated guess at the encryption key. This was useful, of course, only when a good idea of the key could be determined.*

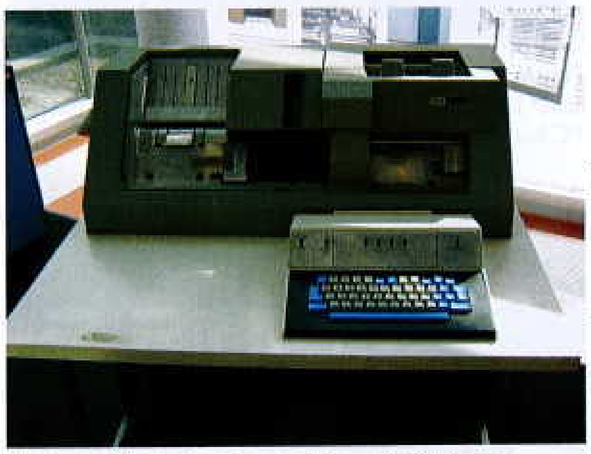


Figure 3. *The IBM 029, one of the most widely used keypunch machines. Cards were loaded into the hopper on the upper right and fed into the punch mechanism. Each card had one line of data or of source code. Once punched, the cards were transferred to the bin at the upper-left part of the machine.*

By the early 1970s, it was possible to buy small computers that occupied only the floor space of an ordinary office or even just a desktop.

Mid-20th-century computers were limited by their circuitry, which used discrete devices hand-soldered into circuit boards and primitive, expensive memory technologies. An example of an early computer board is shown in Figure 4; it doesn't require a lot of imagination to realize that this is, by today's standards, a very slow, expensive circuit. Circuit parasitics were large, memory access times were long, and memory boards required great electronic overhead. Inevitably, the most successful software technologies would be ones that made reasonable demands on speed and memory, did not require special ancillary equipment (e.g., special displays or input–output tools) but still provided great value in eliminating manual labor. Circuit simulation fit these requirements well.

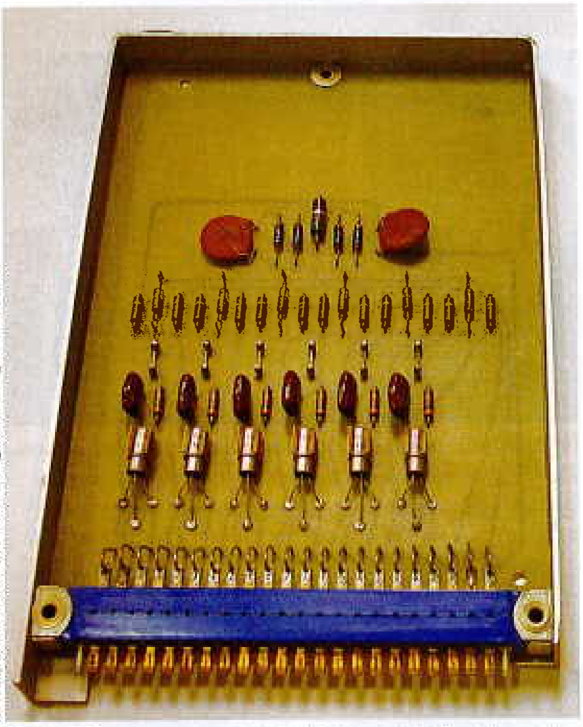


Figure 4. *A computer circuit board—a hex inverter—from about 1960. It used germanium transistors mounted on a PC board.*

Figure 5. *A VAX PDP 11/780 computer from Digital Equipment Corporation (DEC). These machines were especially useful for running early IC-layout software.*

An early, widely used circuit-analysis program was IBM's Electronic Circuit Analysis Program (ECAP). Though its name was not terribly imaginative or clever (it was coined, after all, by engineers), ECAP analyzed linear circuits described by *LCR* components and by controlled and independent sources. ECAP contained no formal transistor models (although occasionally models were added [3], [4]), and it could not handle noise or nonlinearity. Lacking transmission-line elements, it was not very useful for high-frequency design. Other programs existed as well; as this was an important area of circuit-theory research, many noncommercial, academic simulators were also in use. Although it never achieved the rock-star level of fame enjoyed by SPICE (discussed below), ECAP was used extensively in industry, and an advanced version, ECAP II, was reported in 1971 [5]. ECAP II included time-domain nonlinear analysis, sparse-matrix formulation, and other advanced features.

At the same time, the methods for formulating circuit equations for computer analysis were being developed. Initially, methods using such techniques as nodal incidence matrices [6] were used, but a consensus rapidly developed favoring nodal analysis, as it is numerically efficient and much simpler to implement in a computer. Microwave circuit analysis initially used a block-cascade form, described below. It took a few years—and significant improvements in computers—before a consensus for the use of nodal analysis in microwave circuit simulation developed.

By the early 1970s, it was possible to buy small computers that occupied only the floor space of an ordinary office or even just a desktop. An example was the VAX PDP series, one of which is shown in Figure 5. Calculators programmable in a high-level language (usually BASIC, originally a simplified Fortran-like language) had become common, as well. These were possible for one primary reason: the digital integrated circuit. The resulting reduction of size and cost, combined with a significant improvement in speed, brought about rapid improvements in computer technology.

Early Microwave Circuit-Analysis Software

Until the 1960s, microwave systems consisted almost entirely of interconnected waveguide components. Until active microwave devices and hybrid circuits became common, there really wasn't much need for microwave design software that could accommodate an arbitrary circuit. Design software, such as it was, usually analyzed or synthesized specific types of components, such as filters.

In the late 1960s and into the 1970s, microwave hybrid circuits were becoming more common, and it was clear that the old "cut-and-tune" methods for creating high-frequency circuits were too slow and expensive. One motivation was the U.S. space program, whose demand for small, light, rapidly produced components could be met only by hybrids. The advent of gallium arsenide technology, along with really good microwave FETs, made it clear that monolithic microwave ICs would soon be practical as well. You can't "tweak" an IC; precise design is essential. Even hybrid circuits were moving to millimeter wavelengths, becoming so small that tuning was impractical. Something better had to be done.

At the same time, all the pieces were in place to bring about fundamental changes in the way microwave components were designed. These included the following:

- small desktop computers and programmable calculators
- reasonably powerful mainframe computers
- high-level programming languages
- IC technology
- fundamental theoretical understanding of ways to analyze circuits by computer.

In the early 1970s, Les Besser, then an engineer at Fairchild's microwave division, wrote a program called Speedy [7]. Its purpose was primarily to promote Fairchild's transistors, including some of the first commercial GaAs FETs, by providing a practical way to design circuits using them. Three years later, Besser wrote a more advanced program called Computerized Optimization of Microwave Passive and Active Circuits (COMPACT) and formed a company, Compact Engineering, to market, support, and develop the product [8]–[11].

Until the 1960s, microwave systems consisted almost entirely of interconnected waveguide components.

COMPACT was, for many of us, our first introduction to circuit-analysis software and a good lesson in what could be accomplished with it. COMPACT displayed a remarkable combination of simplicity and important features. For an early product, COMPACT was remarkably advanced, including two kinds of circuit optimization, a rudimentary tuning capability, noise analysis, and a plotting capability. Initially it ran on mainframe computers via worldwide commercial time-sharing, but later it was sold for in-house computers. COMPACT was ported to a wide variety of platforms, eventually including programmable calculators and desktop workstations.

COMPACT used a cascade formulation. Figure 6 shows a circuit described in this manner. Each circuit element was treated as a two-port block that could be cascaded with other blocks or connected in series or parallel. This results in a simple way to create a circuit model. In principle, cascaded blocks can be analyzed with ABCD matrices, while parallel and series connections use Y or Z parameters. This would involve repeated matrix conversions, however, and manipulating even small matrices involved a significant computational effort for early machines. Speedy and COMPACT, however, formulated the interconnections directly in S parameters. This obviated much of the matrix conversion.

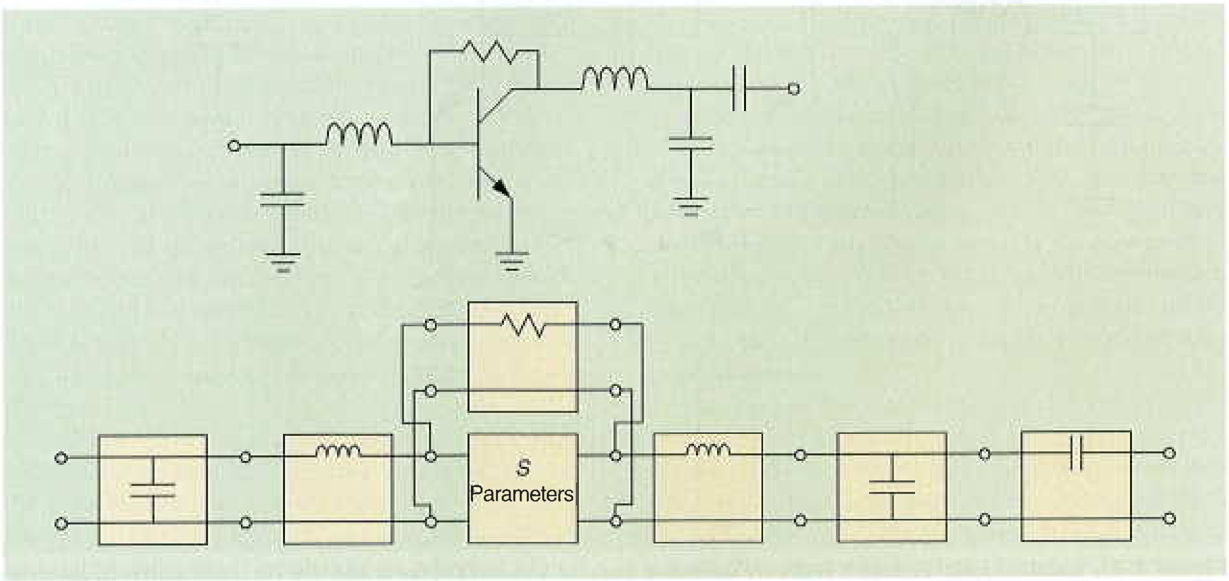


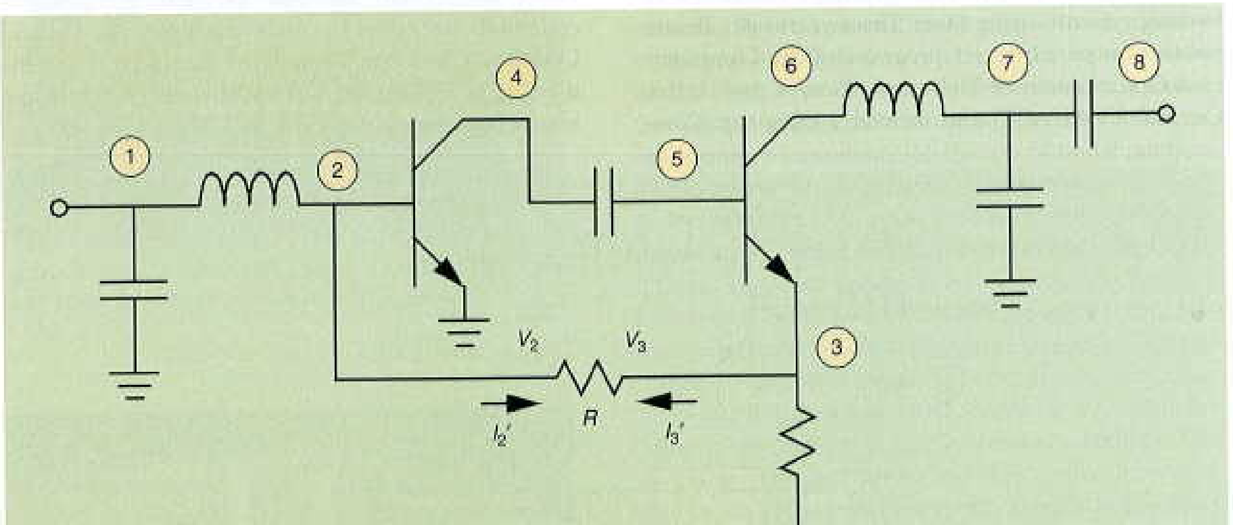
Figure 6. *Many microwave circuits can be described as a cascade, series, or parallel combination of two-ports. Generating an S matrix for the entire structure requires only the application of straightforward block-interconnection rules.*

The idea of using the computer for microwave design in general and optimization in particular did not sit well with some of the older, experienced, but hidebound engineers.

It is not hard to see that certain kinds of circuits cannot be described this way. To accommodate these, COMPACT included the ability to describe a subcircuit by its nodal interconnections. Figure 7 shows an example of a nodal description. Its n nodes are numbered and the admittances of the elements are added to an $n \times n$ nodal matrix according to their nodal connections. The matrix is technically an indefinite admittance matrix, a Y matrix where all the node voltages are referenced to a common ground point. The process is completely mindless, perfect for a computer. The pattern for any element is sometimes called a *stamp*; the idea is that setting up the circuit equations is simply a process of "stamping" the matrix with the element values. Figure 7 shows an example of a circuit and the stamp of one of its elements.

Nodal analysis requires decomposing an $n \times n$ matrix, where n is the number of nodes. Solution of the circuit equations requires decomposing this matrix, not a difficult task if it is small but one that rapidly becomes larger with increased circuit size, as the number of operations, for standard methods, scales as n^3. Furthermore, the nodal matrix is very sparse, so much of that manipulation involves multiplying zero by zero and adding it to zero. That's not an efficient thing to do, and it took some time before efficient sparse-matrix techniques, which largely eliminated this nonsense, became available. Until then, the efficiency of the block-interconnection formulation made it the favored method, when applicable, especially for small computers.

In 1978, I was one of the first in my lab at Hughes Aircraft Co. to use COMPACT. I used it primarily for designing the IF filters and matching circuits for millimeter-wave mixers. Previously, the accepted method for realizing IF circuits involved a classical filter design, usually straight out of Matthai et al. [12]. Analysis by computer showed the limitations of using an unmodified filter design: while it certainly resulted in an IF circuit that would reject the RF and local oscillator (LO), it generally did not provide optimum terminations to the diode at all the important mixing frequencies. Achieving those terminations required a fairly complex optimization process; there was no analytical way to do such a design. At the same time, the idea of using the computer for microwave design in general and optimization in particular did not sit well with some of the older, experienced, but hidebound engineers. It was met with quite a bit of skepticism; one even called the optimization process an "idiot search" and claimed that competent engineers didn't need it.



$$\begin{bmatrix} I_2' \\ I_3' \end{bmatrix} = \begin{bmatrix} \frac{1}{R} & -\frac{1}{R} \\ -\frac{1}{R} & \frac{1}{R} \end{bmatrix} \begin{bmatrix} V_2 \\ V_3 \end{bmatrix}$$

Figure 7. *For nodal analysis, a circuit's nodes are numbered and each element's value is added to the nodal matrix in the form of a submatrix called a rom. As an example, the stamp for the resistor R is shown in the figure. Grounds are always node 0. Note that this circuit cannot be described as in Figure 6.*

In 1980, Compact Engineering merged with Comsat Corp. The new company was called Comsat General Integrated Systems. COMPACT eventually developed into a generalized nodal program called Super-COMPACT, which included a simple layout program called Auto Art and ran on a number of mainframe and minicomputers.

Around 1983, Charles Abronson and Bill Childs created a new company called EEsof, whose first product was a linear simulator called Touchstone. Touchstone ran on the new IBM PC, whose architecture eventually became a standard for personal computers worldwide. Touchstone was a full nodal program and included interactive graphics consistent with the capabilities of the PC. It also included a built-in full-screen text editor and used a format for S parameter files that has since become a de facto standard. This allowed it to import files generated by network analyzers. One of its most attractive features was an ability to tune variables in the network description file simply by tabbing between them. As each parameter was selected, it was highlighted and could be easily modified from the keyboard.

The 8080 processor used in the IBM PC was limited to a little over 1 MB of addressable space, and its MS-DOS operating system allowed only 640K memory for both the program and data. This quickly became insufficient for many kinds of application software, so rather clumsy techniques (and often extra hardware) were needed to address additional memory. These slowed computational speeds considerably. This problem continued through the early versions of the Windows operating system and was not corrected until 32-b processors, with larger address registers, became available in the mid-1990s. For this reason, much design work in the 1980s moved to engineering workstations: graphically oriented desktop computers using the UNIX operating system. Those were much more powerful and had fewer limitations.

In response to competition from EEsof, Super-COMPACT was ported to the PC. Even so, the developers had difficulty keeping up with the competition, and in any case, a software company did not fit well with Comsat's mission. Thus, in 1985 the company was sold to U. Rohde and relocated to Paterson, New Jersey. It grew to more than 100 employees, and in 1997 was sold to Ansoft Corp., until then a company that exclusively produced electromagnetic simulators. Ansoft renamed the circuit simulator Ansoft Designer.

Another PC-based program was Super Star, from a company at first called Circuit Busters, a name that was wisely changed to Eagleware. Although it included some nice features such as filter synthesis and tightly integrated layout, it never achieved a large market share and was eventually purchased by Agilent Technologies.

The 8080 processor used in the IBM PC was limited to a little over 1 MB of addressable space, and its MS-DOS operating system allowed only 640K memory for both the program and data.

Nonlinear Analysis

1971, SPICE, written at the University of California, Berkeley, was released free of charge to the public. Unlike COMPACT or Touchstone, which were formulated in the frequency domain, SPICE was a time-domain program. While microwave programs provided only steady-state, frequency-domain solutions, SPICE's time-domain integration provided, primarily, a circuit's transient response. This was easy to do with *LCR* circuits, but including transmission lines was difficult, and including S, Y, or Z parameters was seemingly impossible.

While SPICE had major limitations for use with microwave circuits, it was still used to some degree for microwave design. Transmission lines could be approximated by *LC* ladder circuits, and lumped-element, nonlinear device models supplanted S parameters. SPICE was used in certain applications in which there was simply no alternative, such as the simulation of oscillator start-up transients and certain kinds of power amplifiers.

SPICE's pedigree has never been entirely clear to me. It seems to have evolved partly from a course taught by R. Rohrer at the University of California at Berkeley in the late 1960s [13]. The version that was released, however, came from the Ph.D. dissertation of Larry Nagel, who, along with Donald Pederson, his adviser, is viewed as its creator [14]. Pederson received the IEEE Medal of Honor in 1998 for SPICE's development, although it was a surprise to many of us that Larry Nagel was not a corecipient.

SPICE underwent additional development. In 1975, SPICE2 was released, offering a number of improvements. Version 2G6, released in 1983, was the last Fortran version. Subsequently, SPICE was completely rewritten in the C language and was released in 1985. The change from Fortran to C made the program much more portable, and it has been used on a wide variety of platforms. SPICE3 is still available but is no longer in development.

The idea of harmonic-balance (HB) analysis had been in existence for some time before general-purpose HB simulators were developed. It appears to have been conceived almost simultaneously by Nakhla and Vlach [15] and Colon and Trick [16], although neither of these papers described applications to microwave circuits; they were concerned with rapidly finding steady-state conditions, often difficult with SPICE. The value of this type of analysis, for microwave circuit design, was made clear in papers by Egami [17] and Held and Kerr [18], who used very different formulations applied specifically to microwave mixers. Egami's approach was most like the harmonic-balance formulations we use today. It is surprising that it received relatively little attention when it was reported.

A third method that made a brief appearance was Volterra-series analysis [19]. Volterra-series analysis was most useful for intermodulation analysis of weakly nonlinear circuits. It integrated well with linear analysis (which is, in fact, a first-order Volterra method) but was not capable of meeting many ordinary design needs.

In the early 1990s, it was not at all clear whether harmonic-balance, time-domain, or Volterra simulation would dominate microwave design. While all had advantages and disadvantages, harmonic-balance analysis seemed the only method that, while not optimal in many respects, was adequate in virtually all. It therefore took its place among such engineering staples as alumina substrates, coaxial transmission lines, and many of our personal skill sets: not great at anything but adequate at almost everything. This is the stuff of which practical engineering is made, so it's no great surprise that HB became the dominant analytical method.

The first general-purpose harmonic-balance simulator (in the sense that, like SPICE, it could simulate an arbitrary circuit) was written at UC Berkeley by K. Kundert and A. Sangiovanni-Vincentelli [20]. It was Kundert's dissertation work. Originally it was called Harmonica, although the name was later changed to Spectre when it was learned that the name *Harmonica* had been trademarked. (Spectre has since been used as a product name by Cadence Design Systems.) The simulator's development was supported by Hewlett-Packard Co. (HP) and the State of California, under the MICRO program, which makes matching funds available for industry-supported research. Because of the MICRO funding, however, the source code became public domain and thus became the core of other, commercial simulators.

Harmonica/Spectre used SPICE models and an advanced sparse-matrix solver that was similar to the one in SPICE3. Like SPICE3, it used a modified nodal formulation, which differed from the classical, port-based formulation but allowed greater versatility. Although it could clearly be used for high-frequency circuits, the program seemed to be intended, like SPICE, more for analog ICs than for RF or microwave ones. Berkeley was, after all, a digital and analog IC school.

Commercial HB simulators were quickly released. These invariably ran on engineering workstations, as DOS and Windows machines simply did not have the speed and memory address space to support them. They included the following:

- EEsof released a simulator called Libra. It was reputedly based on the Berkeley code, although this has been denied by sources within the company.
- HP released its Microwave Design System (MDS). It was also based on the Berkeley code and originally was supported only on HP computers.
- Compact released a product called Microwave Harmonica, which was based on the work of V. Rizzoli at the University of Bologna [21].

Because of its lack of cross-platform support, MDS never got a strong foothold in the microwave design industry. In contrast, Harmonica and Libra ran on a variety of workstations, and as PC computers increased in capability, both products were ported to them as well. That left Libra and Harmonica to slug it out in the marketplace. The clear winner of that match was Libra, and Libra became the dominant tool in industry for designing microwave components.

HP acquired EEsof in 1993. Libra and MDS were merged into a product called the Advanced Design System (ADS). It became part of Agilent Technologies when the instrument and measurement division split off from HP. Eventually, Agilent also vacuumed up Eagleware and a start-up simulator company called Xpedion.

A new company entered the business in the late 1990s. Started by four circuit designers from Hughes Aircraft Co., Applied Wave Research (now AWR Corporation) in 1998 released its first product, a planar electromagnetic simulator. It eventually offered tightly integrated electromagnetic, system, linear, and nonlinear circuit simulation. Microwave Office, its flagship product, differed from earlier simulation systems by combining advanced software technology with advanced simulation technology. This resulted in a product that was user-friendly and supported an efficient design flow. The company grew rapidly and acquired substantial market share from its competitors. AWR became a wholly owned subsidiary of National Instruments in 2011.

Electromagnetic Analysis

The evolution of electromagnetic software products is even more complicated and extends over a shorter period of time than that of circuit tools. Part of the reason is the tendency of seemingly every Ph.D. graduate in electromagnetics to start an electromagnetic (EM) simulator company. A seemingly uncountable number of these have surfaced and submerged; in the future, there are certain to be many more.

As a minimum, however, three companies should be mentioned, largely because of their long-term success and influence on the way we do things. Sonnet Software, founded in 1983 by Jim Rautio, is the first. Sonnet markets a 3-D moment-based simulator for predominantly planar circuits. (It has become stylish to call these "2.5-D simulators." Since partial dimensions, in this context at least, make no sense, I have always called them "3-D, predominantly planar" simulators, or "3-D planar" for simplicity.) This simulator has been Sonnet's sole product, which has allowed it to focus on implementing advanced capabilities and solid software reliability. While technically a stand-alone simulator, Sonnet's product integrates nicely with AWR's Microwave Office through a COM interface; it also works with ADS. The second company is Ansoft, which we mentioned earlier as the final owner of Compact Engineering. Ansoft markets both planar and full 3-D tools. The third company is Zeland Software, started in 1992, which also provides a simulator called IE3D that is also 3-D planar. Ansoft was acquired by Ansys in 2008, and Zeland became part of Mentor Graphics in 2010. Sonnet is still privately held. Sonnet was the original planar 3-D program, and Jim Rautio won the IEEE MTT Society's Application Award in 2001 for its invention.

EM simulation is numerically intensive and memory-hungry, so its practicality depends strongly on advances in computer technology. For this reason, EM simulation was not integrated with circuit simulation in any meaningful way until the late 1990s. Before that, critical parts of a circuit would be EM-simulated on a separate machine, usually network-mounted, and S parameters for the circuit structure would then be imported into the circuit simulator. Even as computer technology improved, the simulation of even a few simple EM structures from a component design required far more computational effort than its circuit simulation and optimization. Successful integration depended on avoiding unnecessary resimulation of structures that had not changed. The simulator had to retain sophisticated dependency information for all its parts so it could mark the parts it had to resimulate when any particular change was made. This was a burden for the software technology and had little to do with the simulator's computational "engine" itself. It required careful thought for the software design of the system; a kludged-together collection of dissimilar simulators couldn't provide that kind of functionality.

Integration

The products we have been discussing are sometimes called "point tools": single-purpose software that stands alone and is used for a single purpose: electronic design, layout, EM simulation, filter synthesis, matching-network synthesis, or something similar. In the days when a microwave component consisted of a single stage, a set of disconnected point tools was probably adequate. As designs became more complex, however, it led to a clumsy design process. For example, not knowing exactly what the circuit layout would look like while the component was being designed, the engineer could easily find himself with a circuit that couldn't be fabricated.

It worked like this: you design the circuit. You made a layout sketch and gave it to the layout tech. A couple hours later, you got a phone call and were asked to come to the layout lab (a dark, forbidding, hot, underventilated room lit only by the multicolored glow of large CRT displays), where the tech showed you that the layout just wouldn't work. You returned to your office, brought up the design on the computer, changed microstrip dimensions as necessary, and discovered that the circuit no longer worked. You reoptimized and then saw that a critical, EM-simulated part had changed dimensions. You put the EM simulation into the queue, got the results a day later, and optimized the circuit yet again. Then you modified the sketch and returned it to the layout tech. An hour later you got another phone call and discovered that you had to cycle through the process at least once more. It's a miracle that any work got done.

This description isn't an exaggeration. I've been forced into it many times, and I suspect that almost everyone reading this has, as well. A way was needed to integrate the "point tools" so that the effect of a circuit change in, say, layout was immediately clear and potential problems could be avoided early. This meant, in effect, that the design process had to be concurrent, not sequential. That is, the layout, EM simulations, and even the system simulations had to be performed more or less simultaneously. To achieve that, integration among those point tools was necessary.

Early integration efforts involved the creation of supervisory software that passed data, with any necessary conversions, among the tools. This created an extra layer of complexity that slowed the design process and was error-prone. A better method required an entirely new architecture, not just a kludge of the existing pieces.

Fortunately, the computer technologies necessary for an improved architecture existed. Knowledge of those technologies resided in the brains of computer-science specialists, however, not (generally) in those of microwave engineers. It was essential to elevate the software system design to the same level as the simulator engine design. This was a new—and perhaps upsetting—concept for many engineering software specialists. What are those technologies? Here are two:

- **Object-oriented design.** Object programming involves the creation of *objects*, elements that include all their data along with the functions necessary to manipulate those data. Objects can inherit other objects, providing a capability for code reuse and improved reliability, as well-verified code can be employed in a number of places. Object-oriented design simplifies the design and maintenance of complex software systems, allowing greater versatility, as objects can be used in a multiplicity of ways. For example, suppose an HB simulator is configured as an

The best use of technological history is to show us where we are and in what direction we're moving.

object. Another simulator, say, a system simulator, can instantiate it, use it to calculate necessary data for a system model, and then dispose of it. It's frightening to imagine how that would be accomplished through some kind of supervisory software.

- **Component software.** For many years, computer scientists have been concerned with ways to interchange data among processes. Early examples were pipes in UNIX and Dynamic Data Exchange (DDE) in Windows. The holy grail of such efforts has been to develop methods for integrating software at the object level, however, not simply for passing data among processes. The idea was that two programs created entirely separately (i.e., by different individuals or even different companies) should be able to interconnect and operate as if they were a single entity. A Windows technology for this kind of integration is called Component Object Model (COM). It is closely related to the old idea of object linking and embedding. When you embed, say, a Word document in a spreadsheet, you are using COM to do that. Much of the Windows operating system is built on COM technology; it is now being used in simulation systems, too. In its best realization, component software allows a user to select a desired set of simulation tools, perhaps from a variety of vendors, and create from them a seamlessly functioning simulation system that supports the user's own, particular design flow. The user creates the design flow, rather than having the software force the user into one that may or may not meet the user's needs.

Whatever technologies are used, the trend toward greater integration will continue. This requires that we become accustomed to an environment that is somewhat foreign, however: engineers and computer scientists working together. This requires adaptation on both sides. Software specialists at last need to consider and understand the needs of science and engineering, and microwave engineers must cede some control of their products to the computer guys. This is a clear trend in engineering design software, and following it is the only way we'll continue to make real progress.

Trends

The best use of technological history, as I intimated at the start of this article, is to show us where we are and in what direction we're moving. In view of this, I think a few observations are in order.

- **Speed versus versatility.** Early in the development of these technologies, computational speed was usually the only characteristic of a simulator anyone cared about. The methods we now use, however, are not necessarily the fastest, computationally, but are instead the most versatile. Still, they are fast enough, partly because of advanced computer technology but also because of well-conceived and well-programmed numerical methods. It does us no good to use a fast method that can't analyze some of our circuits. One example is the move from cascade formulations to nodal ones. Nodal analysis is slower than block analysis but necessary to accommodate all kinds of circuits. Indeed, HB simulation today uses a nodal formulation instead of the port formulation of classical methods. This increases the problem size but avoids problems in handling certain kinds of circuits, particularly ones having a large number of nonlinear elements and relatively few linear ones. With modern analytical methods, the nodal formulation can be made as fast as the port one.
- **Change in architecture to favor speed over memory use.** When many of today's simulators were conceived, memory was expensive. As a result, those simulators were designed to save memory at the expense of computational speed. Today, memory is cheap, and it makes sense for new architectures to increase speed by using more memory; for example, by caching results instead of recomputing them. This kind of change is difficult to accomplish by modifying existing software, as it is fundamental to the software system's design. It requires new development.
- **Blurring of the distinction between circuit and system simulation.** Time-domain system simulators are now quite sophisticated, accounting for interface VSWR, feedback, and similar properties. They can automatically invoke a circuit simulator to calculate the parameters of a system model, in a sense including that circuit model in the system simulation. I expect that the system and circuit-simulation architectures will become more and more tightly integrated over time. Indeed, circuit simulation may turn out simply to be a subroutine of system simulation.
- **Blurring of the distinction between time- and frequency-domain simulators.** Techniques such as envelope analysis allow HB simulators to treat nonperiodic signals, while shooting methods allow time-domain simulators to find the steady-state response of a circuit quickly. Frequency-domain models can now be included in time-domain analysis. It seems likely that this kind of interplay will lead to hybrid methods that exploit the best parts of each method.

- **Standardization.** Currently there exist some significant efforts toward standardization and integration of various kinds of simulators. I suspect we will see more of this, as software company managers recognize that they can best sell software by offering customers software that works with competitors' products instead of software that is an "all-or-nothing" deal. Model standardization, another matter of great importance, is actually occurring as well. One example is the series of Berkeley Short-Channel IGFET Model (BSIM) MOSFET models, intended to be a standard for CMOS technologies. Similar efforts for other kinds of devices (e.g., involving the Compact Model Council) have been pursued for many years.
- **Ease of use.** Microwave design 40 years ago was not nearly as intense as it is today. The engineer had plenty of time to design a component and could spend hours in the lab getting it working really well. Those days are over. Today, one person may have to design a multicomponent chip single-handedly, and time-to-market pressure may leave little time for the job. Complex designs create a large amount of data, and thus the project is as much a data management challenge as a technical one. The fastest simulator system in that environment is not likely to be the one with the greatest computational speed but the one that places the necessary data before the user in the simplest and quickest manner.
- **Development models and design flow.** Much of the evolution in software technology has been driven by changes in the way that hardware has been developed. We no longer make approximate designs and "tweak" them into existence. We are more likely to design a circuit, as an IC, without ever setting foot in a lab, not even seeing our creation until it is complete, if at all. Software has had to adapt to this new reality; in short, it has had to accommodate a very large change in the ordinary design flow. It will have to adapt further, in the future, as that design flow continues to evolve.

Conclusions

Sometime around 1970, the way we create microwave products began to change inexorably from a kind of technological cottage industry into a complex industrial technology. That process continued through the years, and we can now design circuits and systems with a degree of accuracy and efficiency that seemed unimaginable only a couple of decades ago. Instead of tweaking in the dark, we make ICs that work; "first-pass success" is the term used, and it is regularly achieved. An essential for this success has been increased software versatility. It may be surprising to note that versatility has driven software evolution as much as computational efficiency or numerical accuracy. As time goes on, we can expect this trend to continue. It will be fun to see where it all goes.

Around 1970, the way we create microwave products began to change into a complex industrial technology.

References

[1] J. G. Brainerd and T. K. Sharpless, "The ENIAC," *Proc. IEEE*, vol. 87, p. 1031, 1999.

[2] B. Jack Copeland, *The Secrets of Bletchley Park's Code-Breaking Computers*. London, U.K.: Oxford Univ. Press, 2010.

[3] R. W. Jensen, "Charge control transistor model for the IBM electronic circuit analysis program," *IEEE Trans. Circuit Theory*, vol. CT-13, p. 428, 1966.

[4] B. D. Roberts and C. O. Harbourt, "Computer models of the field-effect transistor," *Proc. IEEE*, vol. 55, p. 1921, 1967.

[5] F. H. Branin et al., "CAP II—A new electronic circuit analysis program," *IEEE J. Solid State Circuits*, vol. SC-6, p. 146, 1971.

[6] N. Balabian, T. A. Bickart, and S. Seshu, *Electrical Network Theory*. New York: Wiley, 1968.

[7] L. Besser, "A fast computer routine to design high frequency circuits," in *IEEE ICC Conf. Dig.*, 1970.

[8] P. Bodharamik, L. Besser, and R. Newcomb, "Two scattering matrix programs for active circuit analysis," *IEEE Trans. Circuit Theory*, vol. CT-18, p. 610, 1971.

[9] L. Besser, "1987 IEEE MTT-S International Microwave Symposium Keynote Address," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-35, p. 1105, 1987.

[10] L. Besser, "COMPACT—microwave circuit optimization through commercial time sharing," in *IEEE MTT-S Int. Microwave Symp. Dig.*, p. 711, 2008.

[11] L. Besser, F. Ghoul, and C. Hsieh, "Computerized optimization of transistor amplifiers and oscillators using 'COMPACT,'" in *Eur. Microwave Conf. Dig.*, 1973.

[12] G. Matthaei, L. Young, and E. M. T. Jones, *Microwave Filters, Impedance-Matching Networks, and Coupling Structures*. New York: McGraw-Hill, 1964.

[13] R. I. Dowell, "Hijacked by SPICE," *IEEE Solid-State Circuits Mag.*, Spring, 2011.

[14] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," *Univ. California, Berkeley, Tech. Rep. ERL-M520*, May 1975.

[15] M. S. Nakhla and J. Vlach, "A piecewise harmonic balance technique for determination of periodic response of nonlinear systems," *IEEE Trans. Circuits Syst.*, vol. CAS-23, p. 85, 1976.

[16] F. R. Colon and T. N. Trick, "Fast periodic steady-state analysis for large-signal electronic circuits," *IEEE J. Solid-State Circuits*, vol. SC-8, p. 260, 1973.

[17] S. Egami, "Nonlinear, linear analysis and computer-aided design of resistive mixers," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-22, p. 270, 1974.

[18] D. Held and A. R. Kerr, "Conversion loss and noise of microwave and millimeter-wave mixers," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-26, p. 49, 1978.

[19] S. A. Maas, "A general-purpose computer program for the Volterra-series analysis of nonlinear microwave circuits," in *IEEE MTT-S Int. Microwave Symp. Dig.*, 1988.

[20] K. S. Kundert and A. Sangiovanni-Vincentelli, "Simulation of nonlinear circuits in the frequency domain," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, p. 521, 1986.

[21] V. Rizzoli et al., "State-of-the-art harmonic balance simulation of forced nonlinear microwave circuits by the piecewise technique," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-40, p. 12, 1992.