Contents

Chapter 6

Computer-Aided Circuit Design

This chapter introduces Amateur Radio builders and experimenters to fundamentals of using computer-aided design (CAD) tools for RF design. These tools enable the hobbyist to harness the circuit-simulation power employed by professional electronic and RF engineers working in large-scale electronic and RF design and production for industry.

David Newkirk, W9VES, explores generic simulation tools capable of analyzing arbitrary circuit topologies. Dr Ulrich Rohde, N1UL, surveys issues associated with linear and nonlinear RF simulation and contributes three extensive papers on the accompanying CD-ROM. Dale Grover, KD8KYZ, presents a comprehensive introduction to the use of PCB design and layout software.

Software to aid design and analysis in specialized areas, for example RF filters, switchmode power supplies, transmission lines and RF power amplifiers, is covered in other chapters.

6.1 Circuit Simulation Overview

That computers are now used wherever information is generated, communicated and used is everyday knowledge. We expect, or at least can infer if we take a moment to think, that a computer was used to type and typeset these words, and to compose the pages of this book and to design its cover. Thinking a bit more deeply, we realize that computers are used in just about every phase of product design and process design and control, including the electronic devices we use at work and play. The widespread application of computers is possible because computers are, at base, generic math machines. The TV set you watch, the radio transceiver which with you talk to the world, is a math machine — math put to work in the form of a kinetic sculpture built from copper, carbon, plastic, silicon and steel. You feed radio (or audio) and ac-line energy — both of which can be readily described numerically — in. The TV or radio performs mathematical operations on those inputs, and a desired, quantifiable result comes out: a sound, a moving picture, a Field Day contact, a declaration of rook to queen's bishop three from a chess opponent in Spain.

Because radio-electronic circuitry "just does math," math can predict and analyze the action of electromagnetic signals and the radio-electronic circuitry we build to produce and process them. Program an electronic computer — which, at base, is a generic math machine — to do radio/electronics math in practically applicable ways, and you're ready to do computer-aided design (CAD) of radio and electronic circuits. (Program a computer to do radio-electronics math *in real time*, and you're ready to replace radio-electronics hardware with software, as this book's **DSP and Software Radio Design** chapter describes.)

6.1.1 Hobby Circuit Simulation versus Circuit Simulation in Industry

Professional-grade circuit simulation software exists to facilitate the construction of tightly packaged, highly integrated, no-tweaking-required modern electronics/RF products. These products work predictably well even when reproduced by automated processes in large quantities — quantities that may, with sufficient marketing success and buyer uptake, exceed millions of units.

Manufacturers of specialized *electronic design automation* (*EDA*) software serve the engineering needs of this industry. Through comprehensive CAD suites, one may proceed from graphical component-level circuit and/or IC design (*schematic capture*), through simulation of circuit and IC behavior (often using a variant of the simulator called *SPICE*, but increasingly with non-*SPICE* simulators more fluent in issues of RF and electromagnetic design), through design of PC board and IC masks suitable for driving validation, testing and production. Comprehensive EDA CAD reduces costs and speeds time to market with the help of features that can automatically modify circuits to achieve specific performance goals (*optimization*); predict effects of component tolerances and temperature on circuit behavior across large populations of copies (*Monte Carlo analysis*); and generate bills of materials (BOMs) suitable for driving purchasing and procurement at every step of the way.

Free demonstration versions are available for some EDA CAD products (see **Table 6.1**), and a subset of these are especially useful for hobby purposes. Although these *demoware* tools come to us with a large-scale-production pedigree, they are greatly (and strategically)

Table 6.1
Some Sources of Freeware/Demoware Simulation, Schematic and Layout CAD Software

Source	*Address*	*Resource*
Ansoft	**www.ansoft.com**	*Ansoft Designer SV 2* (schematic, linear RF simulator, planar electromagnetic simulator, layout [PCB] design), more. (Students version SV2 has been discontinued in favor of an academic licensing program, but older copies of the program may still be abailable.
Cadence Design Systems	**www.cadence.com**	*OrCAD 16* (schematic, *SPICE* simulator, layout [PCB] design)
CadSoft	**www.cadsoft.de www.cadsoftusa.com**	*EAGLE* schematic and layout design
gEDA	**www.gpleda.org**	GPLed suite of electronic design automation tools
Kicad	**iut-tice.ujf-grenoble.fr/kicad**	GPLed full-function schematic and layout design
Linear Technology Corp	**www.linear.com**	*LTSpice* (schematic, *SPICE* simulator enhanced for power-system design)

SPICE: What's in a Name?

SPICE — Simulation Program with Integrated-Circuit Emphasis — originates from the Electrical Engineering and Computer Sciences Department of the University of California at Berkeley and first appeared under its current name as *SPICE1* in 1972. *"SPICE"*, write the maintainers of the official *SPICE* homepage at **bwrc.eecs.berkeley.edu/Classes/icbook/SPICE/**, "is a general-purpose circuit simulation program for nonlinear dc, nonlinear transient, and linear ac analyses. Circuits may contain resistors, capacitors, inductors, mutual inductors, independent voltage and current sources, four types of dependent sources, lossless and lossy transmission lines (two separate implementations), switches, uniform distributed RC lines, and the five most common semiconductor devices: diodes, BJTs, JFETs, MESFETs, and MOSFETs."

That *SPICE* is "general-purpose" does not mean that its usefulness is unfocused, but rather that it is well-established as a circuit-simulation mainstay of comprehensive power. A wide, deep *SPICE* ecosystem exists as a result of decades of its daily use, maintenance and enhancement by industrial, academic and hobby users. Many excellent commercial versions of *SPICE* exist — versions that may be improved for workhorse use in particular subdisciplines of electronics, power and RF design.

Throughout this chapter we will use a particular commercial variant of the *SPICE* simulator: *PSpice* as embodied in Cadence Design System's *OrCAD 16.0* EDA CAD demonstration software suite. With those particulars declared, we will usually refer to the underlying simulator merely as *SPICE*, expecting and intending that simulations presented in this chapter will be duplicable with any modern commercial *SPICE* variant. Only details associated with value-added features, such schematic and layout editing and simulation-results reporting, will vary.

CAD Software and Your Computer's Operating System

The *OrCAD 16.0* and *Ansoft Designer SV 2* demoware packages used in this chapter, and most other CAD products you're likely to use, are compiled to run under Microsoft *Windows*. So what if you want to run RF and electronics CAD software under *Linux* or on the Mac?

You're in luck. *EAGLE* schematic and layout software is available in native versions for *Windows*, *Linux*, and the Macintosh. The GPLed EDA application suite, *gEDA*, are primarily developed on *Linux*, but are intended to run under, or at least be portable to, *POSIX*-compliant systems in general. The GPLed schematic and layout editor *Kicad* runs natively under *Windows* and *Linux*, and has been tested under *FreeBSD* and *Solaris*. Further, the great strides made in the *Wine* translation layer (**www.winehq.com/**) allow many applications written for *Windows* to run well under the operating systems supported by *Wine*, including *Linux* and the Macintosh.

In your writer's experience, MicroSim *DesignLab 8* (a widely distributed precursor to *OrCAD 16* that can run all of the *SPICE* examples described in this chapter) and Ansoft *Serenade SV 8.5* (a precursor to *Ansoft Designer SV 2*) can be run in *Wine* under *Linux* with few artifacts and their expected schematic-capture and simulation capabilities intact. Cursor handling in *OrCAD 16* installs under *Wine* readily enough. but cursor-handling artifacts in its schematic editor, at least in the computers tried, seems to preclude its use under *Wine* for now. *Ansoft Designer SV 2* installs but does not properly start.

All things considered, however, especially as *Wine* and CAD applications continue to strengthen and mature, running your favorite *Windows*-based applications under *Wine* is well worth a try.

feature-limited. Only a relatively few components, often representing only a subset of available component models, may be used per simulation. Monte Carlo analysis, optimization, BOM generation and similar enhancements are usually unavailable. Demoware is intended to drive software purchasing decisions and serve as college-level learning aids — learning aids in college study toward becoming electronics/RF professionals who will each day work with the unlimited, full versions of the demoware.

The radio hobbyist's circuit-simulation needs are much simpler. Most of us will build only one copy of a given design — a copy that may be lovingly tweaked and refined to our hearts' content far beyond "good enough." Many of us may build as much with the intent of learning about and exploring the behavior of circuits as achieving practical results with them. A demoware circuit simulator can powerfully accelerate such self-driven exploration and education in electronics and RF.

6.1.2 Real Circuits versus Simulations: The Difference

Having just finished stuffing and soldering the PC board for a binaural CW receiver, you find yourself in need of a 7-MHz local oscillator (LO) to drive it. The receiver designer has provided just such a circuit: a junction field-effect transistor (JFET) Hartley oscillator driving a JFET amplifier to an output power of +10 dBm. You obtain the parts, perhaps substituting components with close-enough values here and there as your junk box suggests and your experience allows. Having completed its construction, you interconnect it, the main receiver board, a power supply, headphones and an antenna. You power up the receiver, hear 40-meter signals, and —

perhaps with the aid of a frequency counter or second receiver — adjust the tuning range of your new LO to cover your favorite part of the band. Another project done, another success achieved — what could be simpler? Certainly not modeling the same circuit in circuit-simulation software — precisely because all actions in the real world are always *maximally and unimaginably complex*.

The components we use to build real circuits always operate to the full robustness of their intrinsic properties, their simplicity to our mind's eye notwithstanding — regardless of our relative ignorance of what those properties may be and how and why they operate the way they do. No real-world component operates ideally. Nothing — excepting, perhaps, events in the realm of quantum physics — is sketched in, so far as science has been able to determine. So it is that we may set out to build an amplifier only to discover at power-up that we have instead built a persistent oscillator. So it is that we may design, successfully build, and publish a circuit intended to be an oscillator only to discover that for 3 out of every 100 subsequent reader-builders it does not oscillate at all!

When we take the "just build it" approach with circuit simulation software — that is, when we expect components to "just work" as they do in real-world circuits — we may run into trouble rapidly. This can happen because the electrical and electronic components available in circuit simulators are only mathematical *models* of real-world components — because every modeled behavior of every modeled component is *only sketched in* relative to the behaviors of its real-world counterpart. Simulated component characteristics and behaviors approach those of real-world components only as closely as science may allow *and* only as closely as the simulator's designers, engineers and users need its simulations to mirror real-world behaviors to meet their design goals. Once a design works well enough, reproducibly enough, cheaply enough, it goes into production, ultimately to ship in the hundreds, thousands or millions.

Finalizing an Amateur Radio circuit design is less spectacular in comparison. Even if the designer of our figurative VFO provided a quantitative specification for its performance — +10 dBm output — we very likely will not ultimately confirm whatever numbers we have before happily putting the project into service. We're hearing signals and making radio contacts — the likely object of the exercise, after all — and so we consider our construction project successful.

Were this chapter a textbook, or part of a textbook, on computer-aided circuit design, we might begin our CAD explorations by reviewing the basics of what electronic circuits are and do, following this with a discussion of what computerized circuit simulation is and how it works. An excursion into the arcane world of active-device modeling — the construction and workings of mathematical electrical equivalents to the transistors, diodes and integrated circuits that await us at our favorite electronics suppliers and in our junk boxes — might follow. Finally, we might systematically proceed through a series of simulation examples from the basic to the more complex, progressively building our store of understood, trustworthy and applicable-to-future-work circuit-CAD concepts as we go.

But this is a chapter in a handbook, not a textbook, and following a sequence of abstract basics to concrete practice very likely does not reflect the process most of us have followed, and follow yet, in learning and using what we know about electronics and radio. More realistically, our approach is more like this: We find ourselves in need of a solution to a problem, identify one, and attempt to apply it. If it works, we move on, likely having learned little if we have not had to troubleshoot. If the solution does *not* work, we may merely abandon it and seek another, or — better, if we are open to learning — we may instead seek to understand why, with the happily revised aim of understanding what we need to understand to make the solution work. Even if we must ultimately abandon the solution as unworkable in favor of another, we do not consider our time wasted because we have further accelerated our deepening intuition by taking the initiative to understand why.

This chapter introduces circuit CAD in exactly that spirit: Especially at first, we will learn about circuit simulation through "just building," and simulating with CAD, *buildable circuits we know should work* in quantified and quantifiable ways that we also know well. If they don't work as they should, we will investigate why, fix them and review what we have learned — just as we do when building and learning from real-world hardware.

6.2 Computer-Aided Design Examples

6.2.1 Example 1: A Two-JFET VFO and Buffer

In introducing this chapter, we "just built" — in our mind's eye — a receiver LO that "just worked," as real-world projects are intended and expected to do. Building and simulating the same two-JFET design in the circuit simulator called *SPICE* transports us right to the heart of fundamental circuit-modeling problems and their solutions. If we build through graphical schematic entry — through schematic capture — its oscillator-buffer circuit exactly according to its electrical schematic, *it cannot possibly work!*

Fig 6.1 shows the circuit for real-world duplication. In it, a J310 JFET Hartley oscillator drives a J310 buffer amplifier, which drives a 50-Ω load at +10 dBm via a trifilar broadband transformer that provides an impedance step-down ratio of 9:1. Any experimenter accustomed to building and using such circuits needs no more information than that provided in the Fig 6.1 schematic and its caption to make the circuit work as expected.

Fig 6.2 shows the same circuit successfully modeled in *OrCAD 16.0*, *SPICE* based simulator software from Cadence Design Systems. To understand the differences between Fig 6.1 and Fig 6.2, we'll examine the simulation's components type by type.

RESISTORS

The designer's output power specification (+10 dBm) assumes that the VFO is connected to the 50-Ω load afforded by the mixer system in the receiver it was designed to drive. To simulate this mixer load, we have added R5.

The value of R1, specified as 1 MΩ in the original circuit, is now specified as 1E6 — scientific/engineering notation for one million. We have done this to remind ourselves that *SPICE*'s use of unit suffixes — *scale factors* in *SPICE*-speak — differs from what we are generally accustomed to seeing in electrical schematics, and that we have multiple options for specifying values numerically using integer and decimal floating-point numbers. The scale factors available in *SPICE* include:

- F — 1E–15
- G — 1E9
- K — 1E3
- M — 1E–3
- MEG — 1E6
- MIL (0.001 inch) — 25.4E–6
- N — 1E–9
- P — 1E–12
- T — 1E12
- U — 1E–6

Specifying the value of R1 as 1M would declare its value as 1 milliohm (0.001 Ω), short-circuiting the JFET's gate to common and breaking our simulation. Specifying the

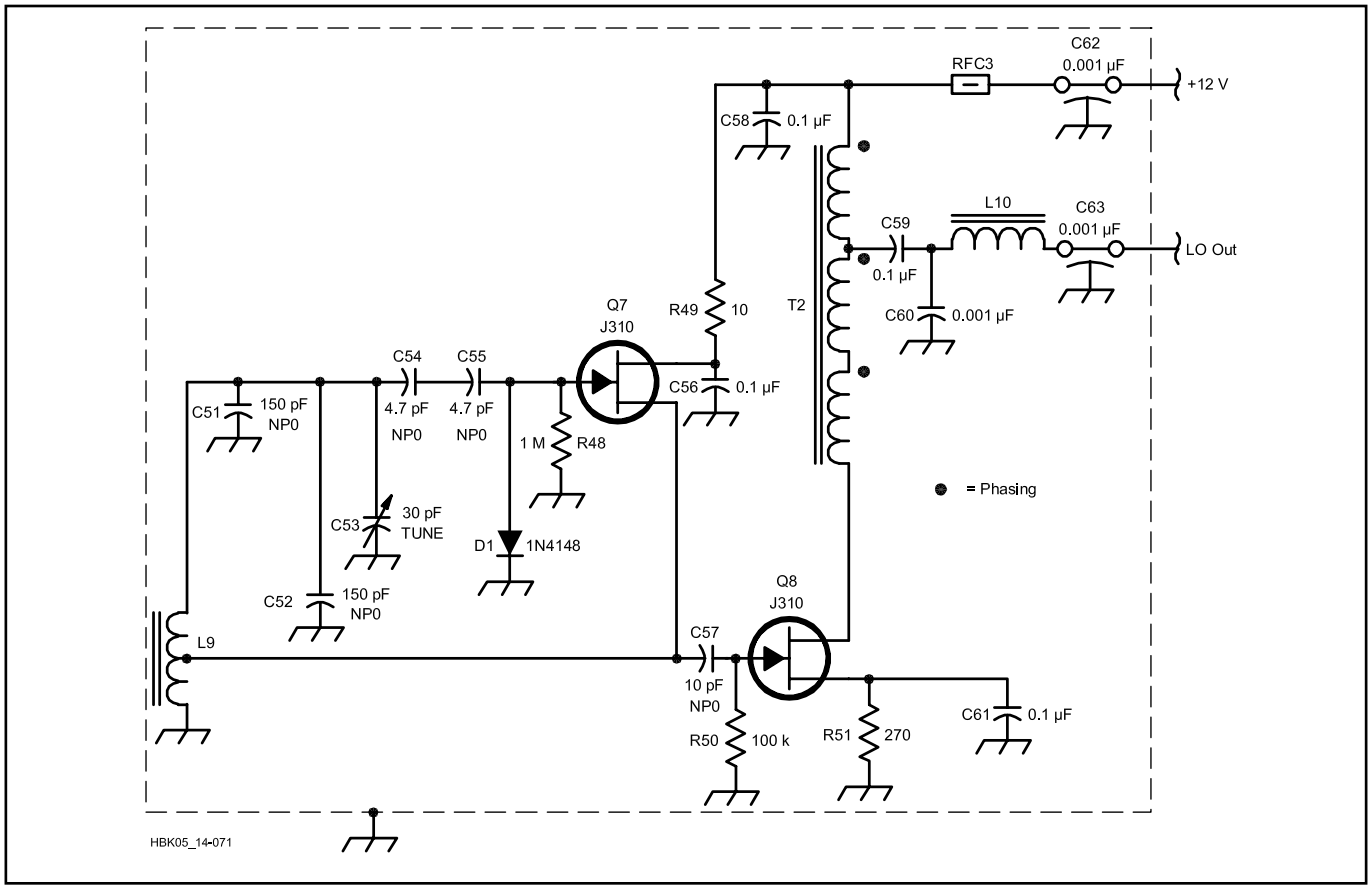


Fig 6.1 — Standard electrical representation of a 7-MHz VFO with buffer amplifier. JFET Q7 operates as a Hartley oscillator; D1, as a limiter that improves frequency stability by keeping Q7's gate voltage from going more positive than about 0.6 V; and Q8, as an amplifier that increases the oscillator output — obtained from the feedback tap on the oscillator inductor by capacitive coupling (C57) — to +10 dBm. The tapped inductor (L9), is 1.2 µH (22 turns of #28 wire on a T-30-6 toroidal core); the trifilar output transformer (T2), 10 trifilar turns of #28 wire on an Amidon FB-43-2401 ferrite bead).

Using Your Computer to Draw Schematics

The art of drawing circuit schematics predates electronic computers; the art of drawing schematics and PC-board layouts with computers predates the art and science of *simulating* circuits with computers. What if you only want to draw a circuit's schematic and design a PC board without simulating it? What computerized tools are available to *you*?

Almost any circuit-simulation program or electronic design automation (EDA) suite that uses schematic circuit capture can, within functionality limits imposed on the demoware version, serve as a first-rate schematic editor. Although demoware component library limitations usually restrict the *types* of components you can use — in CAD-speak, *place* — in a design, part-*count* limitations usually operate only at simulation time. Restrictions in physical size and layer count, and in the materials and metallizations available for substrate specification, will likely apply to whatever layout-design facilities may be available. After all, the main purpose of demoware is to let students and potential buyers taste the candy without giving away the store.

Excellent simulation-free schematic-capture and layout-design products exist, of course. The schematic style long standard in ARRL publications comes from the use of Autodesk *AutoCAD*, a fully professional product with a fully professional price. Long popular with radio amateurs and professionals alike is CadSoft *EAGLE*, a schematic-capture and layout-design product available in freeware and affordable full-version forms. You can even export *EAGLE* schematics to a *SPICE* simulator and back with Beige Bag Software's *B2 Spice*. The full-function freeware schematic and PCB-layout application *Kicad* and the EDA suite *gEDA* come to us from the open-source community.

Do-it-yourself schematic CAD can be as close as the basic drawing utility included with your computer's operating system. Some hobbyists find that cutting, copying, moving and pasting components snipped from favorite graphical schematic files into new configurations and adding new wires as graphical lines is enough for what they want to do. The connection between the world of modern EDA tools and this seemingly primitive approach to schematic creation can be as close as the operating system's clipboard: Some CAD schematic-capture programs represent circuit elements as metafile data during copying, cutting and pasting operations. Copying a schematic to the clipboard with such a program and then pasting the clipboard contents to a suitable drawing program creates a *picture* of the copied schematic — give it a try with *Windows Paint* and the schematic editor in the free demoware version of *OrCAD 16*.

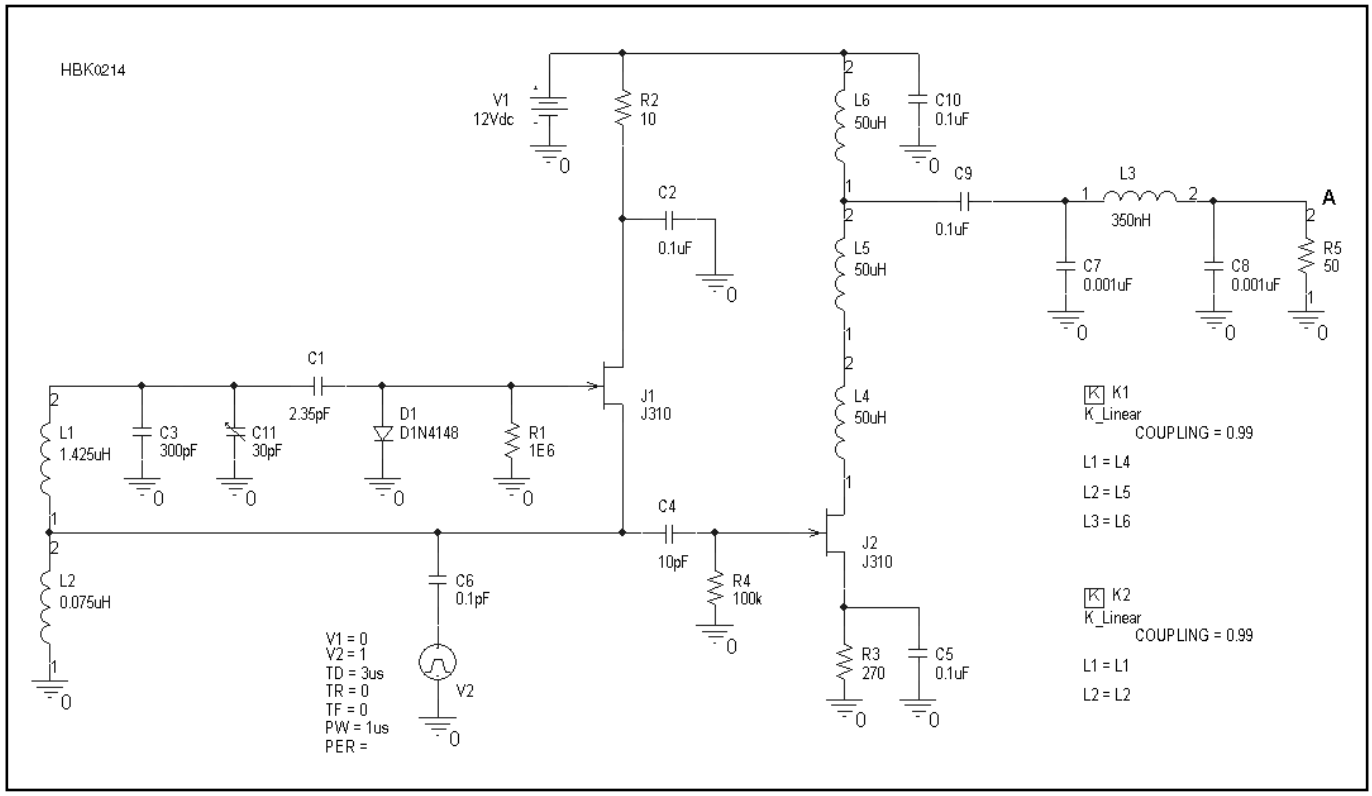


Fig 6.2 — The VFO-and-buffer circuit configured for successful *SPICE* simulation under the *OrCAD 16* CAD suite as drawn — "captured" — in the *OrCAD Capture CIS* schematic editor. To make the circuit work like the real thing, we added several components only implied in Fig 6.1, including a 12-V dc source (V1) and a 50-Ω load resistor (R5). Also new to the circuit, ac-coupled (via C6) to the oscillator JFET (J1) source, is a mysterious second voltage source (V2) — an addition without which the oscillator cannot oscillate. The numbers identifying the leads of each inductor are displayed by default in the *OrCAD* schematic editor to indicate phasing, knowledge of which is essential for properly modeling the behavior of the trifilar transformer formed by L4 through L6. We have also enabled pin-number display for R5 to help us determine the name of the circuit node — labeled A — we must probe to graph the circuit's output waveform.

value of R1 as 1MEG or 1000K would be correct alternatives. *SPICE* scale factors are case-insensitive.

Notice that *SPICE* assumes unit *dimensions* — ohms, farads, henrys and so on — from component-name context; in specifying resistance, we need not specify ohms. In parsing numbers for scale factors, *SPICE* detects only scale factors it knows and, having found one, ignores any additional letters that follow. This lets us make our schematics more human-readable by appending additional characters to values — as long as we don't confuse *SPICE* by running afoul of existing scale factors. We may therefore specify "100pF" or "2.2uF" for a capacitance rather than just "100p" or "2.2u" — a plus for schematic readability. (On the reduced readability side, however, *SPICE* requires that there be *no space* between a value and its scale factor — a limitation that stems from programming expediency and is present in many circuit-simulation programs.)

CAPACITORS

To develop the habit of keeping our simulated circuits' part counts down so we don't run up against the circuit-complexity restrictions of the *OrCAD 16.0* demoware (*all* demoware and student-version CAD packages have such limits) we have combined the seriesed and paralleled capacitor pairs in our real-world tuned circuit into single capacitances. Original C54 and C55 become C1; original C51 and C52, C3. That said, there are two reasons why we may not want to do so. If one of the objects of our simulation is to examine the voltage, current or power at the junction of C54 and C55 in the original circuit, combining them has disallowed achieving that aim. More generally, if we also intend to base a PC-board design on our captured circuit through *OrCAD 16.0 PCB Design Demo* — one of *OrCAD Capture CIS*'s sibling applications in the *OrCAD 16.0* demoware suite — we must engineer our simulation with that practical outcome in mind throughout.

INDUCTORS

Simulating the tapped coil of a Hartley oscillator immediately challenges us to learn more about our real-world circuit than we need to know to successfully build it. The original coil, 1.5 μH, consists of 22 turns of wire tapped at 5 turns, yet a tapped inductor is not available in the *SPICE* model library. Multiple approaches to simulating a tapped inductor can be used, including connecting two inductors in series (as we have done here) and proportioning their values intelligently, or basing the tuned circuit on one winding of an ideal transformer and proportioning the inductance of the secondary to simulate the tap. Either way, we must make the best educated guess we can about the inductance between the tap and ground, as its value directly affects the oscillator feedback, and hence its output.

Especially if your approach to building is more practical than theoretical, proportioning the values of the two coils in the 22:5 ratio reflected in the original's winding information might seem like a fair approximation — until we recall that a coil's inductance-versus-turns ratio is not linear. Taking that approach would give us a larger-than-life inductance value for the lower portion of the coil, resulting in more-than-realistic feedback and higher-

than-realistic output. So what we have done for this simulation is calculate the inductance of 5 turns of wire on a T-30-6 core, taking the answer (0.075 μH) as the value of Fig 6.2's L2, and 1.5 μH – 0.075 μH (1.425 μH) as the value of L1, the upper portion of the coil.

To illustrate another feature of *SPICE* — and to provide one avenue for later experimentation with this simulation — we have also specified near-ideal ($K = 0.99$) coupling between L1 and L2 by means of K_Linear element K1 (in the lower right corner of Fig 6.2). *SPICE* allows us to specify coupling between any subset of inductors in a simulation, including *all* inductors in a simulation. The value of this feature in enabling greater realism in simulations of complex, cross-coupled connector, circuit-board and IC structures is profound — at the expense of requiring the realistic specification of coupling values if the power of this feature is to be realized.

Here we have coupled the two sections of our oscillator tank inductor because (1) we know that they actually *are* coupled in the real thing; (2) we want to experience specifying inductor coupling in *SPICE*; and (3) practical experiments with Hartley inductors consisting of separate toroidal cores nonetheless shows that such coupling is *not* necessary to make real Hartley oscillators work! Assuming we can get the circuit to oscillate as is, reducing the coupling between L1 and L2 would let us simulate the use of separate coils in a real-world oscillator.

Having specified coupling between the sections of the oscillator tank inductor, we are ready to welcome the similar challenge of simulating the trifilar broadband output transformer (T2) in Fig 6.1. Here, as with the tapped oscillator tank coil, no direct equivalent to this transformer topology is available in *SPICE*, but multiple alternatives can get us close enough. One option would be to use a conventional two-coil transformer, such as that available in the *OrCAD Capture CIS* component library. As with the tapped oscillator coil, however, we have decided to use three separate coupled inductors, specifying their coefficients of coupling with another K_Linear element, K2. For the inductance of each, we have drawn on our experience with the "10 multifilar turns on an FT-37-43 core"-class broadband transformers commonly used for just such applications in many ARRL RF projects, specifying an inductance (50 μH) close in value to that of a single such winding for each coil of our simulated transformer. (As a check on the intelligence of using this value, we recall that the rule of thumb for the inductance of a conventional broadband transformer winding calls for a winding reactance of at least 5 to 10 times the impedance at which the winding operates — and the reactance of 50 μH at 7 MHz equates to 2.2 kΩ, over 40 × 50 Ω.) In wiring the three inductors (L4, L5 and L6), we have also taken care to phase them properly, their **1** and **2** labels conveying the winding-sense information communicated by the phasing dots that accompany T2's windings in Fig 6.1.

Specifying for simulation the remaining inductor in Fig 6.1 — in our Fig 6.2 schematic, L3, 350 nH, one of three components in a π low-pass filter network — is straightforward enough to warrant no comment. But comment we shall, for in specifying the electrical performance of an inductor merely by setting a value for its inductance — as we have so far done for all of the inductors in our circuit — we have in no way specified its quality factor (*Q*). In simulation it will there act as an absolutely pure inductance — a component that cannot be built or bought. This will make a difference in how closely our simulation may approach the real thing — but how much of a difference?

All *real* inductors, capacitors, and resistors — all real components of any type — are non-ideal in many ways. For starters, as **Fig 6.3** models for a capacitor, every real *L* also exhibits some *C* and some *R*; every real *C*, some *L* and *R*; every real *R*, some *L* and *C*. These unwanted qualities may be termed *parasitic*, like the parasitic oscillations that sometimes occur in circuits that we want to act only as amplifiers, and in oscillators (which may simultaneously oscillate at multiples frequencies, including the frequency we intend). For experimental and proof-of-concept purposes at audio and HF radio frequencies, parasitic L, C and R can often be ignored. In oscillator and filter circuits and modeled active devices, however, and as a circuit's frequency of operation generally increases, neglecting to account for parasitic L, C and R can result in surprising performance shortfalls in real-world *and* simulated performance. In active-device modeling realistic enough to accurately simulate oscillator phase noise and amplifier phase shift and their effects on modern, phase-error-sensitive data-communication modes, device-equivalent models must even include *nonlinear* parasitic inductances and capacitances — Ls and Cs that vary as their associated voltages and currents change. As active-device operation moves from small-signal — in which the signals handled by a circuit do not significantly shift the dc bias points of its active devices — to large-signal — in which applied signals significantly shift active device dc bias and gain — the reality of *device self-heating* must be included in the device model. Examples: When amplitude stabilization occurs in an oscillator or gain reduction occurs in an amplifier as a result of voltage or current limiting or saturation.

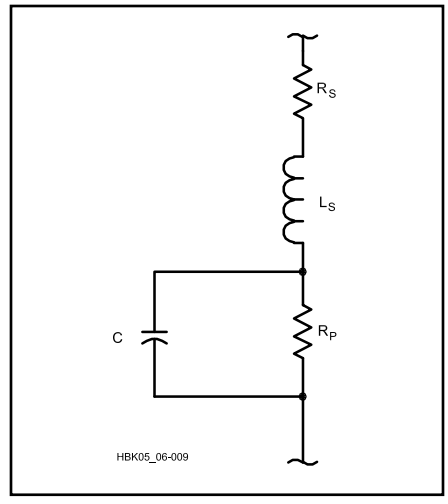


Fig 6.3 — A capacitor model that aims for improved realism at VHF and above. R_S models the net series resistance of the capacitor package; L_S, the net equivalent inductance of the structure. R_P, in parallel with the capacitance, models the effect of leakage that results in self-discharge. Intuiting the topology of this model is one thing; measuring and/or realistically calculating real-world values for R_S, L_S and R_P for application in a circuit simulator is a significant challenge. How and to what degree these parasitic characteristics may cause the electrical behavior of a capacitor to differ from the ideal depends on its role in the circuit that includes it and frequency at which the circuit operates. For simulating many ham-buildable circuits that operate below 30 MHz, the effects of component parasitic R, L and C can usually be ignored unless guidance or experience suggests otherwise.

Designers aiming for realism in simulating power circuits that include magnetic-core inductors face the additional challenge that all real magnetic cores are nonlinear. Their magnetization versus magnetic field strength (*B-H*) characteristics exhibit hysteresis. They can and will saturate (that is, fail to increase their magnetic-field strength commensurately with increasing magnetization) when overdriven. Short of saturation, the permeability of magnetic cores varies, hence changing the inductance of coils that include them, with the flow of dc through their windings. These effects can often be considered negligible in modeling ham-buildable low-power circuits (such as Fig 6.1), but designers using *SPICE* to simulate power supplies and electromechanical systems for mass fabrication and production must harness its ability to model nonlinear magnetics — a capability greatly limited in the demo version of *OrCAD 16.0*. See the **Power Supplies** chapter for more information on this topic.

So what of our non-specification of *Q* for the tuned-circuit inductors — and while we're at it, the tuned-circuit capacitors — in Fig 6.2? With all other factors ignored and with no steps taken to otherwise introduce realistic parasitics and losses into simulations, ideal tuned circuits will generally result in

higher-than-expected output in amplifiers and heavier clipping than we might expect (and therefore only *maybe* higher output than expected) in oscillators. Ideal transformers and *LC* filters will exhibit lower-than-expected losses and sharper-than-actual resonances. Whether this matters depends on our simulation aims. If we merely want to get an idea of the frequency response of a filter, ideal *L*s and *C*s are fine; if we want to compare the efficiencies of competing matching networks or filters or realistically model filter insertion loss, ideal *L*s and *C*s will lead us astray. So, if we want to more realistically simulate the output power of an oscillator, setting a realistic *Q* for at least its tuned-circuit *L* would seem to be a good thing.

Yet we have not yet done so in Fig 6.2. Because we are new to circuit modeling in general and this circuit in particular, and because against all odds we have chosen as our first exercise the modeling of an *LC* oscillator — and merely getting a *SPICE*-simulated oscillator to start can be enough of a challenge — we will work with ideal *L*s and *C*s for now. We will go into issues of specifying realistic *Q* later as part of our exploration into evaluating circuit gain.

SOURCES

We would expect Fig 6.2 to include a 12-V dc source, and it does (V1). Unexpectedly, however, our simulated circuit includes a second source: V2, which we have configured to generate a 1-V, 1-ns-long pulse that occurs 3 µs after simulation begins. We have included this pulse source because our simulated oscillator has a high-*Q* tuned circuit and, simulated in *SPICE*, cannot start oscillating without it.

Real oscillators need help starting, too. The difference between Fig 6.2 and its real-world equivalent is that real-world transistors in well-designed oscillators can usually get themselves started oscillating without our building in any means of kick-starting them.. A real-world *LC* oscillator can do this because its active device or devices generate internal noise, which, fed back and filtered by its tuned circuit, and amplified again and again, becomes less and less noiselike and more and more sinusoidal as it builds, until some mechanism of voltage or current limiting allows the signal to increase no further. That an oscillator can non-self-destructively reach and maintain this condition of *amplitude stabilization* is no less momentous than the occurrence of oscillation startup.

SPICE can simulate active-device noise, but only during ac circuit analysis, in which any active devices present are treated as linear — that is, as operating under small-signal conditions — before circuit behavior with ac signal input is evaluated. As an oscillator starts and then reaches amplitude stabilization, its oscillatory device(s) move from small-signal to large-signal operation. We must therefore use *SPICE*'s time-domain (also known as transient) simulation capabilities to simulate Fig 6.2 if we want to observe its output power. In time-domain simulation, *SPICE* progressively calculates the voltages at and currents through each circuit node — each point of interconnection between components — as the time steps of a simulation advance from the initial conditions at Time Zero.

Our simulation of Fig 6.2 will begin as every *SPICE* time-domain simulation begins if we do nothing to adjust the initial conditions for any of its components away from their defaults: All voltage and current sources will be at their specified levels, all capacitors will not yet be charged, and all inductors and transformers will not yet be energized. Starting such an analysis is very much like suddenly connecting a 12-V battery to the real-world circuit in Fig 6.1.

Because we want to see what happens when we "just build" a real-world circuit in a simulator, we will only mention in passing the availability of the advanced technique of explicitly specifying non-zero initial conditions of key circuit components as an aid to oscillator startup. This technique requires that we first build a high-*Q* oscillator, such as a crystal oscillator, as low-*Q*, get it going well enough to determine steady-state voltage or current values for key components, rebuild it as high-*Q*, and then re-simulate it with the steady-state values in place. That said, rather than first trying Fig 6.2 without V2 only to have it fail to start, we choose with the writer's help to magically learn from that certain failure in advance by kick-starting our simulated oscillator with a voltage pulse from V2. (We also include V2 [and C6] in Fig 6.2 from the get-go as a service to your memory: Introducing the circuit without these components and adding them later in a small, separate schematic would encourage your image-memory capabilities to snapshot a picture of a simulatable circuit *that cannot work*.)

DEVICES, DEVICE MODELS AND DEVICE PARAMETERS

In building a real circuit that uses active devices, we pull the necessary parts out of storage, solder them in, and they "just work." Like the rest of the components in the projects we build, they operate to the full robustness of the intrinsic properties of their constituents and construction regardless of our knowledge of the details. We can, and do, count on it.

Not (likely) being degreed practitioners of either general circuit-simulator mathematics or of the more specialized disciplines of device manufacturing and/or modeling, we will naturally tend to do the same when using a circuit simulator. Just as with real-world devices, once we click **Run** and simulation begins, our modeled devices will act to the fullest degree allowed by their construction. Very differently, however, *absolutely every desired behavior exhibited by a simulated device must be explicitly built into the model*, mathematical atom by mathematical atom. A real-world device always "knows" exactly what to do with whatever conditions confront it (however the resulting behavior may alarm or confound us). A simulated device can reliably simulate real-world behavior only to the extent that it has been programmed and configured to do so. A 1N4148 diode from your junk box "knows" exactly what to do when ac is applied to it, regardless of the polarity and level of the signal. Mathematically *modeling* the forward- and reverse-biased behavior of the real thing is almost like modeling two different devices. Realistically modeling the smooth transition between those modes, especially with increasing frequency, is yet another challenge.

Mathematical transistor modeling approaches the amazingly complex, especially for devices that must handle significant power at increasingly high frequencies, and especially as such devices are used in digital-communication applications where phase relationships among components of the applied signal must be maintained to keep bit error rates low. The effect of nonlinear reactances — for instance, device capacitances that vary with applied-signal level — must be taken into account if circuit simulation is to accurately predict oscillator phase noise and effects of the large-signal phenomenon known as *AM-to-PM conversion*, in which changes in signal amplitude cause shifts in signal phase. In effect, different aspects of device behavior require greatly different models — for instance, a dc model, a small-signal ac model, and a large-signal ac model. Of *SPICE*'s bipolar-junction-transistor (BJT) model, we learn from the *SPICE* web pages that "The bipolar junction transistor model in *SPICE* is an adaptation of the integral charge control model of Gummel and Poon. This modified Gummel-Poon model extends the original model to include several effects at high bias levels. The model automatically simplifies to the simpler Ebers-Moll model when certain parameters are not specified."

As an illustration of device-model complexity, **Fig 6.4** shows as a schematic the BIP linear bipolar junction transistor model from *ARRL Radio Designer*, a linear circuit simulator published by ARRL in the late 1990s. Luckily for those interested mainly in audio and relatively low-frequency RF applications, specifying values just for the parameters A (0.99 for average transistors) and RE (26 ÷ collector current in milliamperes, in which the *26* equates to 26 *millivolts*, the room-temperature value of V_T, the thermal equivalent of voltage in the transistor's semiconductor material) can suffice for good-enough-for-

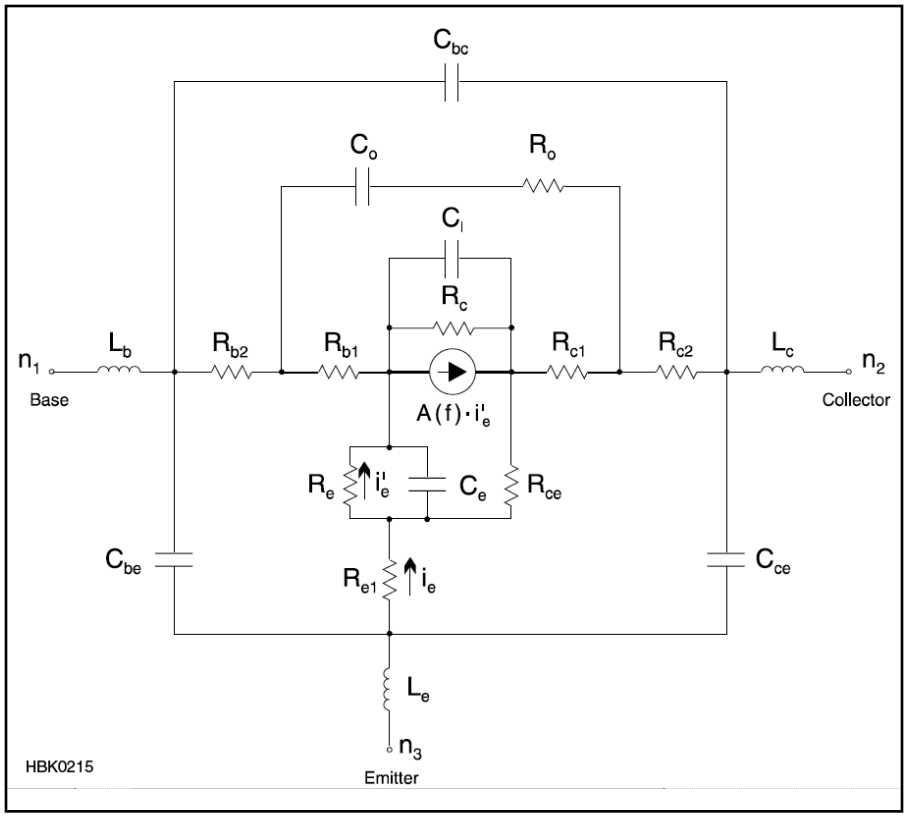


Fig 6.4 — The linear BJT model, BIP, from *ARRL Radio Designer*, a now-discontinued circuit-simulation product published by ARRL in the late 1990s. Frustratingly to users of *ARD*, real-world values for many BIP parameters could not be directly inferred from manufacturers' device datasheets. Scarcity of device parameters is much less of a problem for *SPICE* users, as the widespread use of the simulator by industry has compelled many device manufacturers to extract and publish — free for the downloading — real-world device parameter values that can be plugged directly into *SPICE*. Modern RF-fluent non-*SPICE* simulators like *Ansoft Designer SV 2* may be able to use *SPICE* device parameters directly or with a bit of parameter-renaming and value-resuffixing.

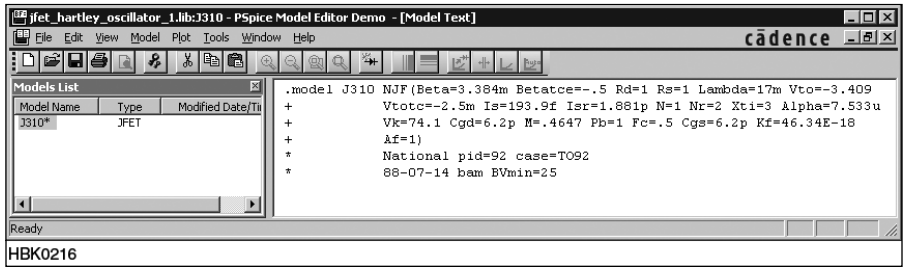


Fig 6.5 — Opening the circuit's J310 device for editing reveals that its model parameters were extracted at National Semiconductor in 1988. In the *OrCAD 16* demoware, both J310 instances in Fig 6.2 must use exactly this parameter-value set; non-demoware simulators allow separate customization of each instance of the same model. Lines that begin with asterisks are comments, ignored by the simulation engine.

basic realism with the BIP model or linear BJT model equivalent to it.

Especially in the area of MOSFET and MESFET device modeling, and large-signal device modeling in general (of critical importance to designers of RF integrated circuits [RFICs] for use at microwave frequencies) *SPICE* and RF-fluent non-*SPICE* simulators include active-device models home experimenters are unlikely, even unable, to use. (Do BSIM3 and BSIM4 ring a bell? MEXTRAM? Statz, Curtice and TriQuint GaAs FET models [levels 1, 2, 3 and 6]? No points off if you can't already answer *yes* to these extra-credit questions. Several, if not most, of these models are of interest only to EE students and their teachers, those who work for a semiconductor foundry that uses them, and those who produce circuit-simulation products that implement them.)

Most of us will go (and need go) no further into the arcanities of device-modeling than using *SPICE*'s JFET model for FETs like the 2N3819, J310, and MPF102, and *SPICE*'s BJT model for bipolar transistors like the 2N3904. Getting the hang of the limitations and quirks of these models may well provide challenge enough for years of modeling exploration. (We'll encounter another device-modeling option — representing devices as black boxes characterized by manufacturer-supplied network parameter datasets — later as we consider the RF-fluent simulator *Ansoft Designer SV 2*.)

Obtaining real-world-useful device-parameter values to plug into even *SPICE*'s standard diode, BJT and JFET models is critically important if we are to creating simulations that work well. *OrCAD 16.0* includes preconfigured 1N914, 1N4148, 2N2222, 2N2907A, 2N3819, 2N3904 and 2N3906 devices, among others, and these will be sufficient for many a ham-radio simulation session. To get parameters for other devices, especially RF devices but also including the J310 JFETs of Fig 6.2, we must search the Internet in general and device-manufacturer websites in particular to find the data we need. The manufacturer sites listed in **Table 6.2** will get you started.

Fig 6.5 shows the Fig 6.2 J310 parameters in *OrCAD 16*'s component editor. Because the demoware version of *OrCAD 16.0* does not allow us to edit multiple instances of a device independently of each other, the J310s in our simulation are identical.

Fig 6.6 illustrates the level of detail involved in more-accurate device modeling for VHF and UHF. The device is a California Eastern Labs NE46134, a surface-mount BJT intended to serve as a broadband linear amplifier at collector currents up to 100 mA and collector voltages up to 12.5. This manufacturer-supplied model embeds the unpackaged device chip (NE46100) within a *netlist*-based subcircuit that models parasitic reactances contributed by the transistor package, including chip-to-lead connections. At MF and HF, where the NE46134 could serve well as a strong post-mixer amplifier, modeling the device with just its basic, bare-chip characteristics (declared in the .MODEL NE46100 statement) would likely be accurate enough for many applications.

That Fig 6.6 illustrates the device parasitics using ASCII art is a side attraction. The main show — aside from the conveyance of the *SPICE* parameters of the NE46100 chip in its .MODEL statement — is the depiction of the NE46134 device as an NE46100 chip embedded in a subcircuit defined in netlist —

Table 6.2
Device Parameter Sources

Source	*Address*	*Resource*
Cadence Design Systems	**www.cadence.com/products/orcad/pages/downloads.aspx#cd**	*OrCAD*-ready libraries of manufacturer-supplied device models
California Eastern Laboratories	**www.cel.com**	data and models NEC RF transistors
Duncan's Amp Pages	**www.duncanamps.com/spice.html**	*SPICE* models for vacuum tubes
Infineon	**www.infineon.com**	data and models for Siemens devices
Fairchild Semiconductor	**www.fairchildsemi.com**	device data and *SPICE* models
National Semiconductor	**www.nsc.com**	data and *SPICE* models for National op amps
NXP	**www.nxp.com/models/index.html**	data and models for Philips devices

The National Semiconductor J310 parameters used in simulating Fig 6.2 came from a QRP-L posting ("Re: More JFET Models") archived at **qrp.kd4ab.org/1999/990616/0098.html**.

```
* FILENAME:        NE46134.CIR
* from http://www.cel.com/pdf/models/ne46134.cir
* NEC PART NUMBER: NE46134
* LAST MODIFIED:   11/97
* BIAS CONDITIONS: Vce=5V, to 12.5V,  Ic=50mA to 100mA
* FREQ RANGE:      0.1GHz TO 2.5GHz
*
*                            CCBpkg
*                      .-------||--------.
*                      |           CCB    |
*                      |       .--||---.  |           COLLECTOR
* BASE                 |       |       | |
*               __Tb_  |       |   /----o-o-------o----o
*  o----------|_____|-o-LB---o--|Q1   |          |
*                      |       B   \   --         |
*                     --           |   -- CCE     --
*                     -- CBEpkg E o----'          --CCEpkg
*                      |           |              |
*                      |          LE              |
*                      `----------o--------------'
*                                 |
*                                 -
*                                | |  Te
*                                | |
*                                 -
*                                 |
*                                 o EMITTER
*
*
*       CCB   = 0.03 pF       LB  = 1.2nH          Tb/Te:
*       CCE   = 0.5 pF        LE  = 1.2nH          z=60 ohms
*       CCBpkg= 0.18pF                             l=50 mils
*       CCEpkg= 0.18pF                             a=0.0001
*       CBEpkg= 0.01pF                             f=0.9GHz
*
*                    c b e
.SUBCKT NE46134/CEL 2 1 3
Q1 2 6 7 NE46100
CCB 6 2  0.03E-12
CCE 2 7  0.5E-12
LB  4 6  1.2E-9
LE  7 5  1.2E-9
TB 1 0 4 0 Z0=60 TD=9.63E-12
TE 5 0 3 0 Z0=60 TD=9.63E-12
CCBPKG 4 2 0.18E-12
CCEPKG 2 5 0.18E-12
CBEPKG 4 5 0.01E-12
.MODEL NE46100 NPN
+( IS=8.7e-16    BF=185.0         NF=0.959       VAF=30.0         IKF=0.20
+ ISE=5.70e-13   NE=1.80          BR=5.0         NR=1.0           VAR=12.4
+ IKR=0.018      ISC=1.0e-14      NC=1.95        RE=0.630         RB=6.0
+ RBM=4.0        IRB=0.004        RC=3.0         CJE=4.9e-12      VJE=0.60
+ MJE=0.450      CJC=2.50e-12     VJC=0.830      MJC=0.330        XCJC=0.20
+ CJS=0.0        VJS=0.750        MJS=0.0        FC=0.50          TF=12.9e-12
+ XTF=1.60       VTF=19.9         ITF=0.40       PTF=0.0          TR=1.70e-8
+ EG=1.11        XTB=0.0          XTI=3.0        KF=0.0           AF=1.0 )
.ENDS
*$
```

Fig 6.6 — ASCII art lives on in the depiction of package parasitics in the California Eastern Laboratories model of the NE46134 linear broadband transistor. In introducing us to the concept of subcircuits — a defined circuit chunk that, once simulated in a given simulation run, can be reused multiple times elsewhere in the simulation. This model also illustrates that *SPICE*, in common with other simulators, uses a network list — netlist — to communicate circuit topology, component values, and analysis instructions to its simulation engine.

```
* source JFET_HARTLEY_OSCILLATOR_1
J_J2          N012082 N01236 N01504 J310
D_D1          N00091 0 D1N4148
Kn_K2          L_L1 L_L2     0.99
C_C8          0 N02935  0.001uF
J_J1          N00193 N00091 N00753 J310
C_C7          0 N09766  0.001uF
L_L6          N19502 N02415  50uH
V_V2          N07095 0
+ PULSE 0 1 3us 0 0 1us
R_R1          0 N00091  1E6
V_V1          N02415 0 12Vdc
C_C11          0 N15520 {30pF*0.5+.001p}
R_R2          N00193 N02415  10
C_C6          N07095 N00753  0.1pF
C_C9          N09766 N19502  0.1uF
L_L1          N00753 N15520  1.425uH
C_C4          N01236 N00753  10pF
L_L4          N012082 N18891  50uH
R_R3          0 N01504  270
C_C1          N00091 N15520  2.35pF
Kn_K1          L_L4 L_L5
+ L_L6    0.99
C_C2          0 N00193  0.1uF
R_R5          0 N02935  50
C_C5          0 N01504  0.1uF
C_C10          0 N02415  0.1uF
L_L5          N18891 N19502  50uH
L_L2          0 N00753  0.075uH
R_R4          0 N01236  100k
C_C3          0 N15520  300pF
L_L3          N09766 N02935  350nH
```

Fig 6.7 — The JFET VFO circuit in *SPICE* netlist form. The N*nnnn* declarations name circuit nodes or *nets* — points of interconnection between components. Each component statement names the part's *primitive* — the leading *J* (for JFET) in *J_J1* — and identifies its instance (*J1*) to form an instance name (*J_J1*). Net numbers, generated automatically and algorithmically by the *netlister* function associated with the schematic editor, are ordered in element statements such that element *pins* — points of electrical interconnection — are connected to the correct nets as graphically depicted in the schematic. With some tedium, display of net names may be togglable in a simulator's schematic editor; net name display in *OrCAD 16.0* must be enabled one net at a time. This can be useful because netlisters occasionally make mistakes, and comparing a circuit's netlist to its schematic may aid in troubleshooting simulation errors.

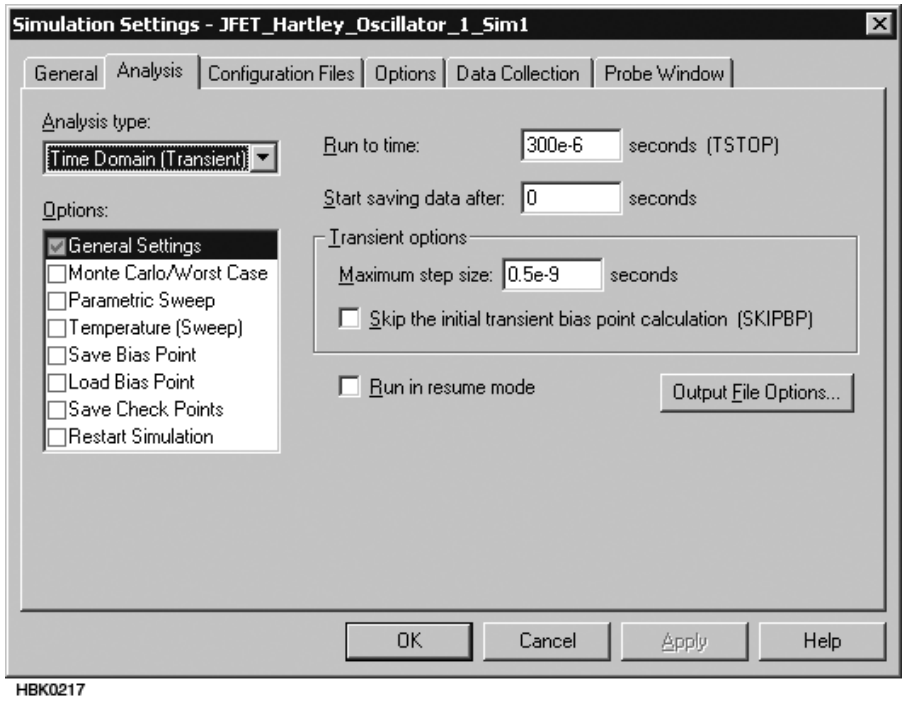


Fig 6.8 — *OrCAD 16 SPICE* setup for transient analysis of the JFET oscillator-buffer circuit. The behavior of the circuit will be progressively simulated every 0.5 ns (at most) from 0 to 300 µs.

network list — form. A netlist is a specialized table that names the circuit's components, specifies their electrical characteristics, and maps in text form the electrical interconnections among them. Uniquely numbered nodes or *nets* — in effect, coordinates in the connectivity space walked by the simulated circuit — serve as interconnects between components, with each component defined by a statement comprising one or more netlist lines. Statements that must span multiple lines include *continuation characters* (+) to tell the netlist parser to join them at line breaks. Asterisk (*) or other non-alphanumeric characters denote comments — informational-to-human lines to be ignored by the simulator. In Fig 6.6, header information and the ASCII-art portrayal of the device-package parasitics are *commented out* in this way.

The netlist served as the original means of circuit capture for all simulators known to the writer, including *SPICE* and the now-discontinued *ARRL Radio Designer* simulation product; *schematic* capture came later. Further reflecting *SPICE*'s pre-graphical heritage is the fact that, to this day a *SPICE* netlist may be referred to by long-time *SPICE* hands as a *SPICE deck*, as in "deck of Hollerith punch cards." In *SPICE*'s early days, circuit definitions and simulation instructions (netlist statements that begin with a period [.]) were commonly conveyed to the simulation engine in punched-paper-card form. All of the circuit simulators known to the writer still use a netlist as a means, if not *the* means, of conveying circuit topology and simulation instructions to the simulator; simulation based on schematicless user-created netlists may be possible with some.

Fig 6.7 shows the VFO circuit rendered as a partial netlist for simulation. This netlist is "partial" in the sense that it omits simulation and output commands we would expect to see in a full *SPICE* deck. At analysis time, the netlist contents we're shown by *OrCAD 16.0* are concatenated in memory with other data, including analysis setup information.

ANALYSIS SETUP

Many of us know from our smattering of learned theory that a complex waveform must be sampled at a frequency at least twice that of the highest-frequency component present if the samples taken are to be acceptably representative of reality. In doing transient (time-domain) simulations in *SPICE*, we have a similar concern: We must sample circuit behavior over time often enough to show us accurately the highest-frequency effects we want to see. Present-day *SPICE* simulators can intelligently size the maximum time step used in transient analysis to a value appropriate for useful representation of the highest-frequency fundamental source in a simulation. That said, manually setting the maximum step

to a smaller-than-automatic value can provide smoother-looking waveform graphs more acceptable to the eye. Smaller time steps come at the expense of longer simulation times and larger simulation-data files — issues more important in the earlier days of *SPICE* than nowadays, when gigahertz-class CPUs, gigabytes of RAM and even terabyte-class hard drives are standard. In this case, to simulate Fig 6.2 we set up a transient analysis out to 300 microseconds and a minimum step size of 0.5 ns (**Fig 6.8**). And then we click **Run**.

The *OrCAD 16.0* reporter opens when the analysis has finished. What we want to see is the circuit's output waveform across R5, the 50-Ω load we added. In reporter-terminology for this simulator, that's V(R5:2). **Fig 6.9** shows the resulting graph, rescaled a bit to center the instant of startup in the frame.

As a goal within our more general goal of getting a feel for "just building" simulated circuits as we often build them in the real world, we undertook this simulation with the aim of confirming the real-world circuit's claimed output power of +10 dBm (10 mW). That we have done, as **Fig 6.10** shows. Interestingly, the power level rises relatively slowly (and actually is still edging higher — ultimately to 9.5 mW — even after 300 µs). Does this reflect the behavior of the real thing? The same question may occur to us after we view the circuit's output spectrum, which **Fig 6.11** shows on a linear voltage scale. Shouldn't the output of our VFO be a pure sine wave? For a discussion of where such assumptions, unconscious and otherwise, may lead us, see the sidebar, "Simulation Goal or Moving Target?"

6.2.2 Example 2: Modeling a Phase-Lead RC Oscillator

The small maximum time step (0.5 ns) and long simulation period (300 µs) we set in simulating Fig 6.2 result in simulation runs that take nearly two minutes on an 868-MHz Pentium III computer with 512 MB of RAM. We chose those settings to give the circuit time to amplitude-stabilize and to allow for smoother display of waveform graphs when we zoom in. If our simulating computer has sufficient RAM and disk space to generate and handle the resulting large data file (170 MB), *OrCAD 16.0* is ready to graph simulation results from Fig 6.2 in the time it takes to start a cup of tea steeping.

As an illustration of a class of oscillators that can simulate much more rapidly, **Fig 6.12** presents an RC phase-lead circuit used as a CW sidetone generator in popular Amateur Radio transceivers of the 1970s and 1980s. In this case, the addition of a kick-start pulse is not required because charging currents in the circuit's 0.012-µF capacitors provide the stimulus for startup; **Fig 6.13** shows its startup to 40 ms.

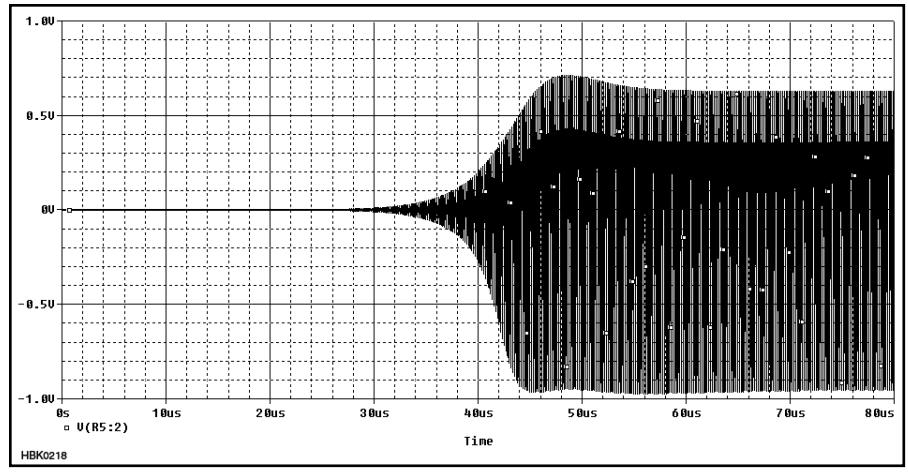


Fig 6.9 — At last: The spectacle of oscillator startup and amplitude stabilization rewards our attention to modeling detail. This graph shows the voltage at point A in Fig 6.2 — that is, the voltage, referred to ground (node 0), at node 2 of R5 [in this simulator's reporter-terminology, V(R5:2)], the circuit's 50-Ω load. The square dots scattered throughout the plot are trace markers. Note that the waveform's peak-to-peak span is not symmetrical around 0 V.

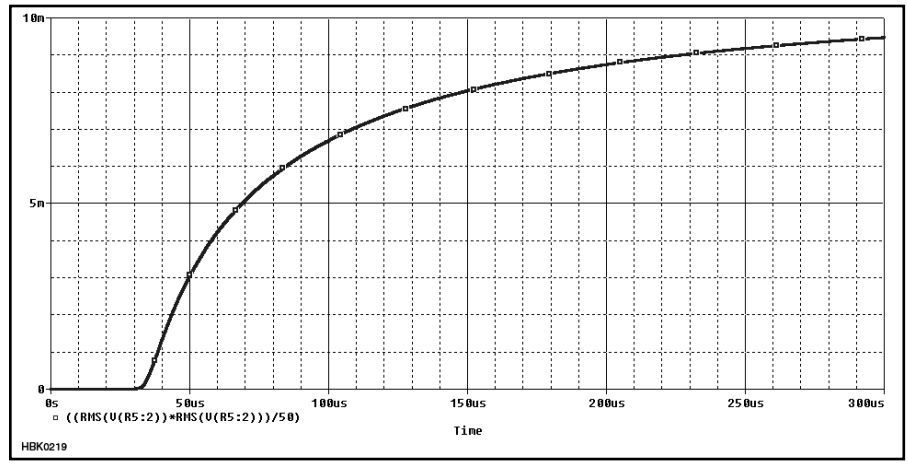


Fig 6.10 — To graph the circuit's power output, we render the voltage across R5 as RMS with the reporter's built-in RMS function, square the result by multiplying it by itself, and then divide the result by 50, the resistance of R5. By the end of the simulation, the output has reached 9 mW (9.5 dBm) — realistically close to the 10 dBm reported for the real-world circuit.

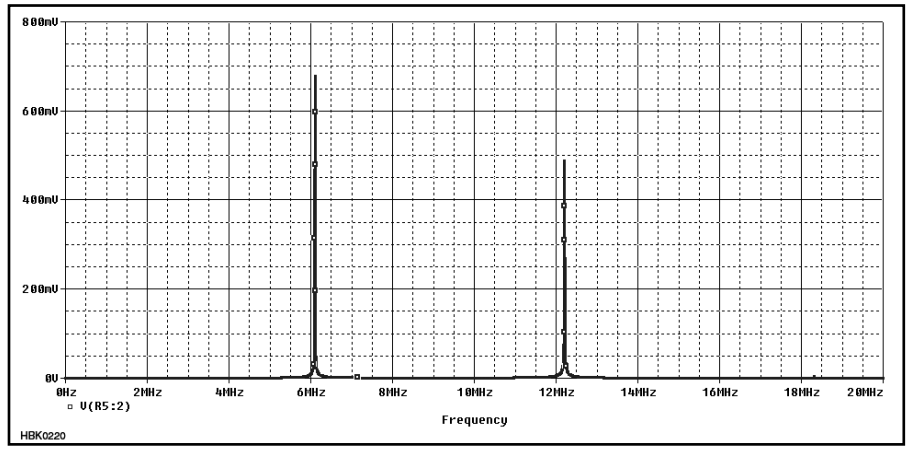


Fig 6.11 — Output spectrum of the simulated VFO. To make this clean spectral plot, we told the reporter's fast Fourier transformer (FFT) function to use only data beyond 60 microseconds after the start of simulation. Including data up through startup and stabilization data would cause the FFT to generate noise and discrete-frequency artifacts that diverge from real-life behavior.

Simulation Goal or Moving Target?

In simulating Fig 6.1, we set out to confirm the real-world circuit's output power, 10 dBm (10 mW). Fig 6.10 shows that we did just that. So, are we done? That depends.

Fig 6.A1 shows what the circuit's output waveform looks like when we zoom in on Fig 6.9 after the circuit has had time to amplitude-stabilize. It's clearly *not* a sine wave! Does our simulated circuit have a problem? What should we do next? That depends.

Circuit simulation allows us to explore circuits and what we know — and what we *think* we know — about them in a highly elastic and dynamic way. We may go into a simulation looking to find the answer to a simple question or questions, and find ourselves inventing new questions, even new modeling goals, along the way. This can bring good news and bad news. The good news is that our intuition and learning can be supercharged by whatever we may encounter. The bad news is that we can be misled by assumptions we may not even know we've been making.

An accurate RF power meter substituted for R5 in Fig 6.2 can indicate the absence or presence of RF at that point and its absolute level, and no more. The simulator's reporting function played the same role in providing us with the output-power curve shown in Fig 6.10. Whether the power-measured signal is spectrally dirty or clean, whether its frequency drifts or jumps — these characteristics, and more — cannot possibly be inferred from power measurement. Luckily — or maybe not — a simulator's reporter can tell us much more.

In real life, we would build our circuit expecting to see 10 mW output, measure that value with our RF power meter — if we even *have* a meter — connect the VFO to the receiver we built it to drive, and tune happily away. Little would we know that behind the deceptive simplicity of our measurement may lurk a signal that's other than a sine wave. (Even more arcane: The VFO output signal may be a sine wave when driving a resistive load but become non-sinusoidal when connected to the more reactive load presented by the receiver mixer's local-oscillator port.)

Perhaps the waveform in Fig 6.A1 really *does* generally reflect what real-world copies of Fig 6.1 do — but at this red-hot second we don't know enough to be sure. The description of the original circuit did not include a picture of its output waveform. A good high-frequency oscilloscope connected to the output of the real thing could tell us; a spectrum analyzer could also tell us, if a bit less directly.

Short of that, we can only intelligently speculate: Maybe the model data we used for the J310 JFET is insufficiently realistic. Maybe *SPICE*'s built-in JFET model folds up somehow as devices modeled with it move into the large-signal operation that oscillator amplitude stabilization involves. Maybe a signal like Fig 6.A1 happens whenever we diode-clamp the gate voltage of a JFET oscillator. Maybe the oscillator is overdriving the buffer amplifier. Maybe we or the author misspecified a value or left out a component. (Note to self: This is the first thing to check.) Maybe we have been unwarrantedly assuming for all of our radio-experimentation lives that unseen sources are sinusoidal until proven guilty.

Industrial-strength circuit simulators come to us as a result of engineering disciplines that seek to understand the world and predict its behavior toward the ultimate goal of achieving practical tools reproducible in quantity. As radio-hobbyist experimenter-builders, we may be just as satisfied with speculating about, and exploring of the quality and behavior of, a tool far beyond our having achieved its sufficiently practical function.

Computerized simulation can empower both approaches. The trick for us experimenters is to know when we've moved from solving a narrowly defined problem to dynamically redefining the problem such that enough is never enough. If you spend a half hour tweaking a bandpass-filter simulation for a –3-dB bandwidth of exactly 200 kHz through specification of capacitance values out to three decimal places, will you be able to achieve exactly that result with parts from your junk box? Will you even be able to know if you have? Will it even matter when you use your circuit on the air?

So what *about* that non-sinusoidality in Fig 6.A1? Is it a problem, an opportunity, or neither? Writing the rest of its story is up to you.

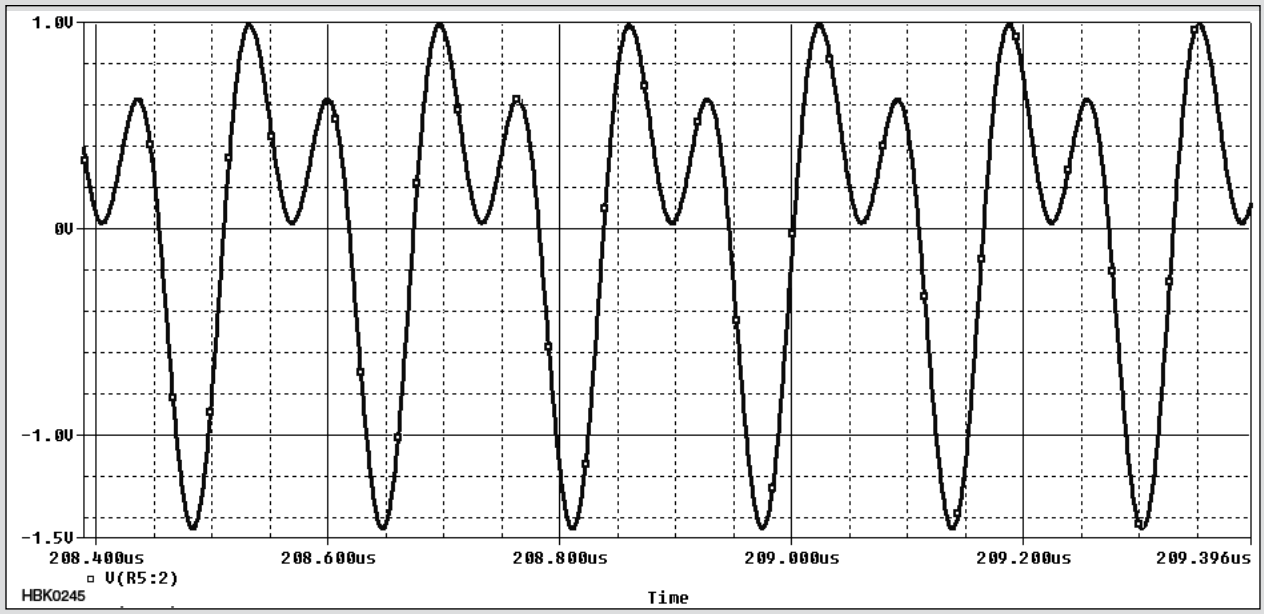


Fig 6.A1 — Zooming in on the JFET VFO's output signal confirms what the waveform asymmetry in Fig 6.9 and the second-harmonic spike in Fig 6.11 imply: that the VFO's output is not a pure sine wave. But is this a problem? That depends.

As we did with the JFET VFO example, we can use the reporter's FFT function to display the circuit's output waveform as an amplitude-vs-frequency (spectral) graph. We must do so with care, however, as the results may vary significantly with the amount and type of data used in the transform. To illustrate this, **Fig 6.14** shows the circuit's output spectrum based on analysis data from 40 ms to 500 ms, and **Fig 6.15** shows us what the FFT reports when we tell it to use only data from 480 to 500 ms. In effect, an FFT becomes surer of its results — the spectral components it reports sharpen — as we give it more data to work with; on the other hand, once the FFT has shown us all there is to see, we encounter diminishing returns — insufficient improvement in resolution as the dataset grows — if we make the simulation longer than it needs to be.

6.2.3 Example 3: Exploring Issues of Modeling Gain, Link Coupling and Q with a 7-MHz Filter

In evaluating the simulated oscillators in Figs 6.2 and 6.12, we had little difficulty in identifying the schematic junction — *node* — at which to probe the signal we wanted to evaluate. Other than working through the issue of graphing RMS power rather than peak power for our simulated VFO, we were able to graph our simulations' output without difficulty. Choosing the circuit point at which we would monitor our simulated circuits' behavior was simplified because an oscillator has only one *port* — excluding power and bias sources and control lines, only one point of interaction with the outside world.

Obtaining simulation results that we can both understand and trust becomes more complex when we simulate circuits with two or more ports. This is so because reporting the behavior of a simulated circuit is form of Q & A: Through the simulator's reporting functions we formulate a question, receiving as our answer a graph or table of circuit responses as numerical values, manipulated by such additional mathematical operations as we may specify.

From life experience we know that asking the wrong question will necessarily give us a wrong — though not necessary useless — answer. Especially if we are not electrical or electronics engineers, however, asking the wrong question of a simulator's reporting functions — its *reporter* — is deceptively easy. Our real-world experience in evaluating circuit behavior may not have prepared us for the subtleties of correctly posing even the most basic question to our simulator's reporter. The most immediate and far-reaching example of this is the evaluation of circuit gain.

The *concept* of gain as a ratio of output power, voltage or current to input power, voltage or current, perhaps expressed in decibels, is straightforward enough: Gain is at base a ratio that expresses the difference between the level of a signal at a circuit's output and the level of the same signal at the circuit's input. Sometimes we express gain numbers directly ("a voltage gain of 10"); sometimes we express gains terms of decibels ("a gain of 20 dB"). And we usually, but not always, call negative gain *loss*.

Measuring gain in real life is straightforward as well — at least gain as we are accustomed to thinking about it when we evaluate mixers, amplifiers, and filters for many ham-radio purposes. **Fig 6.16** shows a test-bench setup for measuring and displaying the gain (or loss), of a two-port circuit — a *two-port*, to put it generically — across a range of frequencies. In it, a signal generator — the tracking generator — applies a test signal to the input of the device under test (DUT), and an output-level-calibrated receiver — the spec-

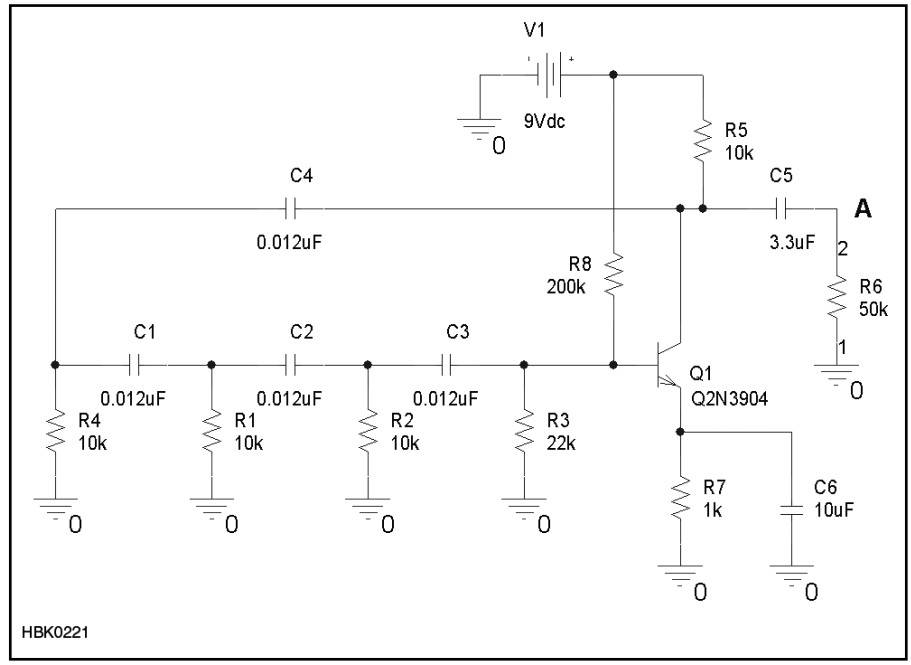


Fig 6.12 — Popular ham transceivers of the 1970s and 1980s included an RC phase-lead oscillator very much like this one as a CW sidetone generator. This modeled oscillator does not need a kick-start pulse, as the cascading disturbance of charging currents progressing through its *RC* phase-shift network is sufficient to start oscillation. Real-world builders may find that the lossiness of the circuit's feedback loop may require careful selection of the 2N3904 to ensure reliable starting.

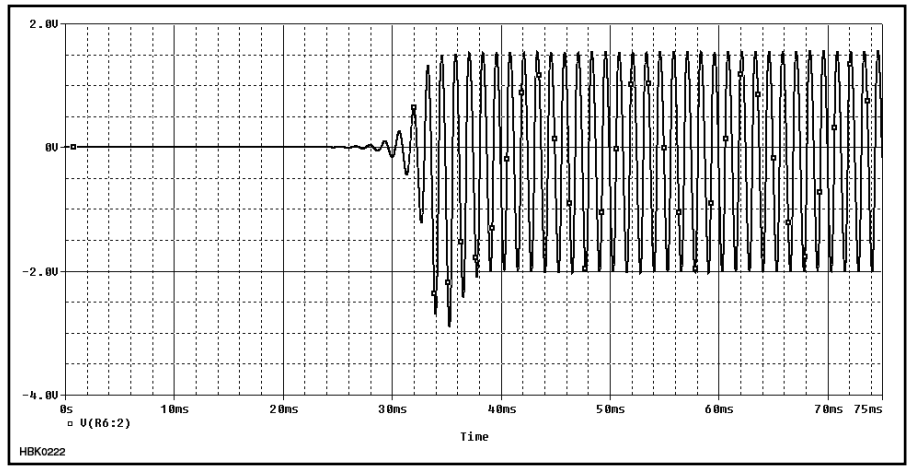


Fig 6.13 — Oscillator startup as embodied by the *RC* oscillator. This simulation (10-µs steps to 75 ms) takes only a few tens of seconds in a relatively slow computer; the Fig 6.2 simulation, *minutes*.

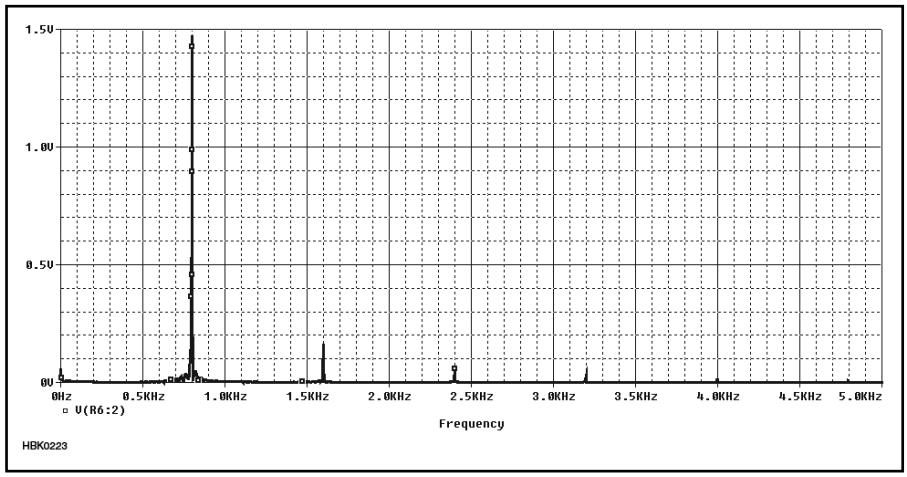


Fig 6.14 — Output spectrum of the RC phase-lead oscillator on a linear voltage as returned by the *OrCAD 16* reporter's fast Fourier transform (FFT) function. For a cleaner response — that is, to avoid displaying mathematical resultants from the circuit's rapidly changing spectral characteristics before and through startup and amplitude stabilization, and to give the FFT a larger periodic dataset to digest — we have extended the analysis to 0.5 s and excluded from the FFT analysis data between 0 and 40 ms.

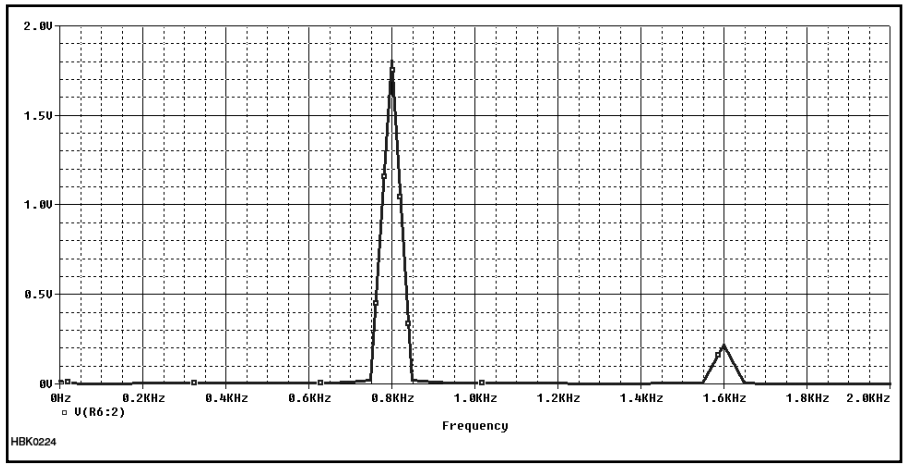


Fig 6.15 — Greatly restricting the dataset digested by the FFT degrades the amplitude-vs-frequency resolution of the report to the point of unrealism.

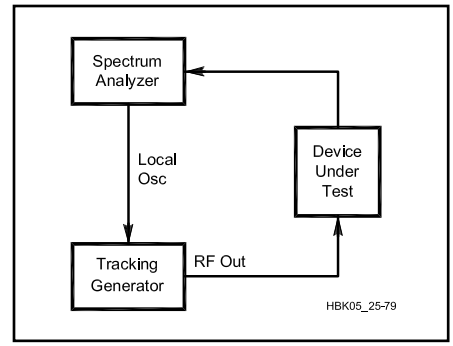


Fig 6.16 — One possible test setup for measuring the gain-versus-frequency response of a two-port device under test (DUT). This simplified diagram does not show the additional input and/or output attenuation that would be present in many actual measurement scenarios. For example, the presence of an active DUT (one expected to exhibit positive gain, as opposed to negative gain [loss]) would compel us to add attenuation between its output and the spectrum-analyzer input — not only to keep the spectrum-analyzer receiver from overloading and giving us false results, but also to protect the analyzer from damage if the DUT were to start oscillating rather than just amplifying.

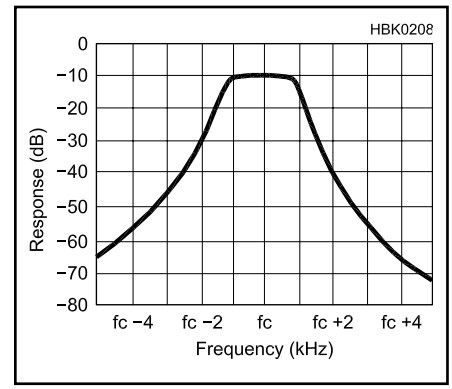


Fig 6.17 — The passband response of a crystal ladder filter as displayed by a spectrum analyzer.

trum analyzer in Fig 6.16 — receives the test signal as modified by the DUT and displays the results as an output-vs-frequency graph (**Fig 6.17**). To determine the gain or loss of the DUT at a given frequency, we'd compare that graph to the graph we get when we connect the tracking generator directly to the spectrum analyzer. When in ARRL publications we say that an RF circuit has "2 dB of loss" or "16 dB of gain," we almost always mean the type of gain derivable from measurements made by this process or its equivalent.

Type of gain? There's more than one? Yes: See the sidebar, "Defining Gain." It turns out that the gain we measure with a system like that shown in Fig 6.16 returns values that correspond to just one among multiple possible definitions of gain. In this case, our test system measures *insertion gain*: the difference between output measurements made at the load with the source connected directly to the load and with the device under test inserted between source and load. If we want to be able to compare the gain of a CAD-simulated two-port device with the gain of a counterpart real-life device measured in a test setup like Fig 6.16, we must tell our CAD program to report our circuit's insertion gain.

Using the Fig 6.16 system to measure insertion gain is straightforward because it is hard-configured to produce two-port insertion-gain measurements. Its spectrum analyzer can report only the signal level present at the load, and its TEST GENERATOR OUTPUT and SPECTRUM ANALYZER INPUT jacks enforce the insertion of the DUT at a particular point between source and load. We must merely take care not to connect the DUT backwards — unless doing so might return some other value of interest.

Reporting the insertion gain of circuit simulated in *SPICE* (even a very simple one) is fundamentally trickier because the relationships between source, DUT and load, and between signal-level metering and load, that can be safely assumed to exist in Fig 6.16 are neither predefined nor enforced in *SPICE* simulations. We must build a test-signal generator into our circuit, and we may report the signal level at one circuit node as easily as another. To explore how this complication

Defining Gain

Although the concept of gain as the ratio of the voltage, current or power at the output of a circuit to the power, voltage or current at the input of a circuit may seem to require no qualification, exactly where we measure these values in a system under test can greatly affect the gain value returned. Determining the point at which to measure circuit output is relatively straightforward: We measure output power in the load, output voltage across the load, and output current through the load. But what about input power, voltage, or current? Considering the problem only in terms of power, **Fig 6.A2** lays the groundwork for an understanding of this issue by depicting a gain-measurement setup in generic form.

Writing in NTIA Publication TR-04-410, *Gain Characterization of the RF Measurement Path*, J. Wayde Allen shows that four possible definitions can be proposed for the power gain of the 2-port in Fig 6.A2:

$$G_1 = \frac{P_{2a}}{P_{1a}} \quad (1)$$

$$G_2 = \frac{P_{2a}}{P_{1d}} \quad (2)$$

$$G_3 = \frac{P_{2d}}{P_{1a}} \quad (3)$$

$$G_4 = \frac{P_{2d}}{P_{1d}} \quad (4)$$

where *a* denotes available power and *d* denotes delivered power.

Equation 1 is commonly considered as describing the *available gain* (G_a) of the 2-port; equation 3 as describing the 2-port's *signal gain* (G_s) or transducer gain (G_t); and equation 4 as describing *power gain* (G). Further qualification of measurement conditions leads to additional, more specialized definitions for gain.

When we simulate circuits with the aim of directly comparing the results with real-world measurements, we need to simulate measurements made with a test set like that shown in Fig 6.16. To do that, we must know exactly which circuit points to probe to give a ratio that, expressed in decibels, gives the same results we'd see if we tested our simulation's real-world counterpart in the Fig 6.16 setup. Keeping Fig 6.A2 in mind, knowing which of the equations above corresponds to the gain measured by the Fig 6.16 setup could help us determine where to probe in our simulations. (By the way, considering that throughout our example simulation discussions we have chosen to generate graphs in terms of signal voltage, here we should mention that equations 1 through 4 are just as valid for voltage and current as they are for power; it's only when we want to render G in decibels that we must remember whether to multiply the logarithm of *G* by 10 or 20.)

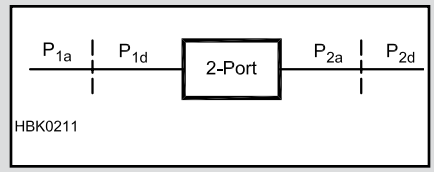


Fig 6.A2 — Generalized two-port gain evaluation in terms of available (subscript *a*) and delivered (subscript *d*) power (P) at port 1 (subscript *1*) and port 2 (subscript *2*). The power available from a source can differ from the power delivered to its load as a result of impedance mismatch between source and load. Although the subtleties of the issues involved are beyond the scope of this chapter, we must note that measuring gain becomes even more complex in the presence of sources and loads that are not purely resistive — that is, when voltage and current are not in phase.

Reading further in Allen, it turns out that none of those equations will suffice, for what the Fig 6.16 setup actually measures is *insertion gain* (G_i), which, thinking in terms of power, can be defined as

$$G_i = \frac{P_d}{P_r} \quad (5)$$

where P_d is the power delivered to the load when the 2-port under test is connected between the signal generator and the load, and P_r, the reference power, is the power delivered to the load when the 2-port under test is absent and the signal generator is connected directly to the load.

Working with a spectrum analyzer to determine gain or loss involves exactly this two-step operation. What a test setup like that shown in Fig 6.16 determines is insertion gain. As we'll see in simulating a 7-MHz band-pass filter, having access to the internals of the test-signal generator lets us report a circuit's insertion gain in just *one* step with the help of a bit of math.

can affect the results we see, we'll examine a simple 7-MHz filter in *OrCAD 16.0* — that is, as simulated by *SPICE* — and in a more-RF-friendly simulator, *Ansoft Designer SV 2*.

Fig 6.18 shows the circuit, a top-coupled double-tuned-circuit (DTC) design intended to provide a 3-dB bandwidth of 200 kHz centered at 7.1 MHz. From its description, we learn that its inductors consist of 17-turn windings on T-50-6 powdered-iron cores, with 2-turn links for input and output coupling. Per the inductance-from-turns equation associated with the toroidal core properties tables in this *Handbook*'s **Component Data and References** chapter, we calculate the inductance of the resonators as 1.156 μH; of the coupling links, 0.02 μH. **Fig 6.19** shows its response as published by Hayward.

Because we want the response of our simulated filter to approach Hayward's results as closely as possible, we must somehow specify the quality factor, *Q*, of its resonator inductors —their *loaded Q* (Q_U). In this case, the most direct approach would be to construct the coils and measure their *Q* on a *Q* meter. Not having access to one, we do the next best thing and extrapolate from another's careful *Q* measurements on a similar coil. From the **Measured Toroidal Inductor *Q* Values** table in Zack Lau's "RF" column in March 1995 *QEX*, we estimate our resonators' *Q* as 238 based on his data for a very similar 1.165-μH coil.

Having convinced ourselves of the importance of working inductor *Q* into our simulation and having obtained a trustworthy *Q* value for our inductors, we face a more fundamental challenge: *SPICE*'s inductor (and capacitor) models afford *no* built-in means of specifying *Q*! Recalling that the *Q* of a coil can be equated to its reactance, *X*, divided by its (equivalent series) resistance, R_S, we realize further that R_S can be determined by dividing *X* by *Q*. For a *Q* of 238 in a 1.156-μH coil at 7.1 MHz, R_S works out to 0.22 Ω. It seems that all we must do to model realistic resonator-inductor *Q* in our simulation of Fig 6.18 is add a 0.22-Ω resistance in series with each tuned-circuit coil.

In this case, yes — but a few sentences ago we said "*equivalent* series resistance" for good reason. The less-than-infinite *Q* of inductors at ham-band frequencies is a result not of ohmic resistance, but ac resistance due to *skin effect,* the tendency for ac at frequencies higher than a few hundred kilohertz to flow mainly at and near the *surface* of a conductor. Adding 0.22 Ω in series with each of our resonators therefore has the unwanted side-effect of making the coils unrealistically lossy at dc and low frequencies. In this simple HF-filter-modeling case, dc accuracy doesn't matter because our coils carry no dc. If, however, we wanted to model realistic *Q* in a coil that also carried dc to, say, a tran-

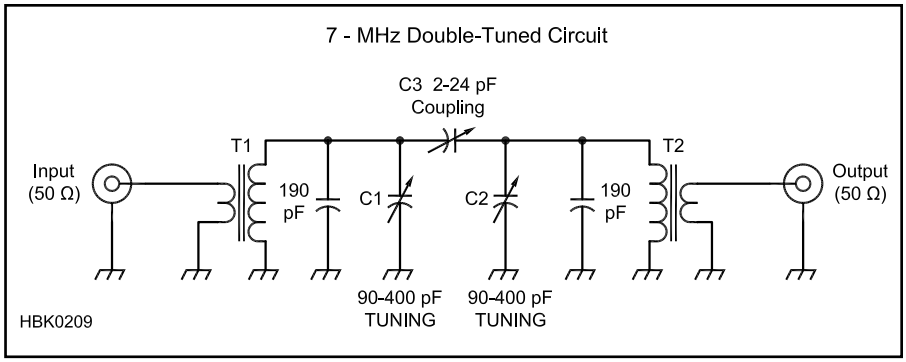


Fig 6.18 — A 7-MHz double-tuned-circuit filter as drawn for real-world builders. Each resonator consists of 17 turns of #22 wire on a T-50-6 toroidal, powdered-iron core; the coupling links consist of 3 turns of insulated wire. Per the article that described this circuit, the filter is intended to have "a 7.1-MHz center frequency and a 200-kHz bandwidth."

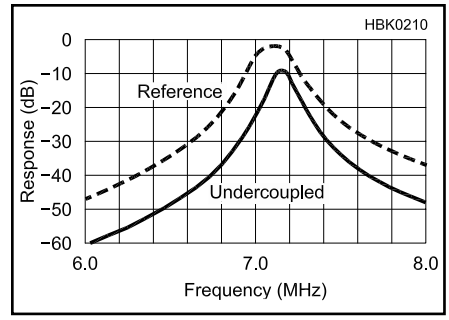


Fig 6.19 — Published response (Reference curve) of the 7-MHz filter. Careful reading reveals that these and other response graphs in the source article were generated by CAD software ("a computer generated the data for this and the other graphs in this article; experiment confirmed the data"); additional reading provides measurement results that provide real-world modeling goals: "The result: a critically coupled filter that's 178 kHz wide, with just over 2 dB of insertion loss."

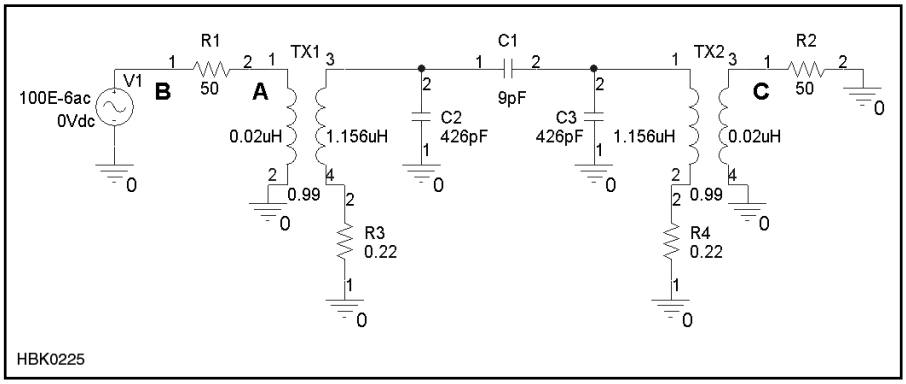


Fig 6.20 — The 7-MHz filter ready for SPICE simulation in *OrCAD 16.0 Capture CIS*. Transformer (TX) components simulate the circuit's link-coupled resonators (note the coupling coefficient, *K*, of 0.99), and 0.22-Ω resistors added in series with the tuned-circuit windings to model realistic *Q*. (No attempt is made to simulate realistic *Q* in the coupling links [0.02 µH] as real-world data is unavailable for that characteristic.) Paralleled capacitances in Fig 6.18 are represented by single values — an approach worth cultivating as a habit to stay within the component-count limitations of feature-limited demoware. Sinusoidal voltage source V1 (100 µV) and R1 serve as the test-signal generator, with R1 also serving as the filter's input termination; R2 serves as the filter load. Although intuition tells us that probing node R2:1 (labeled C) will give the circuit's output voltage, real-world test-bench experience does not immediately suggest where (point A or B?) to obtain the input-voltage value necessary for calculating the circuit's insertion gain as a real-world test setup would report it.

sistorized power amplifier, the voltage drop across any inductor R_S added merely as a *Q*-modeling workaround might well make our simulation unacceptably inaccurate at dc. One workaround to this new problem might be to shunt the *R*-equipped *L* with a high-value ideal *L* — a dc-bypass choke — but doing so would likely introduce yet other side-effects, including parasitic resonances and reduced realism in simulating the circuit in the time domain. A better approach would be to use an inductor model that implements skin-effect-based *Q*, such as that available in *Ansoft Designer SV2*.

Fig 6.20 shows the filter in *OrCAD Capture CIS*, ready for modeling in *SPICE*, with our *Q*-modeling resistances included as R3 and R4. Three more additional components, V1, R1 and R2, provide our test setup for measuring the circuit's insertion gain. V1 and R1 form the test-signal generator, with R1, which corresponds to the internal impedance of the generator, serving as the filter's input termination, and R2 serving as the filter's output termination and test load.

That the test-signal generator we build in treats as separate the signal source and its internal impedance gives us access to generator internals that we can't access in the real thing. Specifically, we can probe point B, the node at which the test generator's full output is available without modification by loss in the generator's internal impedance, R1 — a loss that will vary inversely with the impedance presented by the filter and its load. This means that we always have access to the reference level we need for calculating the insertion gain of whatever two-port circuit we connect between the test generator (V1R1) and its load (R2). Because R1 = R2, we can determine the insertion gain of whatever we connect between R1 and R2 with the equation

$$\text{Gain (dB)} = 20 \log\left(2\left[\frac{V_C}{V_B}\right]\right) \qquad (1)$$

where V_C is the voltage at point C of Fig 6.20 and V_B is the voltage at point B. Assuming that R1 = R2 — exactly the case in a real-world gain-measurement setup, in which the values of both will usually be 50 Ω — equation 1 returns a value of 0 when the test generator (V1R1) is directly connected to its load (R2).

Fig 6.21 shows the analysis setup settings that tell *SPICE* to do *ac* analysis, in which *SPICE* first determines a circuit's dc characteristics before determining its ac characteristics in response to whatever ac sources it may contain. In this case, our ac source puts out 100E–6 volts — 100 µV, a level that corresponds to a strong on-the-air signal.

Fig 6.22 graphs the analysis results — as returned by equation 1 (curve B) and (in curve A) as returned by merely expressing in decibels the ratio of the voltage across the filter output to the voltage across the filter input. Curve B reports the circuit's insertion gain as a real-world gain-measurement setup would display it. As we shall also see, Curve

B reports the circuit's insertion gain as an RF-fluent circuit simulator reports it "out of the box" as one of a very important set of tools called *S parameters*. Before we meet such a simulator in the form of *Ansoft Designer SV 2*, we'll use *SPICE* to reality-check our virtual insertion-gain-measurement setup.

6.2.4 Example 4: Checking Reality with a 10-dB Attenuator

To confirm in a general way that we're on the right track with the insertion-gain probing and reporting regime we arrived at for our simulation of Fig 6.20, we'll simulate a simple circuit of known gain and frequency response. The circuit, **Fig 6.23**, consists of little more than a pi-network attenuator with its resistances configured to exhibit 10 dB of loss in a 50-Ω system. As in Fig 6.20, we add 50-Ω source and load resistors, and a voltage source as a test-signal generator. **Fig 6.24** shows the attenuator's gain: –10 dB. We have indeed built and calibrated our virtual insertion-gain test setup to duplicate the behavior of the real thing, confirming as well the validity of the results we obtained for the insertion gain of the filter in Fig 6.20.

6.2.5 Example 5: Simulating the 40-Meter Filter in an RF-Fluent Simulator

Realistically simulating Fig 6.18 in *SPICE* required the addition of resistors to set its resonators' unloaded *Q* to 238. Such workarounds are not necessary when we use a circuit simulator specialized to speak RF, as we'll illustrate by simulating Fig 6.18 in *Ansoft Designer SV 2*, the student version of *Ansoft Designer*, a linear, nonlinear, electromagnetic and system EDA CAD suite engineered for modern RF, MMIC and RFIC design. **Fig 6.25** shows the filter in the *Ansoft Designer SV 2* schematic editor. Now, instead of using ideal transformer windings, we can model the filter's resonators as inductors with realistic *Q* based on skin effect (**Fig 6.26**). But using stand-alone inductors rather than transformers presents a new challenge: How will we model the filter's input and output coupling links?

In addition to including many realistically non-ideal components (**Fig 6.27**), the *Ansoft Designer SV 2* component library includes ideal transformers — transformers with a coupling coefficient (*K*) of 1 — that can be characterized by turns ratio. The resonators in the real-world circuit consist of 17-turn coils with 2-turn links, so we can simulate the links by using transformers with turns ratios of 2:17 (0.118) and 17:2 (8.5) at the filter input and output, respectively.

Absent from our schematic are a signal source and hardwired input and output terminations. Sources are unnecessary for linear simulation using *Ansoft Designer SV 2*. Terminations, including the 50-Ω default set within the circuit's port elements, are applied at reporting time, *after* analysis has concluded.

Fig 6.28 shows the responses available for our simulation in the *Ansoft Designer SV 2* reporter. Four-port responses are shown because the full circuit analyzed actually consists of the Fig 6.18 circuit (ports 1 and 2) and Fig 6.25 circuit (port 3 and 4) place side by side. **Fig 6.29** compares the two circuit's responses, with the responses of the Fig 6.20 circuit shown as dotted lines.

Unsurprisingly, the two modeling approaches produce slightly different responses, both of which meet the goals of the filter designer. Which one is "better" can therefore

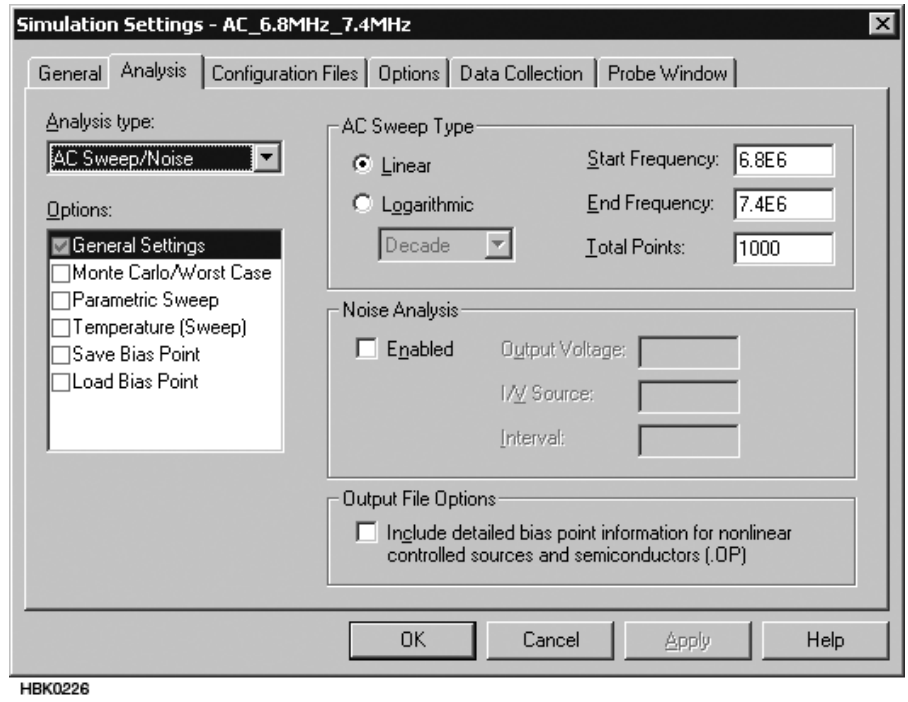


Fig 6.21 — Setup for a *SPICE* ac analysis at 1000 evenly spaced points from 6.8 MHz to 7.4 MHz.

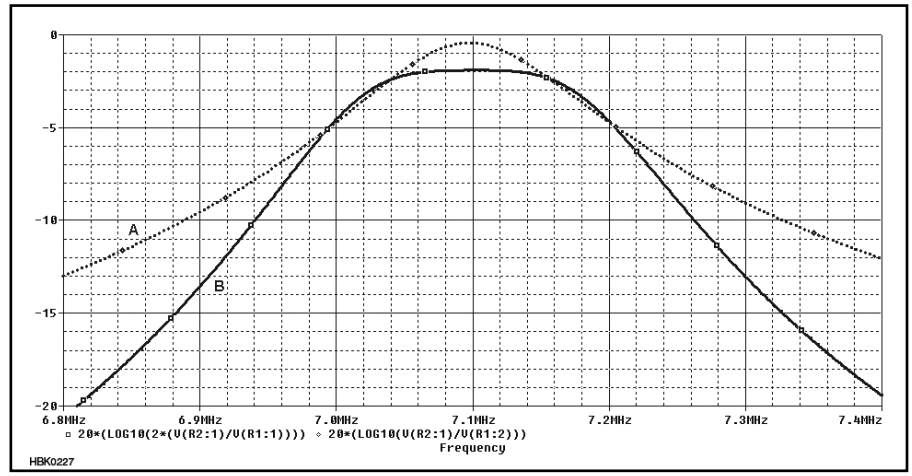


Fig 6.22 — Comparison of right and wrong approaches to reporting the filter gain in decibels. The dotted curve, A, merely plots the ratio, expressed in decibels, of the voltages at the filter output and input — points C and A — in Fig 6.20. The solid curve, B, plots values based on the ratio of the voltages at points C and B as determined by the trace definition 20*(LOG10(2*(V(R2:1)/V(R1:1)))) — the main text's Eq insertion gain calculation rendered in a form understandable by the reporter. Curve B depicts the circuit's insertion gain in decibels as the Fig 6.16 test setup would report it.

Symbols, Circuits and Hierarchies

We discover something quite interesting if we happen to view the *SPICE* netlist for Fig 6.20: Its linear transformer (TX) components (**Fig 6.A3**) are actually symbols that represent subcircuits — in this case, subcircuits that consist of two inductor (L) components coupled with a K_Linear coupling component. **Fig 6.A4** displays the evidence.

Because it includes subcircuits, Fig 6.20 is a *hierarchical* circuit — a circuit with multiple levels. Although support for hierarchies is usually limited in the demoware versions of CAD software that radio amateurs are likely to encounter, subcircuits are a powerful tool for creating and simulating large-scale circuits that contain other circuits, and circuits that use multiple copies of groups of components — individual identical transistor cells used many times throughout an RFIC, switching transistors with built-in bias resistors, bias and decoupling networks — throughout a design. Creating a schematic symbol that represents a subcircuit lets you in turn add the symbolized subcircuit as a new component for subsequent point-and-click placement in schematics like any other part — just as we do whenever we place a linear transformer component in the *OrCAD 16.0* schematic editor.

The power of symbolized subcircuits also operates at analysis time: However many identical instances of a given subcircuit may be present in an analysis, the simulation engine need evaluate its structure only once, recalculating its electrical behavior as it calculates the behavior of circuits that contain it.

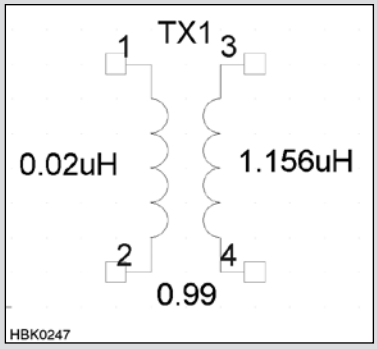


Fig 6.A3 — A linear transformer (TX) component in the *OrCAD 16.0* schematic editor. This part is actually a symbol for a subcircuit that contains three parts.

```
* source ZOI_40_METER_FILTER
V_V1          N07792 0 DC 0Vdc AC 100E-6ac
R_R1          N07792 N07820  50
R_R2          N04311 0  50
X_TX1     N07820 0 N01324 N00387 zoi_40m_filter_orcad16_TX1
X_TX2     N01606 N00537 N04311 0 zoi_40m_filter_orcad16_TX2
R_R3          0 N00387  0.22
R_R4          0 N00537  0.22
C_C1          N01324 N01606  9pF
C_C2          0 N01324  426pF
C_C3          0 N01606  426pF

.subckt zoi_40m_filter_orcad16_TX1 1 2 3 4
K_TX1          L1_TX1 L2_TX1 0.99
L1_TX1          1 2 0.02uH
L2_TX1          3 4 1.156uH
.ends zoi_40m_filter_orcad16_TX1

.subckt zoi_40m_filter_orcad16_TX2 1 2 3 4
K_TX2          L1_TX2 L2_TX2 0.99
L1_TX2          1 2 1.156uH
L2_TX2          3 4 0.02uH
.ends zoi_40m_filter_orcad16_TX2
```

Fig 6.A4 — Inspecting the *SPICE* netlist for Fig 6.20 reveals the reality behind the TX symbol: Placing a TX element actually places two inductors (L) and a coupling element (K).

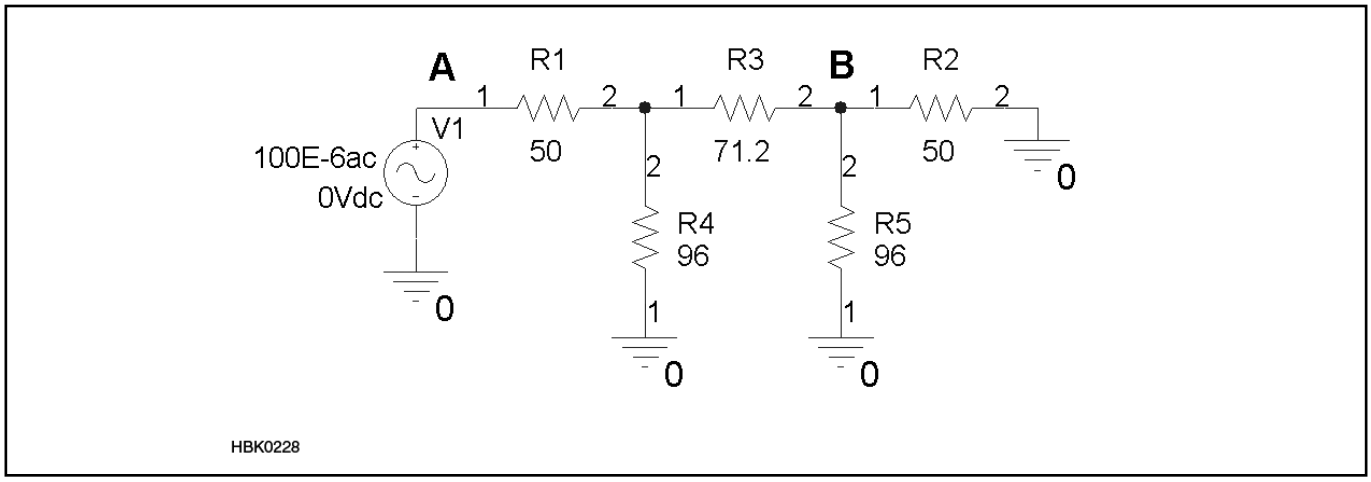


Fig 6.23 — To confirm what we think we now understand about modeling insertion gain with *SPICE*, we model a 10-dB attenuator using the same test generator and load arrangement as in Fig 6.18. Once again, we will calculate the gain of the circuit based on voltages probed between R1:1 (point A) and common (node 0) and R2:1 (point B) and common.

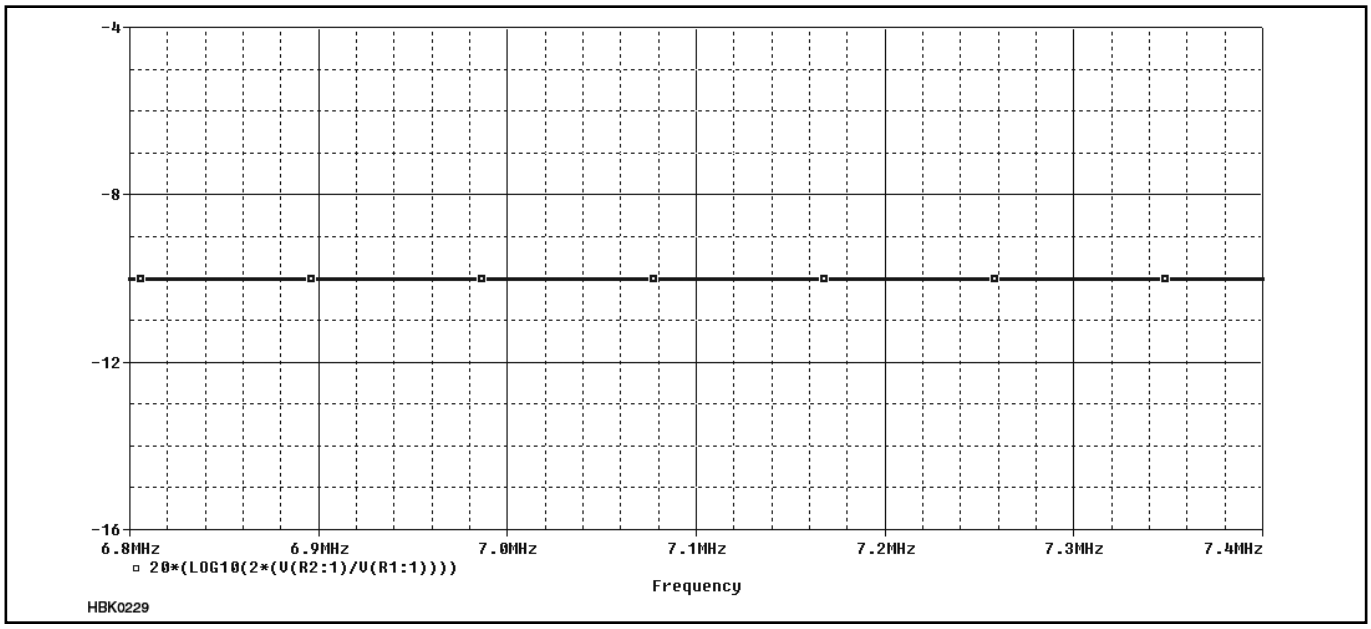


Fig 6.24 — As intended, our simulated 10-dB attenuator exhibits 10 dB of loss. In achieving this result, we have confirmed that our method of probing the circuit at R1:1 and R2:1 and common, in conjunction with reporting the circuit's gain as 20*(LOG10(2*(V(R2:1)/V(R1:1)))), validly returns the insertion gain we would measure in a real-world setup like that of Fig 6.16. In confirming this, we have also confirmed our finding of insertion gain through simulating Fig 6.20.

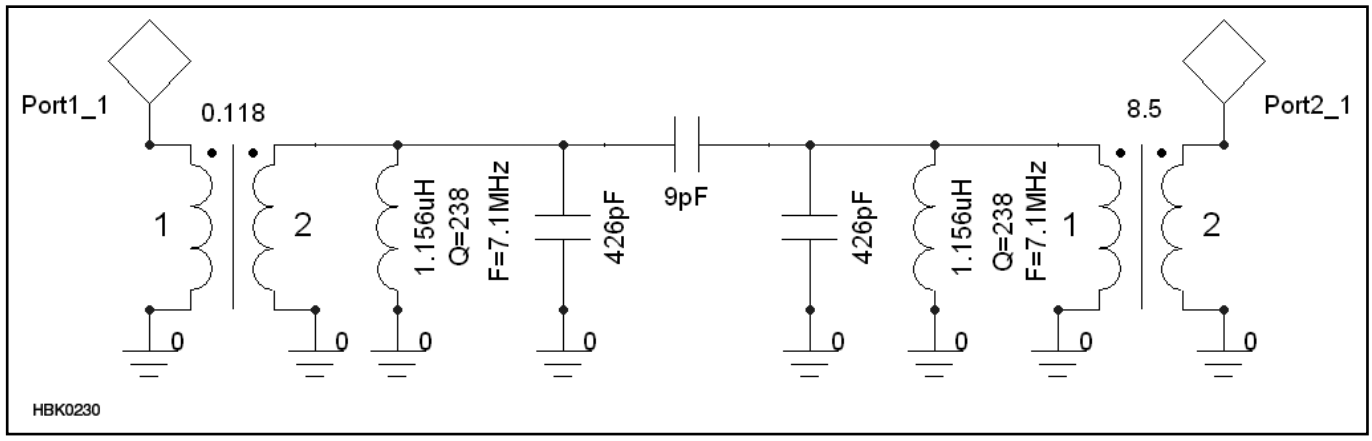


Fig 6.25 — Simulating the 7-MHz filter in *Ansoft Designer SV 2*, lets us use inductors that realistically model *Q*. Doing so then requires us to model the filter's input and output links with ideal transformers characterized in terms of turns ratio. The resonators in the real-world circuit are 17-turn toroidal coils with 2-turn links. The test generator and terminations necessary for *SPICE* analysis are absent for the reasons described in the text.

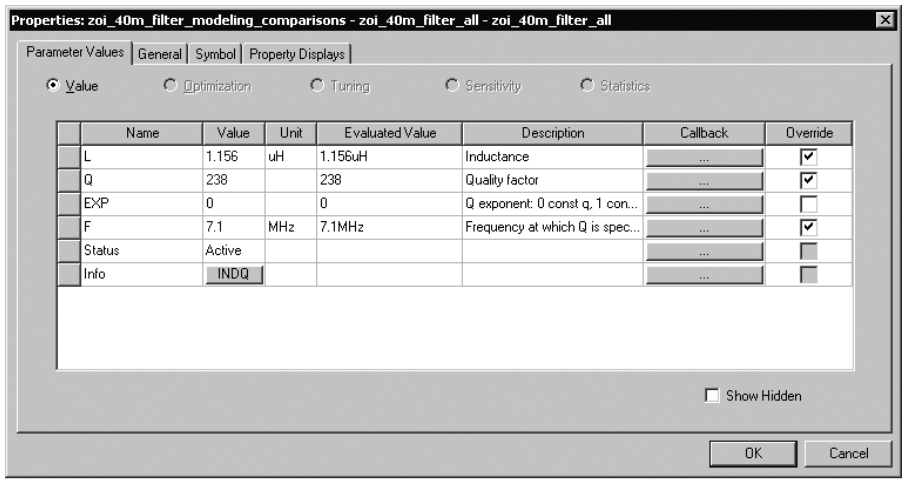


Fig 6.26 — Properties dialog for the *Ansoft Designer SV 2* INDQ component, showing settings for *Q*.

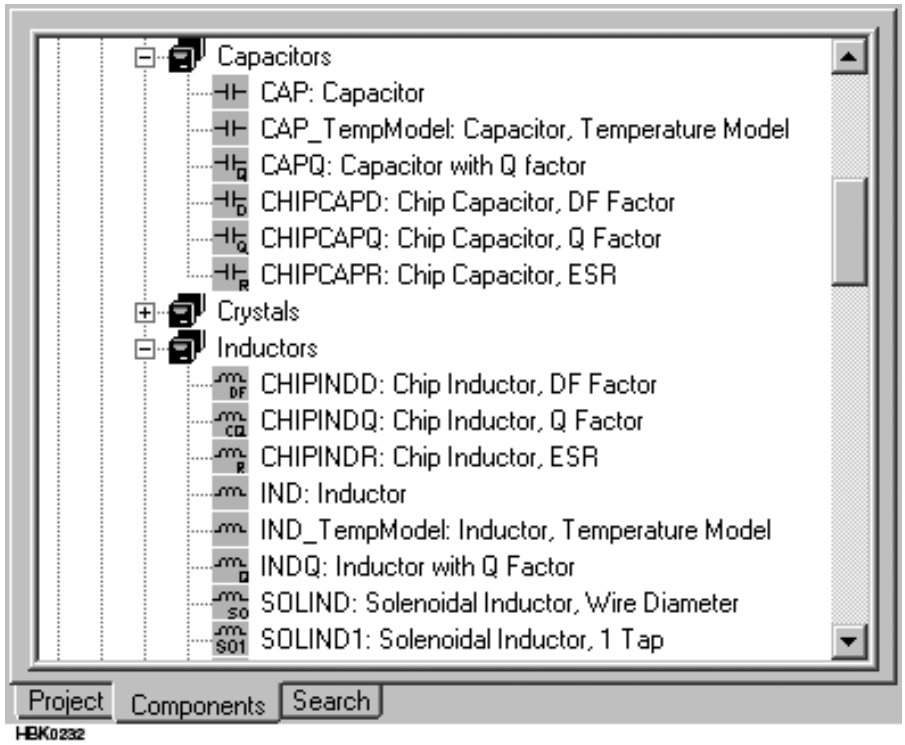


Fig 6.27 — The *Ansoft Designer SV 2* component chooser includes inductor and capacitor models that realistically model lossiness at RF.

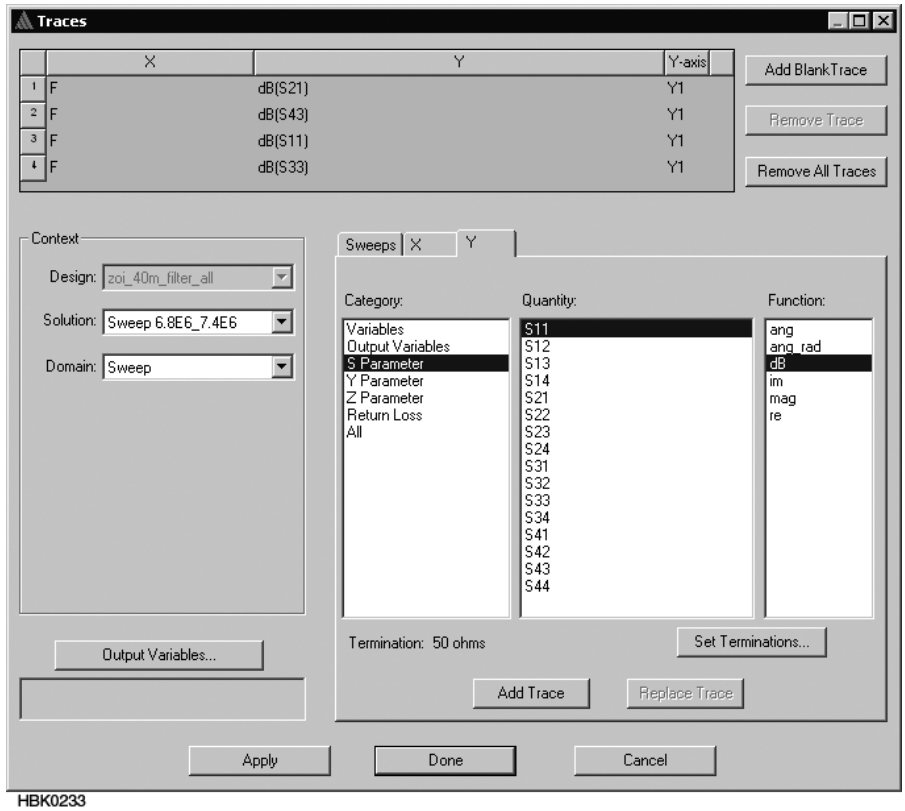


Fig 6.28 — The reporter of the *Ansoft Designer* RF-fluent linear simulator lets us evaluate circuit behavior not in terms of voltages and currents but rather in terms of *network parameters* — scattering parameters (*S*), admittance parameters (*Y*), and impedance parameters (*Z*) — and return loss. These reporter settings, used to produce the comparative graph in Fig 6.29, show responses for a four-port circuit rather than a two-port because the full analysis run included as a separate circuit copy the *Q*-from-added-resistors circuit from Fig 6.20.

be considered to be academic; discerning the difference between two real-world filters that exhibit exactly these responses would be a measurement challenge. On the air we could tell them apart only with difficulty.

Our true purpose in reevaluating the Fig 6.20 filter in *Ansoft Designer SV 2* is to illustrate the difference between RF-fluent CAD and the general-purpose *SPICE*-based CAD tools to which CAD-minded hams tend to default for circuit design. In graphing its responses in *Ansoft Designer SV2* we stand at the threshold of a class of muscular CAD tools that bring great power to hobbyists interested in RF CAD. Rather than plotting just gain, Fig 6.29 actually plots gain and *return loss*, a value of practical importance in many RF-design applications. (The higher the return loss, the more input-signal energy the filter accepts, and the less input-signal energy is reflected back to its source. For a filter of this design, we want and expect return loss to be high in its passband and low in its stopband.) Rather than merely evaluating the circuit's voltage gain, we report its response in terms of standardized *network parameters* — in this case the *S* (scattering) parameters S_{21} and S_{11} (for Fig 6.20, and S_{43} and S_{33} for Fig 6.25 in our two-two-ports-at-once simulation) expressed in decibels. The sidebar "*S*-Parameter Basics" explains more about why *S* parameters are important and how RF engineers use them.

The ability to handle network parameters is a profound differentiator between *SPICE* and more RF-fluent simulators. Although it's possible to derive *S* parameter from *SPICE* analysis through post-processing and/or the use of special subcircuits that stimulate an *n*-port for the purpose of more network-parameter-literate reporting, fluency in small-signal network parameters is not among *SPICE*'s simulation and reporting strengths. As we'll see in the next section, *SPICE* can only limitedly simulate intermodulation distortion, a signal-handling flaw in which multiple signals present at a circuit's input interact in circuit components — in active devices, especially — to produce output spectral components not present at a circuit's input.

6.2.6 Example 6: *SPICE* and Intermodulation Modeling

As an example of the limitations of *SPICE* for critical RF-fluent analyses, we will attempt to simulate intermodulation distortion (IMD) with *OrCAD 16.0*. IMD is of great importance to RF engineers because the span of signal levels — the *dynamic range* — we expect modern communications circuitry to handle is so wide that communication possible by means of weak legitimate signals can easily be made impossible by the weak false signals produced by IMD.

Fig 6.30 shows the circuit: the widely used

S-Parameter Basics

The tool called *S parameters* provides a standardized way of characterizing how a device behaves in response to signal energy applied to its ports — its signal inputs and outputs — usually with all of its ports terminated in identical, standard impedances (commonly, 50 Ω, resistive). A transistor, for instance, is a two-port device. By convention, the ports are labeled with numbers, Port 1 being the input and Port 2 being the output.

Signal energy applied to one port of a two-port device comes out two places: at the same port (because the device reflects some of the energy back to the generator) and at the other port. How much signal comes out relative to the applied signal tells us the device's gain (which can be negative — that is, a loss); how much signal reflects back out tells us something about the impedance match between that port's impedance and our signal generator. Determining the phase of the output or reflection signals relative to the phase of the applied signals tells us even more about the device or subcircuit under test.

Fig 6.A5 shows this idea graphically. An *S* parameter is a voltage ratio (commonly, but not always, expressed in decibels) annotated with two subscript numbers that indicate the ports involved. For instance, a device's forward transmission gain, S_{21}, ("S sub two one"), is the ratio of the voltage at Port 2 to the voltage applied to Port 1 — a value that must be expressed as a vector to convey the two signals' relative phase. To discuss *S* parameters more readily and to communicate them in tabular form, we can split each of the four basic parameters into separate components — real and imaginary parts, or, especially useful for device modeling with *S* parameters, magnitude and phase: MS_{11} (magnitude of input reflection) and PS_{11} (phase of input reflection); MS_{21} (magnitude of forward transmission gain) and PS_{21} (phase of forward transmission gain); MS_{12} (magnitude of reverse transmission gain) and PS_{12} (phase of reverse transmission gain); and MS_{22} (magnitude of output reflection) and PS_{22} (phase of output reflection).

From the standpoint of circuit evaluation and modeling, the great power of *S* parameters is that they can convey usefully realistic information about the ac behavior of a device or subcircuit through relatively few standardized numerical values. For instance, if we know the *S* parameters of a given transistor operating under known conditions of power- and bias-supply voltage and current, we have a very useful picture of how it looks to the outside world — a picture we can paste directly into an *S*-parameter-fluent linear circuit simulator. Of great value to the modeling efforts of professionals and radio amateurs alike is the fact that RF-device manufacturers commonly make their products' *S* parameters freely available in data formats widely used by industry-standard simulators. **Table 6.3** shows *S*-parameter data for the NE46134 transistor in an *S*-parameter format that most RF-fluent simulators can handle.

To use device *S*-parameter data in a simulation, we place a generic *black-box* component in our circuit and tell the simulator to read the *S*-parameter data when it calculates the behavior of the black box. Using black-box *n*-port parts in place of detailed mathematical device models extends the power of linear simulation for use in modeling the network responses, noise, and stability characteristics of, and developing matching networks for, devices that might otherwise not be modelable without recourse to nonlinear simulation and detailed mathematic device models configured with realistic mathematical parameter values. A significant limitation of black-box modeling is that an *S*-parameter dataset is static, and most accurately reflects real-world device behavior only under the conditions of voltage and current used in generating it.

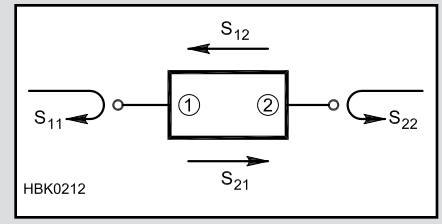


Fig 6.A5 — *S* parameters corresponding to input reflection, forward transmission gain, reverse transmission gain and output reflection can quite closely characterize a small-signal linear device — in this case, a two-port device. Expressing a two-port's forward transmission gain, S_{21}, in decibels returns the same number we would report for its insertion gain.

Table 6.3
Two-Port *S*-Parameter Data Equivalent to an NE46134 Transistor (Operating at V_{CE}=12.5 and I_C=50 mA) from the California Eastern Laboratories File *NE46134G.S2P*

```
! FILENAME:      NE46134G.S2P   VERSION: 8.0
! NEC PART NUMBER: NE46134      DATE:   07/94
! BIAS CONDITIONS: VCE=12.5V, IC= 50mA
# GHZ S MA R 50
```

0.050	0.432	–91.9	34.140	125.6	0.024	63.0	0.586	–56.7
0.100	0.372	–129.4	19.834	106.3	0.036	63.5	0.362	–78.4
0.200	0.348	–161.2	10.444	92.4	0.058	65.4	0.223	–97.1
0.300	0.344	–175.7	7.086	85.1	0.081	67.7	0.183	–106.7
0.400	0.343	173.5	5.332	79.3	0.104	68.1	0.170	–112.8
0.500	0.341	164.1	4.282	74.5	0.127	67.5	0.168	–116.6
0.600	0.343	157.1	3.610	70.1	0.150	66.3	0.172	–119.3
0.700	0.344	149.7	3.114	65.7	0.173	64.7	0.179	–121.0
0.800	0.349	143.8	2.755	61.8	0.196	63.1	0.188	–122.4
0.900	0.347	137.4	2.471	57.9	0.218	61.2	0.198	–123.4
1.000	0.352	132.4	2.254	54.4	0.239	59.2	0.210	–124.2
1.100	0.351	126.5	2.075	50.7	0.260	57.2	0.223	–125.0
1.200	0.353	121.2	1.927	47.1	0.280	55.0	0.236	–125.8
1.300	0.351	116.3	1.803	43.7	0.300	52.9	0.251	–126.5
1.400	0.350	111.6	1.710	40.4	0.320	50.7	0.267	–127.4
1.500	0.346	106.6	1.607	37.2	0.337	48.4	0.283	–128.0
1.600	0.343	102.0	1.523	34.0	0.354	46.2	0.300	–128.9
1.700	0.339	97.0	1.447	31.4	0.369	44.1	0.317	–130.2
1.800	0.339	93.1	1.388	29.0	0.385	42.4	0.330	–131.5
1.900	0.338	88.4	1.340	26.2	0.402	40.4	0.343	–132.1
2.000	0.336	83.3	1.295	23.5	0.418	38.2	0.357	–132.7
2.100	0.331	78.4	1.252	20.7	0.432	36.1	0.372	–133.4
2.200	0.325	73.3	1.206	18.1	0.445	34.0	0.386	–134.0
2.300	0.320	68.4	1.172	16.1	0.458	32.0	0.399	–134.7
2.400	0.318	63.0	1.140	13.6	0.471	29.9	0.411	–135.3
2.500	0.314	57.3	1.109	11.7	0.482	28.0	0.423	–135.9

Each of these lines conveys frequency and the *S* parameter magnitude and phase values M*S*11, P*S*11, M*S*21, P*S*21, M*S*12, P*S*12, M*S*22 and P*S*22. This S*n*P format (where *n* is the number of ports in the device characterized) was originally used by the EESOF Touchstone circuit simulator, and is one of several *S*-parameter data formats now widely used in the RF engineering community. An S*n*P file may also contain noise-modeling data.

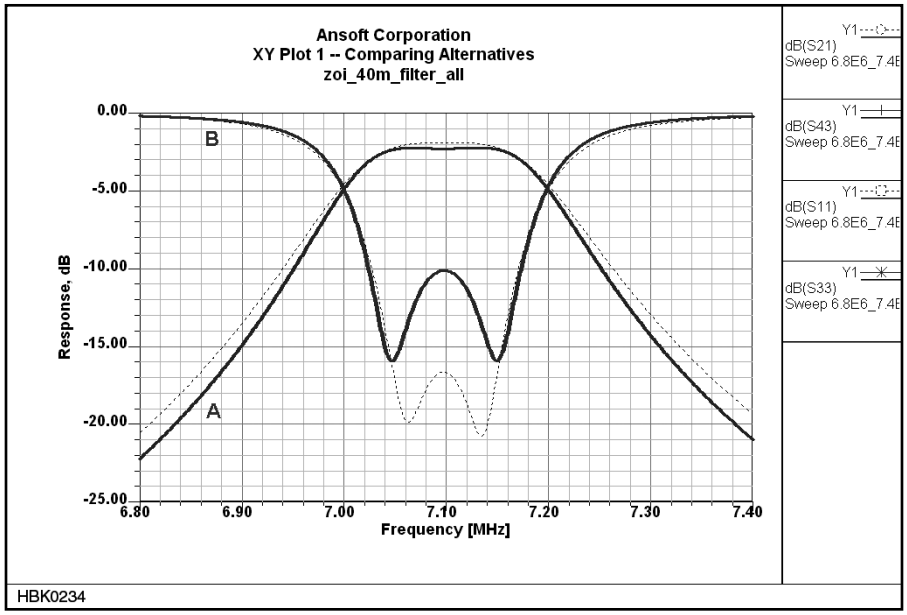


Fig 6.29 — *Ansoft Designer SV 2* report for the realistic-*L* filter showing (A) insertion gain and (B) return loss, both in decibels. (Return loss, the magnitude of S_{11}, is a positive value in dB although in this plot the Y-axis is calibrated in negative dB.) The dotted lines show the same responses for the *Q*-from-series-*R* circuit of Fig 6.20. Graphing the filter's return loss provides information about how closely the impedance of the terminated filter matches the impedance terminating the filter's input. The higher the return loss, the more closely the impedance presented by the input of the terminated filter approaches the impedance of its input termination (in this case, 50 Ω). Return Loss associated with a passive circuit, such as a filter, is always positive.

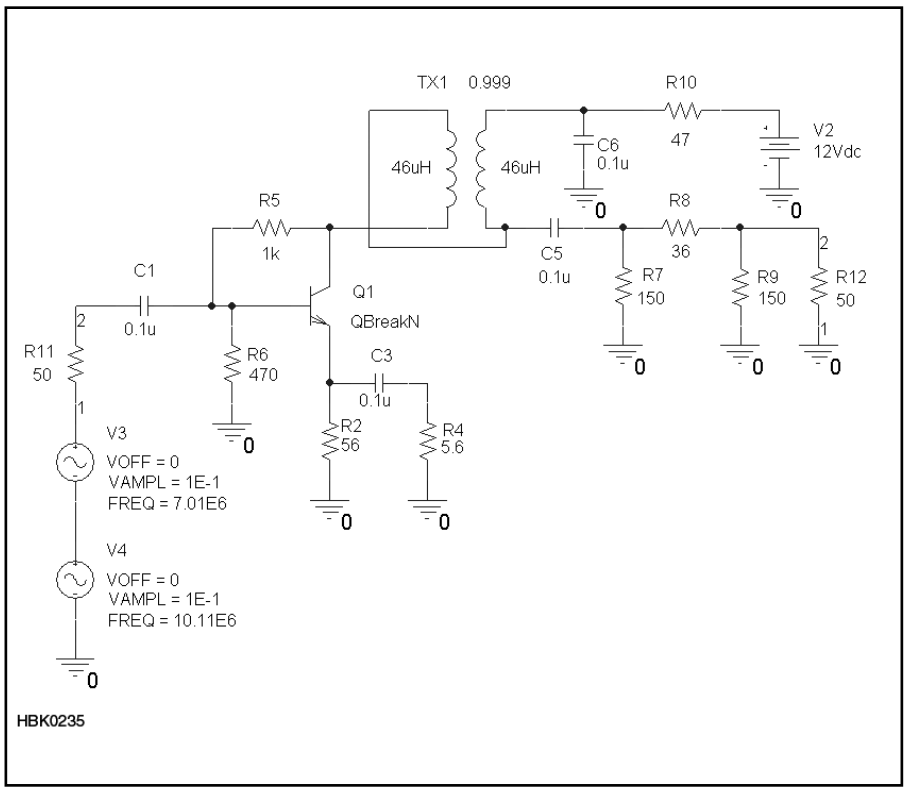


Fig 6.30 — The Progressive Communications Receiver post-mixer amplifier configured for two-tone IMD analysis in *OrCAD 16.0 SPICE*. The level of both test signals is 0.1 V — very strong, "other hams in the immediate neighborhood"-class signals. The transistor is modeled with *SPICE* data for the California Eastern Laboratories NE46134. Using a value of 47 Ω for R10 sets its collector current to 40 mA; 100 Ω, 31 mA.

and well-characterized post-mixer feedback amplifier introduced by Hayward and Lawson in their 1981 Progressive Communications Receiver. This implementation uses the *SPICE* parameters shown in Fig 6.6 for the California Eastern Laboratories NE46134 transistor and includes two signal sources in series for the purpose of generating IMD products.

Simulating the circuit's gain vs frequency response (**Fig 6.31**) returns realistic numbers; turning on nodal voltage and current display in the schematic editor (**Fig 6.32**) confirms that the device's bias point (for a BJT, collector current) is realistic and that we are not exceeding the device's collector-to-emitter voltage rating (15).

Fig 6.33 shows the FFTed output spectrum on a linear scale. Spectral components other than those attributable to the two test signals are absent. **Fig 6.34** displays the same data on a logarithmic voltage scale, with the X-axis zoomed in on the 0- to 40-MHz span. If we know what to look for, we can see responses at frequencies attributable to harmonics of the input tones; to second-order IMD (the frequency of each tone plus and minus the frequency of the other); and third-order IMD (twice the frequency of each tone plus and minus the frequency of the other), but the graph is complicated by higher-order products — arguably good from the standpoint of realism — a high noise floor, and a rising response toward 0 Hz.

As a comparison of simulation techniques, **Fig 6.35** shows the output spectrum for the same circuit as predicted by the harmonic-balance nonlinear simulator in *Serenade SV 8.5*, the now-discontinued predecessor of *Ansoft Serenade SV 2*. Harmonic-balance simulation treats the linear and nonlinear portions of a circuit as separately solvable subsystems, analyzing the linear portion in the frequency domain and the nonlinear portion in the (steady-state) time domain. Harmonic-balance analysis offers significant speed and dynamic-range advantages over *SPICE* for circuits that include transmission lines, long time constants relative to operating frequency, and many reactive components (such as RF circuits and systems commonly contain). Alas, at this writing, harmonic-balance analysis is unavailable to hobbyists and students in free demoware form as discussed in the sidebar, "RF-fluent CAD: What We're Missing."

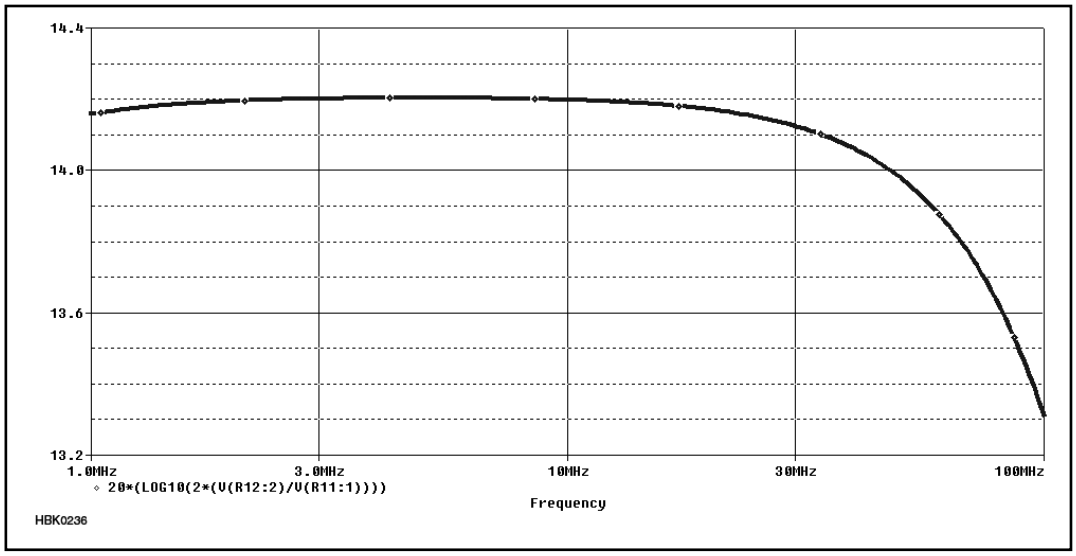


Fig 6.31 —Simulating the gain of the post-mixer amplifier with *OrCAD 16.0 SPICE* produces a realistic gain vs frequency response.

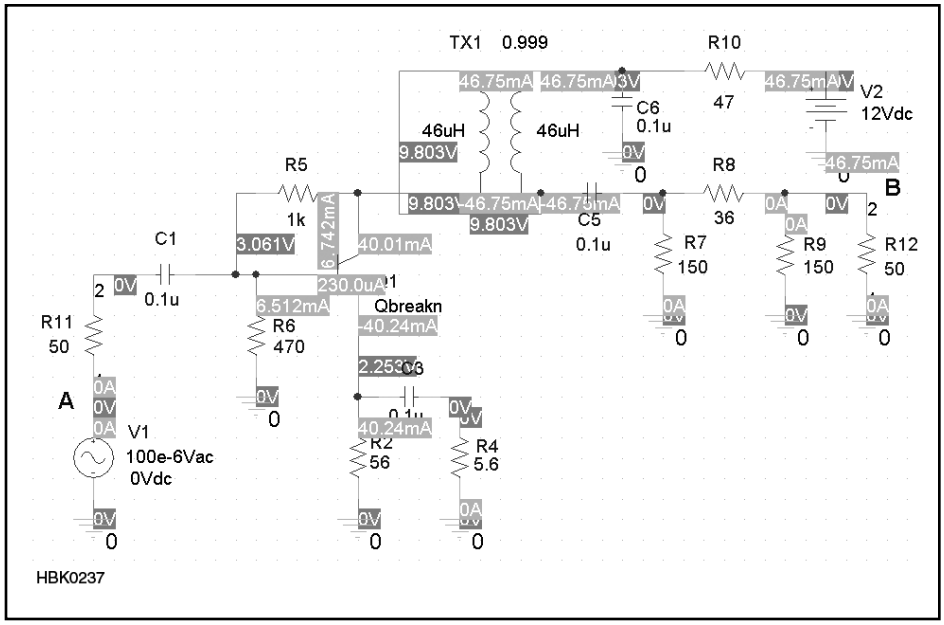


Fig 6.32 — After running at least one analysis, we can turn on voltage and current display in the schematic editor to reality-check the device bias point (for a BJT, collector current). Comparing the voltages displayed for the simulated BJT's collector, base, and emitter lets us ensure that we're not exceeding the published ratings of the real-world device.

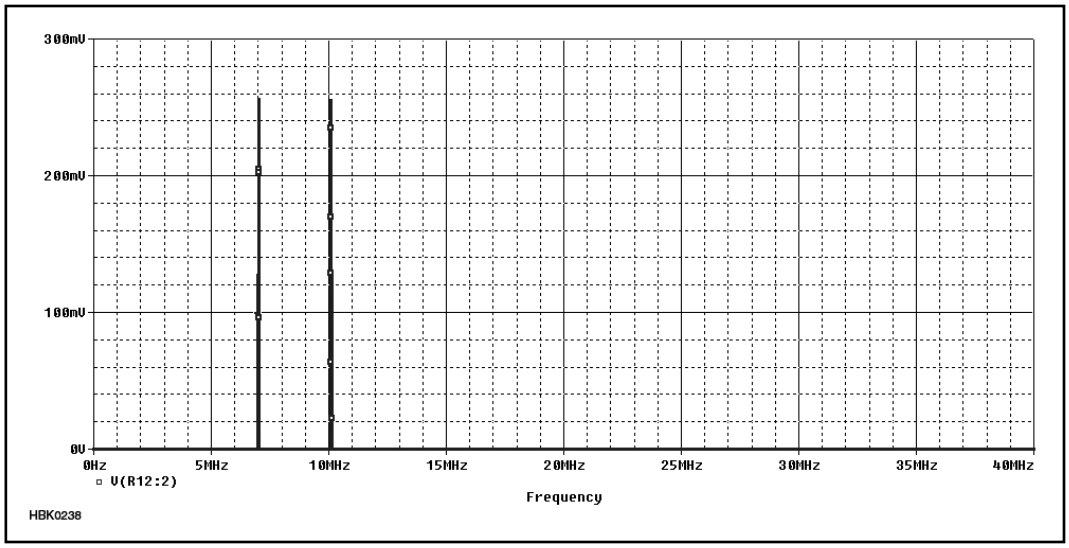


Fig 6.33 — Output spectrum of the feedback amplifier on a linear voltage scale. To generate this graph, we analyzed the Fig 6.30 circuit for 100 µs with a maximum time step of 10e–10 s, and used the *OrCAD 16.0* reporter's FFT function. Components attributable to IMD are superficially absent because of the linear *y*-axis scale.

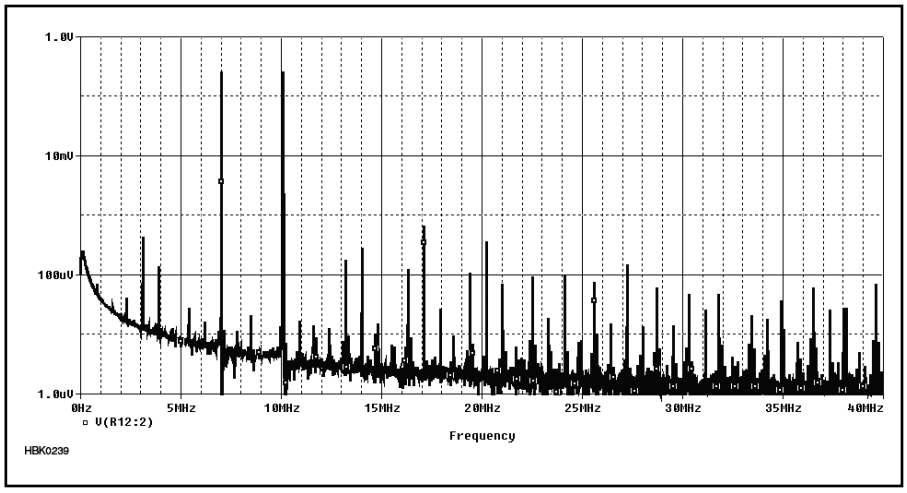


Fig 6.34 — Switching to a logarithmic voltage scale and zooming in on the 0- to 40-MHz range lets us discern spectral components at frequencies corresponding to harmonics and second- and third-order IMD products, but significant artifacts — a relatively high noise floor and a rising response toward time zero — are apparent.

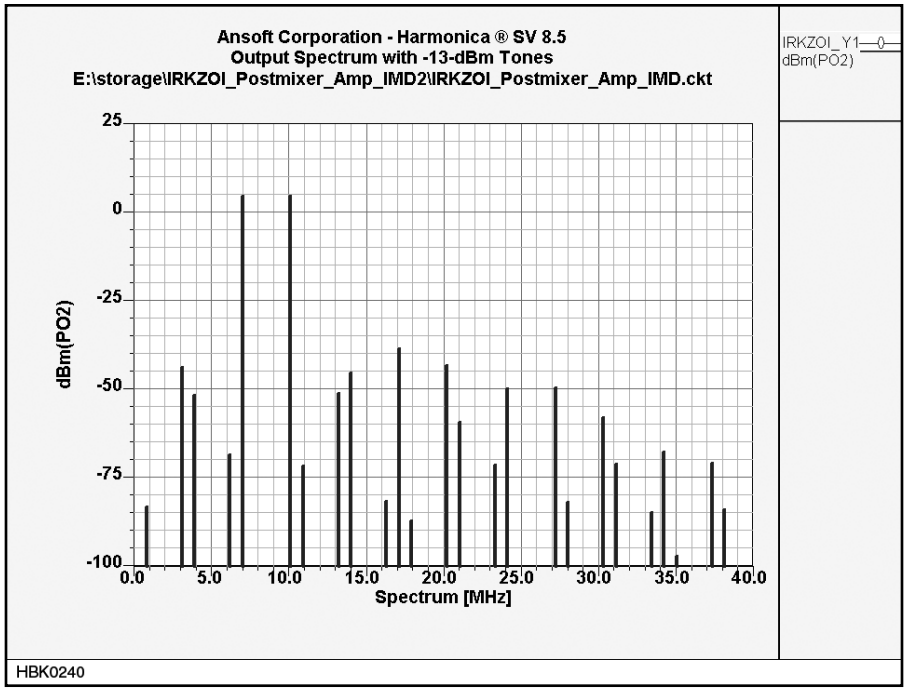


Fig 6.35 — Output power spectrum of the same circuit as predicted by the harmonic-balance nonlinear simulator in the no-longer-available Ansoft *Serenade SV 8.5.* This spectrum is simpler compared to Fig 6.34 because IM calculations were limited to fifth-order products in the student version of *Serenade 8.5*. Most striking is the large simulation dynamic range achieved without the appearance of math-noise artifacts, and the ability to report output power in decibels relative to a milliwatt (dBm). One more thing: The *SPICE* analysis for Figs 6.33 and 6.34 took over five minutes; the *Serenade SV 8.5* run that produced this graph, *under two seconds*.

Simulating Keying with a *SPICE* Behavioral Model

This chapter begins by illustrating that electronic circuits "just do math," responding to and processing electronic signals — also describable mathematically — by, in effect, performing mathematical operations on them. If, for instance, what you require in a simulation is, say, the mathematically idealized behavior of a comparator or 555 timer IC rather than detailed, step-by-time-step nodal analysis of the behaviors of its internal circuitry, you can use (after building it yourself, if necessary) a *behavioral model* of the part instead. Behavioral modeling can be used to simulate analog and digital parts.

The *OrCAD 16.0 Capture CIS* demoware component library includes quite a few behavioral models — mostly 7400-series TTL ICs, but also several analog ICs, including LF411, LM324 and 741 op amps, an LM111 comparator, and a 555 timer. As an example of what behavior models allow us to do, **Fig 6.A6** presents an analysis designed by John Seboldt, KØJD, to simulate on-off keying of a broadband feedback amplifier (Q1, QBreakN, an *OrCAD Capture CIS* "breakout" device characterized with data for an Infineon BFG135 transistor) by means of a series dc-supply switch (Q2, a 2N2907A) keyed via a 2N3904 switch by a 555 timer configured as a "ditter." So realistic is this simulation (**Fig 6.A7**) that it even models "short first dit" and backwave!

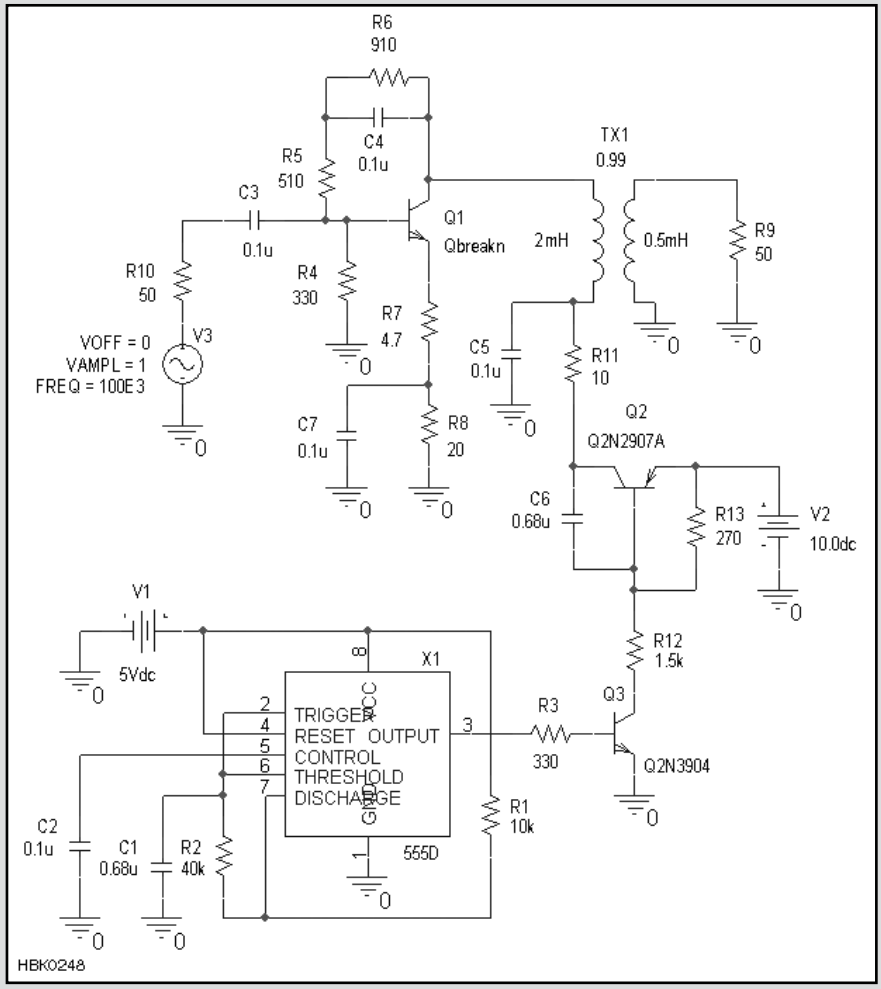


Fig 6.A6 — To simulate the keying waveform of a QRP transmitter stage in *SPICE*, John Seboldt, KØJD, built a ditter — a circuit that, powered up, sends a continuous string of Morse code dots — using a SPICE behavioral model for the 555 timer IC. The timer output drives 2N3904 and 2N2907A switches to interrupt the collector power supply of amplifier transistor QBreakN, an *OrCAD Capture CIS* "breakout" device characterized by NXP Semiconductor data for a Philips BFG135 (BFR194 chip) broadband transistor. Although a low-frequency (100-kHz) source is used to reduce the analysis time and datafile size (and we have increased the inductances in transformer TX1 appropriately relative to their values at HF), coupling and bypass capacitances are kept at their HF-appropriate values to keep RC-time-constant-related settling times consistent with the behavior of the real-world circuit at HF. To make the keyed signal's rise and fall times slower and more easily discerned in a graph, we have also increased the value of shaping capacitor C6 from Seboldt's value of 0.47 µF.

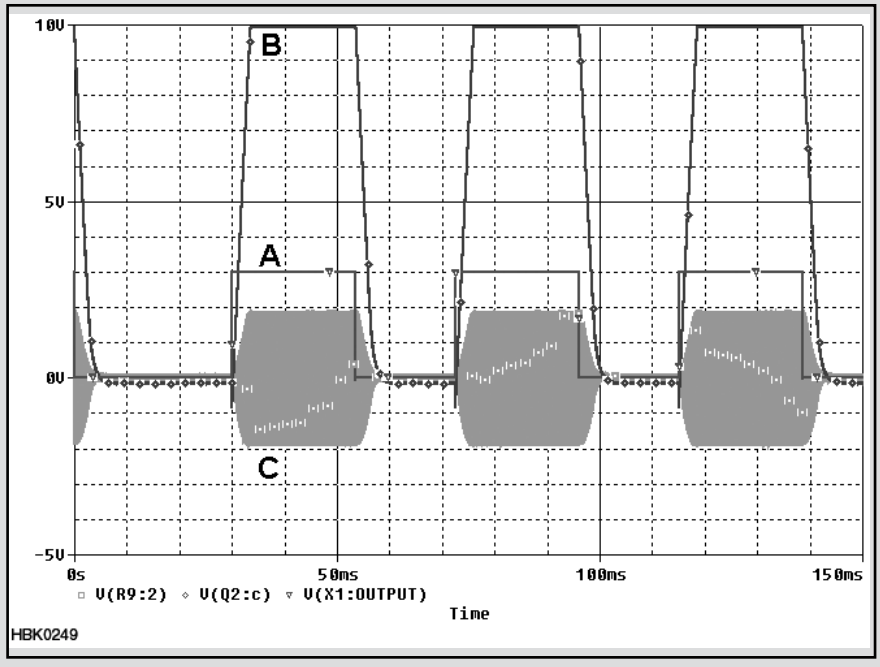


Fig 6.A7 — Results of the keying simulation, showing (A), the voltage at the 555 OUTPUT pin; (B), the keyed collector supply applied to the amplifier; and (C), the keyed amplifier output; time steps are 0.2 µs. So realistic is this simulation that it also reflects the "short first dit" problem shared by some real-world transmitters (in this case, it's an artifact of powering up the keyer and amplifier together as the analysis begins) and significant *backwave* (discernible output during key-up periods). In a real-world transmitter, we would reduce backwave to an acceptable level by keying multiple amplifier stages, a mixer, or — at the risk of degrading signal quality in additional ways — an oscillator.

RF-Fluent CAD: What We're Missing

SPICE-based simulators can do wonders in many classes of circuit simulation. For RF use, however, *SPICE* has significant drawbacks. For starters, *SPICE* is not RF-fluent in that it does not realistically model *physical* distributed circuit elements — microstrip, stripline, and other distributed circuit elements based on transmission lines. It cannot directly speak network parameters (*S*, *Y*, *Z* and more), stability factor and group delay. It cannot simulate component *Q* attributable to skin effect. It cannot simulate noise in nonlinear circuits, including oscillator phase noise. It cannot realistically simulate intermodulation and distortion in high-dynamic-range circuits intended to operate linearly. This also means that it cannot simulate RF mixing and intermodulation with critical accuracy.

The feature-unlimited version of *Ansoft Designer* and competing RF-fluent simulation products can do these things and more excellently — but many of these features, especially those related to nonlinear simulation, are unavailable in the student/demoware versions of these packages where such versions exist.

For awhile, from 2000 to 2005, the free demoware precursor of *Ansoft Designer SV 2*, Ansoft *Serenade SV 8.5*, brought limited use of nonlinear-simulation tools to students and experimenters. With *Serenade SV 8.5*, you could simulate mixers, and you could simulate amplifier IMD — IMD from two tones only, to be sure — to a maximum of four nonlinear ports, meaning that *Serenade SV 8.5* could simulate mixers with up to four diodes or up to two transistors (or one transistor and two diodes — you see the strategy of the limitation). See **Figs 6.A8**, **6.A9** and **6.A10**. You could simulate the conversion gain and noise figure of a mixer. Optimization was enabled. Realistic nonlinear libraries were included for several Siemens — now Infineon — parts. You could accurately predict whether or not a circuit you hoped would oscillate would *actually* oscillate, and assuming that it would, you could accurately predict its output power and frequency.

The harmonic-balance techniques used by Ansoft's nonlinear solver — and by the nonlinear solvers at the core of competing RF-fluent CAD products, such as Agilent *Advanced Design System* (ADS) — allowed you to simulate crystal oscillators as rapidly as you can simulate lower-*Q* oscillators based on LC circuits. (In *SPICE*, getting a crystal oscillator to start may be impossible without presetting current and/or voltages in key components to nonzero values, even if you try kick-starting it with a pulse as we do in this chapter's JFET VFO simulation.)

Students and CAD-minded radio amateurs alike miss the features made available in *Serenade SV 8.5* and hope that they will one day return in some form to the world of free demoware CAD software. In the meantime, if you're serious about pushing into RF CAD beyond what *SPICE* can do and someone you know has no use for their copy of *Serenade SV 8.5*, see if you talk them out of it!

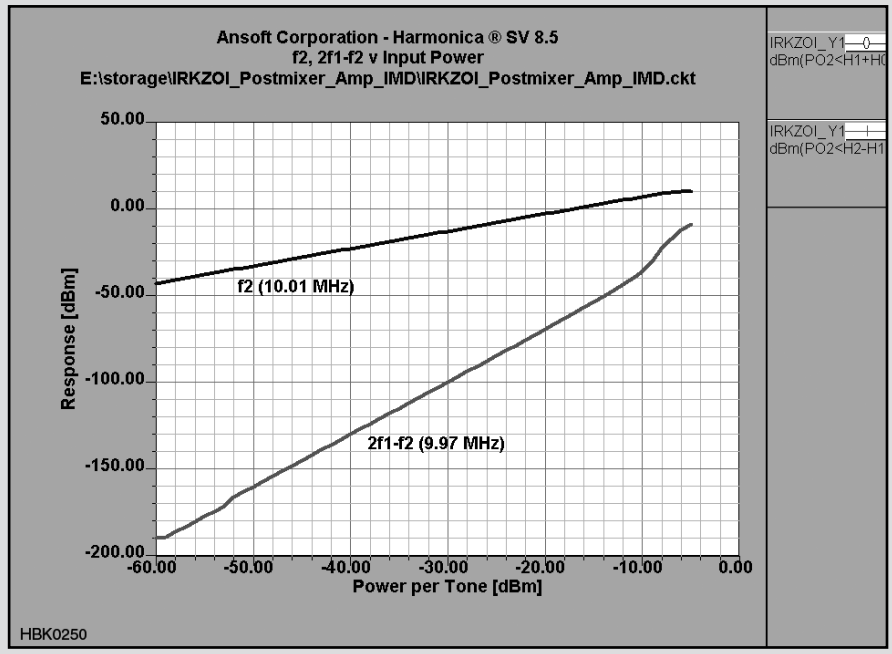


Fig 6.A8 — At this writing, the two-tone nonlinear simulation capabilities necessary to model third-order IMD were unavailable in free-demoware form. This simulation was done by Ansoft *Serenade Designer SV 8.5*, available from 2000 to 2005.

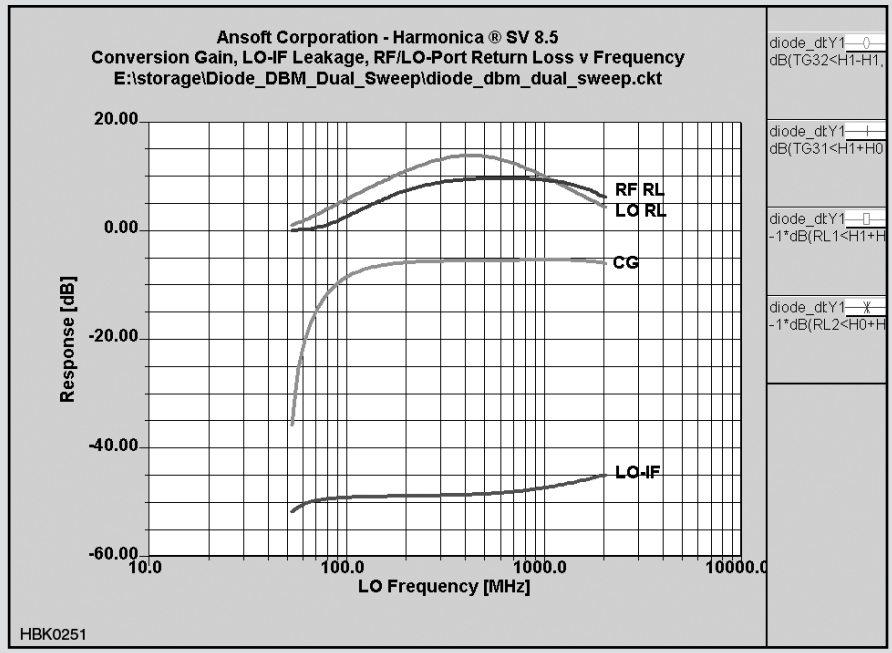


Fig 6.A9 — Professional RF circuit simulators can also simulate mixing and the small-signal characteristics of mixers, such as port return loss, conversion gain, and port-to-port isolation. (Serenade SV 8.5 *simulation*)

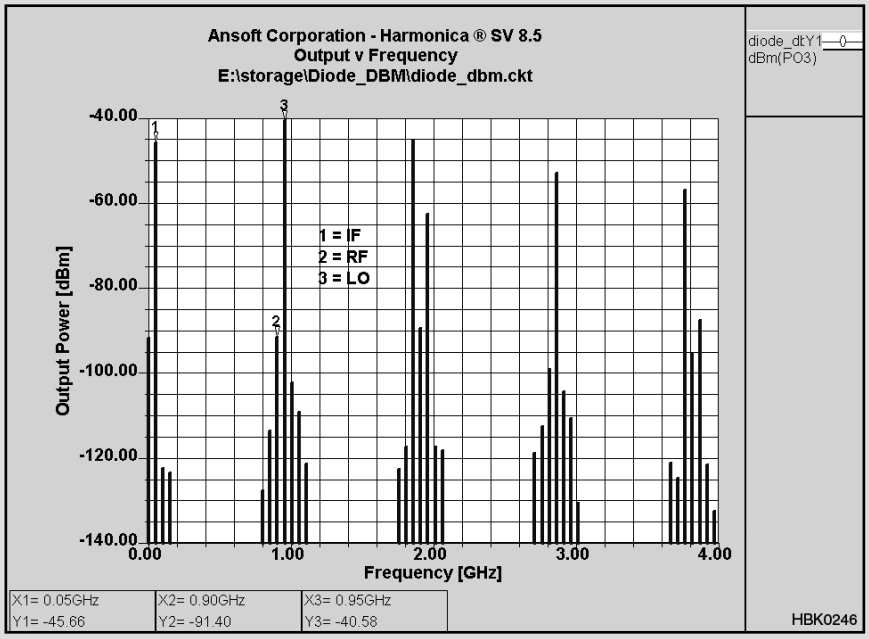


Fig 6.A10 — Output spectrum of a diode-ring doubly balanced mixer as simulated by *Serenade SV 8.5*. Note the dynamic range implicit in this graph: In a simulation that includes a local-oscillator (LO) signal at 7 dBm, we're seeing accurate values for IMD products nearly *140 dB weaker* without encountering mathematical noise — an achievement unapproachable with *SPICE*-based simulators.

6.2.7 Circuit CAD in the Radio Amateur's Toolbox

This chapter emphasizes the use of *SPICE* for circuit simulation because radio amateurs interested in circuit CAD will likely first experience it with a *SPICE*-based simulator and keep using *SPICE*. At this writing, *SPICE* is the only *nonlinear* simulator freely available to hobbyists. Radio amateurs seeking to enhance their knowledge of RF design techniques through circuit simulation will want to use an RF-fluent simulator instead of or in addition to *SPICE*, and *Ansoft Designer SV 2* can serve as a linear-simulation workhorse for this purpose. (Here we must differentiate between trialware and demoware: Although RF-fluent nonlinear simulation may be available in the feature-limited *trialware* versions of some EDA products, hobbyists need CAD capabilities that won't stop working in 30 to 90 days. We hasten to add that radio amateurs do not expect that such capabilities need be free, just affordable; the full versions of the RF-fluent simulators known to the author sell for thousands to tens of thousands of dollars.)

An expensive full-version simulator can mislead as, or more, easily than its freeware version in the absence of designer know-how teamed with *carefully and fully characterized performance data describing the actual behavior of simulatable real-world circuits*. Without tempering by experimental experience and constant comparison with real-world performance data, results obtained through CAD can lose their necessary real-world anchoring. Well-applied, however, computerized circuit simulation can greatly accelerate one's acquisition of the intuition and RF "street smarts" that make RF design an art *and* science.

6.3 Limitations of Simulation at RF

[*Experienced users of circuit simulation software are wary of using any software near or outside the boundaries of circuits and parameters for which it was intended and tested. RF simulation can present just such situations, leading to software failure and unrealistic results. Introduced and summarized in this section, several detailed papers by Dr Ulrich Rohde, N1UL, exploring simulation at RF are provided on the CD-ROM accompanying this Handbook. The papers are:*

- *"Using Simulation at RF" by Rohde, a survey of the issues of RF simulation and the techniques used in current modeling programs.*
- *"The Dangers of simple usage of Microwave Software" by Rohde and Hartnagel, a discussion of inaccuracies introduced by device parameter measurement and model characteristics.*
- *"Mathematical stability problems in modern non-linear simulations programs" by Rohde and Lakhe, presenting various approaches to dealing with non-linear circuit simulation. — Ed.]*

While the precise lower bound of "RF" is ill-defined, RF effects start already at about 100 kHz. This was first noticed as self-resonance of high-Q inductors for receivers. In response, Litz wire was invented in which braided copper wires were covered with cotton and then braided again to reduce self-resonance effects.

As frequencies get higher, passive elements will show the effects of parasitic elements such as lead inductance and stray capacitance. At very high frequencies, the physical dimensions of components and their interconnections reach an appreciable fraction of the signal wavelength and their RF performance can change drastically.

RF simulators fall in the categories of *SPICE*, harmonic-balance (HB) programs and EM (electromagnetic) programs. The EM simulators are more exotic programs. Two types are common, the 2D (2.5) or 2-dimensional and the full 3-dimensional versions. They are used to analyze planar circuits, including vias (connections between layers) and wraparounds (top-to-ground plane connections), and solid-shapes at RF. They go far beyond the *SPICE* concept.

6.3.1 *SPICE*-based Simulators

SPICE was originally developed for low-frequency and dc analysis. (Modern *SPICE* programs are based on *SPICE3* from University of California – Berkeley.) While doing dc, frequency, and time domain simulations

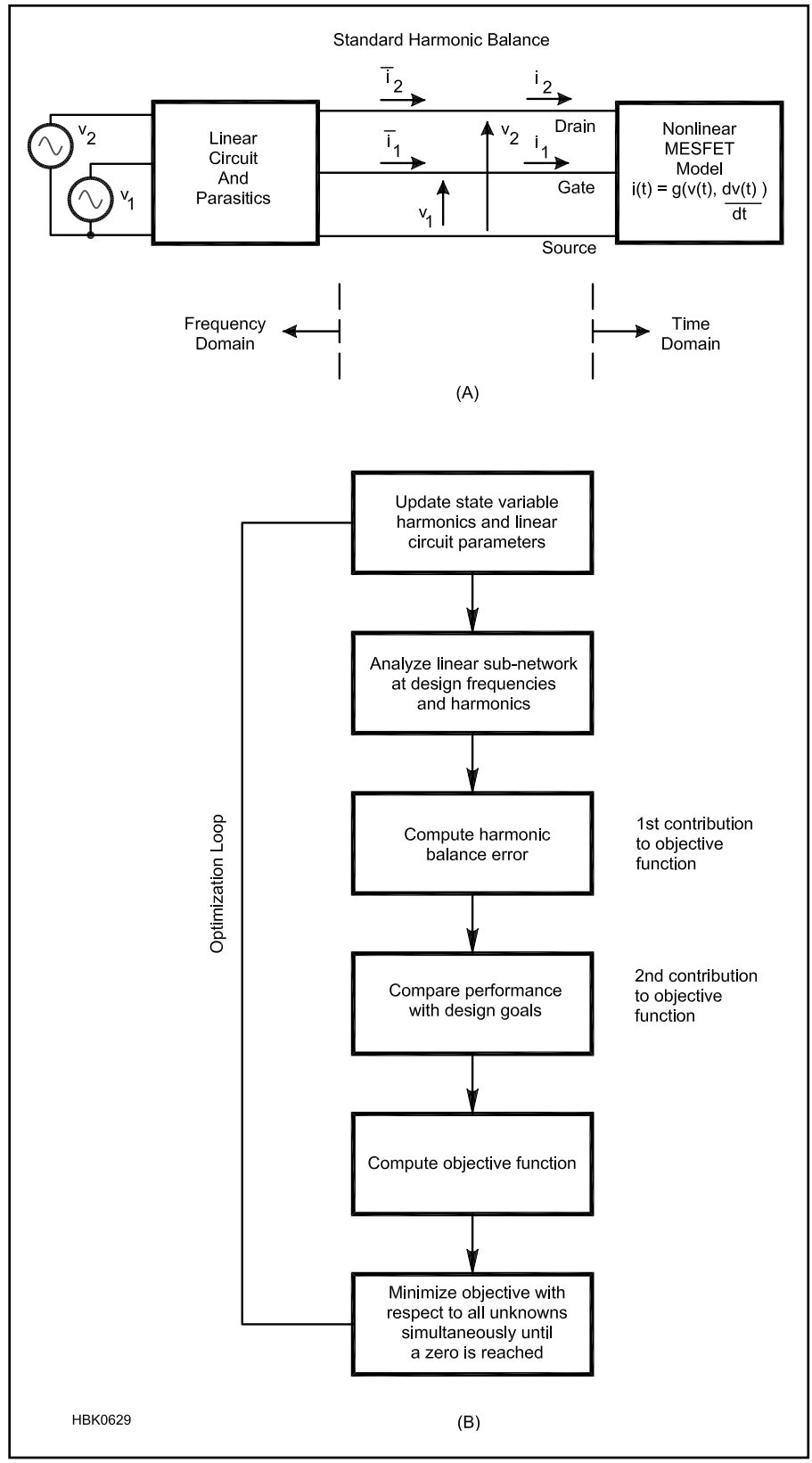


Fig 6.36 — (A) MESFET circuit partitioned into linear and nonlinear sub-circuits for harmonic-balance analysis. Applied gate and drain voltages, and relevant terminal voltages and currents, are indicated. (B) Flowchart of a general-purpose harmonic-balance design algorithm that includes optimization.

very well, *SPICE*-based simulation has some problems. The time domain calculation uses the very complex mathematics of the Newton-Raphson solution to nonlinear equations. These methods are not always stable. All kind of adjustments to the program settings may be necessary for the calculations to converge properly. Knowledge of the specifics of different types of electronic circuits can assist the user in finding an accurate solution by specifying appropriate analysis modes, options, tolerances, and suitable model parameters. For example, oscillators require certain initializations not necessary for amplifiers and bipolar transistors may need different convergence tolerances than do MOS circuits. Generally, *SPICE* finds a solution to most circuit problems. However, because of the nonlinearity of the circuit equations and a few imperfections in the analytical device models, a solution is not always guaranteed when the circuit and its specification are otherwise correct.

The next problem at RF is that the basic *SPICE* simulator uses ideal elements and some transmission line models. As we approach higher frequencies where the lumped elements turn into distributed elements and special connecting elements become necessary, the use of the standard elements ends. To complicate matters, active elements such as diodes and transistors force the designer to more complex simulators. Adding the missing component elements leads to highly complex models and problems of convergence in which the simulator gives an error advising of a numerical problem or more likely by failing to generate a solution.

SPICE also has problems with very high-Q circuits and noise analysis. Questions of the noise figure of amplifiers or phase noise of an oscillator cannot be answered by a *SPICE*-based program accurately. Noise analysis, if not based on the noise correlation matrix approach, will not be correct if the feedback capacitance (Im(Y12), the imaginary component of Y-parameter Y_{12}) is significant at the frequencies involved. Analysis of oscillators in *SPICE* does not give a reliable output frequency and some of the latest *SPICE* programs resort to some approximation calculations.

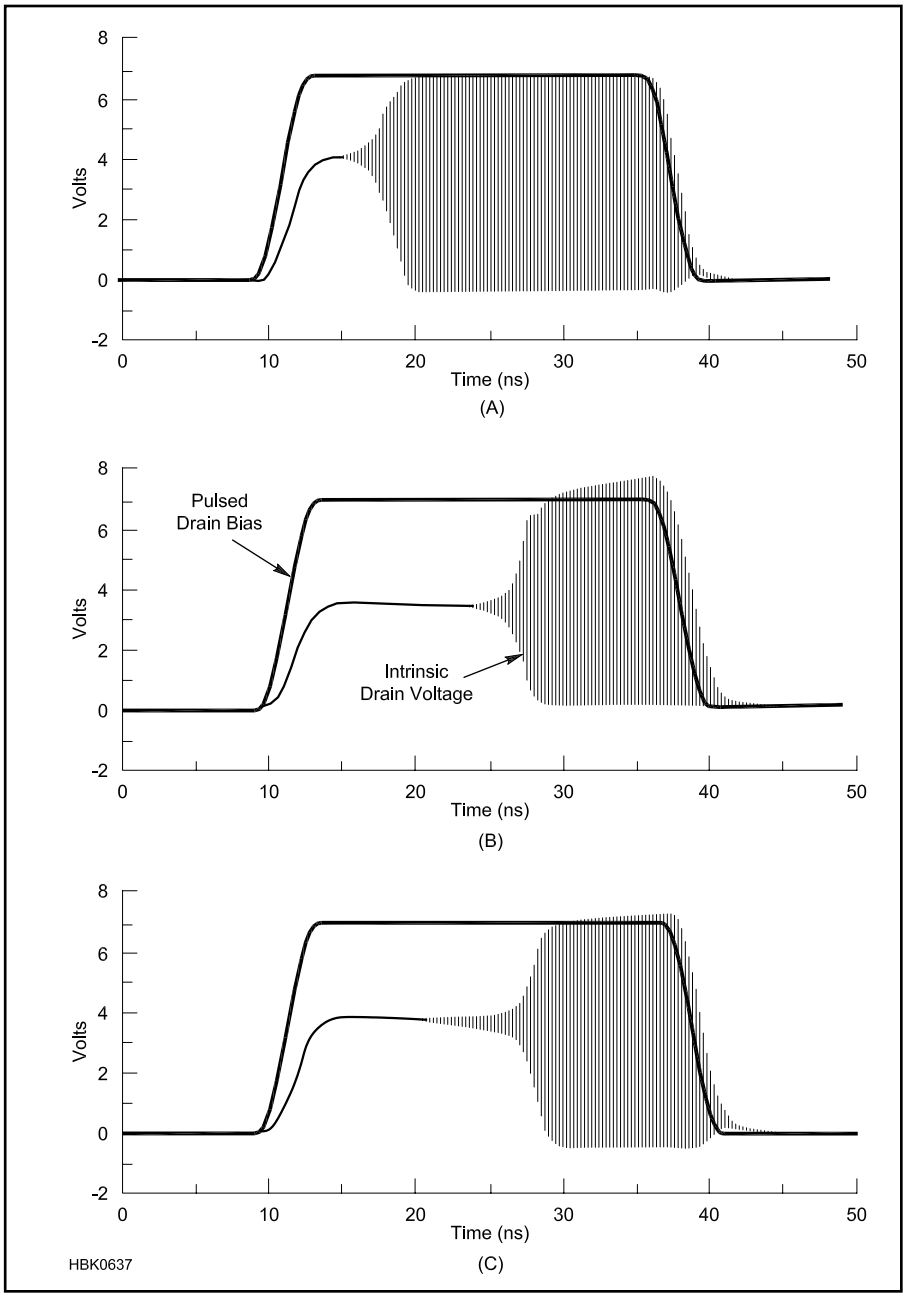


Fig 6.37 — (A) is the initial simulation of a *SPICE*-based simulator. (B) is the correct response of a pulsed microwave oscillator obtained by harmonic balance simulation using the Krylov-subspace solution. (C) is the *SPICE*-based simulation after 80 pulses of the drain voltage.

6.3.2 Harmonic-Balance Simulators

Harmonic balance (HB) analysis is performed using a spectrum of harmonically related frequencies, similar to what you would see by measuring signals on a spectrum analyzer. The fundamental frequencies are the frequencies whose integral combinations form the spectrum of harmonic frequency components used in the analysis. On a spectrum analyzer you may see a large number of signals, even if the input to your circuit is only one or two tones. The harmonic balance analysis must truncate the number of harmonically related signals so it can be analyzed on a computer.

The modern HB programs have found better solutions for handling very large numbers of transistors (>1 million transistors) and their math solutions are much more efficient, leading to major speed improvements. Memory management through the use of matrix formulations reduces the number of internal nodes and solving non-linear equations for transient analysis are some of the key factors to this success.

HB analysis performs steady-state analysis of periodically excited circuits. The circuit to be analyzed is split into linear and nonlinear sub-circuits. The linear sub-circuit is analyzed in the frequency domain by using distributed models. In particular, this enables straightforward intermodulation calculations and mixer analysis. The nonlinear sub-circuit is calculated in the time domain by using non-linear models derived directly from device physics. This allows a more intuitive and logical circuit representation.

Fig 6.36A diagrams the harmonic-balance approach for a MESFET amplifier. Fig 6.36B charts a general-purpose nonlinear design

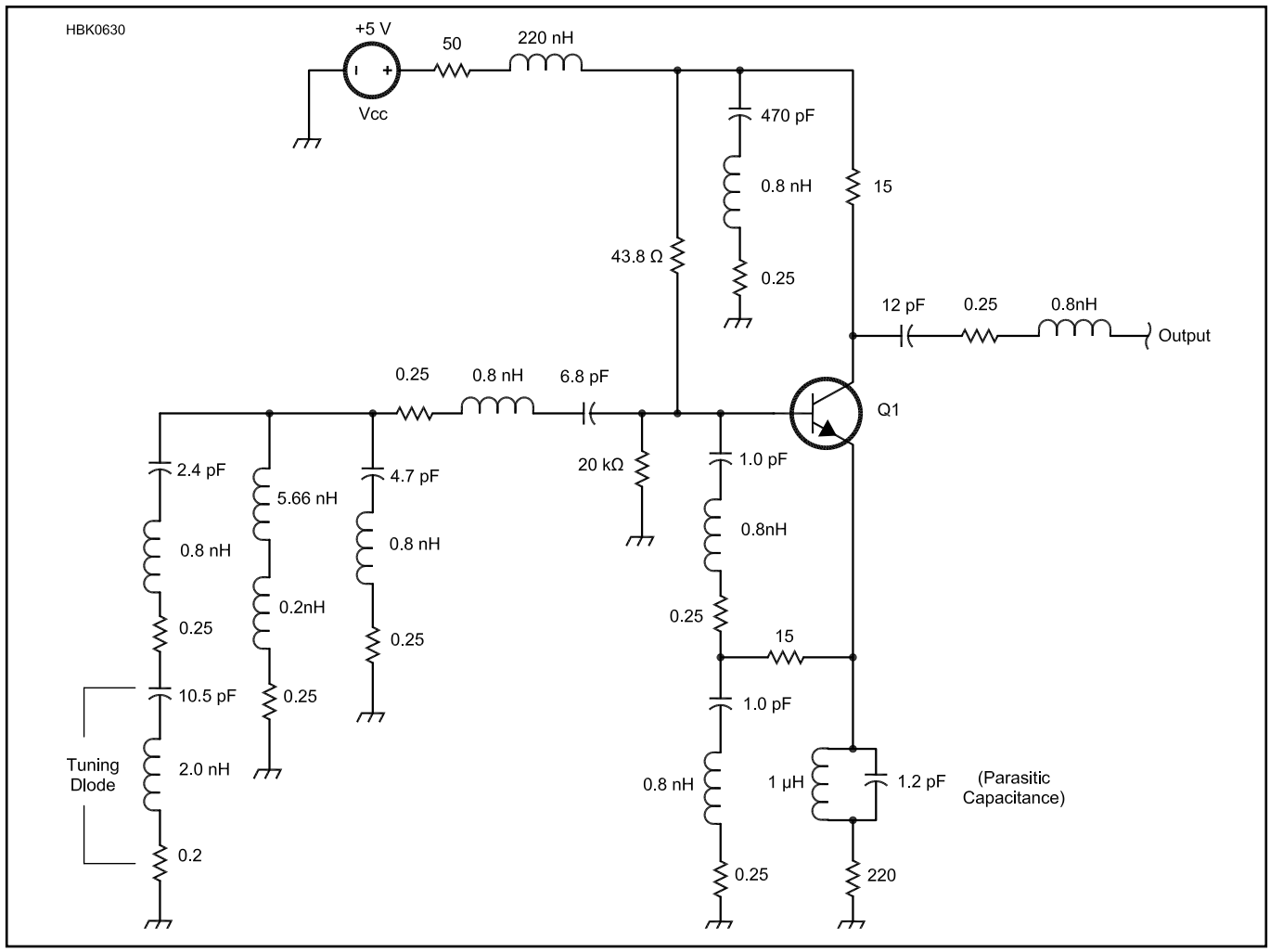


Fig 6.38 — Colpitts oscillator for 800 MHz with lumped elements modeled by their real values.

algorithm that includes optimization. Modern analysis tools that must provide accurate phase-noise calculation should be based on the principle of harmonic balance.

Analysis parameters such as Number of Harmonics specify the truncation and the set of fundamental frequencies used in the analysis. The fundamental frequencies are typically not the lowest frequencies (except in the single-tone case) nor must they be the frequencies of the excitation sources. They simply define the base frequencies upon which the complete analysis spectrum is built.

6.3.3 Contrasts in Results

The following time domain analysis is a good example of differences between *SPICE* and harmonic balance simulation. A microwave oscillator is keyed on and off and a transient analysis is performed. When using the standard *SPICE* based on *SPICE3*, the initial calculation shows an incorrect response after one iteration as seen in **Fig 6.37A**. It takes

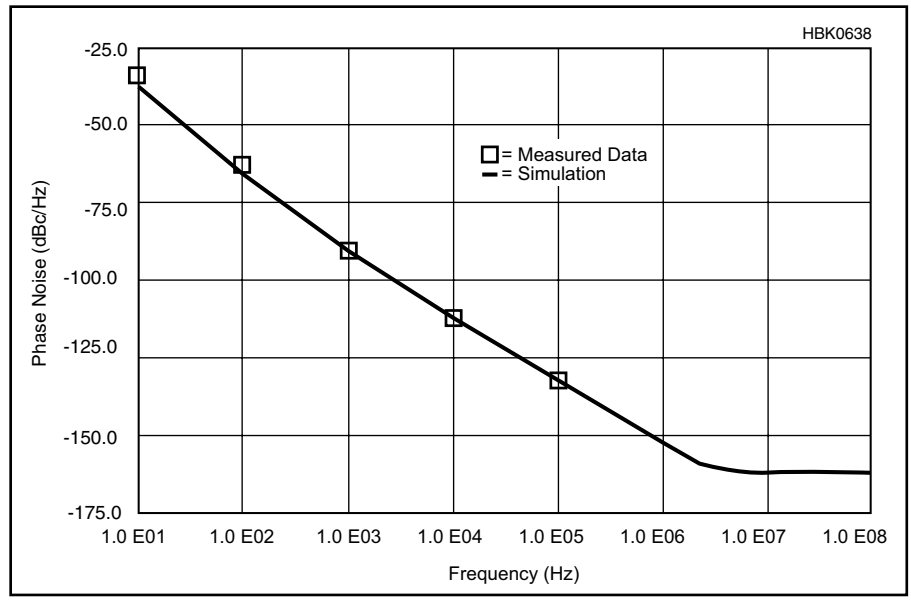


Fig 6.39 — Comparison between predicted and measured phase noise for the oscillator shown in Fig 6.38.

about 80 pulses (80th period of the pulsed drain voltage) until the simulation attains the correct answer (Fig 6.37C) of the Krylov-subspace-based harmonic balance in Fig 6.37B.

The frequencies involved need not be in the GHz range. Oscillators, in particular, can be very difficult to analyze at any frequency as shown by simulations of a low-MHz phase-shift oscillator and a 10 MHz Colpitts oscillator in the referenced papers.

Validating the harmonic-balance approach, **Fig 6.38** shows a BJT microwave oscillator entered into the schematic-capture module of a commercially available HB simulator (Ansoft Serenade 8.0); **Fig 6.39** compares this oscillator's simulated phase noise to measured data. HB analysis gives similarly accurate results for mixers.

6.3.4 RF Simulation Tools

PSPICE: This popular version of *SPICE*, available from Orcad (now Cadence, **www.cadence.com**) runs under the PC and Macintosh platforms. An evaluation version, which can handle small circuits with up to 10 transistors, is freely available, such as from **www.electronics-lab.com**. Contact Cadence for a full version or for more information. *AIM-spice* (**www.aimspice.com**) is a PC-version of *SPICE* with a revised user interface, simulation control, and with extra models. A student version can be downloaded. Table 6.1 earlier in this chapter shows other free *SPICE* offerings.

There are a number of PC-based *SPICE* programs in the $1000 range but they are designed more for switching power supplies and logic circuits optimization than RF. *ICAP4* (**www.intusoft.com/demos.htm**) and *MICROCAP9* (**www.spectrum-soft.com/index.shtm**) both have demonstration/evaluation versions available for download

Agilent, AWR, Ansys, and Synopsis offer very modern mixed-mode CAD tools and they combine the concept of *SPICE* with the advanced technologies. These are professional-quality tools, but if one can arrange to make use of them through a friend or associate, the results are worth investing the time to learn their use.

6.4 CAD for PCB Design

[With numerous PCB design software packages available and low-quantity, low-cost PCB manufacturing services accepting orders electronically, the development of PCBs has never been easier for the amateur. As with any assembly or manufacturing process, it is important to understand the vocabulary and technology in order to achieve the desired result. Thus, this section provides a detailed description of the entire process of PCB design. — Ed.]

The primary goal of using software for printed circuit board (PCB) design is the production of so-called PCB artwork — the graphic design used to create the patterns of traces that establish connectivity on the PCB. Historically, PCB artwork was created by hand on clear film using black tape and special decals which were then photographically reduced. However, free and low-cost programs specifically for the PCB design process are now widely available. These programs not only allow the creation of artwork efficiently and accurately, but produce the required ancillary files for commercial production, exchange information with schematic capture software, produce Bills of Materials (parts lists), and even include such features as three-dimensional visualization of the finished board. While artwork files can be shared with other people for PCB production, the "source" files used by the CAD program can typically only be used by other people who share the same program.

The decision to produce a PCB must take into account the nature of the circuit itself (for example, high frequency, low noise and high current circuits require additional care). Other considerations are time available, expense, available alternatives, quantity required, ability to share and replicate the design, and non-electrical characteristics such as thermal and mechanical, as well as desired robustness.

6.4.1 Overview of the PCB Design Process

The PCB design process begins with establishing the list of components in the circuit, the connections between the components, the physical outline/size of the board, and any other physical, thermal and electrical constraints or design goals. Much of the connectivity and component information is reflected in the schematic for a circuit, so in many cases the PCB layout process begins by entering the schematic in a schematic capture program which may be integrated with the PCB CAD program or standalone. (Schematic capture is not required for PCB layout.) Once the schematic is entered, there may be other options possible such as simulating the circuit as described in the preceding sections. A clean, well-organized schematic that is easily modified is an asset regardless of the circuit production and construction methods.

With input from the schematic and other information, the board outline is created, mounting and other holes placed, the components positioned, and the pattern of traces created. Once the layout is complete, in many cases it is possible to run a design rules check — the equivalent of a "spell checker. " Design rules include component connections and other information to check for problems related to connectivity and manufacturability. This step can save a great deal of time and expense by catching errors that could be fixed by hand, but would otherwise negate some of the benefits of a PCB.

The final step in the PCB layout program is to produce the collection of up to a dozen or so different files required for PCB production. In brief, the list includes the artwork for the pattern of traces, files for producing the board outline, solder masks, silk screens and holes.

The user then uploads the set of files to a PCB manufacturer. As quickly as two to three days later an envelope will be delivered with the freshly-minted boards ready for assembly! Alternatively, the user may create the board "in house" using photomechanical or other processes based on the output files from the software, as is discussed in the **Construction Techniques** chapter along with PCB assembly techniques.

6.4.2 Types of PCB Design Software

PCB software varies in features, function and cost, but for the radio amateur, the most interesting software for introductory use fall into the following categories:

1) Open-Source: PCB design software such as *GNU PCB* (**pcb.gpleda.org**, for *Linux*, Mac OS X) and *KiCad* (**kicad.sourceforge.net/wiki/index.php/Main_Page**, for *Linux*, Mac OS X, and *Windows*, includes schematic capture) are free to use and have no artificial restrictions. Support is through user-forums. Source code is available for the user to modify. *gschem* is a schematic capture sister program to *GNU PCB*.

2) Free, restricted-use/restricted-feature commercial: At least one company makes a version of their PCB and schematic software that is free to use for non-commercial purposes. Though it is restricted in number of layers (two) and maximum board size (4 × 3.2 inches), *Eagle PCB "Light Edition"*

(**www.cadsoftusa.com**, for *Linux*, Mac OS X, *Windows*) is very popular among hobbyists. Files can be shared with others; the resulting industry-standard files can be sent to nearly any PCB manufacturer. *Eagle* also contains a schematic entry program.

3) Free, restricted-output commercial: Several PCB manufacturers offer schematic and PCB software with a proprietary output format tied to their PCB manufacturing service. *PCB123* from Sunstone Circuits (**www.sunstone.com**, for *Windows*) is one such offering, including schematic capture and layout software with up to four layers and board sizes up to 12 × 18 inches (double sided). For an additional fee (per design), industry-standard files can be exported. Schematic entry is included. *Express PCB* (**www.expresspcb.com**, for *Windows*) also provides schematic capture and PCB layout capability, tied to the Express PCB board fabrication services, including the fixed-size (3.8 × 2.5 inches) Miniboard service. Advanced Circuit's proprietary *PCB Artist* software (**www.4pcb.com**, for *Windows*) includes the ability to import netlists.

4. Low-cost commercial: Eagle and many other companies offer PCB and schematic software at a range of prices from $50 to many thousands of dollars. Several versions are typically offered from each company, usually based on limitations on board size, schematic size/complexity and features such as auto-routing. Schematic entry may be included in some packages, or be a separate purchase.

PCB design software manuals and tutorials discuss the basic operation but also special keystrokes and other shortcuts that make operations such as routing traces much more efficient.

The first time designing and ordering a PCB can be daunting, so keep the initial job simple and pay attention to details (and read the instructions). When starting to use a specific software package, join a user's support group or forum if one is available. Request sample designs from other users and experiment with them to see how they are constructed and what files are required in the output data set. Once you are comfortable with the tools, you can begin on a design of your own.

6.4.3 Schematic Capture

The first step in PCB design is to create a schematic. It is possible to design a layout directly from a paper schematic, but it is much easier if the schematic is entered (or "captured") in electronic form. Schematic capture software has two outputs — the visual schematic and the component and connectivity data for subsequent PCB layout. These two separate requirements can make some operations during schematic entry more complicated than what would seem at first glance necessary. Bear in mind however, that the user is creating not only a clear graphic representation of the circuit, but of the underlying electrical connectivity.

Schematics are generally entered on a (virtual) page usually corresponding to common paper sizes — for example, 11 × 17 inches. More complicated schematics can span multiple pages, using special labels or components to indicate both visual and electrical connectivity. Often one can group logically-related elements into a module that can then be referenced as a "black box" on a higher-level schematic. For complex circuits, these features are extremely useful and make the difference between a jumbled diagram that is difficult to use and an organized, compact diagram that efficiently communicates the function and operation of the circuit.

COMPONENTS

The components (resistors, capacitors, etc) on a schematic are either selected from an existing library or created by the user and stored in a custom library. It is also possible to find components and/or additional libraries on the Internet, although each program has its own specific format.

Each component includes a great deal more than shape and pin numbers. A typical component library entry includes:

Symbol — This is the graphic representation shown on the schematic. Many components may have the same symbol (eg, the op amp symbol may be shared by many different types of op amps)

Pins — For each pin or point of electrical connection, the component model may specify the pin number, label (eg, "V_{DD}"), pin type (inverting, non-inverting) or pin functions (common).

PCB footprint — A given component may be available in a number of different packages (eg, DIP or surface mount). Many components may have the same physical footprint (eg, op amps, comparators and optoisolators could all map to the same eight-pin DIP footprint). Footprints include the electrical connections (pins) as well as mechanical mounting holes and pad sizes, and the component outline.

Value — Many components such as resistors and capacitors will have identical information except for a difference in value. All ¼-watt resistors may be instances of the same component, differing only in value and designator.

Designator — The unique reference to the component, such as R1, C7, D3. This is assigned when the component is used (often automatically and in sequence).

Source information — Part number, vendor, cost, etc. This information is for the Bill of Materials.

Components are typically placed on the schematic by opening a library and searching for the desired component. It may be tempting for the beginner to select a component that looks "about right" when faced by a long list of components in some libraries. However, even at this early stage, the physical PCB often must be taken into account. For example, either "1/8W Resistor, Axial" or "1W Resistor, Upright" will result in the same neatly drawn resistor symbol on the schematic but in the subsequent step of using the component data to create a PCB, the footprints will be dramatically different.

It is not at all uncommon to add new components to the library in the course of creating a schematic. Since many components are closely related to existing devices, the process often consists of selecting an existing schematic symbol, editing the shape and/or component data, creating a new label, and associating the part with an existing footprint. Adding a specific type of op amp is an example. This usually only needs to be done once since symbols can be saved in a personal library (and shared with others). It is usually easier to modify a part that is close to what is desired than to "build" a new part from scratch.

Component symbols can generally be rotated and flipped when placing the component instance on the schematic. Designators (R1, T34, etc) can be assigned and modified by the user although the default designators are usually selected sequentially.

CONNECTIONS

The schematic software will have a mode for making electrical connections, called "nets." For example, one might click on the "draw net" symbol then draw a line using the mouse from one pin to another pin, using intermediate mouse clicks to route the line neatly with 90° turns on a uniform grid. Internally, the software must not only draw the visual line, but recognize what electrical connectivity that connection represents. So one must click (exactly) on a component pin to start or end a line or when making a connection between two lines that intersect, explicitly indicate a net-to-net connection (often with a special "dot" component). The connections on a schematic can often be assigned additional information, such as the desired width of the trace for this connection on the PCB or a name assigned by the user, such as "input signal."

Not all connections on a schematic are drawn. To make any schematic — electronic or hand-drawn — more readable, conventions are often employed such as ground or power symbols or grouping similar connections into busses. Schematic capture software often supports these conventions. In some cases, components may be created with implicit power connections; in these cases the connections may not even be noted on the

schematic but will be exported to the PCB software. However, as a general rule, software aimed at beginning PCB designers will not require the use of these advanced features.

Since it is often possible for component pins to be assigned attributes such as "power input", "output," "input," and so on, some schematic entry programs allow one to do an early design check. The program can then flag connections between two outputs, inputs that are missing connections, and so on. This is not nearly as helpful or complete as the Design Rule Check discussed below.

Free text can be placed on the schematic and there will be a text block in a corner for date, designer, version, title and the other information that identifies the schematic.

NETLISTS

Once the components are placed and connections made, the schematic may be printed and any output files for the PCB layout software produced. The connectivity and component information needed for PCB layout is captured in a *netlist* file. The flow from schematic entry to PCB may be tightly integrated, in which case the user may switch between schematic and PCB like two views of the same design (which they are). However, most schematic software will generate a separate netlist to be used by PCB layout software, whether integrated or a separate program. The netlist can also be exported to an external circuit simulation program or be used by an integrated simulator program. (See the first part of this chapter for more information on circuit simulation.)

Netlists are often human-readable text files and in most cases it is possible to create a netlist file manually. In the absence of a schematic entry program, this allows the user to take a hand-drawn schematic, extract the connectivity information, and create the netlist for the PCB program to perform design rule checks. However, a netlist is generally not required for the PCB layout software; the user will also have the option to create a PCB on-the-fly, adding components and connections as they wish.

ANNOTATION AND BILL OF MATERIALS

The important features of *forward* and *backward annotation* enter at the interface between schematic entry and PCB layout. It is not uncommon during the PCB layout process to either come across some design deficiency or realize that a change to the schematic could produce a design that would be easier to lay out. Likewise, a review of the schematic partway through the PCB layout process could reveal some needed design change. In the case of changes to the PCB (perhaps changing some pins on a connector to make routing easier), back annotation can propagate the changes "backward" to the schematic. The connectivity data will be updated; however the user may need to manually route the connection lines to neaten up the schematic. Likewise, changes to the schematic when the PCB is already (partially) routed are known as forward annotations and like the schematic, while the connectivity is updated the user will likely need to manually route the traces. Neither forward nor back annotation is necessary, but is useful in keeping the schematic and PCB consistent. In their absence, the user is strongly urged to keep the schematic and PCB up to date manually to avoid time-consuming problems later on.

Finally, the underlying data in the schematic can be used to produce a *Bill of Materials* (BOM). A BOM lists all the components of the schematic, typically ordered by reference designator(s), and may even be exportable for online ordering.

6.4.4 PCB Characteristics

PCB CONSTRUCTION

It is useful to know a little bit about PCB construction in order to make sense of the PCB design process. **Fig 6.40** shows the basic structure of a PCB and some of its design elements (discussed in later sections). The laminate material provides a stable, insulating substrate with other known characteristics (thermal, dielectric, etc). Copper is bonded to one or both sides and selectively removed (usually chemically) to leave traces and pads. The pads provide points of connection for components. Though electrical connectivity is crucial, it is important to remember that the solder and pads provide mechanical and thermal connectivity as well. Pads may be drilled for mounting *through-hole* components or left undrilled for *surface-mount* components.

A separate electrochemical process plates the inside surface of *plated-through holes* to provide connectivity between upper and lower pads. Plated-through holes whose sole purpose is to provide electrical connectivity between layers of a PCB are known as *vias*, shown in **Fig 6.41**. Since they do not need to accommodate a component lead, their hole and pad size are smaller.

While two-layer boards can mount components on either side, most PCBs will have a primary side called the *component side* upon which most of the components will be placed, and a *solder side* dominated by soldered pins and traces. Where high density is required, surface mount (and sometimes through-hole) devices are mounted on both sides, but this is considerably more complex.

Multi-layer boards are essentially a stack of two or more two-layer boards, with an insulating layer between each board. Plated-through holes make connections possible on every layer, and the laminate material is proportionally thinner so the entire multi-layer board is roughly the same thickness as a regular two-layer board. Vias that join selected, adjacent copper layers without connecting the entire stack of layers are called "buried" or "blind" vias and are typically only needed for very dense designs. Multi-layer boards provide much more flexibility in routing signals and some other benefits such as dedicated layers for power distribution and grounding, but at often substantial additional cost.

PCB MANUFACTURING SPECIFICATIONS

Unless the board is manufactured by the hobbyist, the PCB files are sent out to be manufactured by a *board house*. The most important issue for the amateur may be the pricing policies of the board house. Board size, quantity, delivery time, number of layers, number and/or density of holes, presence of solder masks and silk screens, minimum trace/separation width, type of board material, and thickness of copper will all influence pricing. One cost-saving option of the past, a single-layer board, may not be offered with low-cost, low-volume services — two layers may be the simplest option and it results in a more robust board. [Note that most ordering specifications use English units of inches and ounces. Offshore board houses may use both English and metric units, or be metric-only. English units are used here because they are the most common encountered by hobbyists. — *Ed.*]

The second issue to consider is manufacturing capabilities and ordering options. These will vary with pricing and delivery times, but include the following:

Board material and thickness — FR-4 is the most popular board material for low volume PCBs; it consists of flame-resistant woven fiberglass with epoxy binder. Typical thickness is 0.062 inch (1/16 inch), but thinner material is sometimes available. Flexible laminates are also available at greater cost and longer delivery time. Special board laminates for microwave use or high-temperature applications are also available.

Copper thickness — Expressed in ounces per square foot, typical values are 1-2 oz (1 oz corresponds to 0.0014 inch of thickness.) Other values may not be available inexpensively for small volumes. Inner layers on multi-layer boards may be thinner — check if this is important. Most board designs can assume at least 1 oz copper for double-sided boards; trace width is then varied to accommodate any high current requirements.

Layers — Two-layer boards are the most common. Because of the way PCBs are manufactured, the number of copper layers will be multiples of two. For quick-turn board houses, usually only two or four layer boards are available. PCBs with more than two layers will always be more expensive and

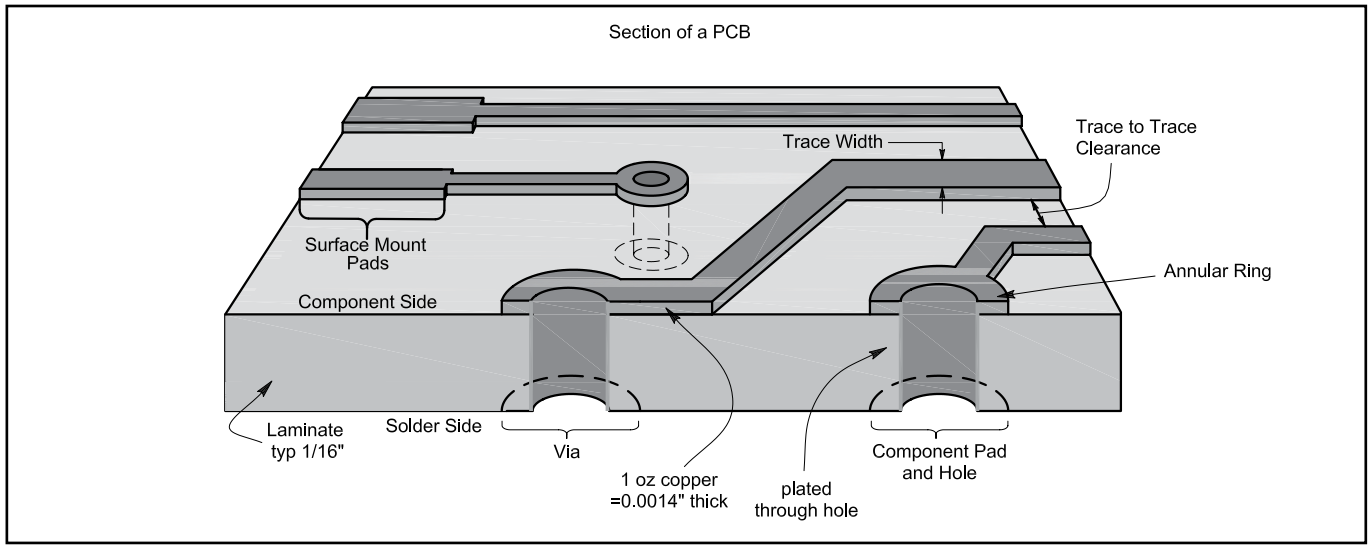


Fig 6.40 — The various elements of PCB construction and specification.

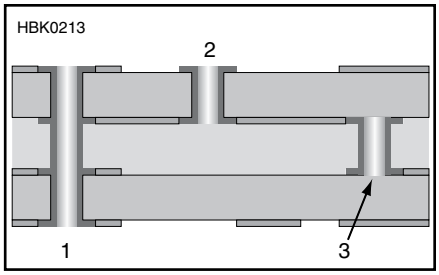


Fig 6.41 — *Via* refers to a plated-through hole that connects one board layer to another. Vias are used for signal, power, or ground connections and even for ventilation. Different via types include through-hole (1); blind (2), and buried (3).

often take longer to manufacture.

Minimum hole size, number, and density of holes — Minimum hole size will rarely be an issue, but unusual board designs with high hole density or many different hole sizes may incur additional costs. Be sure to include vias when specifying minimum hole size. Some board houses may have a specific list of drill sizes they support. Note that you can often just edit the drill file to reduce the number of different drill sizes.

Minimum trace width and clearance — Often these two numbers are close in value. Most board houses are comfortable with traces at least 0.010 inch in width, but 0.008 and 0.006 inch are often available, sometimes at a higher cost.

Minimum annular ring — A minimum amount of copper is required around each plated-through hole, since the PCB manufacturing process has variations. This may be expressed as the ratio of the pad size to hole size, but more commonly as the width of the ring.

Edge clearances — Holes, pads, and traces may not be too close to the edge of the board.

Board outline and copies — There may be options to route the outline of the board in other shapes than a rectangle, perhaps to accommodate a specific enclosure or optimize space. If multiple copies of a board are ordered, some board houses can *panelize* a PCB, duplicating it multiple times on a single larger PCB (with a reduction in cost per board). These copies may be cut apart at the board house or small tabs left to connect the boards so assembly of multiple boards can be done as a single unit.

Tin plating — Once the traces and pads have been etched and drilled, tin plating is usually applied to the exposed copper surfaces for good soldering.

Solder mask — This is a solder-resistant coating applied after tin plating to both sides of the board covering everything except the component mounting pads. It prevents molten solder from bridging the gaps between pads and traces. Solder mask is offered except by the quickest-turn services. Green is the most common color, but other colors may be available.

Silkscreen — This is the ink layer, usually white, on top of the solder mask that lays out component shapes and designators and other symbols or text. A minimum line width may be specified — if not specified, try to avoid thin lines. All but the quickest-turn services typically offer silk screening on one or both sides of the PCB.

6.4.5 PCB Design Elements

The schematic may not note the specific package of a part, nor the width or length of a connection. The PCB, being a physical object, is composed of specific instances of components (not just "a resistor," but a "¼-watt, axial-lead resistor mounted horizontally," for example) plus traces — connections between pins of components with a specific width and separation from other conductors. Before discussing the process of layout, we briefly discuss the nature of components and connections in a PCB.

COMPONENTS

A component in a PCB design is very similar to its counterpart on the schematic. **Fig 6.42** shows the PCB footprint of an opto-interrupter, including graphics and connectivity information. The footprint of a component needs to specify what the footprint is like on all applicable copper layers, any necessary holes including non-electrical mounting holes or slots, and any additional graphics such a silkscreen layer.

Take a common ¼-watt axial-lead resistor as an example. This footprint will have two pins, each associated with a pad, corresponding to the resistor's two leads. This pad will appear on both the top and bottom layers of the board, but will also have a smaller pad associated with inner layers, should there be any. The hole's size will be based on the nominal lead diameter, plus some allowance (typically 0.006 inch). The pad size will be big enough to provide a reasonable annular ring, but is usually much larger so as to allow good quality soldering. The pins will be labeled in a way that corresponds to the pin numbering on the schematic symbol (even though for this component, there is no polarity). A silkscreen layer will be defined, usually a box within which the value or designator will appear. The silkscreen layer is particularly useful for indicating orientation of parts with polarity.

More complicated parts may require additional holes which will not be associated with a schematic pin (mounting hole, for example). These are usually added to the part differently

than adding a hole with a pad — in this case, the hole is desired without any annular ring or plating. The silkscreen layer may be used to outline the part above and beyond what is obvious from the pads, for example, a TO-220 power transistor laying on its back, or the plastic packaging around the opto-interrupter in Fig 6.42.

As with schematic entry, it is not uncommon to have to modify or create a new PCB footprint. Good technical drawings are often available for electronic parts; when possible the user should verify these dimensions against a real part with an inexpensive dial caliper. It is also useful to print out the finished circuit board artwork at actual size and do a quick check against any new or unusual parts.

TRACES

Traces are the other main element of PCB construction — the copper pathways that connect components electrically. PCB traces are merely planar, flat wires — they have no magical properties when compared to an equivalent thickness of copper wire. At VHF/UHF/microwave frequencies and for high-speed digital signals, PCB traces act as transmission lines and these properties need to be accounted for, and can be used to advantage, in the design.

There are few constraints on traces apart from those such as minimum width and clearance imposed by the board house. They are created by chemical etching and can take arbitrary shapes. In fact, text and symbols may be created on copper layers which may be handy if a silkscreen is not included. Traces may be of any length, vary in width, incorporate turns or curves, and so on. However, most traces will be a uniform width their entire length (a width they will likely share with other traces carrying similar signals), make neat 45° or 90° corners, and on two-layer boards have a general preference for either horizontal travel on the component side or vertical travel on the solder side.

The same considerations when building a circuit in other methods applies to PCB design, including current capacity (width of trace, thickness of copper), voltage (clearance to other signals), noise (shielding, guarding, proximity to other signals), impedance of ground and power supplies, and so on.

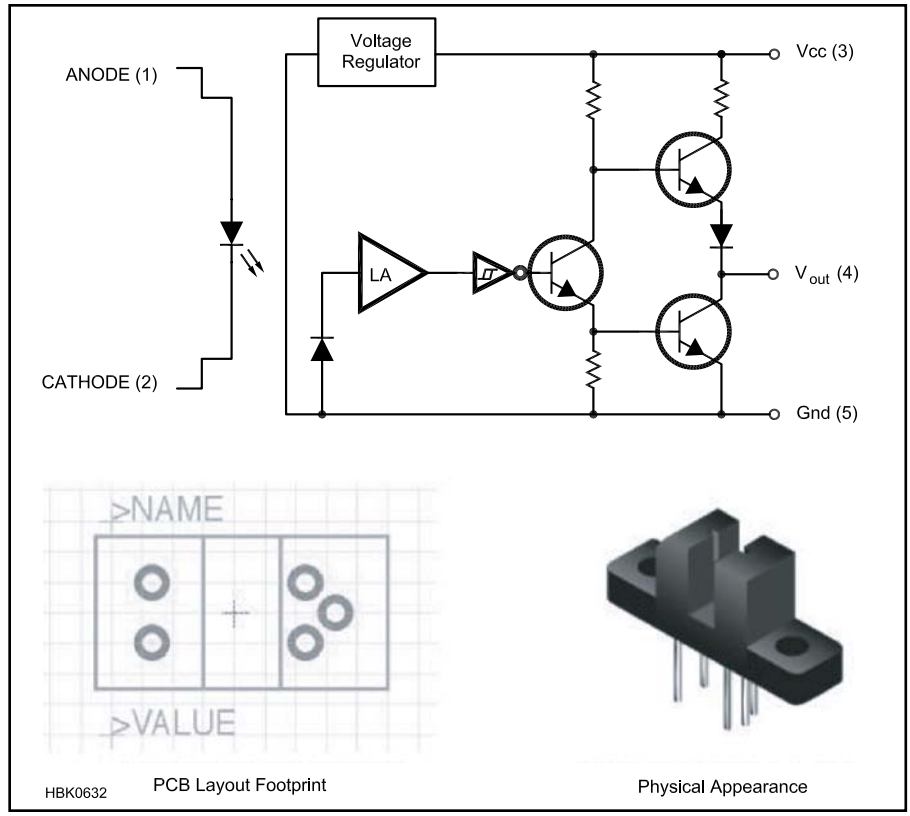


Fig 6.42 — The PCB footprint for a component, such as the opto-interrupter shown here, combines electrical connectivity as defined in the part's schematic and the part's physical attributes.

6.4.6 PCB Layout

With a schematic and netlist ready and all of the PCB characteristics defined, the actual layout of the PCB can begin.

BOARD SIZE AND LAYERS

The first step in PCB layout is to create the board outline to contain not only the circuit itself and any additional features such as mounting holes. For prototype or one-off designs, the board is often best made a bit larger to allow more space between components for ease in testing and debugging. (Some low-cost or freeware commercial PCB software imposes limits on board size and number of components.) The board outline may be provided in a default size that the user can modify, or the user may need to enter the outline from scratch.

As discussed above, rectangular board shapes are generally acceptable, but many board houses can accommodate more complex outlines, including curves. These outlines will be routed with reasonable accuracy and may save an assembly step if the PCB needs a cutout or odd shape to fit in a specific location.

While the software may not require deciding at the start how many layers the PCB will use, this is a decision the user should make as early as possible, since the jump from two to four or more will have a big impact on routing the traces as well as cost!

For your initial design, start with a two-layer board for a simple circuit that you have already built and tested. This will reduce the number of decisions you have to make and remove some of the unknowns from the design process.

COMPONENT PLACEMENT

Good component placement is more than half the battle of PCB layout. Poor placement will require complicated routing of traces and make assembly difficult, while good placement can lead to clean, easy-to-assemble designs.

The first elements placed should be mounting holes or other fixed-location features. These are often placed using a special option selected from a palette of tools in the software rather than as parts from a library. Holes sufficient for a #4 or #6 screw are usually fine; be sure to leave room around them for the heads of the screws and nut driver or standoff below. These will be non-plated-through holes with no pad (though the board house may plate all the holes in a board, regardless).

Depending on the software and whether schematic capture was performed, the board outline may already contain the footprints of all the circuit components (sometimes stacked in a heap in one corner of the board) and the netlist will already be loaded. In this case, components may be placed by clicking and dragging the components to the desired location on the board. Most PCB programs have a "rat's-nest" option that draws a straight line

for each netlist connection of a component, and this is a great aid in placement as the connections between components are apparent as the components are moved around. (See **Fig 6.43**) However, connections are shown to the nearest pin sharing that electrical connection; thus, components such as decoupling capacitors (which are often meant to be near a specific component) will show rats-nest connections to the nearest power and ground pins and not the pins the designer may have intended. These will have to be manually edited.

The PCB layout software may offer *auto-placement* in which the components are initially arranged automatically. The beginner should certainly feel free to experiment and see how well this tool performs, but it is likely not useful for the majority of designs.

PCBs need not be arranged to precisely mimic the schematic, but it is appropriate to place components in a logical flow when possible so as to minimize the length of traces in the signal path. Sensitive components may need to be isolated or shielded from other components, and grounding and decoupling attended to, just as one would do with a point-to-point soldered version.

If the PCB is being designed "on-the-fly" or using an imported netlist, components may need to be selected and placed on the board manually using the libraries of parts in the PCB software. Not all design software makes this task simple or fast — in particular, the description of component footprints may be confusing. The use of highly-condensed industry standard or non-uniform naming conventions often means the user needs to browse through the component library to see the different types of components. Resistors, diodes and capacitors seem particularly prone to a propagation of perplexing options. One solution is to open an example PCB layout and see what library element that designer used for resistors, LEDs and so on. Here, the PCB layout software directed at hobbyists may be superior in that there are fewer options than in professional programs.

During placement the user will find that different orientations of components simplify routing (for example, minimizing the number of traces that have to cross over each other or reducing the trace lengths). Components are generally rotated in increments of 90°, although free rotation may be an option. The user is strongly urged to maintain the same orientation on like devices as much as possible. Mixing the position of pin 1 on IC packages, or placing capacitors, diodes, and LEDs with random orientation invites time-consuming problems during assembly and testing that can be minimized by consistent, logical layout.

Placement and orientation of components can also affect how easily the final PCB can be assembled. Allow plenty of space for sockets, for example, and for ICs to be inserted and removed. Components with a mechanical interface such as potentiometers and switches should be positioned to allow access for adjustment. Any components such as connectors, switches, or indicators (eg, LEDs) should be positioned carefully, especially if they are to protrude through a panel. Often this will involve having the component overhang the edge of the PCB. (Beware of the required clearance between copper traces and pads to the edge of the PCB.)

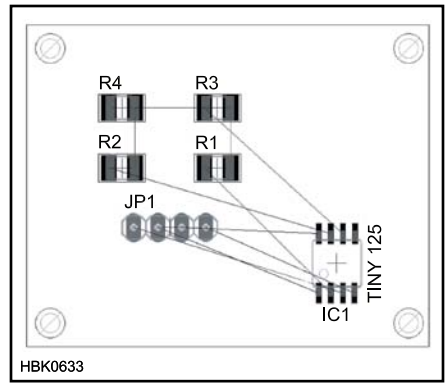


Fig 6.43 — The rat's nest view during PCB layout shows the direct connections between component pins. This helps the designer with component placement and orientation for the most convenient routing.

Components should include a silkscreen outline that shows the size of the whole component — for example, a transistor in a TO-220 package mounted flat against the PCB should have an outline that shows the mounting hole and the extent of the mounting tab. The user should also consider the clearance required by any additional hardware for mounting a component, such as nuts and bolts or heatsinks — including clearance for nut-drivers or other assembly tools.

Take care to minimize the mechanical stress on the PCB, since this can result in cracked traces, separated pads, or other problems. Utilize mounting holes or tabs when possible for components such as connectors, switches, pots. Use two-layer boards with plated-through holes even if the design can be single-layer. Component leads soldered to plated-through holes produce much stronger mechanical connections than single-layer boards in which the soldered pad is held only by the bond between copper and laminate and is easily lifted if too much heat or stress is imposed.

When prototyping a new design, add a few unconnected pads on the circuit board for extra components (eg, a 16-pin DIP, 0.4-inch spaced pads for resistors and other discrete components). Include test points and ground connections. These can be simply pads to which cut-off leads can be soldered to provide convenient test points for ground clips or to monitor signals.

Wires or cables can be directly soldered to the PCB, but this is inconvenient when swapping out boards, and is not very robust. Connectors are much preferred when possible and often provide strain relief for the wire or cable. However, if a wire is directly soldered to the PCB, the user should consider adding an unplated hole nearby just large enough to pass the wire including insulation. The wire can then be passed from the solder side through the unplated hole, then soldered into the regular plated-through hole. This provides some measure of strain relief which can be augmented with a dollop of glue if desired.

ROUTING TRACES

After placing components, mounting holes and other fixed-location features that limit component or trace placement, traces can be *routed*. That is, to complete all the connections between pins without producing short circuits.

Most PCB design programs allow components and traces to be placed on a regular grid, similarly to drawing programs. There may be two grids — a visible coarse-pitch grid, and a "snap" fine-pitch grid, to which components and other objects will be aligned when placed. It is good practice to use a 0.1- or 0.050-inch grid for component placement and to route traces on a 0.025-inch grid. While the "snap to grid" feature can usually be turned off to allow fine adjustment of placement, a board routed on a grid is likely to look cleaner and be easier to route.

The trace starts at a component pin and wends its way to any other pins to which it should be connected. Traces should start and end at the center of pads, not at the edge of a pad, so that the connection is properly recorded in the program's database. If a netlist has been loaded, most PCB software will display a rats-nest line showing a direct connection between pads. Once the route is completed, the rats-nest line for that connection disappears. The rat's-nest line is rarely the desired path for the trace and often not the correct destination. For example, when routing power traces, the user should use good design sense rather than blindly constructing a Byzantine route linking pins together in random order. For this reason, routing the power and ground early is a good practice.

High-speed, high-frequency, and low-noise circuits will require additional care in routing. In general, traces connecting digital circuits such as microprocessors and memories should not cross or be in close proximity to traces carrying analog or RF signals. Please refer to the **RF Techniques** and **Construction Techniques** chapters of this *Handbook*, and the references listed at the end of this section.

Manual routing is a core skill of PCB de-

sign, whether or not auto-routing is used. The process is generally made as simple as possible in the software, since routing will take up most of the PCB design time. A trace will be routed on the copper layer currently selected. For a single-sided board, there is only one layer for routing; for a two-layer board the component side and solder side can have traces; and for multi-layer boards additional inner layers can have traces. Often a single keystroke can change the active copper layer (sometimes automatically inserting a via if a route is in progress). The trace is drawn in straight segments and ends at the destination pin. When routing, 90° corners are normally avoided — a pair of 45° angles is the norm. **Fig 6.44** shows some sample traces.

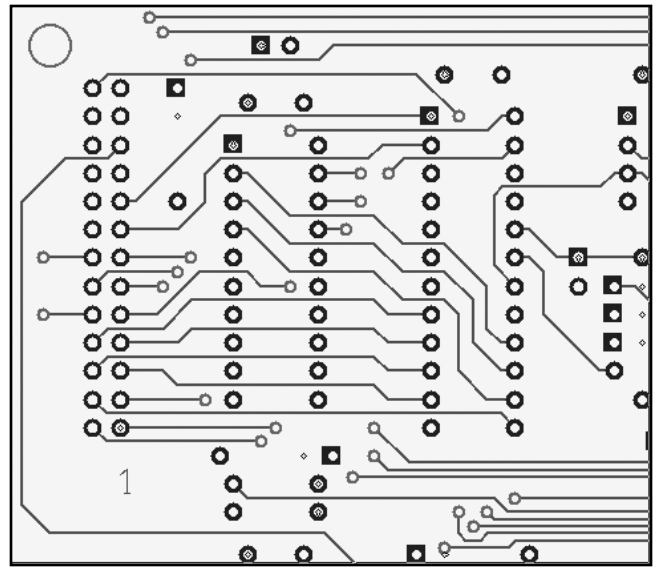


Fig 6.44 — This example shows traces on the side of the PCB for horizontal routing. Traces are routed between pins of ICs. The smaller pads are for vias to a different layer of the PCB.

It is good practice on a double-sided board to have one side of the board laid out with mostly horizontal traces, and the other side laid out with mostly vertical traces. A trace that needs to travel primarily vertically can do so on the side with vertical traces and use a via to move to the other side to complete the horizontal part of the route.

It is easiest during testing and debugging to route most traces on the bottom (solder) side of the board — traces on the component side often run under ICs or other components, making them hard to access or follow. It is often much clearer to connect adjacent IC or connector pins by routing a trace that leaves one pad, moves away from the IC or connector, then heads back in to the adjacent pad to connect. This makes it clear the connection on the assembled board is not a solder bridge, which a direct connection between the two pads would resemble.

It may be the case that no amount of vias or wending paths can complete a route. The one remaining tool for the PCB designer is a jumper — a wire added as a component during assembly just for the purpose of making a connection between two points on the board. Jumpers are most often required for single-sided boards; when the "jump" is rather small, uninsulated wire can be used. Jumpers are usually straight lines, and can be horizontal or vertical. Professional production PCBs use machine-insertable zero-ohm resistors as jumpers. Jumpers on double-sided boards are usually not viewed very favorably, but this is an aesthetic and efficiency issue, not a functional one.

Multi-layer boards clearly offer additional routing options, but again having some dominant routing direction (vertical or horizontal) on each layer is recommended, since mixing directions tends to cause routing problems. However, it is not uncommon to devote one or two inner layers to power and ground, rather than merely be additional layers for routing signals. This allows power and ground to be routed with minimal resistance and exposes the traces carrying interesting signals on the component and solder sides where they are available to be probed or modified. It is very difficult to modify traces on inner layers, needless to say!

Before routing too many traces, it is helpful to run the *Design Rule Check (DRC)* on the board. (See the section on Design Rule Checking below.) Applied early and often, DRC can identify areas of concern when it is easiest to correct. For example, a given trace width may provide insufficient clearance when passing between two IC pads.

Some PCB design packages offer *auto-router* capability in which the software uses the component and connectivity data of the netlist and attempts to route the traces automatically. There are some circumstances when they save time, but view these tools with some caution. Auto-routers are good at solving the routing puzzle for a given board, but merely connecting all the pins correctly does not produce a good PCB design. Traces carrying critical signals may take "noisy" routes; components that should have short, low-resistance connections to each other may have lengthy traces instead, and so on. More sophisticated auto-routers can be provided with extensive lists of "hints" to minimize these problems. For the beginner, the time spent conveying this design information to the auto-router is likely better spent manually routing the traces.

If an auto-router is used, at a minimum, critical connections should be first routed manually. These include sensitive signals, connections whose length should be minimized, and often power and ground (for both RFI and trace width reasons). Better still is to develop a sense of what a good layout looks like (which will come with practice and analyzing well-designed boards), and learn at what stage the auto-router can be "turned loose" to finish the routing puzzle.

TRACE WIDTH AND SPACING

All traces will have some width — the width may be the default width, the last width selected, or a width provided from data in the netlist. It may be tempting to route all but the power traces using the smallest trace width available from the board house (0.008 inch or smaller), since this allows the highest density of traces and eases routing. A better design practice is to use wider traces to avoid hard-to-detect trace cracking and improve board reliability. The more common traces 0.012-inch wide can be run in parallel on a 0.025-inch grid and can pass between many pads on 0.1-inch centers. Even wider traces will make the board easier to produce "in house," though the exact process used (CNC routing, chemical etching, etc) will limit the resolution. Note

PCB Design and EMC

While amateur projects are rarely subject to electromagnetic compatibility (EMC) standards, using good engineering practices when designing the board still reduces unwanted RF emissions and susceptibility to RF interference. For example, proper layout of a microprocessor circuit's power and ground traces can reduce RF emissions substantially. Proper application of ground planes, bypass capacitors and especially shield connections can have a dramatic effect on RFI performance. (See the **RFI** chapter for more on RFI.) A good reference on RFI and PCB design is *Electromagnetic Compatibility Engineering*, by Henry Ott, WA2IRQ.

that it is possible to "neck down" traces where they pass between IC or connector pads — that is, the regular, thick trace is run up close to the narrow gap between the pads, passes between the pads with a narrow width, then expands back to the original width. There is little reason to use traces wider than 0.030 inch or so for most signals (see **Table 6.4**) but power and ground trace widths should be appropriate for the current.

All traces have resistance, and this resistance is a function of the cross section of the trace (width times thickness) and the length. This resistance will convert electrical power to heat. If the heat exceeds a relatively high threshold, the trace becomes a fairly expensive and difficult-to-replace fuse. The trace width should be selected such that for the worst-case expected current, heat rise is limited to some threshold, often 10 °C. In practice, power traces (especially grounds) are often made as wide as practical to reduce resistance, and they greatly exceed the width required by heat rise limits alone.

Table 6.4 summarizes maximum currents for external (component and solder side) and internal traces for some common trace widths. Internal traces (on inner layers of multi-layer boards) can carry only about half the current of external traces for the same width since the internal layers do not dissipate heat to the ambient air like external traces can. (Note that trace widths are also sometimes expressed in "mils." 1 mil = 0.001 inch; it is shorthand for "milli-inch", not millimeter!)

There is no upper bound on the effective trace width. It is common to have large areas of the board left as solid copper. These *copper fill* areas can serve as grounds, heat sinks, or may just simplify board production (especially home-made boards). It is not a good idea to place a component hole in the middle of a copper fill — the copper is a very efficient heat sink when soldering. Instead, a "wagon wheel" pattern known as a thermal relief is placed (sometimes automatically) around the solder pad, providing good electrical connectivity but reducing the heat sinking. Often, copper fill areas can be specified using a polygon and the fill will automatically flow around pads and traces in that area, but can lead to isolated pads of copper.

In practice, most boards will have only two or perhaps three different trace widths; narrow widths for signals, and a thicker width for power (usually with a healthy margin).

One final note on trace width — vias are typically one size (ie, small), but multiple vias can be used to create low-resistance connections between layers. Spacing the vias so their pads do not touch works well; the pads are then shorted on both top and bottom layers.

Voltage also figures into the routing equation, but instead of trace width, higher voltages should be met with an increased clearance between the trace and other copper. The IPC-2221 standard calls for a clearance of 0.024 inch for traces carrying 31-150 V (peak) and 0.050 inch for traces carrying 151-300 V (peak); these are external traces with no coating. (With the appropriate polymer solder mask coating, the clearances are 0.006 inch for 31-100 V and 0.016 inch for 101-300 V. Internal traces also have reduced clearance requirements.) Fully addressing the safety (and regulatory) issues around high voltage wiring is outside the scope of this brief review, however, and the reader is urged to consult UL or IPC standards.

Table 6.4
Maximum Current for 10 °C Rise, 1 oz/ft² Copper
Based on IPC-2221 standards (not an official IPC table)

Trace Width (inches)	*Max. Current (External Trace) (A)*	*Max. Current (Internal Trace) (A)*	*Resistance (ohms/inch)*
0.004	0.46	0.23	0.13
0.008	0.75	0.38	0.063
0.012	1.0	0.51	0.042
0.020	1.5	0.73	0.025
0.040	2.4	1.2	0.013
0.050	2.8	1.4	0.010
0.100	4.7	2.4	0.0051
0.200	7.8	3.9	0.0025
0.400	13	6.4	0.0013

IPC-2221 Generic Standard on Printed Circuit Design,
Institute for Interconnecting and Packaging Electronic Circuits, **www.ipc.org**

SILKSCREEN AND SOLDER MASK

The silkscreen (or "silk") layer contains the text and graphics that will be silkscreened on the top of the board, shown in white in **Fig 6.45**. Components will generally have elements on the silkscreen layer that will automatically appear, such as designators and values, but other elements must be created and placed manually. Common silkscreen elements include: Circuit name, date, version, designer (and call sign), company name, power requirements (voltage, current, and fusing), labels for connections (eg, "Mic input"), warnings and cautions, labels for adjustments and switches. A solid white rectangle on the silkscreen layer can provide a good space to write a serial number or test information.

The board house will specify the minimum width for silkscreen lines, including the width of text. Text and graphics can be placed anywhere on the solder mask, but not on solder pads and holes.

Many quick-turn board houses omit the silkscreen for prototype boards. As noted earlier, many of the text elements above can be placed on the external copper layers. Component outlines are not possible since the resulting copper would short out traces, but component polarization can be noted with symbols such as a hand-made "+" made from two short traces, or a "1" from a single short trace. (Note that some component footprints follow a practice of marking the pad for pin 1 with a square pad while others are round or oval.)

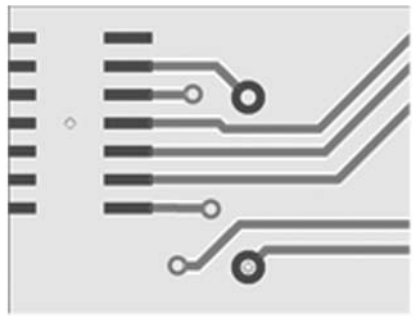
The solder mask is a polymer coating that is screened onto the board before the silkscreen graphics. As shown in Fig 6.45, it covers the entire surface of the board except for pads and vias. Solder masking prevents solder bridges between pads and from pads to traces during assembly and is particularly important for production processes that use wave soldering or reflow soldering. There is one solder mask layer for the top layer and another for the bottom layer. Internal layers do not need a solder mask. Solder masking may be omitted for a prototype board, but care must be taken to keep solder from creating unwanted bridges or short circuits.

During the PCB layout process, solder mask layers are generally not shown because they do not affect connectivity. **Fig 6.46** shows a typical PCB as it appears when the PCB layout process is complete.

DESIGN RULE CHECK

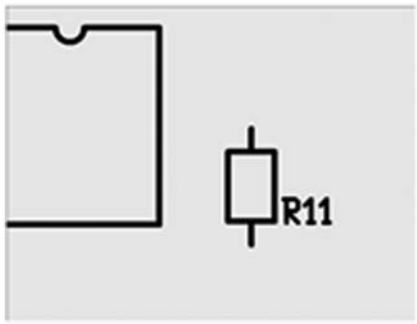
If a netlist has been provided from the schematic capture program, a *design rule check* (*DRC*) can be made of the board's layout. The PCB software will apply a list of rules to the PCB, verifying that all the connections in the netlist are made, that there is sufficient clearance between all the traces, and so on. These rules can be modified based on the specific board house requirements. As stated above, it is useful to run the DRC even before all the traces have been routed — this can identify clearance or other issues that might require substantial re-routing or a different approach.

If the user has waited until all the routing is done before running the DRC, the list of violations can be daunting. However, it is often the case that many if not all of the violations represent issues that may prevent the board from operating as wished. Whenever possible, all DRC violations should be rectified before fabrication.



Component Side (top) of PCB Traces

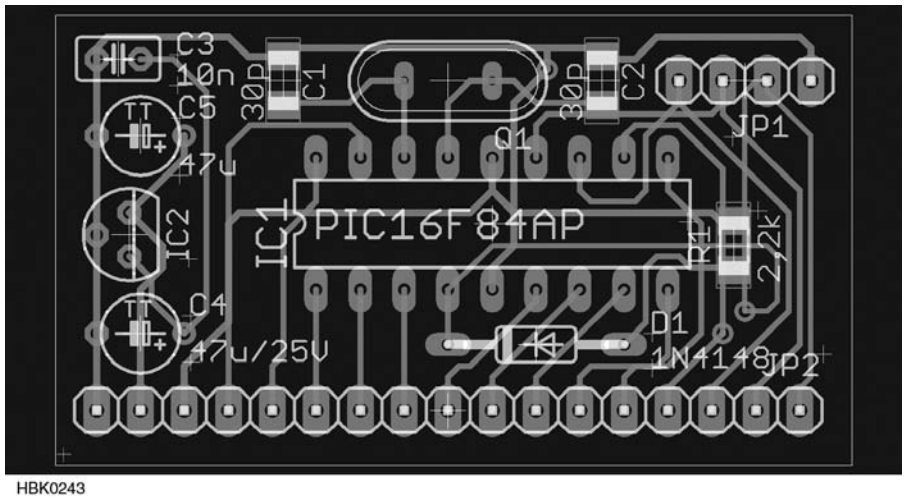
Solder Mask (top)



Silkscreen (top only)

HBK0635

Fig 6.45 — The relationship between the layout's top copper layer with traces and pads, the solder mask that covers the copper (a separate solder mask is required for the top and bottom layers of the PCB) and the silkscreen information that shows component outlines and designators.



HBK0243

Fig 6.46 — A completed microprocessor board as it is seen in a typical PCB layout editor (*Eagle*). Solder mask layers are omitted for visibility. Traces that appear to cross each other are on different sides of the board and are in different colors in the layout software. The silkscreen layer is shown in white.

6.4.7 Preparation for Fabrication

LAYOUT REVIEW

Once the board has passed DRC, the electrical connectivity and basic requirements for manufacturability have likely been satisfied. However, the design may benefit from an additional review pass. Turn off all the layers but one copper layer and examine the traces — often simplifications in routing will be apparent without the distractions of the other copper layers. For example, a trace can be moved to avoid going between two closely-spaced pins. Densely spaced traces could be spaced farther apart. There may be opportunities to reduce vias by routing traces primarily on one layer even if that now means both vertical and horizontal travel. Repeat the exercise for all the copper layers.

Review the mechanical aspects of the board as well, including the proximity of traces to hardware. If your prototype PCB does not have a solder mask, traces that run underneath components such as crystals in a conductive case or too close to mounting hardware can form a short circuit. An insulator must be provided or the trace can be re-routed.

GENERATING OUTPUT FILES

Once the PCB design is complete, the complete set of design description files can be generated for producing the PCB. These are:

Copper layers — One file per copper layer. These are known as *Gerber files* and were text files of commands originally intended to drive a *photoplotter*. Gerber was the primary manufacturer of photoplotters, machines that moved a light source of variable width (apertures) from one location to another to draw patterns on photographic film. While photoplotters have been replaced by digital technology, the format used by Gerber has been standardized as RS-274X and is universally used except by PCB software tied to a specific manufacturer. RS-274X is related to RS-274D ("G-Code") used by machinists to program CNC machinery but is an *additive* description (essentially saying "put copper here"), rather than describing the movements of a tool to remove material. A program is thus required to translate between Gerber and G-Code if a CNC machine is used to make a PCB by mechanically removing copper.

Drill file — The file containing the coordinates and drill sizes for all the holes, plated or not. Also called the *NC* or *Excellon* file, some board houses may require a specific format for the coordinates, but these are usually available to be set as options in the PCB program. There is only one drill file for a PCB, since the holes are drilled from one side. (Exotic options such as buried vias will require more information.) Like RS-274X apertures, the drill file will generally contain a drill table.

Silkscreen — Also in RS-274 format. Some board houses can provide silkscreen on both sides of the board, which will require two files.

Solder mask — The solder mask file is used by the board house to create the solder mask. One file per side is required.

A Gerber preview program such as *Gerbv* (**gerbv.gpleda.org**, open source, *Linux*, Mac OS X) or *GC-Prevue* (**www.graphicode.com**, *Windows*) can be used to review the trace layout Gerber files. This is a good test — the board house will make the boards from the Gerber files, not the PCB design file. Gerber previewers can import the copper layers, silkscreen and drill files to verify they correspond and make sense.

Any of the layers can usually be printed out within the PCB program (and/or Gerber preview program) for reference and further inspection.

In addition to files for PCB production, PCB layout programs can also generate assembly diagrams, and in some cases can provide 3D views of what the assembled board will look like. These can be useful for documentation as well as verification of mechanical issues such as height clearance.

Sending the files to the PCB manufacturer or board house and ordering PCBs is explained on the manufacturing Web site or a customer service representative can walk you through the process. Some firms accept sets of files on CD-ROM and may also offer a design review service for first-time customers or on a fee basis.

6.5 References and Bibliography

J. Wayde Allen, "Gain Characterization of the RF Measurement Path," *NTIA Report TR-04-410* (Washington: US Department of Commerce: 2004). Available as **www.its.bldrdoc.gov/pub/ntia-rpt/04-410/04-410.pdf**.

W. Hayward, W7ZOI, and J. Lawson, K5IRK, "A Progressive Communications Receiver," *QST*, Nov 1981, pp 11-21. Also see *QST* Feedback: Jan 1982, p 47; Apr 1982, p 54; Oct 1982, p 41.

W. Hayward, W7ZOI, "Designing and Building Simple Crystal Filters," *QST*, Jul 1987, pp 24-29.

W. Hayward, W7ZOI, "The Double-Tuned Circuit: An Experimenter's Tutorial," *QST*, Dec 1991, pp 29-34. Included in J. Kleinman and Z. Lau, compilers, *QRP Power* (ARRL: Newington, 1991), pp 5-29 to 5-34.

W. Hayward, W7ZOI; R. Campbell, KK7B; and B. Larkin, W7PUA, *Experimental Methods in RF Design*, 2nd ed (ARRL: Newington, 2009).

P. Horowitz and W. Hill, *The Art of Electronics*, 2nd ed (Cambridge: Cambridge University Press, 1989).

NXP Semiconductor: **www.nxp.com/models/bi_models/mextram/**

Z. Lau, W1VT, "Calculating the Power Limit of Circuits with Toroids," RF, *QEX*, Mar 1995, p 24.

D. Newkirk, WJ1Z, "Modeling a Direct-Conversion Receiver's Audio Response and Gain with ARRL Radio Designer," Exploring RF, *QST*, Mar 1995 pp 76-82.

D. Newkirk, WJ1Z, "Optimizing Circuit Performance with ARRL Radio Designer," Exploring RF, *QST*, Mar 1995, pp 76-78.

D. Newkirk, WJ1Z, "Math in a Box, Transistor Modeling, and a New Meeting Place," Exploring RF, *QST*, May 1995, pp 90-92.

D. Newkirk, WJ1Z, "Transistor Modeling with ARRL Radio Designer, Part 2: Optimization Produces Realistic Transistor Simulations," Exploring RF, *QST*, Jul 1995, pp 79-81.

D. Newkirk, WJ1Z, "Transistor Modeling, Part 3: Constraints in Optimization, Two-Port Data with Noise Parameters, and Introducing ARRL Radio Designer, Exploring RF," *QST*, Sep 1995, pp 99-101.

D. Newkirk, WJ1Z, "ARRL Radio Designer as a Learning (and Just Plain Snooping) Tool," Exploring RF, *QST*, Nov 1995, pp 89-91.

D. Newkirk, WJ1Z, "An ARRL Radio Designer Voltage Probe Mystery: The 3-dB Pad that Loses 9 dB," Exploring RF, *QST*, Jan 1996, pp 79-80.

D. Newkirk, WJ1Z, "Gyrating with ARRL Radio Designer," Exploring RF, *QST*, Mar 1996, pp 67-68.

D. Newkirk, WJ1Z, "Frequently Asked Questions About ARRL Radio Designer Reports and Report Editor," Exploring RF, *QST*, May 1996, pp 78-79.

D. Newkirk, WJ1Z, "ARRL Radio Designer versus Oscillators, Part 1," Exploring RF, *QST*, Jul 1996, pp 68-69.

D. Newkirk, WJ1Z, "ARRL Radio Designer Versus Oscillators, Part 2," Exploring RF, *QST*, Sep 1996, pp 79-80.

D. Newkirk, W9VES, "Simulating Circuits and Systems with *Serenade* SV," *QST*, Jan 2001, pp 37-43.

"The Spice Page," (**bwrc.eecs.berkeley.edu/Classes/icbook/SPICE/**). The home page for *SPICE*.

Paul W. Tuinenga, "*Spice: A Guide to Circuit Simulation and Analysis Using Pspice*, 2nd ed (New York: Prentice-Hall, 1992).

Andrei Vladimirescu, *The SPICE Book* (New York: John Wiley and Sons, 1994).

PCB CAD REFERENCES

Pease, Robert A, *Troubleshooting Analog Circuits*, Butterworth-Heinemann, 1991

Analog Devices, *High Speed Design Techniques*, Analog Devices, 1996

Johnson, Howard, and Graham, Martin, *High Speed Digital Design: A Handbook of Black Magic*, Prentice Hall, 1993

Ott, Henry, *Electromagnetic Compatibility Engineering*, Wiley Press, 2009