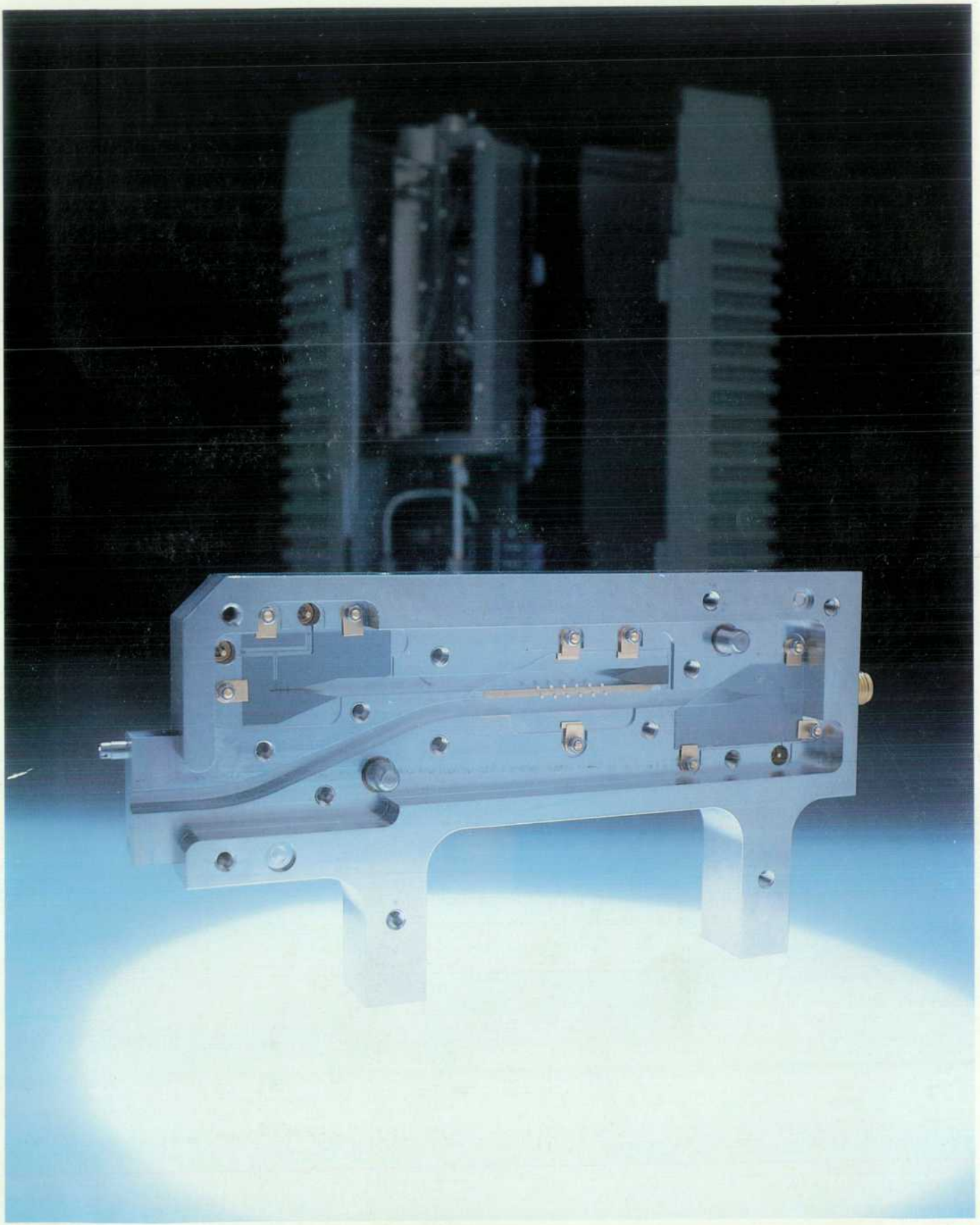
HEWLETT-PACKARD JOURNAL

APRIL 1988



HEWLETT-PACKARD JOURNAL

April 1988 Volume 39 • Number 2

Articles

Departments

Editor, Richard P. Dolan • Associate Editor, Business Manager, Kenneth A. Shaw • Art Director, Photographer, Arvid A. Danielson • Support Supervisor, Susan E. Wright
Administrative Services, Typography, Anne S. LoPresti • European Production Supervisor, Michael Zandwijken

In This Issue

Millimeter waves are electromagnetic energy in a frequency range of roughly 30 to 300 gigahertz, where wavelengths range from 1 to 10 millimeters. Millimeter-wave technology offers relief from the spectrum crowding that exists at lower frequencies, and many new radar, communications, navigation, and other systems are being built to operate at these higher frequencies. Millimeter-wave sources and test instrumentation for applications up to 110 GHz are the subjects of several papers this month. Besides sources (pages 6, 18, and 26), topics covered include detectors (page 31), power sensors (page 35), calibration and verification standards (page 8), and network analysis (page 12). At 110 GHz, a thousandth of an inch is about a hundredth of a wavelength or about four degrees of phase shift, enough to change a shorted quarter-wave transmission line from an open circuit to a troublesome inductance or capacitance. Cables are too imprecise, so energy propagates in rigid waveguide. Critical technologies are precision machining, thin-film circuits, and high-frequency diodes of the type described in our November 1986 issue. For the millimeter-wave sources, the interplay of these technologies makes possible an intricate and precise structure called a finline, which is shown in the cover photograph and discussed in the paper on page 18.

When a testing application requires signals so complex that they are difficult or impossible to produce with conventional signal generators, an arbitrary waveform synthesizer may be the answer. These synthesizers are capable of producing virtually any finite-length waveform within their specified bandwidth. For the HP 8770A Arbitrary Waveform Synthesizer, the specified bandwidth is dc to 50 megahertz. The HP 8770A's "front panel" is an HP 9000 Series 200 or 300 Computer workstation running the HP Waveform Generation Language (WGL). Waveforms are specified in WGL by the user and then downloaded from the computer to the synthesizer, where an advanced digital-to-analog converter produces the desired output waveform. The technology is similar to the compact disc player, but the bandwidth is much higher, of course. The architecture, operation, and performance of the HP 8770A are discussed in the paper on page 69. WGL and its advantages in waveform synthesis over conventional programming languages are treated in the paper on page 94. Because the projected performance of commercial digital-to-analog converters fell short of the requirements, a special one was developed using advanced metal-oxide-semiconductor integrated circuit technology. As described in the paper on page 78, an MOS IC switches precision currents generated off-chip, while a gallium arsenide sampler removes the distortion of switching transients from the MOS chip's output. Applications for the HP 8770A in magnetic recording and radar are presented in the article on page 86. Other applications can be found in television and many other fields.

AT&T's UNIX® operating system is a multitasking, tool-based system that seems ideally suited for multiple-user applications in manufacturing and business. Most of these applications need to be friendly to novice users. Although the UNIX system provides tools that initialize a system at power-on (boot programs) and allow users to log onto the system (logon programs), these tools in their standard forms are not friendly enough for many applications. If a programmer wants to use these tools rather than write completely new programs, adaptations have to be made. In the paper on page 39, Marv Watkins discusses the problem and offers several examples of adaptations for the manufacturing environment.

At the annual HP Software Engineering Productivity Conference, the HP software engineering community shares accomplishments, challenges, and future directions in software engineering. Three papers in this issue are from the 1987 conference. "A Virtual User Simulation Utility," page 48, describes a software testing tool that simulates one or several users of a system under test, which can be a computer or any other device that communicates by means of byte streams. After

writing a script that executes exactly the same test that a real user would perform, the test engineer can go home or do something else while testing proceeds automatically. The virtual user utility is being used at several HP Divisions around the world. "An HP-UX Kernel Load and Measurement System," page 54, describes a system that measures and controls the amount of stress applied by tests or test packages of any type to the kernel of the HP-UX operating system on HP Precision Architecture Computers. The HP-UX system is HP's version of AT&T's UNIX System V operating system with extensions. The Kernel Load and Measurement System has load measurement and load generation components that interact to form a sampled feedback control system whose output is a user-specifiable amount of kernel loading. "Process Measures to Improve R&D Scheduling Accuracy," page 61, tells of efforts to address a perennial problem in software engineering—project slip, or failure to meet completion date commitments. At HP's Roseville Networks Division, scheduling is looked at as a process and is subjected to continuous measurement. R&D managers are provided with two special process measures, which are updated monthly. Since the program began, the average lab-phase overrun has been reduced from 70% to 20%.

R. P. Dolan

What's Ahead

In the June issue, we'll have five papers on applications of statistics; topics include designing, specifying, and calibrating instruments, computing computer system availability, and managing software projects. Four papers will discuss the concepts and technology used for the 1-GHz front end of the HP 54111D Digitizing Oscilloscope. The design of the compact, autoloading HP 7980A Half-Inch Tape Drive will be discussed in two articles.

The **Hewlett-Packard Journal** is published bimonthly by the Hewlett-Packard Company to recognize technical contributions made by Hewlett-Packard (HP) personnel. While the information found in this publication is believed to be accurate, the Hewlett-Packard Company makes no warranties, express or implied, as to the accuracy or reliability of such information. The Hewlett-Packard Company disclaims all warranties of merchantability and fitness for a particular purpose and all obligations and liabilities for damages, including but not limited to indirect, special, or consequential damages, attorney's and expert's fees, and court costs, arising out of or in connection with this publication.

Subscriptions: The Hewlett-Packard Journal is distributed free of charge to HP research, design, and manufacturing engineering personnel, as well as to qualified non-HP individuals, libraries, and educational institutions. Please address subscription or change of address requests on printed letterhead (or include a business card) to the HP address on the back cover that is closest to you. When submitting a change of address, please include your zip or postal code and a copy of your old label.

Submissions: Although articles in the Hewlett-Packard Journal are primarily authored by HP employees, articles from non-HP authors dealing with HP-related research or solutions to technical problems made possible by using HP equipment are also considered for publication. Please contact the Editor before submitting such articles. Also, the Hewlett-Packard Journal encourages technical discussions of the topics presenting in recent articles and may publish letters expected to be of interest to readers. Letters should be brief, and are subject to editing by HP.

Please address inquiries, submissions, and requests to: Editor, Hewlett-Packard Journal, 3200 Hillview Avenue, Palo Alto, CA 94304, U.S.A.

Millimeter-Wave Sources and Instrumentation

The recent growth in developing and applying millimeter-wave systems has created a corresponding demand for millimeter-wave test instrumentation.

by Mohamed M. Sayed and John R. Regazzi

THE PAST SEVERAL YEARS have seen a strong increase of research and applications in the millimeter-wave region. Spectrum crowding in the microwave frequency range has pushed this development. Advantages of millimeter-wave applications are small component size, increased available bandwidth, narrow beamwidth, and the possibility of taking advantage of the atmospheric attenuation and absorption that occur at some millimeter-wave frequencies. Some typical applications are radar, communication transceivers, navigation equipment, and scientific radiometers. Hence, the demand for appropriate test instrumentation is expected to grow significantly in the future.

Millimeter-wave sources with full waveguide coverage are key instruments for many systems in this new frequency range. Some examples are sweepers, synthesizers, vector analyzers, and scalar analyzers. HP's design objectives for such millimeter-wave systems are to extend present microwave measurement capabilities to 60 GHz and provide the foundation for measurements up to 110 GHz.

Two techniques (Fig. 1) were considered for achieving these goals—the fundamental oscillator and the frequency multiplier. However, to accomplish the expandability goal of operation up to 110 GHz with sufficient power output, the diode multiplier technique became the only choice. The greater insertion losses of typical signal processing components such as modulators, couplers, and connectors at millimeter-wave frequencies offset the power that can be achieved by a fundamental oscillator. The present difficulties associated with medium-power solid-state amplifiers also favor the multiplier technique. In addition, a frequency multiplier allows the millimeter-wave system specifications to be derived from those of the microwave source used. For example, the spectral purity, frequency resolution, stability, and modulation characteristics of a microwave synthesizer are all preserved at the millimeter-wave frequency.

Fig. 2 shows five HP millimeter-wave source modules and the caption lists the frequency ranges, waveguide band designations, and HP source module model numbers. The main contributions of these solid-state millimeter-wave sources are user convenience by using an existing microwave source as a driver, small size, full waveguide coverage, lower phase noise, reliability, and system specifications for different applications.

System Block Diagram

Fig. 3 shows a basic system block diagram. A microwave synthesizer drives a microwave power amplifier in the frequency range from 13.33 to 20 GHz. The output of the amplifier provides enough power to drive the millimeter-wave source module. The dc power for the source module is supplied by the HP 8349B Microwave Amplifier. The microwave signal is filtered, amplified, and then tripled inside the source module (HP 83556A). Coverage from 40 to 60 GHz is available from the output of the HP 83556A with

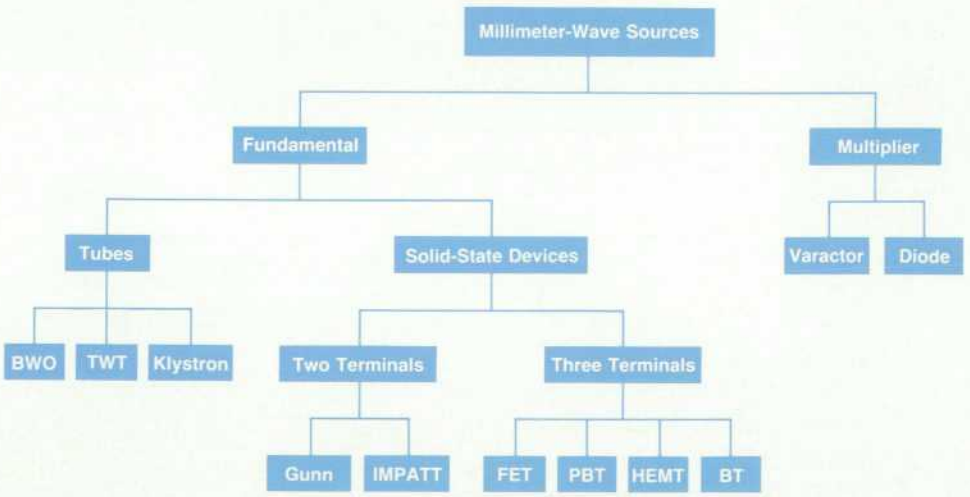


Fig. 1. *Millimeter-wave source designs. The diode multiplier approach is used by HP.*

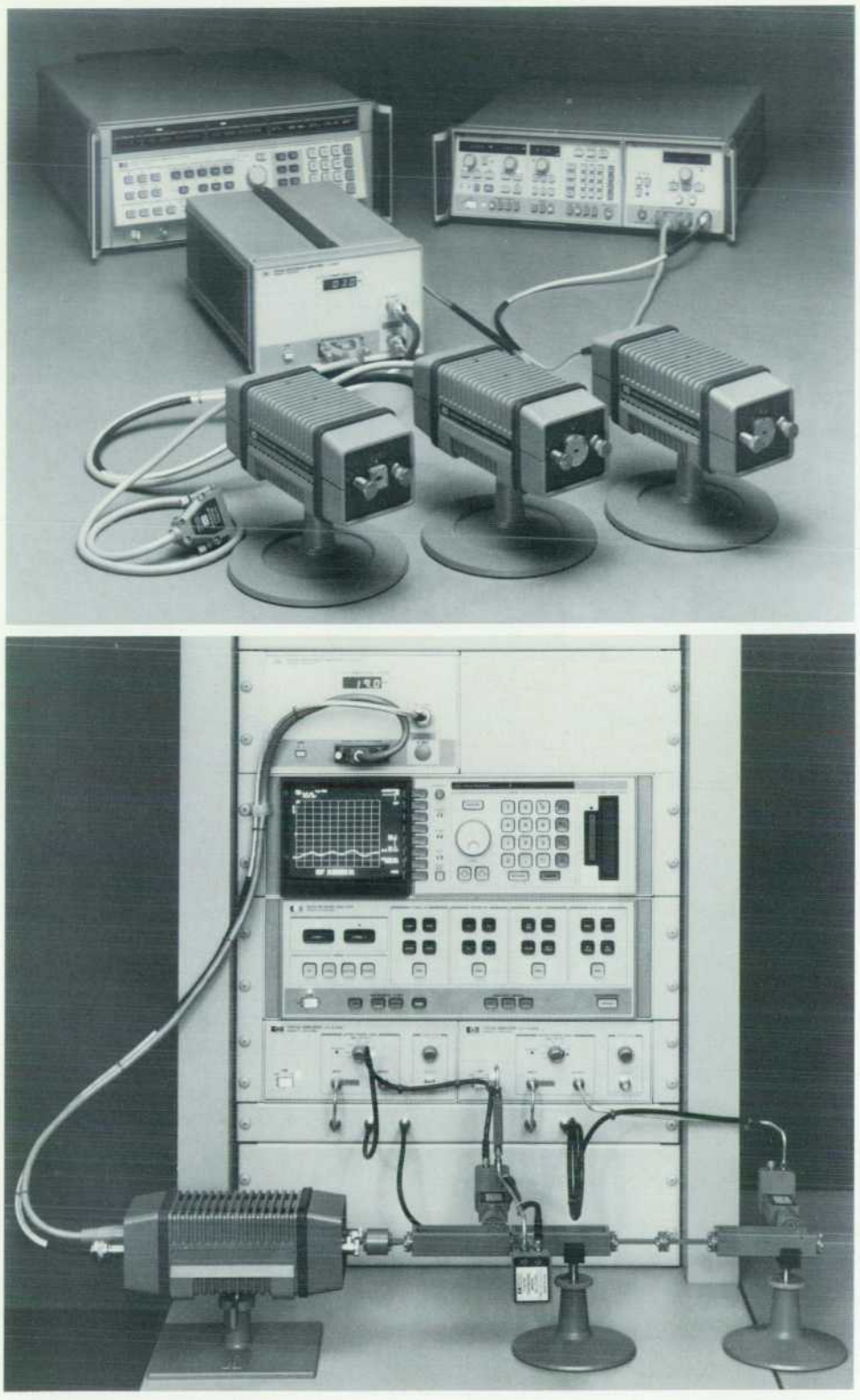


Fig. 2. *(Top) Three HP millimeter-wave source modules: HP 83554A (R band, 26.5 to 40 GHz), HP 83555A (Q band, 33 to 50 GHz), and HP 83556A (U band, 40 to 60 GHz). (Bottom) The HP 85100V (V band, 50 to 75 GHz) is used as part of the HP 8510 System. The HP 85100W (W band, 75 to 100 GHz) is similar in appearance.*

12-Hz resolution and 3-dBm leveled output power. The R and Q bands (26.5 to 50 GHz) can be covered by using the HP 83554A and HP 83555A Source Modules.

Millimeter-Wave Systems

Sources. Fig. 4 shows two ways to configure a millimeter-wave source. The millimeter-wave source driver (HP 83550A) interfaces directly to the millimeter-wave source modules and the frequency range (Q-band in this example) and output power are displayed by the HP 83550A. However, if a user already owns a synthesizer (HP 8340/41 or HP 8673) or a sweeper such as the HP 83592A/B/C, then only the HP 8349B and a source module are required to

(continued on page 9)

A New Generation of Millimeter-Wave Calibration and Verification Standards

The availability of fast and affordable computation power and data storage has made possible instruments such as the HP 8510 Network Analyzer with real-time error correction. Just as improved dynamic range and linearity demanded a new generation of coaxial microwave standards with improved connection repeatability and accuracy, a new generation of waveguide millimeter-wave standards had to be designed to take advantage of the instrument's capabilities for frequencies up to 100 GHz.

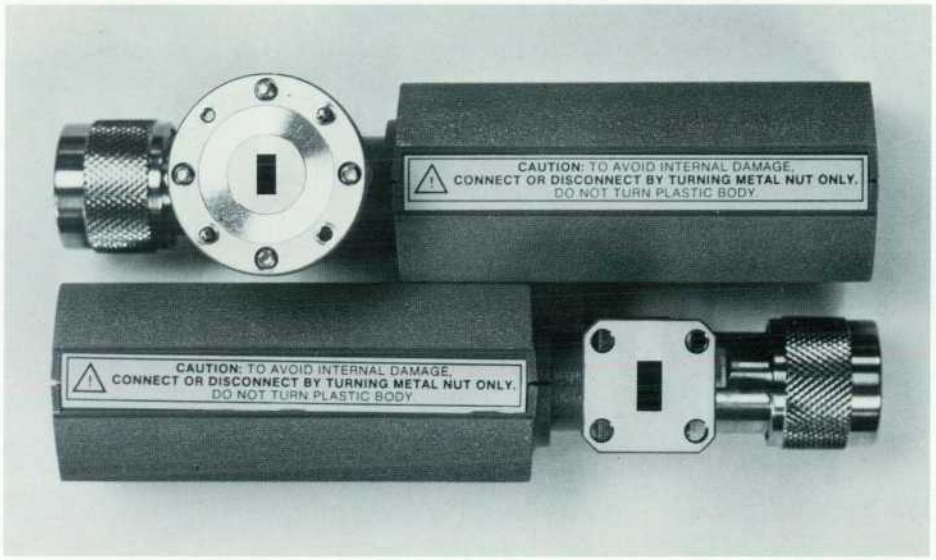


Fig. 1. *Close-up view of HP precision flanges.*

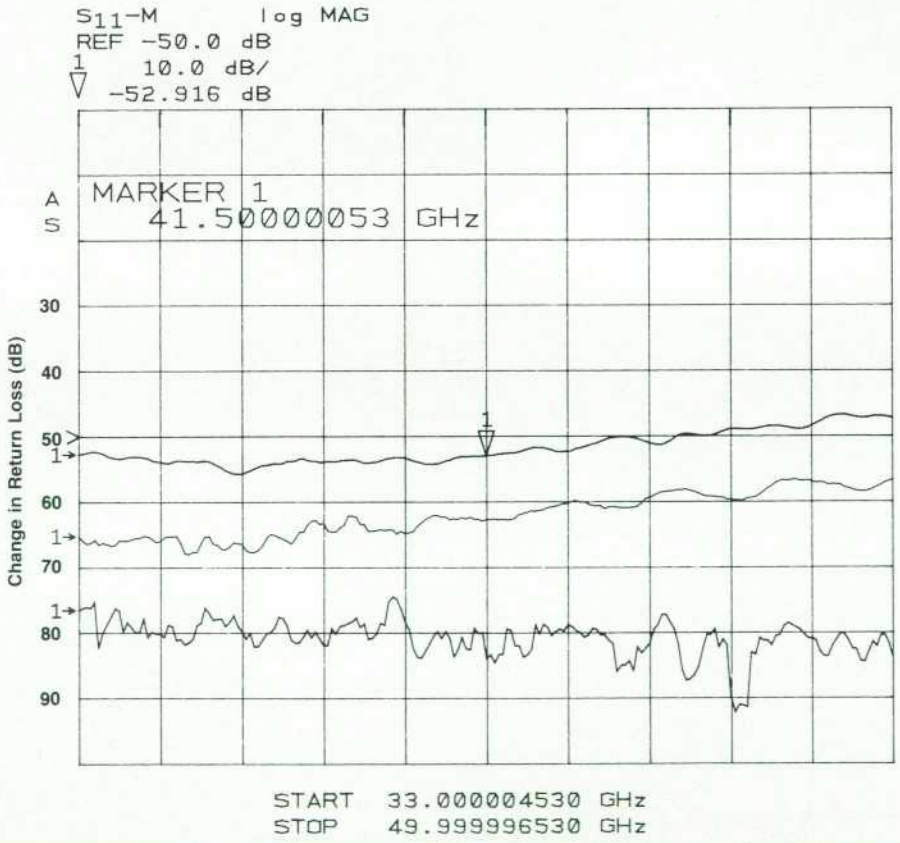


Fig. 2. *Typical repeatability of military-specified millimeter-wave (Q-band) waveguide flanges and HP's new precision waveguide flanges. The top curve is for two coupled military-specified flanges. The middle curve is for an HP precision flange coupled to a military-specified flange. The bottom curve is for two coupled HP precision flanges. In each case, the pair of flanges was connected, return loss was measured, and a normalization was done so that subsequent measurements were normalized to the first measurement for that pair. Each pair of flanges was then reconnected and measured again, and the plot shows the change in return loss between the two measurements.*

Two basic problems we had to deal with were the same as for the older generation of coaxial standards—a lack of sufficient connection repeatability and accuracy. First we found a way to improve the performance of any waveguide flange while retaining compatibility with military-specified flanges and other flanges compatible with them. The use of the latest manufacturing technology allows us to hold tolerances on the size and position of features on the flanges in such a way as to achieve all our goals. By adding an outer ring on the face of the round flanges using a raised contact area around the guide opening (see Fig. 1), we solve the parallelism problem encountered when connecting to a similar or planar flange. Fig. 2 shows the typical repeatability of military-specified and the new HP precision flanges.

It became clear that, when the new flange was used on both the HP 8510's test port and the calibration standard, further improvements in measurement accuracy were possible by improving the size and position of the guide opening with respect to the alignment pins. Hence, we developed several capabilities for manufacturing high-precision waveguide standard sections, sliding loads, and fixed loads. Waveguide aperture tolerances of ±0.0003 inch and precision flange hole tolerances within 0.001 inch of the true position to the aperture result in excellent electrical performance. For example, in U-band operation the specification for return loss of the precision sliding load is 46 dB, but the typical performance is greater than 50 dB. Parts for the higher-frequency bands have similar electrical performance.

In V band and W band, fixed loads are used to eliminate sliding load mechanical stability problems that would become more prominent because of the smaller physical size. (For information on calibration methods, see the article on page 12.)

Adapters

To facilitate measurements in the millimeter-wave region in coaxial line, we developed a series of precision coax-to-waveguide adapters using HP's new 2.4-millimeter coaxial connector family. At present, R and Q-band adapters are available (see Fig. 3).

Acknowledgments

The development work described above was accomplished through the dedicated efforts of manufacturing engineering, materials, and accounting at Stanford Park Division.

Julius K. Botka
R&D Project Manager
Paul B. Watson
Development Engineer
Network Measurements Division

Doug Halbert
Production Manager
Stanford Park Division

Fig. 3. *R and Q-band waveguide standards and adapters to the 2.4-mm coaxial connector.*

(continued from page 7)

upgrade the user's microwave system to millimeter-wave frequencies.

The article on page 18 describes the R, Q, and U-band source modules covering the frequency range from 26.5 to 60 GHz. The two HP source modules covering the frequency range from 50 to 100 GHz are described at the end of this article.

Scalar Analyzer. The HP 8757A Scalar Analyzer[1] can be used to measure millimeter-wave components using the millimeter-wave detector described on page 31. Both ac and dc detectors can be used in this application. However, for ac modulation the source module modulation response is critical.

Vector Analyzer. The source output power is critical when determining the system dynamic range. The dynamic range varies from over 95 dB for the R-band system to about 75 dB for W band (see article on page 12).

Millimeter-Wave Source System Interface. The design goal of the millimeter-wave source system is to make any HP 8355x Source Module work with any HP 83550A Sweep Oscillator or with any HP 8349B Microwave Amplifier and sweeper (HP 8359X) or synthesizer (HP 8340/41 or HP

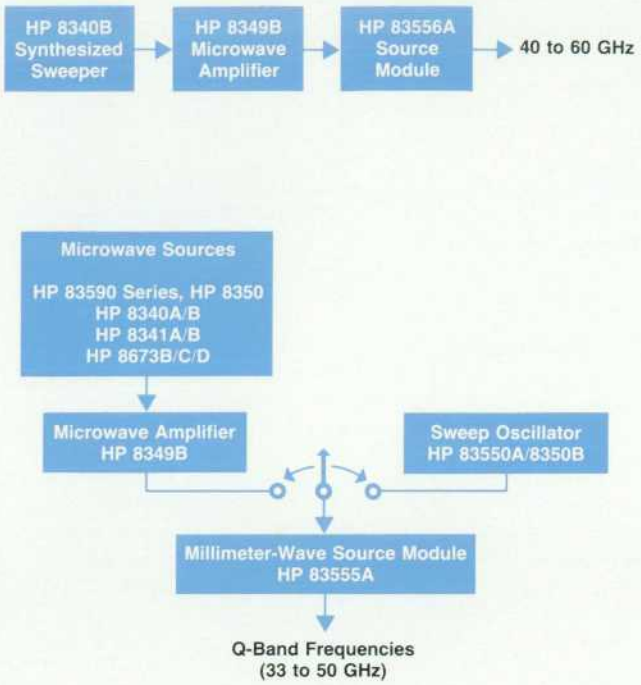


Fig. 4. *Two Q-band millimeter-wave source configurations.*

8673). The interface between these instruments is both analog and digital. Details of this interface are described on page 20.

Millimeter-Wave Source Specification

The critical specifications for any millimeter-wave source are output power, power accuracy, output power flatness, frequency accuracy, dynamic range, in-band harmonics, and modulation characteristics. The first three parameters are mainly functions of the source module and the last three parameters depend on the microwave source driver and microwave power amplifier. To guarantee the system specifications, a comprehensive characterization

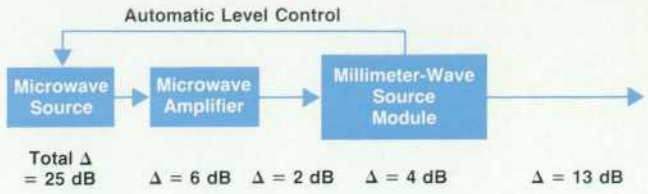


Fig. 5. *ALC range required for HP 83554A Source Module.*

was performed with the different systems and with different millimeter-wave source modules.

Here, dynamic range is determined by the automatic level control (ALC) circuitry and is defined as the difference between maximum and minimum leveled output power. The maximum leveled output power depends on the minimum input power to the source module and the maximum conversion losses of the source module. Any gain slope of the combined system must be compensated by the microwave source modulator range as shown in Fig. 5.

The modulator dynamic range of the microwave source is 25 dB. The 13-dB dynamic range of the millimeter-wave source module is the difference between a maximum output power of 8 dBm and a minimum leveled output power of −5 dBm. The variation of the millimeter-wave source module conversion efficiency across the waveguide band is 4 dB and the variation of the cable losses is 2 dB. To guarantee a dynamic range of 13 dB for the HP 83554A Module, the maximum gain slope of the HP 8349B is set to reduce the gain by 6 dB from the low end of the frequency range to the high end. The HP 8340/41 and HP 8673 perform satisfactorily without any adjustment. It is verified in production that the HP 8359X plug-ins are capable of delivering −20 dBm without oscillation of their ALC loop.

A similar analysis was performed for the harmonically related specification, and a two-tone test was developed for the source modules.

V and W-Band Sources

The R, Q, and U-band source modules used for the HP

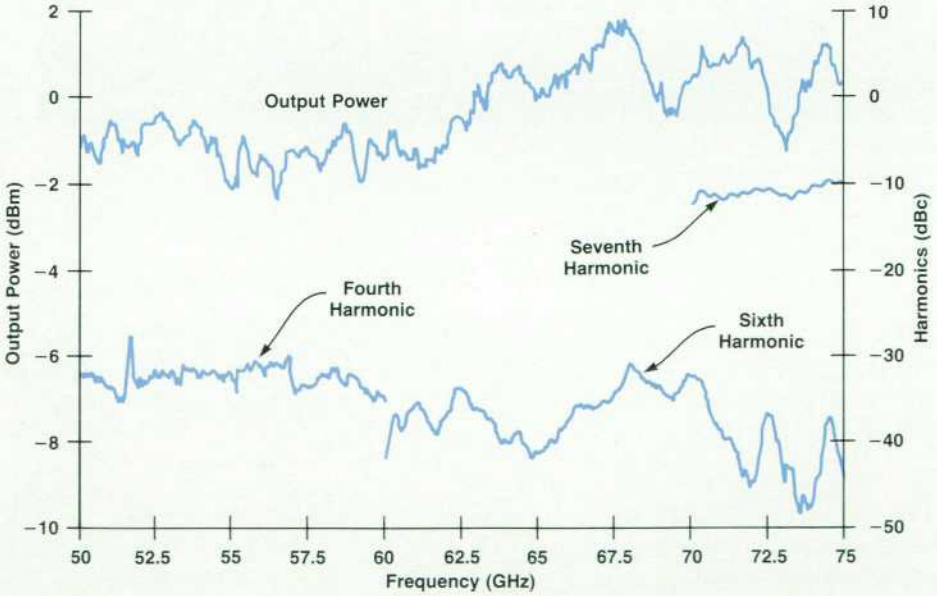


Fig. 6. *V-band source (HP 85100V) performance.*

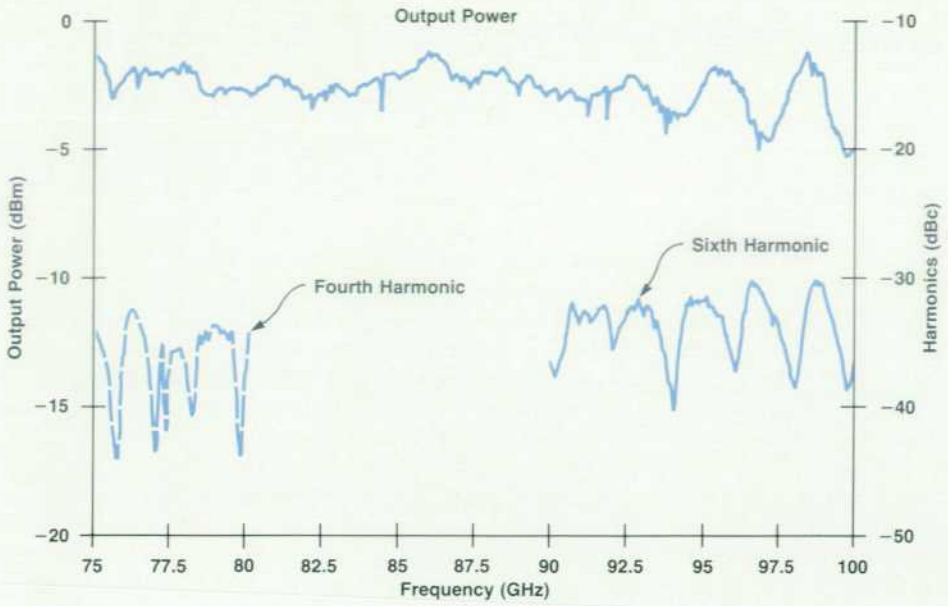


Fig. 7. *W-band source (HP 85100W) performance.*

8510 millimeter-wave system cover the frequency range up to 60 GHz. However, to cover the V and W bands, new source modules had to be developed. A constraint on these V and W-band multipliers is that the available input frequency is less than 20 GHz. Thus a quintupler circuit is required. Since no directional detector is required for the HP 8510 millimeter-wave system, the HP 85100V/W Source Modules deliver unleveled output power to the HP 8510 system.

The HP 85100V and HP 85100W consist of a microwave amplifier and a millimeter-wave quintupler. The microwave amplifiers for the HP 85100V and HP 85100W span the frequency ranges of 10 to 15 GHz and 15 to 20 GHz, respectively. The output power of the microwave amplifier is more than 24 dBm across its frequency range. The quintupler consists of a low-pass filter, an antiparallel-diode IC, and a reduced-height-and-width waveguide output. The low-pass filter has well-controlled reflection characteristics over the frequency range, which extends to the lower edge of the waveguide band. The last element of the low-pass filter is a resonance-free metal-insulator-semiconductor (MIS) capacitor and the multiplier diode IC is mounted across the reduced-height waveguide.[2]

The harmonics of the input microwave frequency and the millimeter-wave frequency generated by the diode IC can result in periodic cancellation of power across the frequency range of interest. Thus special care was taken to design the microwave filter such that its cutoff frequency is high enough to pass up to 15 or 20 GHz with minimum attenuation and at the same time low enough to attenuate any unwanted harmonics.

Two antiparallel diodes are integrated on one chip with gold beam leads to reduce the parasitic components and improve the connection reliability and ruggedness.[3] This diode IC is also used for frequency tripling as described on page 19. A 200-ohm resistor stabilizes the output power from the diode and protects it by creating a reverse bias when high microwave power is applied.

The waveguide impedance is chosen to present an optimum match to the diode multiplier. The height and width tapers are integrated into the same section of the waveguide.[4] The output power and harmonics for the V and W-band sources are shown in Fig. 6 and Fig. 7, respectively.

Acknowledgments

We would like to thank John Noble and Steve Handley for helping in the system evaluation, Giovannae Anderson and Vinh Nguyen who designed the HP 85100V/W Modules, and all the NPI and production teams.

References

1. J.H. Egbert, et al, "Advanced Scalar Analyzer System Improves Precision and Productivity in R&D and Production Testing," *Hewlett-Packard Journal*, Vol. 37, no. 2, February 1986, pp. 24-39.
2. R. Dalichow and M. Sayed, "Millimeter Wave Sources," *Measurement Science Conference*, January 1987.
3. G. Anderson, et al, "GaAs Beam-Lead Antiparallel Diodes for MMW Sub-Harmonic Mixers," *IEEE International Electron Devices Meeting*, Washington, D.C., December 1981.
4. F. David, "Analysis and Synthesis of Exponentially Tapered, Non-Uniform Transmission Line Impedance Transformers," *MSEE Thesis*, Oregon State University, 1975.

Millimeter-Wave Vector Network Analysis

by Robert G. Dildine and James D. Grace

NETWORK ANALYSIS at millimeter-wave frequencies has long been a difficult and time-consuming process with limited accuracy and a trade-off of time for results. Recently, a method of making vector network measurements at frequencies above 26 GHz was developed that retains the error-correction capability, time-domain analysis, and accuracy of the HP 8510B Microwave Network Analyzer at microwave frequencies.[1]

The system makes reflection and transmission measurements and is based on the HP 8510B, but no HP 8510B Series test set is used. Full waveguide band coverage is obtained from 26.5 to 100 GHz, and the system features accuracy and speed similar to that of the HP 8510B in stepped sweep operation. In addition, the system's waveguide test ports can be moved, eliminating the need for custom or flexible waveguide to connect the device under test.

System Block Diagram

The system block diagram is shown in Fig. 1. A microwave synthesizer driving a frequency multiplier for the appropriate waveguide band is used as the RF source. The output of the multiplier is fed to an isolator to improve the source match, then to a reflectometer made up of two directional couplers (a dual directional coupler can also be used). The couplers sample the incident and reflected signals for the device under test and feed them to harmonic mixers[2] that convert the signals directly to the 20-MHz IF of the HP 8510B. Isolators placed ahead of the mixers prevent unwanted harmonics generated by the mixers from producing spurious responses that would interfere with the measurement. An improvement of 15 to 30 dB in dynamic range results from effectively dealing with these spurs. A microwave sweep oscillator operating in the 3-to-6-GHz range acts as the local oscillator for the mixers (a microwave synthesizer is required to be used as the LO for the V and W bands: 50 to 100 GHz). Preamplifiers are used in the IF path to overcome some of the mixer conversion loss and bring the IF signal level up to the optimum input range for the HP 8510B.

The return port is another directional coupler terminated with a fixed waveguide load to improve the load match. The coupled arm is fed via an isolator to a harmonic mixer driven by the same local oscillator as the other two mixers. The return port mixer local oscillator uses a long 0.141-inch-diameter semirigid cable that can be flexed to allow

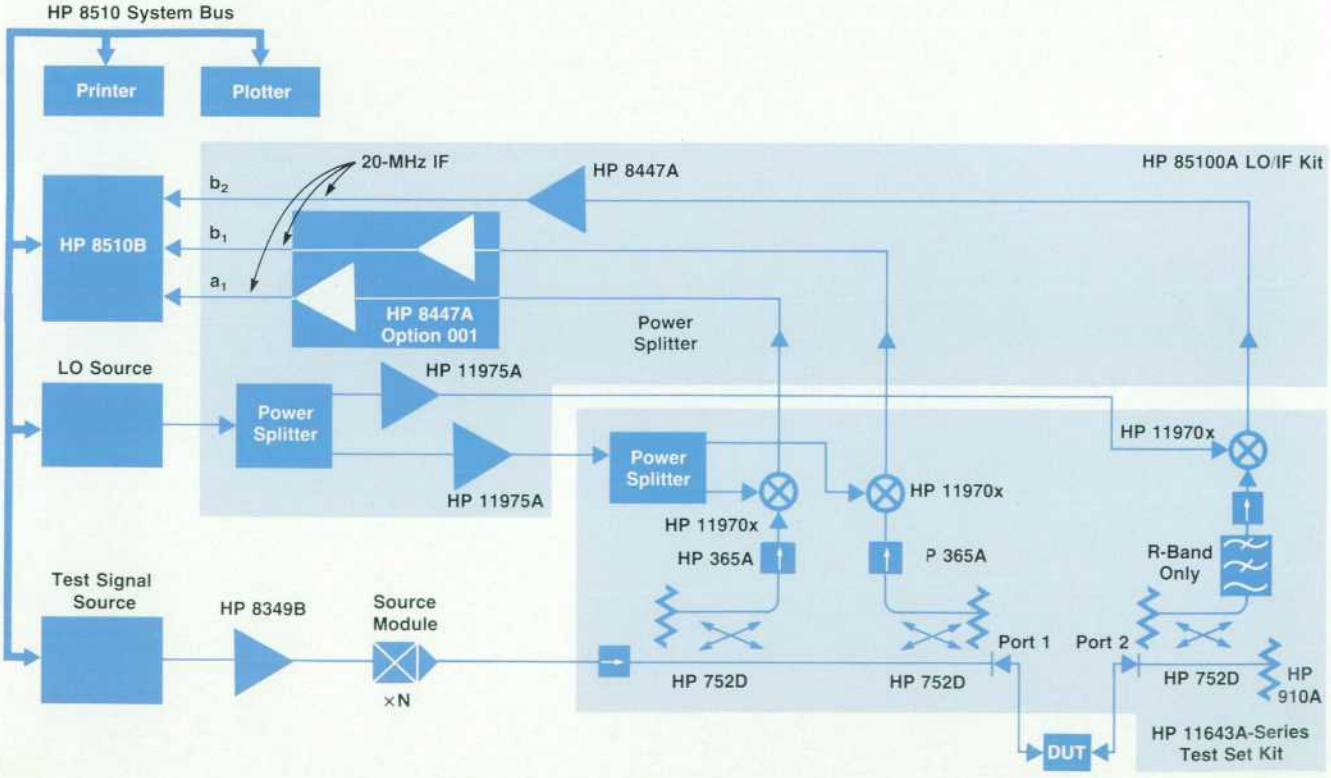


Fig. 1. *Millimeter-wave vector network analyzer system block diagram.*

insertion of the device under test. A low-pass filter with a cutoff frequency just above the upper band edge can be used in front of the mixer if desired and will help reduce the unwanted harmonics generated by the harmonic mixer. It has been found, however, that good performance can be obtained in the bands above 40 GHz without the filter.

The HP 8510B provides the user interface for the system, allowing the user to set up the measurement conditions and display the results. It controls the RF and LO sources, setting up the correct frequencies and commanding them to step along as the measurement is being made, and phase-locking the sweep oscillator used for the LO to the RF source with the proper offset that gives a 20-MHz IF. The relative magnitudes and phases of the three IF signals are measured and displayed by the HP 8510B.

Sources of Error

Two types of errors, systematic errors and noncorrectable errors, exist in a network analyzer. Systematic errors repeat from measurement to measurement and are independent of time and of the device being measured. Because these errors repeat, they can be measured and their effects removed from the device measurement.[3] The systematic errors most often encountered in network analysis are frequency response tracking, directivity, source match, load match, and isolation errors. Other errors such as connection repeatability, noise, compression, and the effects of spurious signals are not repeatable and are noncorrectable.

Systematic Errors

Frequency response tracking errors are caused by differences in frequency response between the test and reference signal paths. These differences are usually a result of the different components in the signal paths and affect both transmission and reflection measurements.

Directivity error is the inability of the directional device in the reflectometer to separate the incident and reflected waves absolutely. For example, the directivity of a directional coupler may be only 30 or 35 dB. This means that the coupled arm also samples a small amount of the energy traveling in the reverse direction, the amount being 30 or 35 dB lower than the sample of energy traveling in the forward direction. The contribution of directivity error to the uncertainty of the measurement depends on the device and shows up most in reflection measurements, especially for devices with low reflection coefficients.

Source match error is caused by a mismatch between the test port impedance and the system impedance. If the test port impedance is not equal to the system impedance, reflections from the test port can be re-reflected by the device under test and introduce errors into the measurement. The contribution of source match error to the uncertainty of the measurement depends on the device and shows up most in reflection measurements, especially for devices with high reflection coefficients.

Load match error is caused by a mismatch between the return port impedance and the system impedance. If the return port impedance is not equal to the system impedance, reflected energy from the return port can be reflected again back to the return port by the device under test and introduce errors into the measurement. The contribution of load match error to the uncertainty of the measurement depends on the device and shows up most when measuring two-port devices with high output reflection coefficients and/or low transmission loss.

Isolation error is the leakage between the test and reference signals through the test set and usually is caused by crosstalk between the test and reference paths in the test set or network analyzer. The contribution of isolation error to the uncertainty of the measurement can depend on the device under test and occurs most often when making transmission measurements on devices with high transmission loss.

Fig. 2 is a flow diagram of the one-port error model showing the effects of directivity, source match, and reflection frequency response errors. Applying simple flow graph analysis techniques, we see that the measured reflection coefficient is a function of the true reflection coefficient and the three error terms. If these error terms are known, the true reflection coefficient can be calculated from the measured reflection coefficient. Fig. 3 is a flow diagram of the two-port error model showing the effects of load match, transmission frequency response, and isolation errors. Likewise, if these error terms are known, the true scattering parameters of the device under test can be calculated from the measured scattering parameters.

Calibration

Several calibration techniques can be used to measure and correct for systematic errors depending on the accuracy desired. The techniques that account for the most errors require the most measurements and there is often a trade-off of effort for results. The three calibration techniques used in the millimeter-wave system are frequency response, one-port, and one-path two-port.

Frequency response calibration only corrects for the frequency response tracking errors of the system and assumes perfect directivity and perfect source match. It is useful for

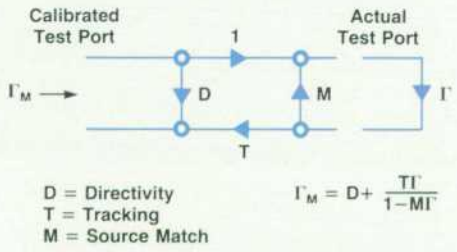


Fig. 2. *Flow diagram of one-port error model.*

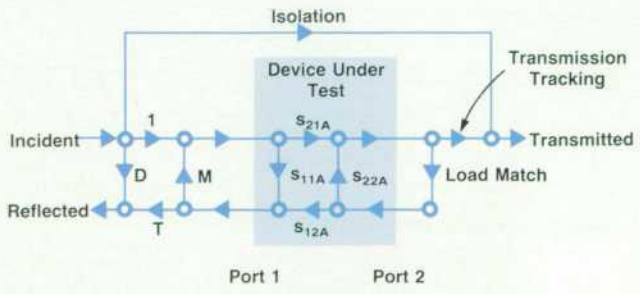


Fig. 3. *Flow diagram of two-port error model.*

measuring well-matched devices and for just taking a quick look at almost any device. The frequency response calibration technique is nothing more than a vector normalization.

One-port calibration corrects for directivity, source match, and reflection frequency response errors. This technique is useful for making high-accuracy reflection measurements on one-port devices. Fig. 4 shows the return loss of a fixed load measured with no calibration and with a one-port calibration. Note the large amount of ripple created by the interference of the directivity error and the load reflection when there is no calibration.

The one-path, two-port calibration technique corrects for directivity, source match, load match, transmission frequency response, reflection frequency response, and isolation errors. This technique is useful for measuring two-port devices with high reflection coefficients on each port or those devices whose performance depends on how they are terminated. Fig. 5 shows the return loss of a bandpass filter measured with a one-port calibration compared to the same measurement with a one-path two-port calibration. Note the large resonance in the passband that is caused by the imperfect load match to the filter. When the load match is accounted for, the passband is found to be well-behaved. The one-path two-port calibration provides the best accuracy for measuring any two-port device when using a reflection/transmission-type test set.

The one-port and one-path, two-port calibration techniques resident in the HP 8510B are redefined for waveguide and are used to calibrate the system. The calibration method is the standard sliding load method commonly used for coaxial cable at microwave frequencies.

At higher frequencies, the waveguide dimensions become so small that it is very difficult to manufacture a sliding load with the required precision for use as a calibration standard. An alternate calibration method, called the short-line method,[4] is provided and requires only a fixed load with good repeatability, the same load offset by a precisely known quarter-wave delay, a flush short, and a short offset by a quarter wavelength. The same quarter-wave delay can be used for offsetting both the load and the short, minimizing the number of components in the calibration kit. This delay is made from a waveguide transmission line about a quarter wavelength long at midband. For best accuracy, the line's physical length should be measured accurately and this value entered into the calibration routine.

Referring to Fig. 2, it can be seen that the measured reflection coefficient of each calibration standard is a function of the standard's actual reflection coefficient and the error terms:

$$\Gamma_M = D + T\Gamma/(1 - M\Gamma)$$

Let the true reflection coefficient of the fixed load be Γ_l and the measured value be Γ_L. Then:

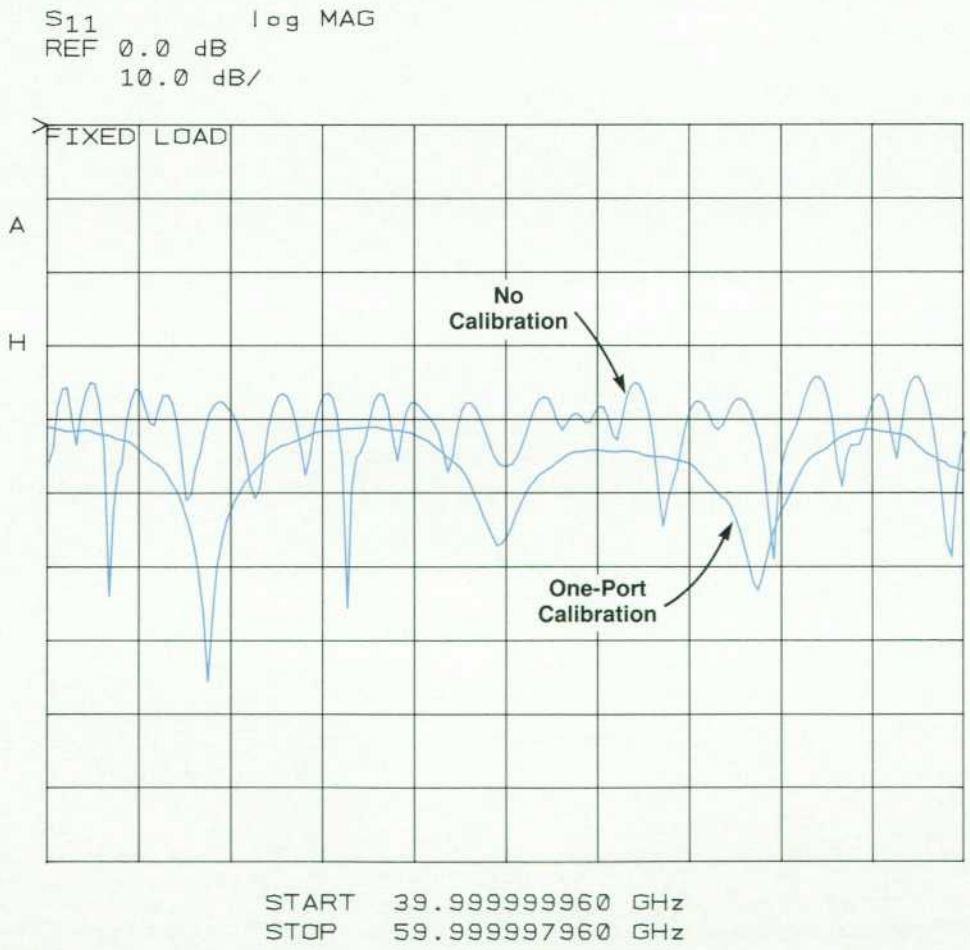


Fig. 4. *Return loss of a fixed load measured with no calibration and with a one-port calibration.*

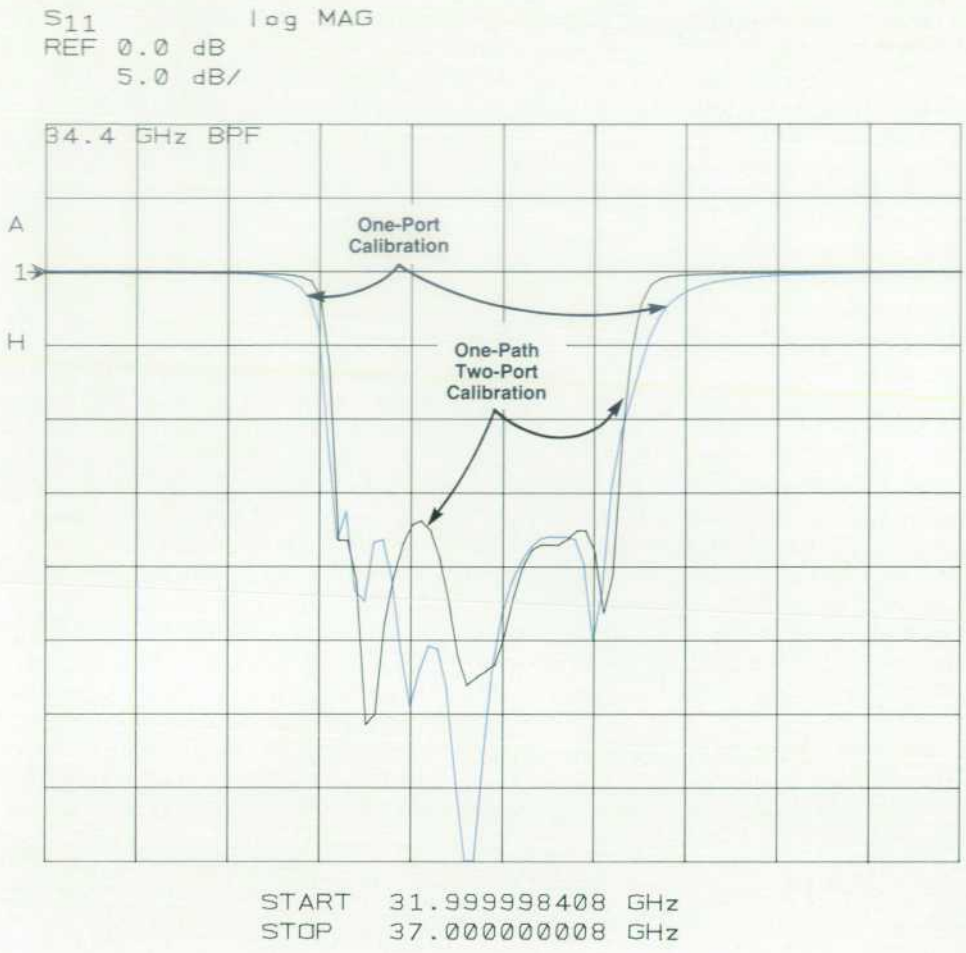


Fig. 5. *Return loss of a bandpass filter measured with a one-port calibration and with a one-path, two-port calibration.*

Let the propagation constant of the quarter-wave line be e^x where $x = (\alpha + j\beta)l$, the round trip loss and electrical length of the line. Then the measured reflection coefficient of the load offset by the quarter-wave line is:

$$\Gamma_{LD} = D + T\Gamma_l e^x/(1 - M\Gamma_l e^x)$$

The short is assumed to be perfect, so its reflection coefficient is -1. The measured reflection coefficient of the short is then:

$$\Gamma_S = D + T(-1)/[1 - M(-1)] = D - T/(1 + M)$$

Finally, the measured reflection coefficient of the short offset by the quarter-wave line is:

$$\Gamma_{SD} = D + T(-1)e^x/[1 - M(-1)e^x] = D - Te^x/(1 + Me^x)$$

The above steps result in four independent equations in four unknowns: directivity, source match, tracking, and the reflection coefficient of the fixed load. Solving these equations and accounting for the electrical length of the quarter-wave line as a function of frequency yields the error coefficients that are then used by the HP 8510B for correcting the measured data.

To simplify the computation required to solve the four

Fig. 6. *Repeatability of magnitude response for three consecutive measurements of a 10-cm waveguide section using a 101-data-point, one-path, two-port calibration.*

simultaneous equations, several assumptions are made. The quarter-wave delay is considered to be a perfect waveguide whose electrical parameters can be calculated from its physical dimensions, although loss can be accounted for if desired. In addition, it is assumed that the uncorrected directivity and source match are reasonably good and that the fixed load is of reasonably good quality. Errors in the calibration results are on the order of the third-order products of these terms. Thus, if directivity, source match, and the return loss of the fixed load are at least 35 dB, the errors introduced by the assumptions will be more than 105 dB down or about six parts per million.

Noncorrectable Errors

Noncorrectable errors are not repeatable over time or from device to device and therefore cannot be measured and subtracted from future measurements. Examples of noncorrectable errors are noise, spurious responses, changes in magnitude or phase caused by cables flexing within the system, and connection repeatability.

Trace noise is typically a few thousandths of a dB at 26.5 GHz, increasing to approximately 0.02 dB at 100 GHz. The noise floor ranges from approximately −90 dB with respect to a through connection at 26.5 GHz to approximately −75 dB at 75 GHz.

Flange connection repeatability and cable repeatability are major contributions to error after calibration. By improving the precision of the waveguide flange, repeatability errors are reduced to below the level of residual systematic errors after correction.[5]

Cable repeatability is also important, especially in the cable supplying the local oscillator signal to the port 2 mixer. Cable phase stability is sensitive to movement and temperature. Cable phase also "creeps" for hours after major deformations, often resulting in three to five degrees of phase shift at millimeter-wave frequencies. If the elastic limit is not exceeded, performance is much better—about ±1 degree.

System Performance

Noncorrectable or random errors are minimized by using a low-noise synthesized signal source, waveguide components with good mechanical stability and low signal leakage, and precision waveguide flanges on the test ports for excellent connection repeatability.

Systematic errors are minimized by calibrating the system using a precisely dimensioned sliding load, quarter-wave line, and fixed load from the HP 11644A millimeter-wave calibration kit.

Two measurements emphasize the advantages of error correction and precision connections at millimeter-wave frequencies. Fig. 6 shows the repeatability of the magnitude response for three consecutive measurements of a 10-cm waveguide section using a 101-data-point, one-path, two-port calibration. The 10-cm section is disconnected and reconnected for each measurement. Fig. 7 shows the mag-

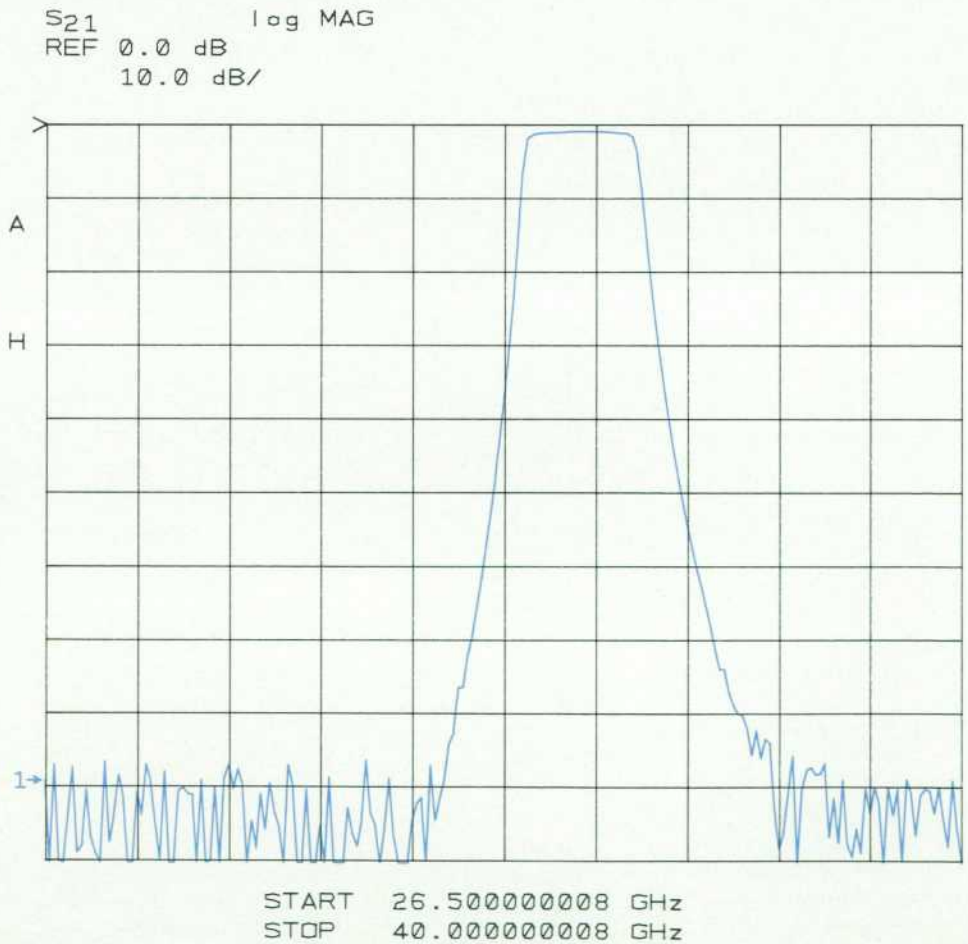


Fig. 7. *Magnitude response of a bandpass filter centered on 34.4 GHz using a 201-data-point, one-path, two-port calibration and an averaging factor of 2048.*

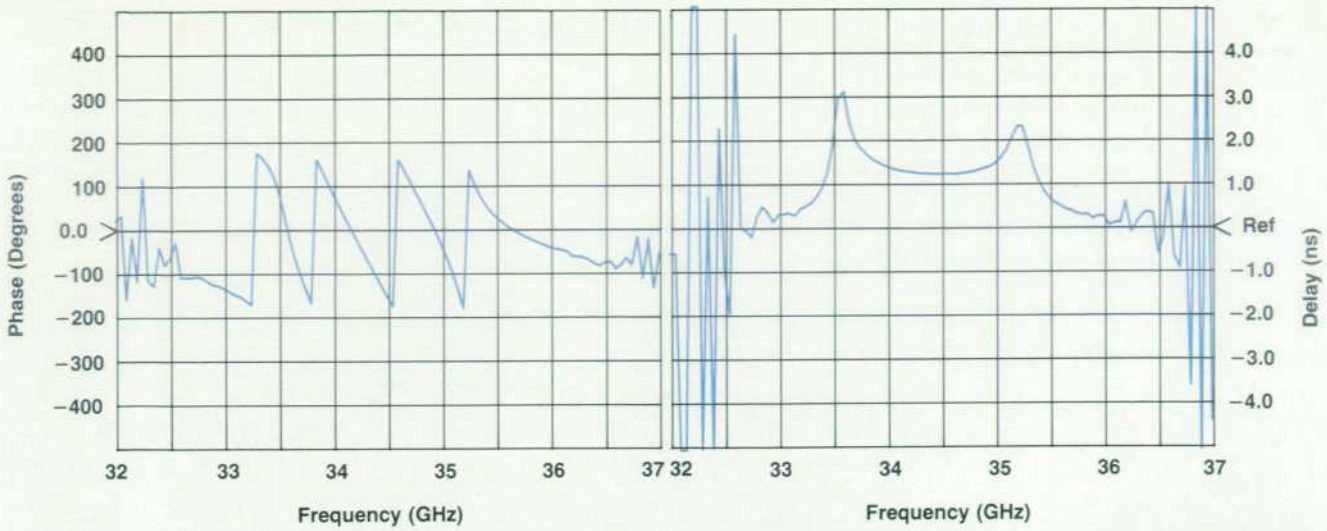


Fig. 8. *Phase and group delay response for the filter of Fig. 7.*

nitude response of a bandpass filter centered on 34.4 GHz using a 201-data point, one-path, two-port calibration and a averaging factor of 2048. Note the high dynamic range, low noise floor, and absence of spurious responses. Fig. 8 shows the phase and group delay response of the same filter.

Time-Domain Measurements

The power of time-domain analysis is shown in Fig. 9, which shows the reflection coefficient in the time domain of a waveguide-to-2.4-mm-coaxial-line adapter terminated with a precision air line and a coaxial load. By loosening connections, it was verified that the large response was caused by the waveguide-to-coax transition. The markers show the location of the time-domain gate used to separate this response from the others. Fig. 10 shows the frequency-domain plot of the transition after gating out the reflections caused by the waveguide flange and fixed load. Using this setup, it was possible to optimize the waveguide-to-coax transition without interference from the waveguide flange or the fixed termination.

Acknowledgments

We would like to thank the many people who contributed to the success of the millimeter-wave network analysis system. Jeff Cauffield did much of the original investigation and first got the system going. Julius Botka contributed

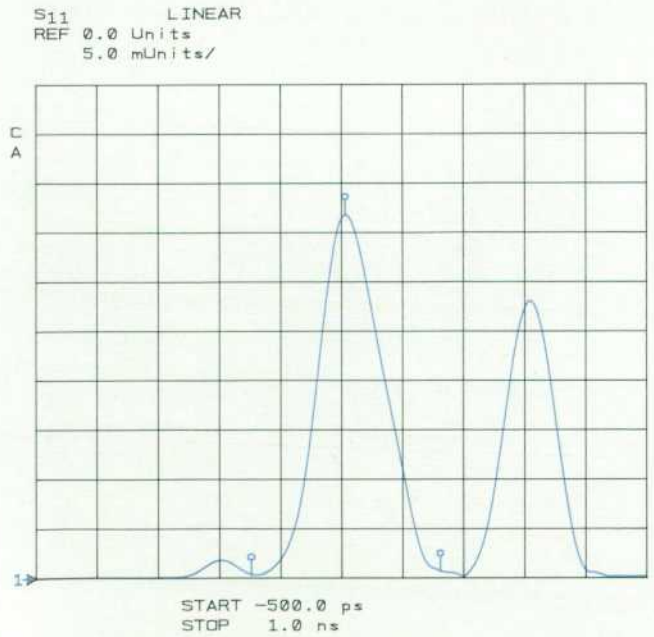


Fig. 9. *Time-domain measurement of the reflection coefficient of a waveguide-to-2.4-mm-coaxial-line adapter terminated with a precision air line and coaxial load.*

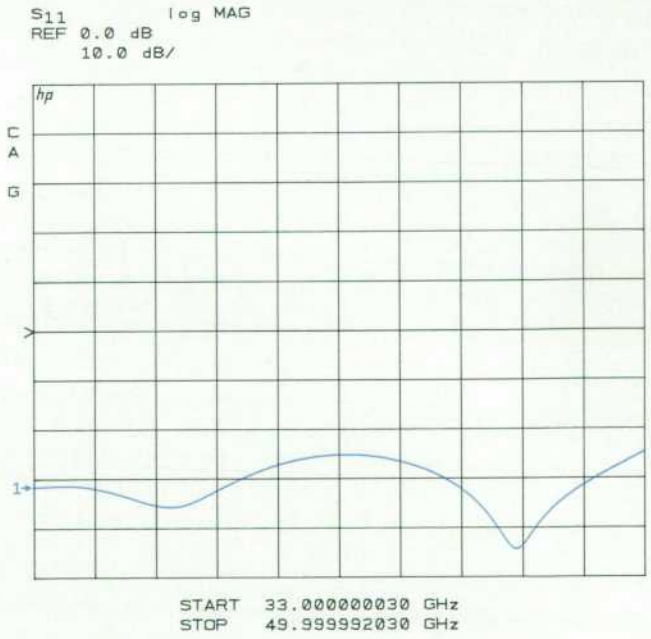


Fig. 10. *Frequency-domain plot of waveguide-to-coax transition for adapter of Fig. 9 after gating out the relfections caused by the waveguide flange and the fixed load.*

many ideas and much design effort toward the precision calibration standards that make the accuracy of this system possible. Bob Hasenick was responsible for supplying the high-quality millimeter-wave waveguide hardware used in the system and Doug Rytting played a key role in providing encouragement and lending his expertise in network analysis to the project. We would also like to thank Dr. Roger Pollard of the University of Leeds for his assistance with calibration techniques and software.

References

1. *Product Note 8510-12*, Hewlett-Packard Company.
2. R.J. Matreci, "Unbiased Subharmonic Mixers for Millimeter-Wave Spectrum Analysis," *Hewlett-Packard Journal*, Vol. 37, no. 11, November 1986.
3. J. Fitzpatrick, "Error Models for Systems Measurement," *Microwave Journal*, May 1978.
4. R. Pollard, "Verification of System Specifications of a High Performance Network Analyzer," *23rd ARFTG Conference Digest*, Spring 1984.
5. J. Botka, et al., "A New Generation of Millimeter-Wave Calibration and Verification Standards," *this issue*, pp. 8-9.

Millimeter-Wave Source Modules

Driven by a microwave source, these modules double or triple the input frequency to generate output frequencies in the millimeter-wave range.

by Robert D. Albin

THE PRIMARY GOAL of Hewlett-Packard's millimeter-wave source modules is to provide performance at millimeter-wave frequencies similar to that available in the microwave bands. This includes leveled output power exceeding 3 dBm at frequencies up to 60 GHz with good frequency flatness. The design goals also included a leveled source match comparable to HP's current microwave sweep generator family. Convenience of use was another consideration. The output of the source modules is in waveguide. Connecting instruments in the rigid environment of waveguide can be cumbersome. By providing input to a module via a flexible coaxial cable, the module can be positioned easily near the waveguide test system.

The block diagram of a millimeter-wave source system is shown in Fig. 1. The microwave source shown can be any source such as a sweep generator or synthesized source that covers the 11-to-20-GHz range. The output of the microwave source is amplified by the HP 8349B Microwave Amplifier. The output of this amplifier drives the input of the millimeter-wave source module through some length of coaxial line. Since the coaxial line introduces loss, it is necessary to amplify the signal inside the source module. The amplified signal drives a diode multiplier which has its output in waveguide. A portion of the multiplied output signal is directionally coupled to a leveling detector. The output of this detector is then fed back to the modulator in the microwave source. This approach eliminates the need for a modulator at millimeter-wave frequencies, an approach that would have lowered the power available at the source module output significantly because of the losses introduced by the modulator.

The frequency of the signal required to drive the source module depends on the waveguide band. The HP 83554A Source Module uses a frequency doubler that converts an input signal of 13.25 to 20 GHz to an output of 26.5 to 40 GHz (R band). The HP 83555A Module uses a tripler to convert an 11-to-16.7-GHz input signal to an output of 33 to 50 GHz (Q band). Similarly, the HP 83556A triples an input of 13.33 to 20 GHz to an output of 40 to 60 GHz (U band).

The transmission medium used for the frequency multi-

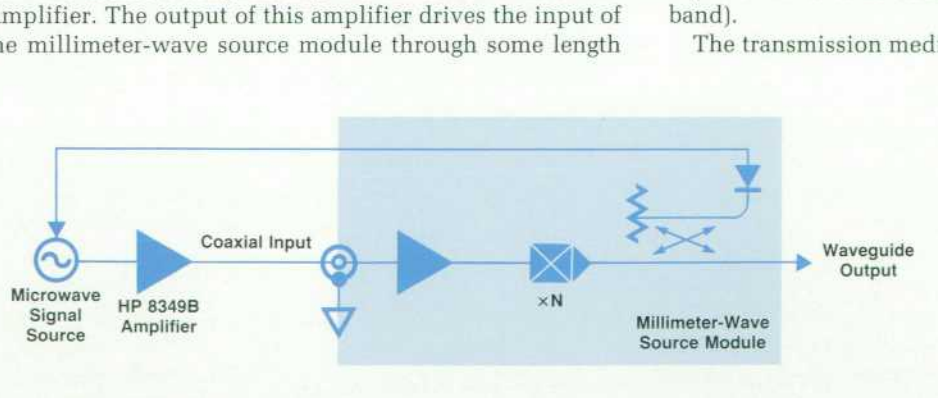


Fig. 1. *Simplified block diagram of a millimeter-wave source system.*

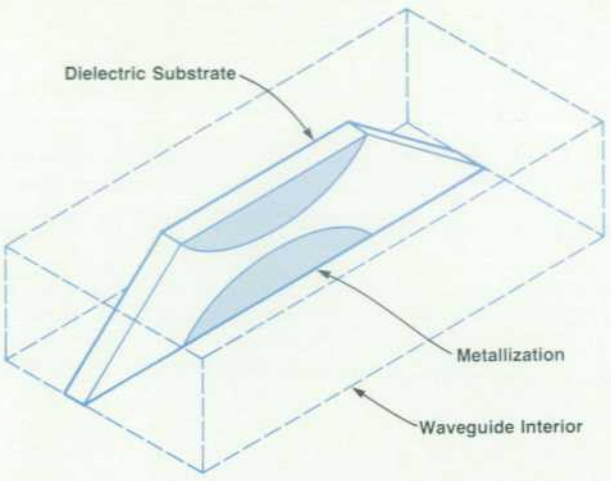


Fig. 2. *Example of a finline structure. A dielectric substrate with an exponential impedance transition metallization pattern located in the center of the waveguide is used to couple the RF energy into the circuitry that can be mounted on the substrate.*

plier, leveling coupler, and leveling detector is finline. A fused-silica substrate is mounted in the maximum electric field region of the waveguide. This occurs in the center of the guide (see Fig. 2 for a simplified finline structure). Energy propagating in the waveguide is coupled first into the dielectrically loaded waveguide. It is necessary to introduce the substrate material gradually or reflections will occur. This is accomplished with a linear cut of the dielectric substrate. A linear slice results in good return loss and is relatively easy to manufacture.

After the energy is coupled into the dielectrically loaded guide, metallization is introduced from the top and bottom of the waveguide. This thin-film metallization provides an exponential transition of impedance from the dielectrically loaded guide to a narrow gap formed by the metallization. The narrow gap is the finline region in which the RF energy propagates in a transverse electric (TE) mode. This gap is convenient for mounting beam-lead multiplier or detector diodes.

Frequency multiplication to 60 GHz is accomplished using a diode tripler. Generation of odd harmonics is usually done by symmetrical distortion of a sinusoidal waveform. If the distortion is perfectly symmetrical, no even-order harmonics of the input signal will be present. A self-biasing antiparallel diode topology (Fig. 3) is used for the HP 83555A and HP 83556A Modules. The tripler is driven by a source impedance which is typically 50 ohms. The input signal propagates through a distributed low-pass filter that provides proper signal routing and termination of the various harmonics. When the input signal achieves an adequate negative or positive amplitude, one of the diodes turns on. This results in clipping of the waveform and generation of odd harmonics. A bandpass filter at the output ensures that only the third harmonic reaches the load resistor.

The self-biasing arrangement allows the tripler to attain low conversion loss for frequencies up to the third harmonic at very high input power (25 dBm). The time constant of the parallel resistors and capacitors is long compared to the incoming signal. This rectifies the incoming signal and generates a dc voltage at the internal nodes of the tripler (V and −V). This dc voltage increases with increasing RF power to the tripler. Without the self-biasing feature, the diodes would clip the incoming signal when its amplitude exceeded the barrier height of the diodes. This would result in saturation at high power levels. The presence of the floating dc voltage causes clipping at higher voltage levels and greatly reduces tripler conversion loss over a wide input power range. Conversion loss for the tripler is typically 16 dB at 60 GHz.

The tripler is realized in a finline structure using a beam-lead diode mounted across the finline gap. A close-up view of this gap inside the waveguide is shown in Fig. 4. The fundamental frequency enters through a coplanar wave-

(continued on page 21)

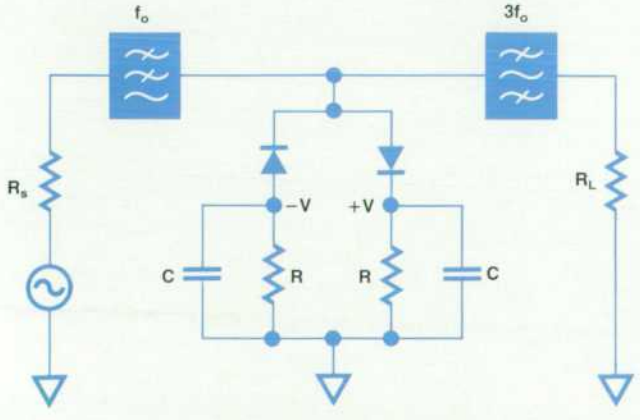


Fig. 3. *Self-biasing tripler circuit.*

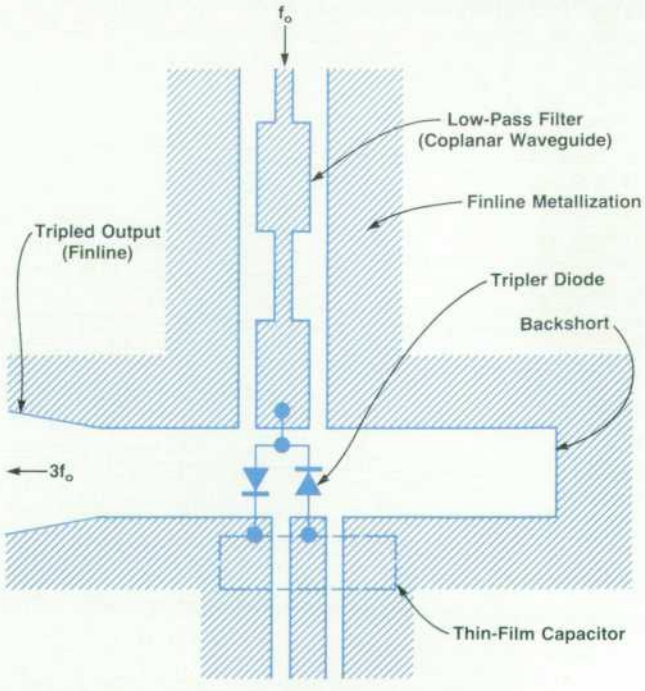


Fig. 4. *Finline tripler structure.*

Millimeter-Wave Source Module Interface

A major design consideration for HP's millimeter-wave source modules was the user interface. In operation the modules are to be functionally transparent to the user. Frequency, power, and modulation control of the modules via the front panel of the source driver was a primary design requirement. As many of these features as possible are made available to owners of existing HP microwave sources, yet the interface architecture is sufficiently general to support future expansion of the remote module concept.

Mechanical Design

The source module interface uses a multiwire cable and a D subminiature connector to route signals between the module and the source driver. This connector was chosen for its low cost, ease of assembly, and field-proven ruggedness. Extensive EMI testing performed on other HP products using this connector provided a high degree of confidence in its RF shielding. The interface cable used with the D connector was designed specifically for this application. This cable contains 20 wires and one low-capacitance coaxial cable, with the size of each wire tailored to suit the signal requirements. In this way, the cable is kept small and flexible without compromising performance. The coaxial cable is intended for returning low-level signals from the module's power detector to the source driver's leveling circuitry. This approach supports present requirements, yet leaves enough spare connections for future needs.

Digital Design

From the beginning, it was known that the driving instrument would need to communicate digitally with the source module. This would allow the controlling instrument to display such things as frequency span and power level, and permit the module to supply bandswitch points, calibration constants, and other data to the instrument. Compelled by the desire to allow module operation with existing instruments, the digital communication path is reserved for enhancements only. Basic features such as frequency and power control or modulation are implemented in such a way as to be backward compatible.

Fitting the digital circuitry into the module was another design constraint. A 4-bit-wide bidirectional data bus was chosen as the best compromise between precious printed circuit board space within the module and minimal pin use on the interface connector. Fig. 1 shows a schematic diagram of the circuitry resident in the module. Data stored permanently in the nonvolatile RAM (NOVRAM) is read by first loading each counter (U2 and U3) with the desired memory address, and then enabling the NOVRAM data output. Using a similiar procedure, permanent programming of the memory is allowed when test points A and B are jumpered together. While this approach burdens the driving instrument's firmware design, it simplifies the hardware required in the module. The module's complete digital interface requires only eight connections and five small ICs.

A nonvolatile RAM was chosen to allow the stored data to be individually tailored to the microcircuits within the module. Since this means that each NOVRAM can contain different data, it is necessary to allow programming and storage of data on the assembly line. Information derived during normal production testing can be downloaded easily into the module. The major concern with this technique is accidental erasure of the NOVRAM. It was found that the simplest, most reliable method of preventing data loss is to require that the NOVRAM programming mode be enabled manually. This prevents entering the programming mode unintentionally. The stored data is recalled immediately following an instrument's power-up cycle to configure the driving source properly for optimum module operation. Any refinement in module performance provided by the digital information is not available to users of older instruments that do not support this feature.

Analog Design

One of the more important features of any microwave source, beyond delivering power over some specified frequency range, is the ability to control that power. Modern microwave sources almost universally employ a negative-feedback scheme requiring a sample of the power taken just before the instrument's output connector. The feedback circuitry, referred to as automatic level control (ALC), typically provides some degree of modulation

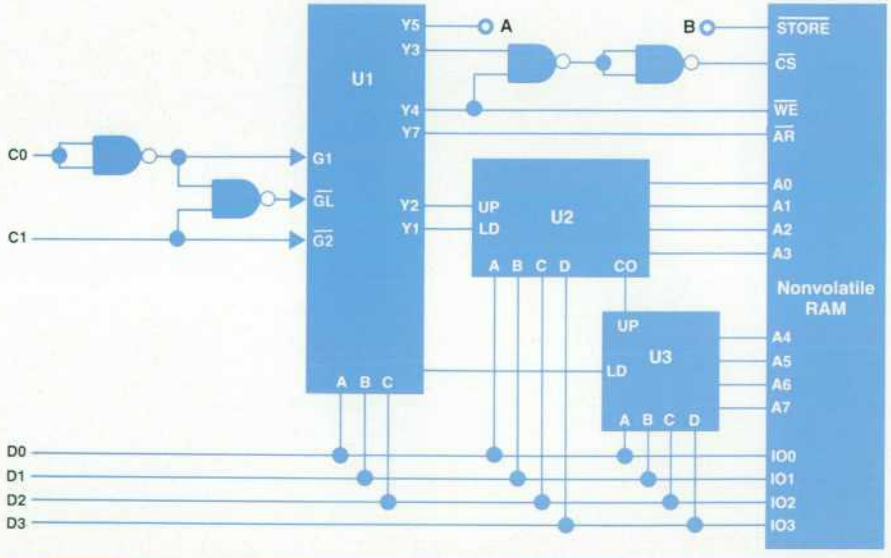


Fig. 1. *Block diagram of millimeter-wave source module digital circuitry.*

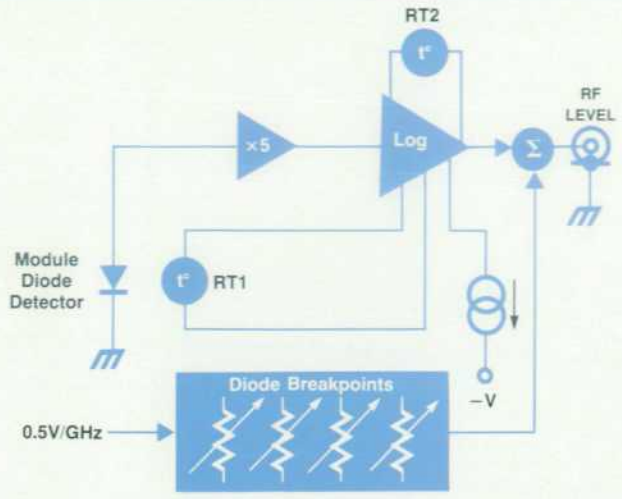


Fig. 2. *Block diagram of millimeter-wave source module analog circuitry.*

capability in addition to simple output power control. In fact, these modulation features can be a significant factor for a user considering a new source. While the basic principles behind ALC operation are simple, most modern designs are not. The millimeter-wave source modules use the source driver's leveling circuitry to conserve space.

The modules incorporate an integrated coupler and detector to sample the module's output power. Because a diode detector's output is typically not linear with power, temperature, or frequency, some conditioning is necessary before sending the signal back to the source. Fig. 2 shows a block diagram of the circuitry used in the source modules. The diode detector output is first amplified by a factor of 5 to improve the bandwidth and slew rate performance at low power levels. The logarithmic amplifier is a special design with external inputs for compensating the temperature dependence of both its own logging transistor and the diode detector. Two thermistors, RT1 near the diode detector and RT2 near the logging transistor, are used to sense the ambient temperature of each device. To compensate for the detector's loss in sensitivity at high power levels, an adjustable current source sets the input level at which the logger gain increases. Finally, the frequency response of the diode detector is smoothed by adding a correction voltage to the logger output. This correction signal is derived from the 0.5V/GHz signal sent to the module from the source driver using a simple diode breakpoint scheme.

All of the compensations mentioned above are individually tailored for each detector diode. Thus, the voltage RF LEVEL returned to the source driver is an accurate indication of the module's output power in dBm. The source driver uses this information to display and control the power at the output of the source module.

John R. Regazzi
Development Engineer
Network Measurements Division

(continued from page 19)

guide line. The input low-pass filter is formed by a series of alternating low and high-impedance transmission lines. The tripler diode is an integrated beam-lead device mounted directly across the narrow finline gap. The generated third harmonic propagates in both directions from the diode. Therefore, a quarter-wave backshort in the finline metallization is required to reflect the signal in the desired direction. The tripled signal propagates down the gap and couples into the waveguide via the exponential impedance transition formed by the thin-film metallization. The thin-film capacitor shown in Fig. 4 provides RF continuity and self-bias holding. In addition, the diodes are biased to reduce conversion loss at low power levels.

A probe coupler (Fig. 5) is included in the integrated microcircuit to provide leveled output power. The metallic coupling probes are photolithographically printed on a fused-silica substrate. A clear overlay substrate placed on top of the probe pattern isolates the probes from the waveguide housing and provides symmetry to the electromagnetic fields. The metallic probes extend partially into the main waveguide and sample a portion of the energy in this guide. This sampled energy is transferred through the waveguide housing and radiated by the portion of the probes extending into the coupled waveguide. Since the probes are spaced one quarter-wavelength from each other, a forward wave is transmitted and a reverse wave is suppressed. The coupling probe shape is optimized for frequency flatness and directivity. Typical performance of this type of coupler is 16-dB (±1.5 dB) coupling with 17-dB directivity over a waveguide band.

Once a portion of the main waveguide output power is coupled to the second waveguide it must be detected. In the millimeter-wave source modules this is accomplished by a finline detector. The energy propagating in the coupled guide is coupled first into the fused silica dielectric. Finline metallization similar to the exponential impedance transition previously described is then introduced. A close-up of the narrow finline gap region in the vicinity of the detector diode is shown in Fig. 6.

The incoming RF causes conduction in the diode, result-

(continued on page 23)

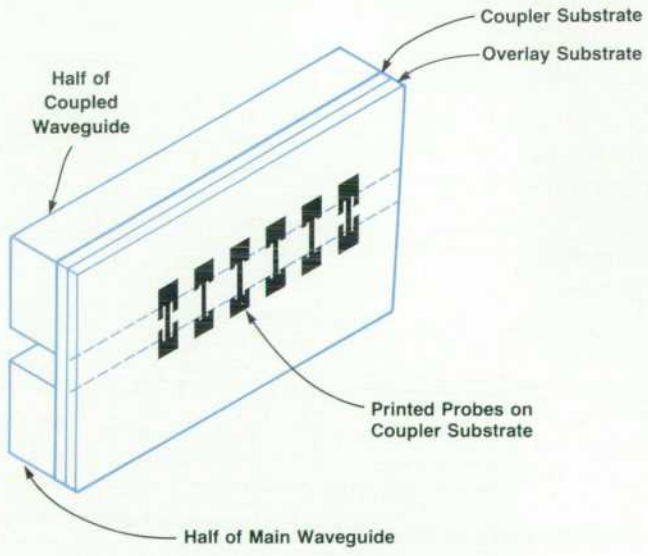


Fig. 5. *Printed probe coupler structure. The foreground half of the coupled and main waveguides has been removed to show the position of the probes.*

2-GHz-to-20-GHz Amplifier

HP's millimeter-wave source program relies upon an interface to a high-power 2-to-20-GHz source. A hybrid microwave amplifier provides this power as the output amplifier in the HP 83550A RF Plug-in and is also available as the stand-alone HP 8349B Microwave Amplifier. For those users who want to upgrade their existing sources with millimeter-wave capability, the HP 8349B Amplifier provides RF and dc power to HP's millimeter-wave source modules. Power can be leveled at either the output of the source module or at the output of the microwave amplifier. The power of the leveling detector is indicated on the HP 8349B's output power display. The HP 8349B also contains interface control circuitry that automatically selects the proper system control when the millimeter-wave source module is connected or when the instrument is used in stand-alone applications.

The heart of the HP 8349B is a 2-to-20-GHz amplifier and a 2-to-20-GHz directional detector realized in a single microwave integrated circuit (see Fig. 1).[1]

Ripple cancellation is the design technique used to create the broadband amplification. Adding the gain ripples of each intermediate stage creates the composite response. By stagger-tuning the interstage ripples, it is possible to produce a level response (Fig. 2). The interstage elements are merely phase rotations of one, three, or five quarter wavelengths at 20 GHz. The lengths result in one, two, or three ripple responses. More ripple interstage responses can be used, but sensitivity suffers. The three-ripple response was found to be the best trade-off between flatness and device sensitivity.

The amplifier design uses six stages to achieve the final gain level (Fig. 3). A 3-dB attenuator is employed in front of the initial stage to improve the input match. The first two amplifier stages, which boost the power level to 11 dBm, are low-power gain sections constructed around 280×0.5-μm GaAs FETs. The next

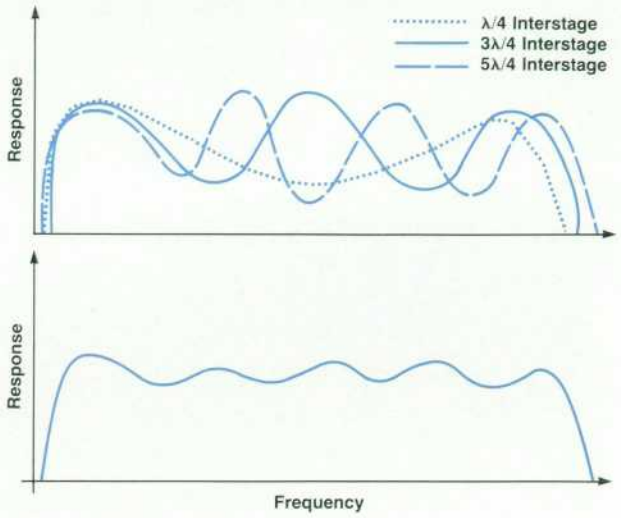


Fig. 2. *By modifying the ripple responses of each stage in an amplifier using different multiples of quarter-wave sections (top), the responses can be summed to yield a linear overall response (bottom). The result using three different ripple responses is shown.*

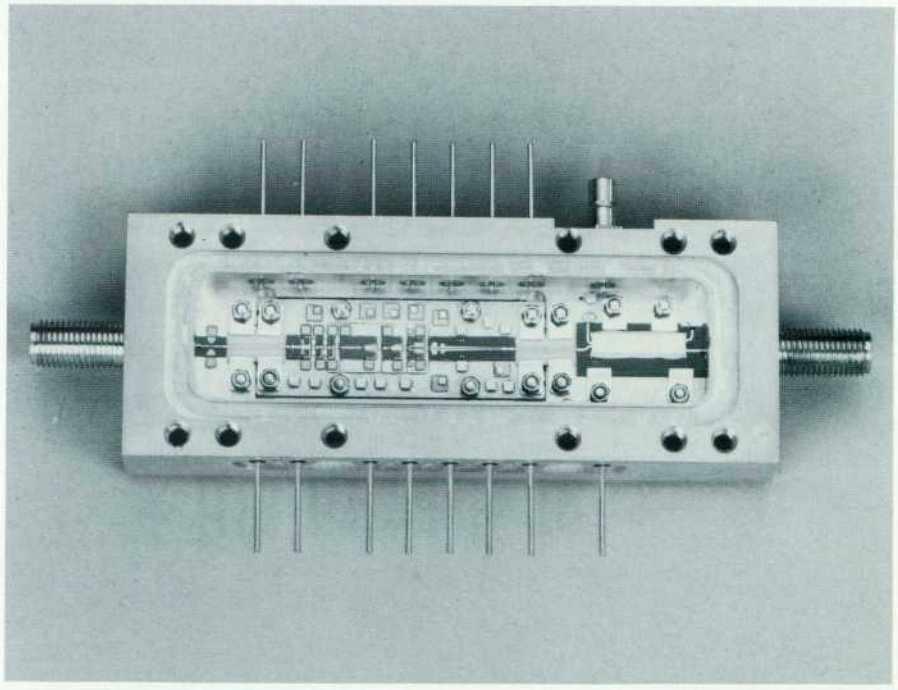


Fig. 1. *The heart of the HP 8349B Microwave Amplifier is this six-stage microwave integrated circuit module.*

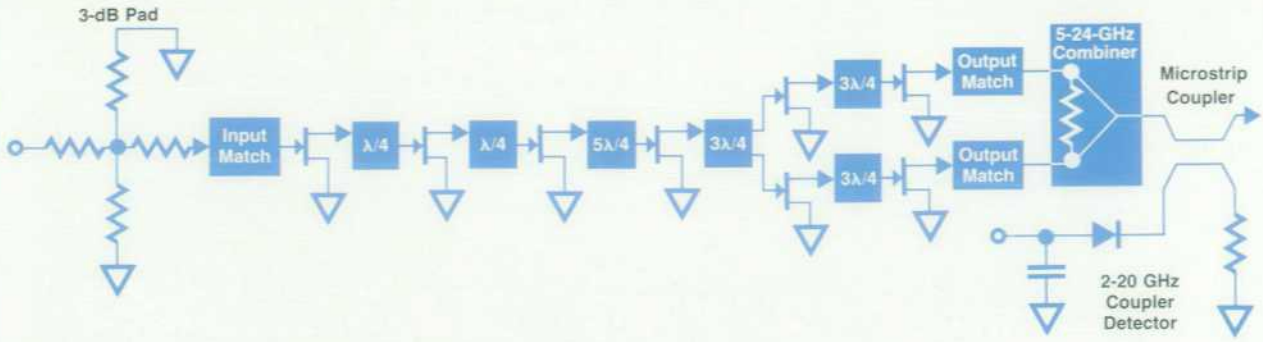


Fig. 3. *Block diagram of 2-to-20-GHz amplifier used in the HP 8349B.*

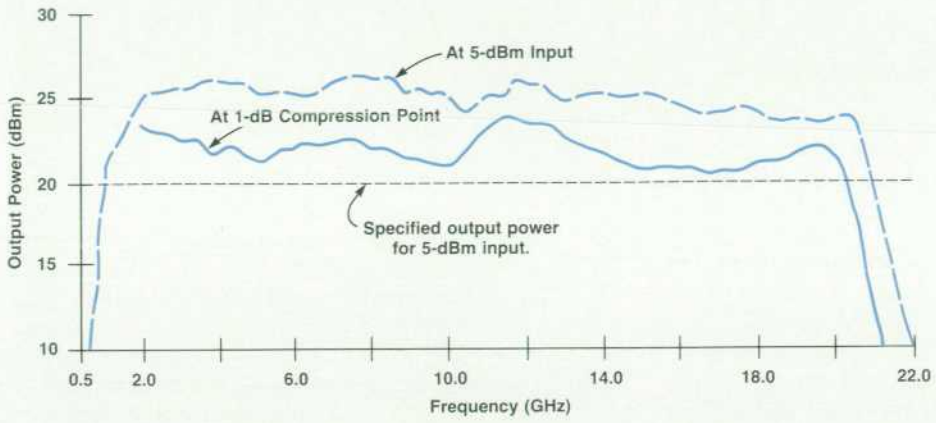


Fig. 4 *HP 8349B Amplifier output performance.*

two stages are medium-power driver stages that employ 750×0.5-μm GaAs FETs. These stages bring the drive level to 18 dBm. The fifth stage relies on a matched dual-cell device with gates driven in parallel and drains tapped individually to form an active power divider. The last stage is a dual-cell 1500×0.5-μm GaAs FET whose cells are driven in parallel and combined through a Klopfenstein tapered power combiner.[2,3] This stage brings the power output level to 23 dBm (Fig. 4). Following this stage, a 22-GHz low-pass filter attenuates the unwanted harmonics generated from fundamental inputs higher than 11 GHz. The directional coupler/detector is the last element in the amplifier package and has a directivity of at least 13 dB from 2 to 20 GHz. A matching network on the coupler circuit matches and flattens the response of the low-barrier silicon diode used in the detector circuitry.

The HP 8349B Microwave Amplifier is the only 2-to-20-GHz amplifier with 20-dBm output power that is currently commercially available. With this component it is possible to move the millimeter-wave source module to the device under test and still have ample RF power to drive the source module.

References

1. J. Meyer, "Broadband Amplifier Aids Test Equipment," *Microwaves & RF*, Vol. 22, no. 13, December 1983, pp. 89-94.
2. R.W. Klopfenstein, "A Transmission Line Taper of Improved Design," *Proceedings of the IRE*, Vol. 44, no. 1, January 1956, pp. 31-35.
3. M.A. Grossberg, "Extremely Rapid Computation of the Klopfenstein Impedance Taper," *Proceedings of the IEEE*, Vol. 56, no. 9, September 1968, pp. 1629-1630.

Jeffrey W. Meyer
Project Manager
Mary K. Koenig
Development Engineer
Network Measurements Division

(continued from page 21)

ing in a dc voltage at the detector output that is proportional to the power incident upon the structure. The resistor shown in Fig. 6 is a thin-film resistor that provides broadband matching. The finline backshort is located less than a quarter wavelength away so that it appears to be inductive at the plane of the detector diode. This inductance resonates with the junction capacitance of the diode to improve detector match. The thin-film capacitor provides RF continuity. By selecting the finline backshort length properly, this type of detector can achieve a 10-dB return loss and a detector flatness of ±1 dB across the 40-to-60-GHz waveguide band. The diode used is a low-barrier planar doped device.[1] This zero-biased gallium-arsenide diode is very stable over temperature. Consequently, the source modules can be operated at low output power levels of −10 dBm with little thermal drift.

The integrated multiplier package is shown in Fig. 7. A mating cover bolted to the base completes the waveguides. The output of the millimeter-wave source module amplifier is input via the coaxial transition at the back of the multiplier package. The fundamental energy propagates through the coplanar waveguide low-pass filter to the multiplier diodes. The main waveguide slot shown in Fig. 7 extends back to the tripler circuit. The third harmonic propagates

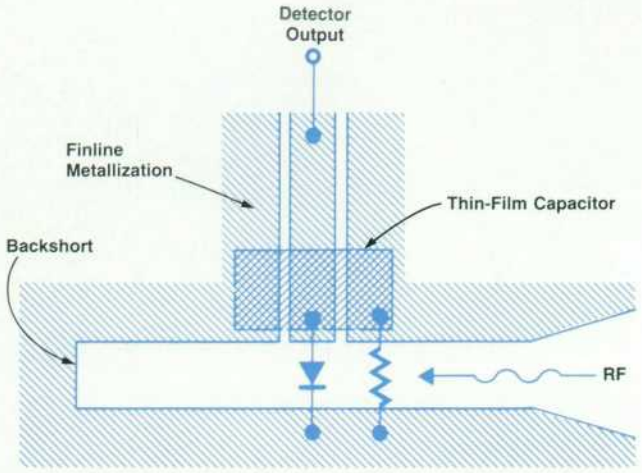


Fig. 6. *Finline detector structure.*

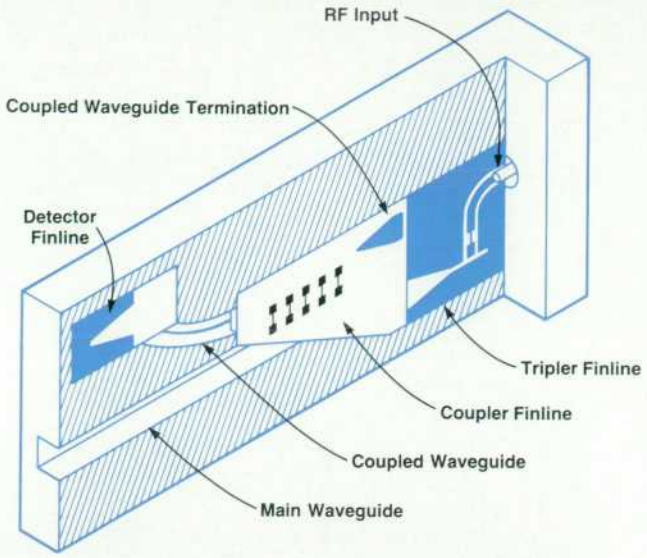


Fig. 7. *Integrated finline multiplier for generating millimeter-wave signals from microwave excitation. See cover for actual device.*

down the finline slot and exponential transition into the dielectrically loaded guide just before the coupler. Most of the tripled signal propagates through the coupler substrate, which is cut at an angle to prevent reflections. After the millimeter-wave signal is in free waveguide, the signal travels to the front of the instrument. The small amount of forward signal coupled to the second waveguide enters the finline detector. The output of this detector is fed back to the modulator in the microwave source for amplitude leveling. Signals reflected back into the source module in the main waveguide are also coupled into the second waveguide. Therefore, it is necessary to terminate the second waveguide with a lossy material. This is accomplished by printing a film of tantalum nitride on the coupler substrate. The pattern introduces this lossy material gradually to prevent reflections. Typical performance of a 40-to-60-GHz HP 83556A Source Module is shown in Fig. 8.

Source Module Amplifier

The microwave signal that reaches the input to the millimeter-wave source module is attenuated by the loss in the coaxial cable that drives the instrument. To achieve maximum output power in the millimeter band it is necessary to amplify this signal before driving the diode multiplier. Because the microwave sweep generators used in the millimeter system are themselves multiplied sources, unwanted multiplied products will be present at their output. These unwanted signals can be mixed by the diode multiplier up into the millimeter-wave band, resulting in degraded harmonic performance of the source module. For this reason it is necessary that the amplifier inside the HP 8355x Source Modules reject the unwanted subharmonics. This internal amplifier is completely realized in microstrip on sapphire substrates.

The amplifier is required to operate in the 11-to-20-GHz range. Since it is used with several different multipliers, depending on the band, it must have good output match. For this reason, the quadrature splitting topology shown in Fig. 9 was chosen. The incoming microwave signal is split by a single-section Wilkinson splitter. The input and output matching circuits are designed to match the gallium arsenide field-effect transistors (GaAs FETs) into 50 ohms. M1 and M2 are distributed transmission lines in microstrip. Assuming the devices are symmetrical, there are equal re-

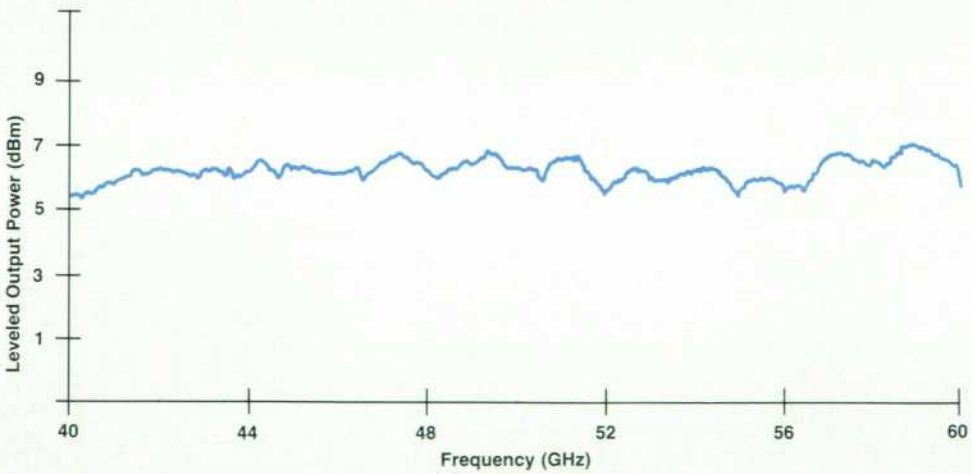


Fig. 8. *Typical performance of HP 83556A Source Module.*

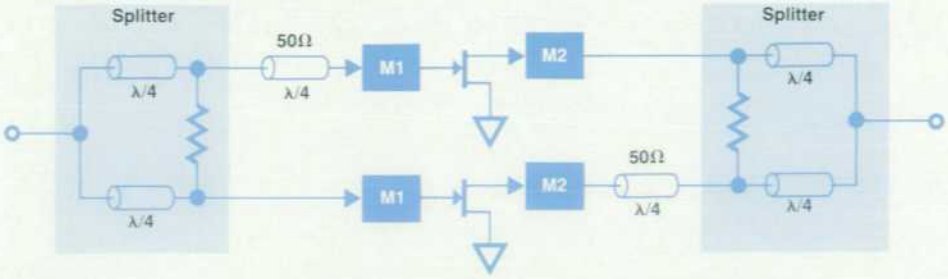


Fig. 9. *Quadrature amplifier circuit.*

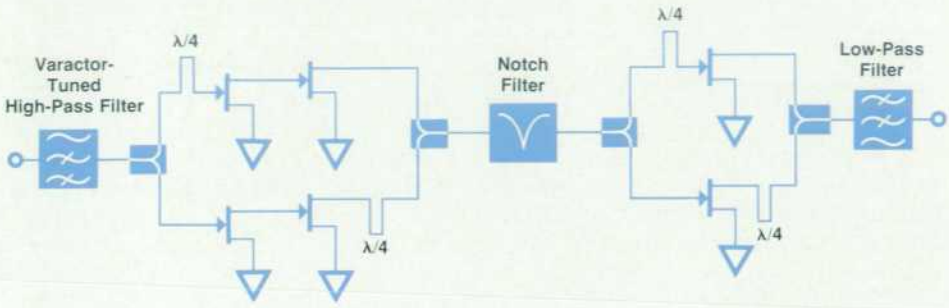


Fig. 10. *Millimeter-wave source module amplifier.*

flections at the inputs to the matching circuits. An additional length of 50-ohm transmission line that is one quarter wavelength long at midband is seen by one path. This means that the two reflected waves will arrive back at the splitter 180 degrees out of phase and be cancelled by the resistor. The forward wave sees an additional phase shift caused by the extra 50-ohm line before the output splitter and recombines constructively. The output match for this type of topology is 10 dB across the 11-to-20-GHz range.

A basic layout of the millimeter-wave source module amplifier is shown in Fig. 10. At the input of the amplifier is a tunable high-pass filter to reject subharmonics. This filter uses gallium arsenide varactor diodes to tune the corner frequency from 12 to 19 GHz. The signal is split after the tunable filter and quadrature shifted to drive two GaAs FETs with 500-μm gate widths in parallel. The output of the two devices of the first stage drives the input of two 900-μm-gate GaAs FETs. The output of the second stage is recombined after phase shift. Before the third stage is a notch filter which also serves to reject subharmonics. The final stage is another quadrature topology using two 900-μm-gate GaAs FETs. To attenuate any second harmonic generated by the amplifier, a fixed low-pass filter is placed at the output. The millimeter amplifier is specified to have 9 dB of gain with 23-dBm output power at 20 GHz. Saturated output power typically is 25 dBm.

Acknowledgments

George Baker, Frank David, Roy Marciulionis, Vinh Nguyen, Bob Schaefer, and Dave Wilson made valuable contributions to the development of these source modules.

Reference

1. E.R. Ehlers, S.W. Johnsen, and D.A. Gray, "Extending Millimeter-Wave Diode Operation to 110 GHz," *Hewlett-Packard Journal*, Vol. 37, no. 11, November 1986.

High-Power Microwave Source for Millimeter-Wave Generation

This plug-in can serve directly as a high-power 8-to-20-GHz source or as a driver for a family of millimeter-wave sources.

by Alan R. Bloom, Roger R. Graeber, Kenneth A. Richter, Andrew N. Smith, and Ronald T. Yamada

THE HP 83550A RF PLUG-IN for the HP 8350B Sweep Oscillator was conceived as a low-cost microwave driver for the HP 8355x family of millimeter-wave source modules. The HP 83550A also functions well as a stand-alone high-power 8-to-20-GHz source. Incorporating this dual personality into one instrument without compromising performance presented a major design challenge.

When connected to an HP 8355x Source Module, the HP 8350B with the HP 83550A displays the millimeter-wave output frequency directly. The detector and associated power-level correction circuitry are located in the source module for best accuracy—the HP 83550A displays the power level at the millimeter-wave output, not at the output of the HP 83550A itself. The source module thus becomes completely transparent to the user, with the front panels of the HP 8350B and HP 83550A acting exactly as if the plug-in were operating directly at millimeter-wave frequencies. Complete programmability of all functions is also maintained because the instrument firmware performs all the needed conversions.

The heart of the HP 83550A is an 8-to-20-GHz YIG (yttrium-iron-garnet) oscillator (Fig. 1). Because this oscillator operates directly at the microwave output frequency, nonharmonic spurious signals are reduced to a minimum. The ALC (automatic level control) modulator, the output coupler/detector, and the associated processing circuitry accurately control the output power level. The pulse modulator is used for both high-speed external pulse modulation and internal square-wave modulation to ensure compatibility with HP scalar network analyzers. For a more detailed description of these and other standard features of HP 835xx plug-ins, see references 1 and 2.

The power amplifier provides 20-dBm output power from 8 to 18 GHz, and 18 dBm up to 20 GHz on no-option units. Since less than 17 dBm is required to drive the millimeter-wave source modules, an external amplifier is not needed. For those applications requiring outputs less than 0 dBm, an optional output attenuator can be used to decrease the minimum power output to −50 dBm.

System Considerations

Before design work on the plug-in could begin, we needed a detailed system specification that described the RF, analog, power supply, and digital interfaces between the microwave source and millimeter-wave source modules. Careful attention to interface specifications guarantees that each millimeter-wave source module will meet its specifications when connected to any HP 83550A in

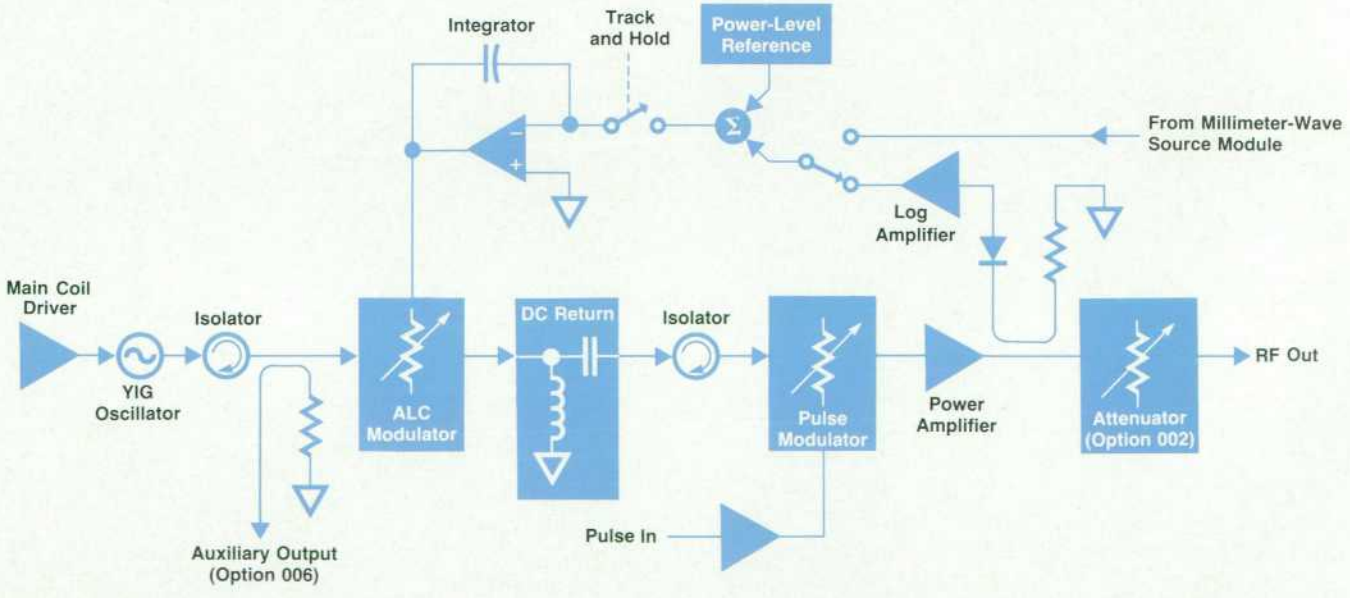


Fig. 1. *Simplified block diagram of the HP 83550A RF Plug-in.*

any HP 8350B.

System requirements affected nearly every circuit in the plug-in. For example, harmonic suppression is important for a microwave sweeper, but is not a major concern when using the HP 83550A to drive the millimeter-wave source modules since the modules include filtering of their own. Above 11 GHz, harmonics are specified at −20 dBc. The HP 83550A performance is typically better than this because the amplifier's built-in low-pass filter attenuates signals above 22 GHz (see box on page 22).

To achieve acceptable stand-alone source performance below 11 GHz, we had to modify the amplifier to reduce the harmonics. Both the unmodified amplifier and the YIG oscillator are specified at −20 dBc. A worst-case analysis assumes that at some frequency the harmonics generated from each source will be in phase. This means the total harmonic signal can have up to twice the voltage amplitude of either source alone, resulting in a net instrument performance of only −14 dBc.

Since the modulators can contribute some harmonic distortion of their own, clearly we were going to have trouble guaranteeing the instrument specification of −15 dBc below 11 GHz. The amplifier was modified to increase its gain at the low end of the band (8 to 11 GHz). This gain slope in effect turns the amplifier into a low-pass filter that attenuates the second and higher harmonics by several dB relative to the fundamental. The extra 5-dB gain at the low end of the band does, however, increase the attenuation requirements of the ALC modulator. Attenuation requirements are stretched further by the millimeter-wave source modules. Since the output of some modules is specified down to −5 dBm, and their insertion loss can be as low as 0 dB, the HP 83550A must be able to attenuate its output down to −5 dBm, not 0 dBm as the instrument specification would imply. When the effects of gain compression in the HP 83550A's internal amplifier and the sum of the gain variations and tolerances of all the other components in the RF chain are added to the above considerations, the plug-in requires a 60-dB modulator to meet its 20-dB output ALC range specification.

The effective frequency accuracy of a nonsynthesized sweeper such as the HP 83550A can be improved by measuring the output with a frequency counter or by using an external phase-locked loop. Unfortunately, this would require an expensive millimeter-wave frequency counter and a millimeter-wave coupler which would degrade the source match and add insertion loss. To avoid this requirement, the HP 83550A's optional auxiliary output allows the user to drive a counter or phase-locked loop at the fundamental frequency. Since the internal coupler is placed before the ALC and pulse modulators in the RF chain (Fig. 1), and the YIG oscillator runs continuously, this signal is available even during pulse modulation or RF blanking.

The decision to allow the YIG oscillator to run continuously caused a potential problem when using the HP 83550A as a stand-alone microwave source with the HP 8757A Scalar Network Analyzer. When used in dc mode, the HP 8502x dc detector probes require a calibration cycle with no RF power applied. No RF power, in this case, is defined as less than −70 dBm. Simply turning off the ALC and pulse modulators in the HP 83550A does not guarantee enough attenuation of the YIG oscillator signal under worst-case conditions. Had the HP 83550A been designated as a source module driver only, this would not have been a problem, since the source modules' insertion loss increases nonlinearly at low power levels. The solution is to turn off the power to the internal microwave amplifier in the HP 83550A during the HP 8757A calibration cycle. Since the amplifiers can have enough broadband noise to upset the HP8757A calibration by themselves, turning off the amplifier solves this problem too.

High-Performance Pulse Modulation

We wanted the HP 83550A/8355x millimeter-wave source module system to have good fast pulse performance. To accomplish this goal, a separate fast pulse modulator with typical rise and fall times of 25 ns was added after the slower ALC modulator (see Fig. 1).

The major factor limiting minimum pulse width (1 μs typical) is not the rise and fall times, but the acquisition time of the track-and-hold circuit. Other HP 8350 plug-ins handle this problem by incorporating a special unleveled ALC mode. Whenever the operator sets the instrument to maximum displayed (unleveled) power, the system microprocessor enables a special high-performance pulse mode by forcing the track-and-hold circuit to track continuously.

While this method does allow narrow (typically less than 100 ns) pulse widths, it does so only at maximum unleveled power. The HP 83550A includes a feature called open-loop mode that breaks the ALC feedback path and uses the output of the power level reference circuit to control the ALC modulator directly.

Although open-loop mode was originally intended as a service feature for troubleshooting the ALC circuitry, we found it useful for obtaining high-performance pulse operation while still retaining some control over the power level. Swept power level flatness is poor in this mode, depending on the unleveled flatness and gain slope of the entire RF chain (typically 15 to 20 dB across the band). For CW or narrow sweeps, however, a front-panel adjustment is provided for calibrating the output level with a power meter. Although not specified, the relative accuracy of the power display at a single frequency typically remains within 10% over the 0-to-20-dBm power range.

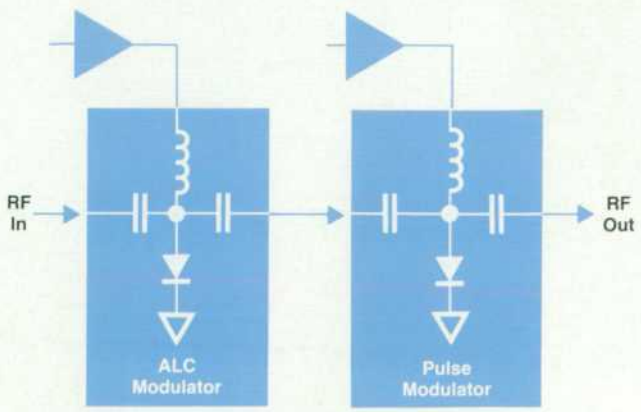


Fig. 2. *Low-frequency equivalent circuits of the pulse and ALC modulators.*

Each modulator consists of a number of pin diodes connected in shunt with the signal path. At low (sub-RF) frequencies, the diodes in each set appear in parallel and can be represented by one equivalent diode connected through the RF coupling capacitors to the diode of the other modulator (Fig. 2). A dc return was introduced to minimize a serious video feedthrough problem between the two modulators. At the instant that the pulse modulator goes from the forward current (RF off) state to the reverse-bias (RF on) state, a 7V peak-to-peak pulse is capacitively coupled to the ALC modulator. Before the dc return was added, this momentary reverse bias caused a full-power spike to appear on the output (Fig. 3a). Since some HP 83550A plug-ins can produce nearly 0.5 watt (27 dBm) at some frequencies, the spike threatened to damage any sensitive device that might be connected. The dc return adds a small RF choke in shunt to ground and a small (1 pF) capacitor in series between the two modulators to eliminate the video feedthrough.

Product Design

The major mechanical design challenges in the HP 83550A were heat control and serviceability. The most important and challenging of these considerations was control of the internal temperature rise. Roughly half of the power dissipated in the plug-in is generated by the current through the main coil of the YIG oscillator. In a YIG oscillator, the frequency of oscillation is directly proportional to the intensity of the magnetic flux imposed on the YIG sphere. That is, a 20-GHz YIG oscillator requires twice the field strength of a 10-GHz oscillator. The flux can be increased without increasing the coil drive current by adding more turns to the coil, but this also increases the inductance. Since the inductive voltage drop is directly proportional to the inductance and the sweep rate ($V = Ldi/dt$), the maximum allowable inductance is limited by the available power supply voltage and maximum sweep speed desired.

In the HP 83550A, approximately 1A at 40V is required to operate the YIG oscillator at 20 GHz, resulting in typically 40 watts of internal dissipation from this source alone. Absolute worst-case power dissipation for the entire plug-in is approximately 88 watts.

All the RF components and major heat-dissipating elements are grouped onto one large aluminum heat sink, known to the project team as the slab. Worst-case power dissipation on the slab is 57 watts. Of all the parts on this assembly, the microwave amplifier is not only the part most sensitive to temperature but also one of the most expensive to replace should it fail. For best reliability, we wanted to keep the amplifier's temperature rise to less than 15°C above ambient.

On our early prototypes, the slab temperature directly

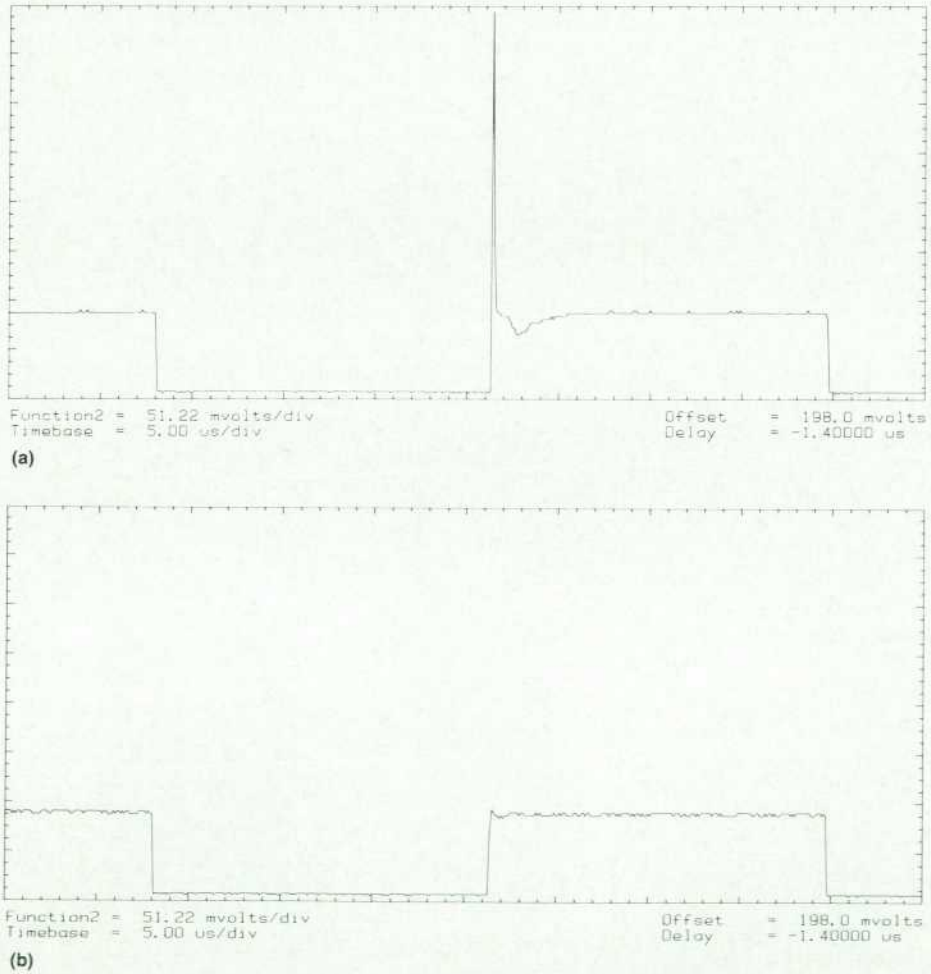


Fig. 3. *A full-power leading-edge spike (a) can be caused by capacitive coupling of the pulse to the ALC modulator. (b) Adding a dc return eliminates the spike.*

under the amplifier was typically 18°C above ambient (full-band 8-to-20-GHz sweep). A slot was cut through the slab to isolate the amplifier thermally from other major heat-generating components. Supported by other modifications to improve the cooling air flow, this change added enough thermal impedance to lower the typical temperature rise of the amplifier to 10.5°C above ambient.

Grouping all the RF components on one easily removable assembly made servicing and manufacturing easier. Frequently, when servicing subtle problems in microwave circuits, simply tightening a connector can have a significant effect on performance. For this reason, it is very useful to be able to gain access to the entire RF chain without disturbing any RF connections.

All the major printed circuit boards are easily accessible and can be mounted on extender cards for convenient troubleshooting. When the plug-in is connected to the HP 8350B mainframe via extender cables, the entire RF assembly can be folded out for testing and repair.

The monolithic microwave assembly also makes production easier. Most of the eight available options for the HP 83550A require changes only to the slab (Fig. 4). The RF portion of the instrument can be assembled and tested separately and then combined as needed with the rest of the plug-in.

Self-Test

When many instruments are combined into a large system, the serviceability issue takes on increased importance. All other things being equal, ten instruments are at least ten times more likely to fail than one. When a failure does occur, localizing the problem to the failed instrument is more difficult. To make the troubleshooting process easier, we designed a series of instrument and assembly-level self-tests into the HP 83550A.

When an HP 83550A is used as a driver for a millimeter-wave source module, three separate instruments make up the source: the HP 8350B mainframe, the HP 83550A microwave plug-in, and the HP 8355x millimeter-wave source module. We felt that we should include at least enough diagnostic capability to allow the user to send back just one instrument for repair, rather than the entire system, thus reducing the cost of ownership.

Some users prefer to do their own repairs. We find, however, that few users attempt to repair microwave instrumentation at the component level. More common is localizing the failure to a specific assembly, and then trading in the bad assembly for a replacement. We decided to emphasize features that would help identify which instrument failed. If the failure occurs in the HP 83550A, further tests are provided to help isolate the failure to a specific assembly.

Circuitry in the HP 83550A is consolidated on a few large printed circuit boards grouped together in a common metal compartment for shielding. This physical layout left enough space to add another printed circuit board for the self-test circuitry. Program memory was doubled by using two ROMs with twice the number of address locations. A hardware bank-select mechanism allows these ROMs to occupy the same logical address space as in former plug-ins. The extra memory allows room for the additional features required to support the source modules and the self-test capability.

There were two primary design objectives for the self-test assembly. One was to keep it inexpensive. The other was to make it as reliable as possible. In this case, reliability means not only avoiding hardware failures in the self-test

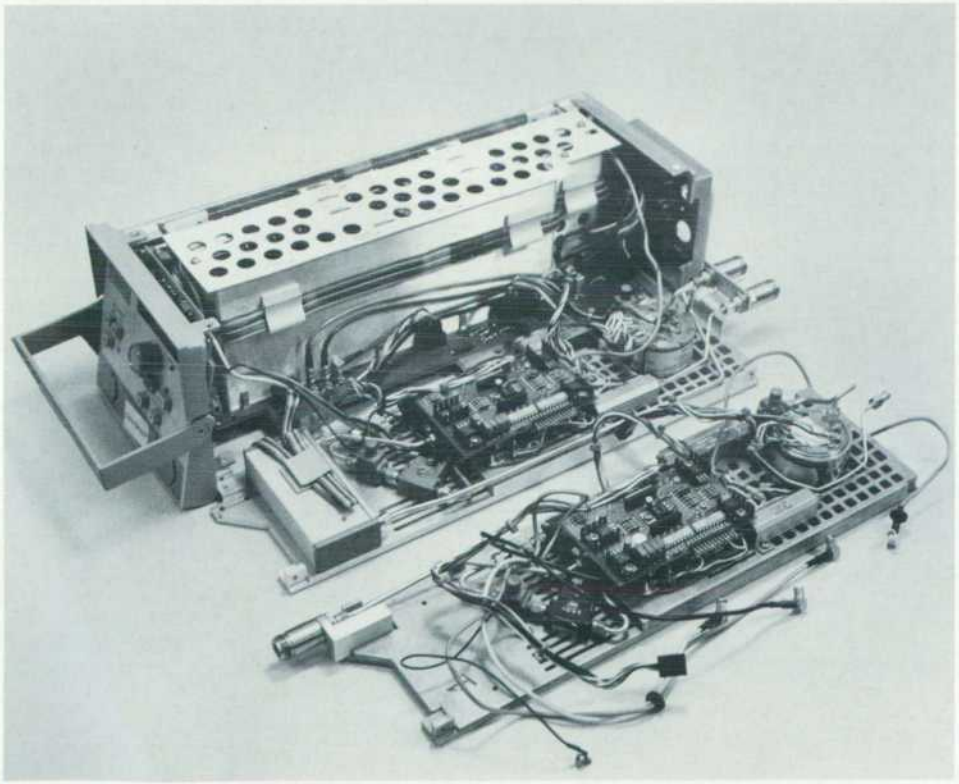


Fig. 4. *All RF components are mounted on a large aluminum heat sink, or slab, which folds out for easy servicing. Different instrument options require changes only to the slab. Two options are shown in the foreground.*

assembly but also avoiding erroneous error detection. To a user, a false error message is the same as a failure. We would gain nothing if an inadequate self-test capability significantly reduced the reliability of the system as a whole.

The self-test assembly consists of a series of analog multiplexers to select the signals to be measured and an analog-to-digital converter (ADC) to measure the various voltage levels and send the information to the system microprocessor. All circuit components run on the instrument's 5V power supply so that the testing circuitry does not fail if one or more of the other power supplies goes down. Relatively wide tolerance bands are used on all measurements to reduce the possibility of false error indications. The modest accuracy requirements allowed us to eschew a high-performance, high-cost ADC in favor of a less-expensive eight-bit part.

The self-test routines run automatically when power is turned on or the **INSTR PRESET** button is pressed. They also can be initiated at any time by an appropriate set of keystrokes. Before any self-testing starts, the firmware checks for the presence of an operable self-test assembly. Hence, should a self-test hardware failure occur, normal operation of the instrument can be restored simply by unplugging the assembly. The order of the tests ensures that the first components checked are those whose failure would invalidate subsequent system tests. If a failure is found, testing stops, and an error code is displayed to indicate the location of the failure. Further diagnostics can then be obtained by executing the remaining tests through a series of front-panel keystrokes. Millimeter-wave source module tests are last in the execution sequence and are performed only if a source module is present.

Acknowledgments

We would like to thank Glenn Hetchler for his invaluable contributions to improving the reliability and producibility of the HP 83550A throughout the design cycle. Thanks also to Rolf Dalichow for his administrative guidance. Dave Copley did much of the original mechanical and thermal design on the project, and Peter Ho made similar contributions to the early design of the self-test hardware and software. Finally, we would like to thank Mike McAndrews, Ruben Mandujano, and Dallas Parr for their invaluable help in testing, troubleshooting, and documenting this product.

References

1. R. Dalichow and D.E. Fullmer, "A Broadband Fully Programmable Microwave Sweep Oscillator," *Hewlett-Packard Journal*, Vol. 33, no. 2, February 1982, pp. 3-10.
2. G.W. Holmlund, G.E. Elmore, and D.C. Wood, "A New Series of Programmable Sweep Oscillator Plug-ins," *ibid*, pp. 11-21.

Millimeter-Wave Detectors Extend Range of Scalar Network Analyzer

by Herbert L. Upham

HEWLETT-PACKARD'S new family of millimeter-wave detectors designed specifically for use with the HP 8757A Scalar Network Analyzer[1] extends the capabilities of this instrument for millimeter-wave system measurements.[2] These detectors operate in both modulated (ac) and unmodulated (dc) modes for optimum performance in various measurement configurations. The unmodulated mode is particularly beneficial in millimeter-wave measurements, because the source does not need to include an accurate modulator. These detectors provide excellent accuracy in a wide variety of applications. All three detectors work over their designated waveguide bands with typical flatness better than ±1.5 dB and return loss greater than 12 dB. This performance provides measurement advantages over most stand-alone detectors in these frequency ranges. The HP 85025C Detector Adapter is used to connect other detectors to the HP 8757A, extending its capabilities to other waveguide bands.

Specifications

Performance specifications for the detectors are given in Table I. To guarantee the dynamic accuracy specification, an HP 8757A with revision 2.0 or later firmware is required.

Table I
Performance of Millimeter-Wave Detectors

Detector	HP R85026A	HP Q85026A	HP U85026A
Frequency range (GHz)	26.5-40	33-50	40-60
Flatness (dB)	±1.5	±1.5	±2.0
Return loss (dB)	≧12	≧12	≧12
Maximum input power	16 dBm	20 dBm	20 dBm
Input waveguide	WR 28	WR 22	WR 19
Dynamic accuracy	±(0.3 dB + 0.03 dB/dB) referenced to 5 dBm typically < ±0.5 dB from 10 to −40 dBm		
Dynamic range	ac mode: 10 to −50 dBm dc mode: 10 to −40 dBm		

Design

The design of these detectors is common except for the microwave detector design of the HP R85026A, which uses an integrated detector diode circuit mounted in a coaxial module. The HP Q and U85026A detectors use a detector diode mounted in a section of waveguide rather than using a waveguide-to-coax transition. All three detectors have circuitry that processes the detected output signal before passing it to the HP 8757A via a front-panel interface cable.

The HP R85026A coaxial detector module is attached to a waveguide-to-coax transition. The waveguide transition is similar to the transition described in the article on page 35. The coaxial output has a characteristic impedance of 50 ohms, which matches the input impedance of the detector circuit. The detector integrated circuit consists of a 50-ohm matching resistor network, a detector diode, and a holding capacitor at the output.[3]

The resistors on the integrated circuit are physically small enough to be considered lumped circuit elements even at 50 GHz. The diode junction capacitance is also small enough so that no special reactive tuning is required for compensation. Thus, a lumped circuit implementation of the detector circuit is achieved.

This circuit has the advantage of presenting a nearly constant input impedance (Z_{in}) as long as the diode video impedance is large compared to 50 ohms. This circuit requires reactive compensation to work at extremely high frequencies where the diode junction capacitance reactance approaches 50 ohms.

The HP Q and U85026A detectors use a finline design technique.[4] The design is illustrated in Figs. 1 and 2. The substrate material is fused silica, chosen for its low dielectric constant and low loss from 30 to 60 GHz. The dielectric substrate is mounted in a reduced-height waveguide section. The height reduction is necessary to compensate for the dielectric loading of the waveguide by the substrate. If the waveguide height is not reduced, higher-order waveguide modes will propagate in the detector and cause return loss and flatness variations. A gold pattern on one side of the substrate captures the electric fields propagating in the waveguide. The pattern tapers from the waveguide height to a width that represents approximately the impedance of the detector diode. At the input to the substrate a diamond-shaped patch of resistive material forms a broadband attenuator. The attenuator improves the return loss

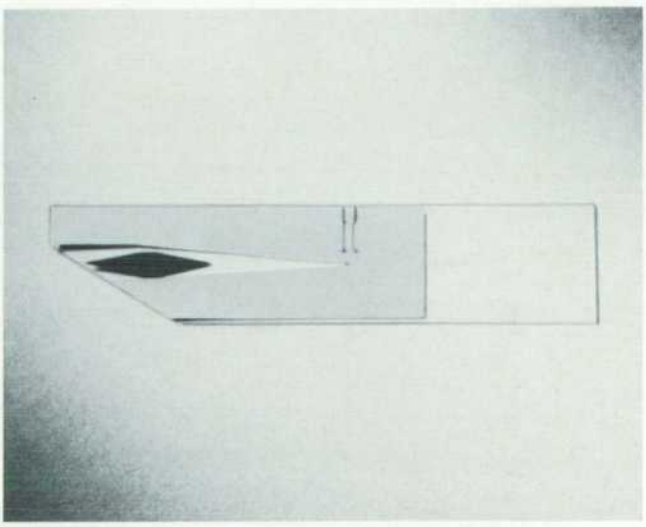


Fig. 1. *Finline substrate for HP Q/U85026A Detectors.*

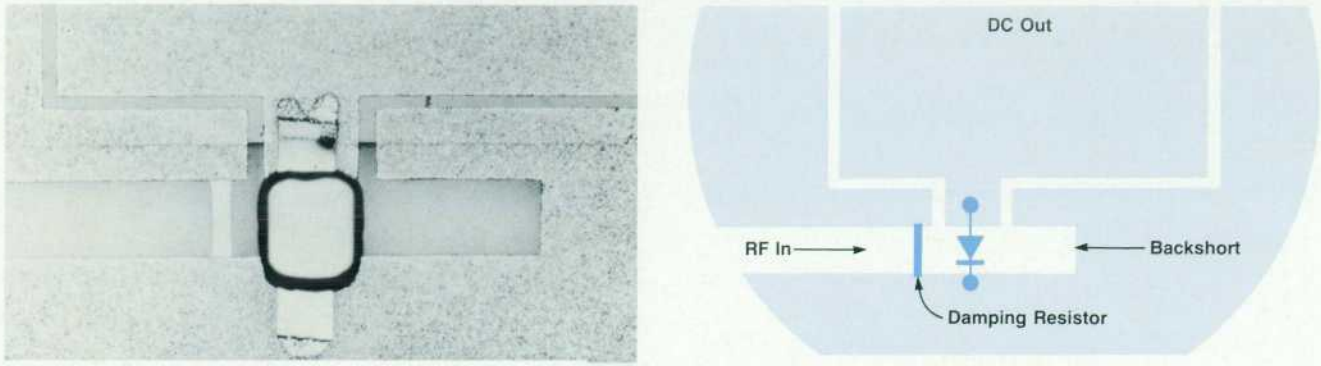


Fig. 2. *Photograph (left) and drawing (right) of beam-lead diode mounting area on finline substrate shown in Fig. 1.*

of the detectors with only a small effect on sensitivity and flatness.

The detector diode used in both products is a modified barrier beam-lead diode.[5] This diode was chosen because of its small physical size and very low junction capacitance. The construction of the detector circuit can be seen in the photograph in Fig. 2. The beam lead diode is positioned across the circuit channel, and is followed by a short circuit. The short can be modeled as an inductance parallel to the junction capacitance of the beam-lead diode. By adjusting the distance from the diode to the short, a resonant circuit is formed that terminates the channel in a very high impedance. A small resistor is placed just before the diode. This resistor reduces the Q of the resonant circuit, which improves the broadband match to the channel impedance.

Application Considerations

These detectors are ideally suited for scalar network analyzer applications in the millimeter-wave frequency range. Their ac or dc detection capability provides flexibility in using sources that may not have the accurate modulation characteristics required for ac detection. The dynamic range and frequency response of these detectors also allow their use as power monitors. A typical measurement system for measuring transmission and reflection simultaneously is shown in Fig. 3.

By using two directional couplers (or a dual directional coupler) the reflection measuring system uncertainty can be determined by the following equation:

$$\Delta\rho \simeq A + B\rho_1 + C\rho_1^2$$

where A is the directivity, B is the calibration error, C represents the effective source match, and ρ_1 is the measured reflection coefficient.

Directional couplers are often used to provide a good

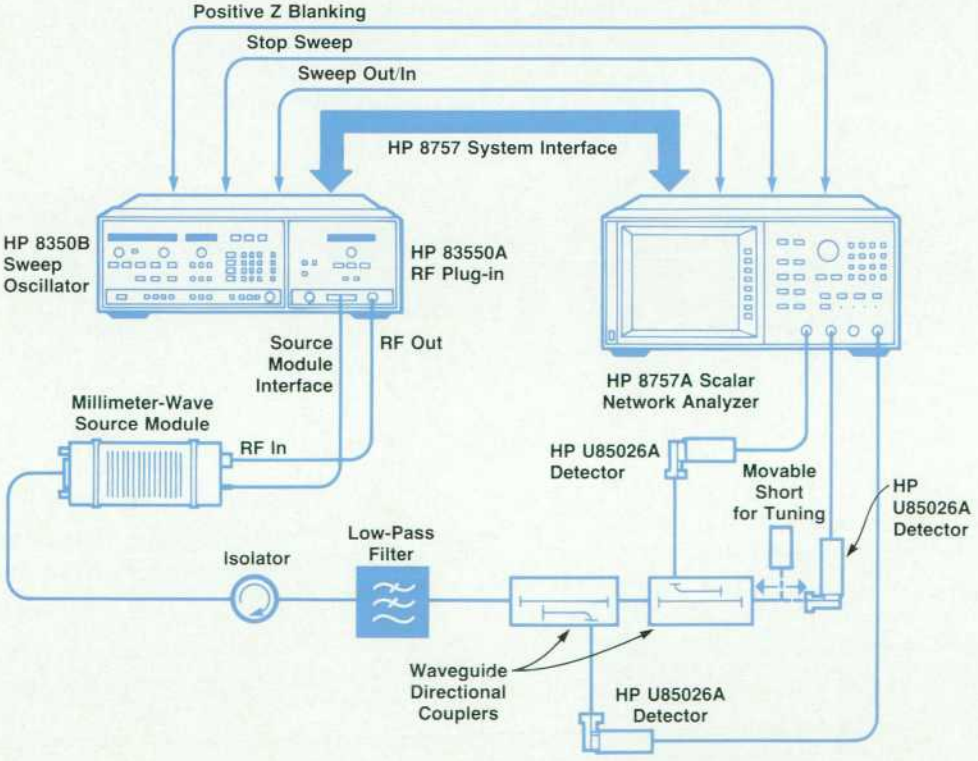


Fig. 3. *Reflection and transmission loss test setup.*

Waveguide Reflectometer Calibration

The HP 8757A Network Analyzer has a built-in calibration routine for establishing the 0-dB reference line. However, the routine does not work for a waveguide reflectometer. Coaxial transmission lines have low dispersion (the velocity of propagation varies little over many octaves of bandwidth) while waveguides are very dispersive[1] and the open end of a waveguide can act as an antenna.[2] Thus if an open/short calibration is attempted, the reference line will be erroneous.

The goal of the calibration is to establish a 0-dB reflection calibration line that is a best estimate when taking the effective source match of the reflectometer into effect. A waveguide sliding short can be used to establish one (open or short) reference, and then can be moved a quarter wavelength to establish the second reference. This works well over a limited portion of a waveguide band where the phase velocity change is small.

Unfortunately, for a full band sweep it has problems. Dispersion will cause the calibration to be poor near the band edges (assuming the open/short references were taken near midband). By moving the short, the amount of error can be seen on the HP 8757A display. If the error is not acceptable, a more sophisticated algorithm must be used to establish the calibration. One method is to make several measurements of the reflection of the moving short. Each measurement is stored in a controlling computer. The average of several measurements is computed and written back to the HP 8757A memory. This average then can be used as the 0-dB reference for the reflectometer.

References

1. S. Ramo, J.R. Whinnery, and T. Van Duzer, *Fields and Waves in Communication Electronics*, Wiley & Sons, 1967, p. 49.
2. *ibid*, pp. 671-673.

trade-off between coupling coefficient, accuracy, and dynamic range. The accuracy of the reflectometer can be determined for a range of measured values from the graph shown in Fig. 4. The calibration error B is assumed to be zero, the directivity A is 40 dB, and the effective source match C is an SWR of 1.15 for this example.

The measurement uncertainty varies with the magnitude of the measured reflection coefficient. The detector's contribution to the uncertainty is through the effective source match of the reflectometer. If the return loss of the reference detector is greater than 12 dB, the detector will contribute a maximum error term of −32 dB (12 dB + 2×10 dB for the incident and return passes through the 10-dB directional coupler) to the effective source match. This contribution is negligible compared to the directional coupler's main line source match and thus does not limit the uncertainty.

Effects of Dynamic Accuracy. Dynamic accuracy introduces errors when measuring large changes in power level. The uncertainty is a result of the detector diodes not responding identically. The HP 8757A contains (in firmware) dynamic calibration tables for several different diode families. These calibration tables are established for average diode performance; hence, small variations in diode structure cause measurable errors in dynamic accuracy. It is also difficult to establish a variable absolute power standard at millimeter-wave frequencies. A specially calibrated rotary vane attenuator is used at the factory to calibrate these detectors, but the dynamic accuracy specification still contains some uncertainty caused by the calibration of the variable attenuator and its repeatability.

Effects of Frequency Response. The frequency response of the detector is only a concern when it cannot be easily removed from the measured value by making a ratioed measurement. This occurs when monitoring the output power from a device under test. In this case the frequency response of the detector applies directly to the measure-

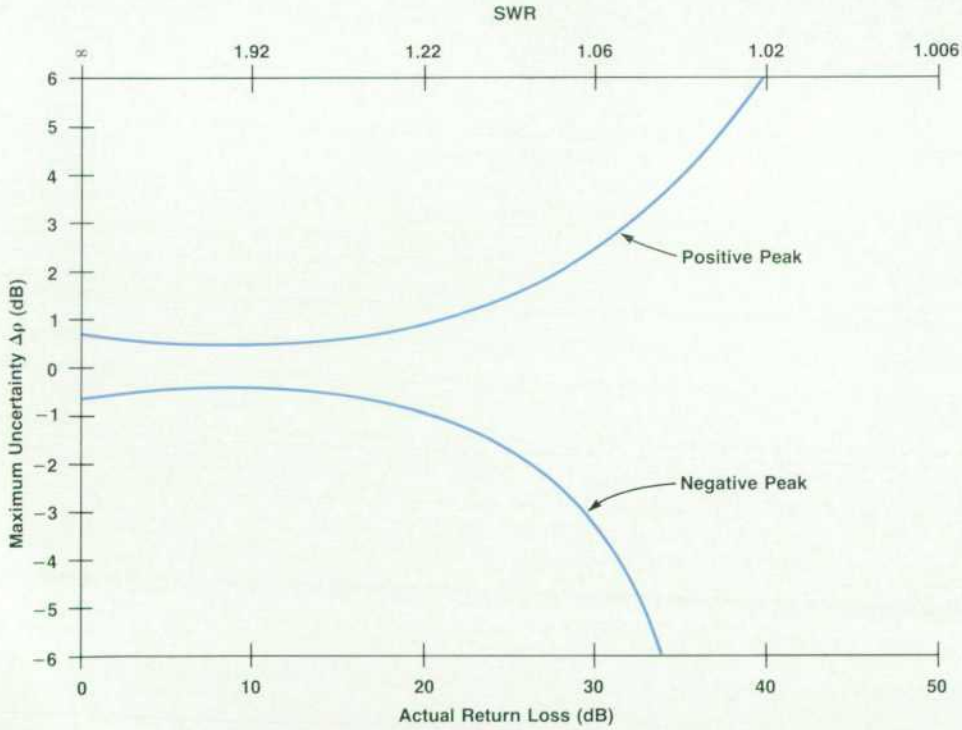


Fig. 4. *Reflection loss error graph for test setup in Fig. 3. The error is a sinusoidal variation around the true value. The upper curve indicates the positive peak and the lower curve indicates the negative peak.*

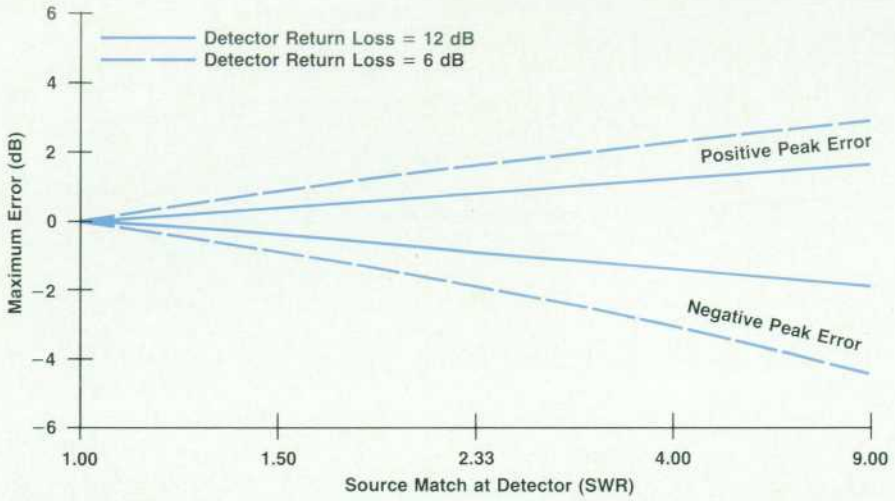


Fig. 5. *Source match error graph.*

ment as an error. To reduce this error an individual calibration of each detector must be made relative to a power standard. The calibration can be stored in a computer and then written to the memory of the HP 8757A. By displaying measured power minus the calibration, a power measurement with enhanced accuracy can be made.

Transmission Test System

When measuring transmission coefficients of devices with losses greater than 10 dB, the greatest factor affecting accuracy is dynamic range. For devices with losses less than 10 dB the return loss of the detector will be a significant error. In Fig. 5, a plot of measurement error versus source match is presented. Loss through the transmission device is assumed to be zero, so the full effect of the detector return loss is seen. Curves for 6-dB and 12-dB detector return losses are shown.

Effect of Dynamic Range. The dynamic range of the detector establishes limits on the maximum gain or attenuation that can be measured by the system. Harmonic content of the input signal can also limit dynamic range, but this is not usually a problem in waveguide systems. In addition, a low-pass filter can be used to reduce the harmonic content of the test signal.

HP 85025C and Other Waveguide Bands

In waveguide bands other than U, Q, and R, it is still possible to make measurements with the scalar analyzer systems described in this article. The HP 85025C detector adapter is an instrument that adapts zero-biased detectors to the HP 8757A Scalar Network Analyzer. Calibration routines are available to optimize dynamic accuracy for each detector.

Acknowledgments

The author would like to thank the many team members whose efforts at Network Measurements Division made the products described here possible.

References

1. J. Egbert, et al, "Advanced Scalar Analyzer System Improves Precision and Productivity in R&D and Production Testing," *Hewlett-Packard Journal*, Vol. 37, no. 2, February 1986, pp. 24-39.
2. *HP 85025C Operating and Service Manual*, Hewlett-Packard Company.
3. C.C. Chang, et al, "A Zero-Biased GaAs Millimeter-Wave Integrated Detector Circuit," *IEEE Microwave Theory & Techniques Symposium Digest*, 1982.
4. K. Solbach, "The Status of Printed Millimeter-Wave E-Plane Circuits," *IEEE Transactions on Microwave Theory & Techniques*, Vol. MTT-31, no. 2, February 1983.
5. E.R. Ehlers, et al, "Extending Millimeter-Wave Diode Operation to 110 GHz," *Hewlett-Packard Journal*, Vol. 37, no. 11, November 1986.

Design and Performance of Millimeter-Wave Thermocouple Sensors

by Lee H. Colby

WITH THE INCREASE in design activity in the 33-to-50-GHz frequency range, Hewlett-Packard wanted to extend HP's power measurements into this millimeter-wave frequency range. The HP Q8486A and HP R8486A Power Sensors shown in Fig. 1 incorporate a special method that inserts a 50-MHz calibrating signal into the waveguide. The HP Q/R8486A sensors have the same 50-ohm thermocouple used by HP's lower-frequency coaxial power mounts and share the same one-microwatt-to-100-milliwatt power range, low drift, low SWR, 50-MHz calibration, and accuracy that these coaxial thermocouple power sensors have. The HP Q8486A operates in the 33-to-50-GHz waveguide band and the HP R8486A operates in the 26.5-to-40-GHz band.

Since the thermocouple was designed for 50-ohm systems, it was necessary to use a TEM structure for the 50-ohm thermocouple and connect it to a low-reflection coax-to-waveguide adapter. The adapter's center conductor, which is normally screwed into the waveguide wall opposite the coaxial entry point, is isolated from ground by a high-frequency choke. The choke allows the 50-MHz energy to be fed across the waveguide through the coax to a coplanar transmission line and into the thermocouple. However, for millimeter-wave power, the choke reflects a short at the center conductor where the center conductor is normally screwed into the waveguide. The millimeter-wave signal coming down the waveguide is transformed from the waveguide impedance to 50 ohms because of the transformer action of the multistepped waveguide.

The ability to diplex the 50-MHz and millimeter-wave signals allows the power meter's gain to be set for the sensor head in use and removes the requirement that the thermocouple and its amplifier have a constant gain versus time, environment, and reasonable overloads.

The thermocouple's low output voltage, approximately 60 nanovolts at one microwatt, is chopped at 220 Hz to eliminate dc drift, amplified, and then fed to the power meter.

Calibration information is typed on each sensor's label and includes the calibration factor, which is the efficiency at which the sensor converts the absorbed power into a power reading, and the reflection coefficient. The calibration data is supplied at 1-GHz intervals across the waveguide band and the calibration factors are traceable to the U.S. National Bureau of Standards (NBS), or to a dry calorimeter for frequencies where NBS does not have standards. The power mount's microwave performance is tested and calibrated using an automated network analyzer.

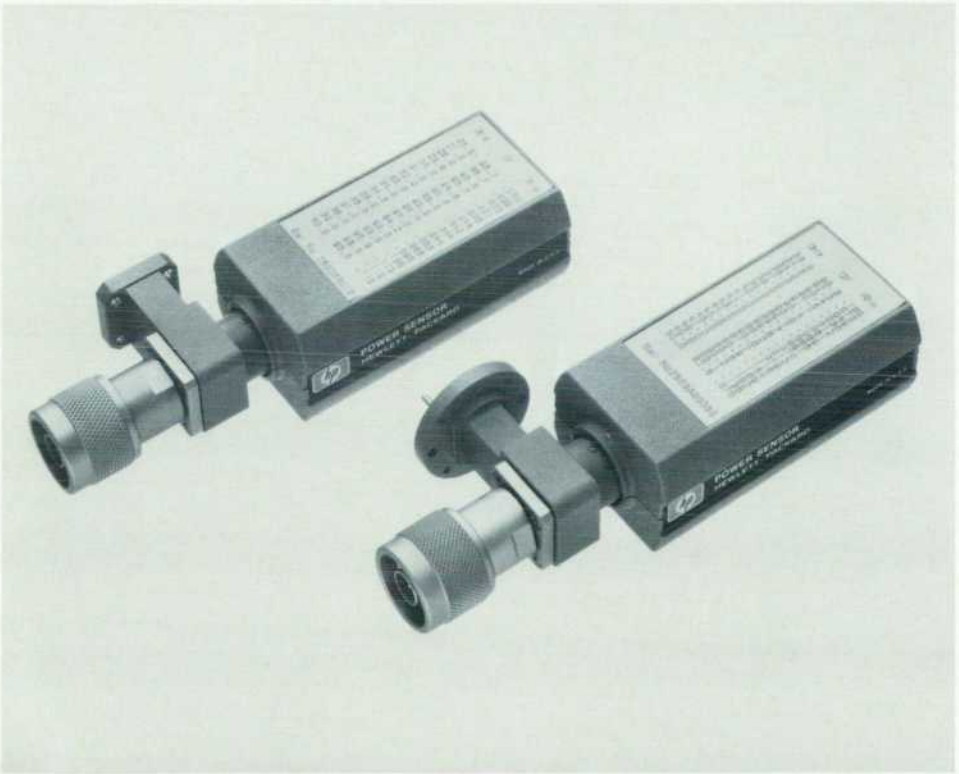


Fig. 1. *HP Q8486A and R8486A Power Sensors for measuring millimeter-wave signal power in frequency ranges from 33 to 50 GHz and 26.5 to 40 GHz, respectively.*

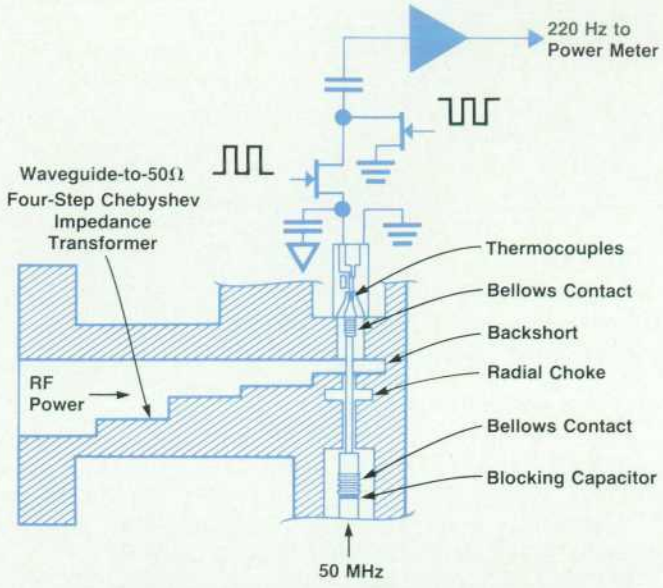


Fig. 2. *Cross section of waveguide-to-coaxial-line transition and associated circuitry.*

Waveguide-to-Coax Transition

The R-band transition shown in Fig. 2 was designed originally as an outgrowth of the lower-frequency X, P, and K281C waveguide-to-coax adapter designs. The Q-band transition was scaled from the P281C design since P band has a similar 2-to-1 waveguide width-to-height ratio.

A combination of empirical and computer-aided modeling was applied in such a manner that the design of the transition and four-step impedance transformer was not done in one step. The adapter design was broken into two parts. First, the coax-to-low-impedance (reduced height) ridge waveguide[1] transition was designed. The waveguide is broadly ridged to decrease impedance variations and increase the bandwidth. Second, sufficient transformer steps were then added to get the low reflection coefficient desired (approximately 0.01). The impedance of the waveguide was modeled using the relationship:

$$Z_o = Z_{o_\infty}/\sqrt{1+(\lambda/\lambda_{ca})^2} \text{ ohms}$$

where $Z_{o_\infty} = 377(\pi/2)(b/a)$ ohms (voltage-current definition) for rectangular waveguide at infinite frequency, a is the wide dimension of the waveguide, b is the narrow dimension, λ is the wavelength in free space at the frequency of interest, and λ_{ca} is the cutoff wavelength of the waveguide.

One goal of the coax-to-waveguide junction design was to arrive at a complex reflection coefficient that would look like the input impedance of the last step before the 50-ohm load of a four-step Chebyshev impedance transformer. After substantial empirical adjustment of the short position in the waveguide, ridge length, and other mechanical dimensions, the desired impedance was obtained. Finally the remaining three transformer steps were added. The combination was measured and the measurements were de-embedded using a computer program called Opnode to obtain the step discontinuity capacitances. The steps were then optimized for length and impedance with Opnode to obtain the desired 0.01 reflection coefficient, machined, and measured again.

50-MHz and Millimeter-Wave Diplexer

The need to calibrate the sensors at a low frequency requires a mechanical layout of the sensor front end that is different from most other waveguide sensors. The coax calibration port (see Fig. 3), beginning at the 50-MHz coaxial input at the bottom and moving upward, consists of a short length of 6-ohm line and a shorted radial transmission line choke[2] whose dimensions are chosen for a high series impedance at the center frequency of the waveguide band on the coax line. This is followed by a quarter-wavelength 6-ohm coax line that transforms the radial choke's high series impedance to a short at the entrance to the waveguide.

The input impedance Z_i at a radius r_b for a radial transmission line shorted at its outer radius r_a is given by:

$$Z_i = jZ_{o_i}\sin(\theta_{kr_b} - \theta_{kr_a})/\cos(\psi_{kr_b} - \theta_{kr_a})$$

where Z_{o_i} is the characteristic wave impedance for radial transmission lines:

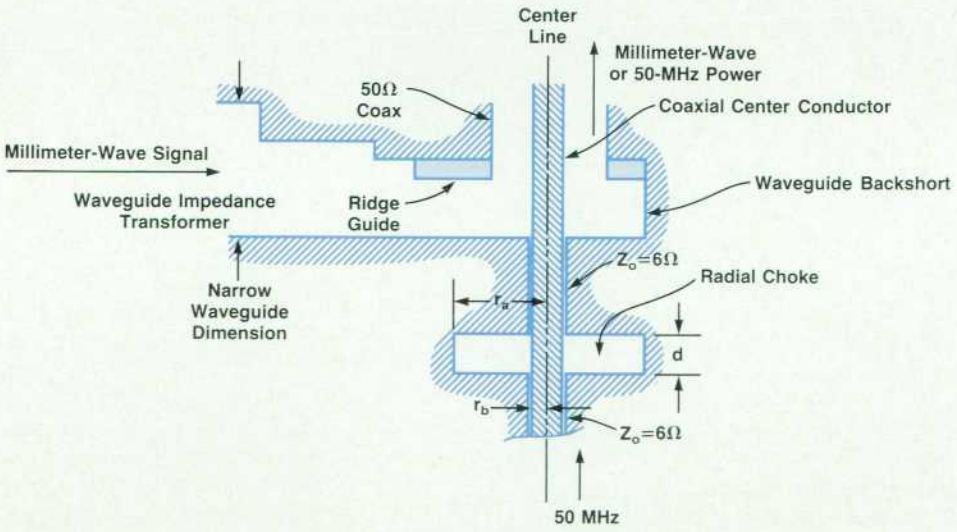


Fig. 3. *Detailed cross section of the junction of the millimeter waveguide, 6-ohm coaxial line with radial choke, and 50-ohm coaxial line to thermocouple.*

$$Z_{o_i} = 377G_0(kr_b)/G_1(kr_b)$$

and

$$\theta_x = \tan^{-1}[N_0(x)/J_0(x)]$$

$$\psi_x = \tan^{-1}[J_1(x)/-N_1(x)]$$

$$G_n(x) = \sqrt{(J_n^2(x) + N_n^2(x))}$$

$$k = 2\pi/\lambda$$

$J_n(x)$ is a Bessel function of the first kind and $N_n(x)$ is a Bessel function of the second kind.

Solving for Z_i resonant at the center of the frequency band, or Z_i equal to infinity by varying r_a, obtains:

$$\cos(\psi_{kr_b} - \theta_{kr_a}) = 0$$

The total impedance at any frequency can be found by taking k at the frequency of interest and solving for the radial choke's input impedance. Then:

$$Z_{total} = dZ_i/2\pi r_b$$

where d is the width of the choke (see Fig. 3).

Other methods of trying to place an electrical high-frequency short at the waveguide were tried, such as a low-high-low-impedance dumbell choke filter, but the radial choke filter reflects an impedance closer to zero ohms at zero degrees at the waveguide entrance. For calibration purposes, a 50-MHz signal can be fed through the choke assembly, across the waveguide with minimal effect, through the coaxial line, and into the thermocouple.

Constant-Impedance Taper

The thermocouple is mounted on a suspended substrate transmission line whose inner and outer conductors are tapered from the wide center conductor at the coaxial end of the transmission line down to a narrow center conductor at the thermocouple end of the transmission line. Usually this is done by using straight tapers for the inner and outer conductors. Straight tapers are a compromise and a better method that maintains a constant impedance as the conductors move closer together is desired.

The suspended substrate is mounted in a circular housing, and for wide outer conductor spacings approaching the outer housing, closed-form equations do not exist for calculating the spacing of the outer-conductor-to-inner-conductor width versus impedance. The required spacing for the wide center conductor cases was determined using a finite-difference program called FCAP.[3] For the narrower center conductor region where the effect of the outer housing is negligible, analysis equations[4] were used to determine the correct spacing because they are faster and more accurate than the finite-difference approach. The results were then curve-fitted into a fourth-order polynomial describing the outer conductor width versus the center conductor width for a constant impedance of 50 ohms.

An arbitrary shape called an ogee curve was selected for the center conductor. An ogee curve is just two opposing arcs that meet at a point where their slopes are identical, giving a smooth continuous curve from the wide to the narrow end of the center conductor length. The beginning and ending center conductor widths and the transition length for the desired center conductor were selected. Then an outer conductor position was located that gave an im-

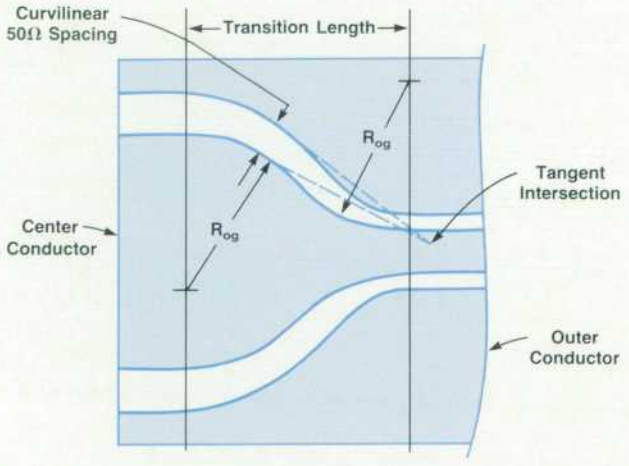


Fig. 4. *Constant-impedance taper from coaxial line to thermocouple connection. (R_{og} is radius of ogee curve.)*

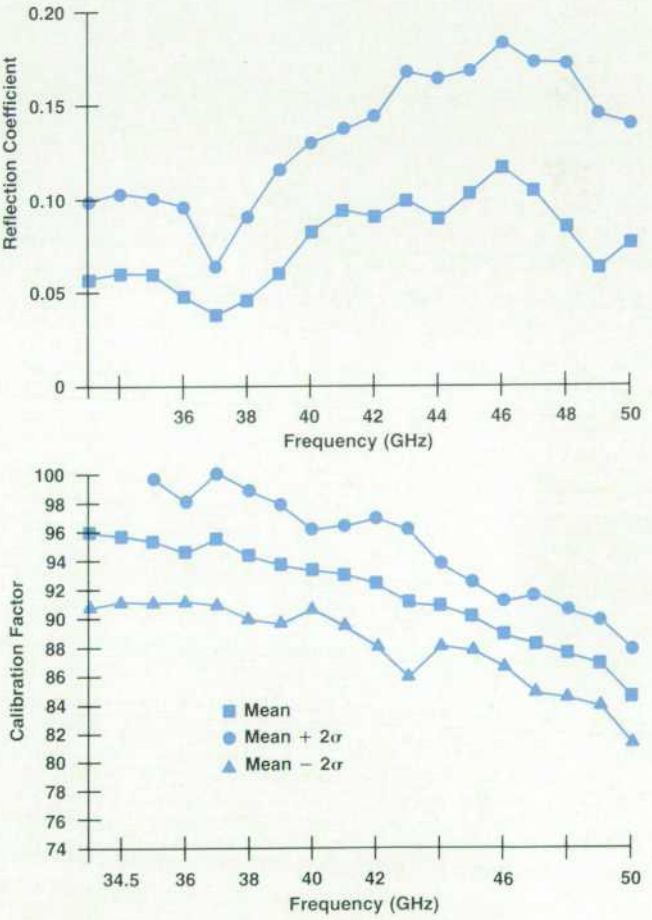


Fig. 5. *Average reflection coefficient and calibration factor versus frequency for a lot of 21 HP Q8486A Power Sensors.*

pedance of 50 ohms for many different closely spaced positions down the length of the center conductor (see Fig. 4).

At each center conductor position, outer conductor positions were located on the arc length of a circle whose center was the intersection of the two tangents to the center and outer conductors, and whose arc distance between the outer and center conductors was equal to the distance the conductors would be spaced if they formed a parallel transmission line. The outer conductor positions must be iterated several times, because the outer tangent and hence the center of the circular arc shifts each time the arc distance is iterated to the 50-ohm distance. The speed of convergence is fast, however, and ten iterations results in a distance giving an impedance within 0.1 ohm of the calculated 50-ohm value.

This approximation positions the outer conductor at the correct spot so that the flux path length between the center and outer conductors is the correct length for 50 ohms as it strikes the conductors normal to the tangent. Therefore, the approximation satisfies both the boundary condition that flux lines terminate normal to the conductors and the 50-ohm constant impedance restraint. This assumes that the field is a curvilinear field between the two conductors. The slight error caused by the flux that terminates on the outer housing instead of the planar conductor is not significant.

Thermocouple

The termination for the transmission line is two 100-ohm tantalum-nitride resistors connected in parallel. They are deposited on a silicon substrate that has a very thin web in the center, which is where one end of each resistor makes contact with a heavily doped diffused conductor. Because of the poor thermal conduction properties of the thin web, the incident RF energy dissipated in the web causes the resistor-to-doped-conductor contacts to run hot. The other end of the diffused conductor makes contact with gold beam leads in a thick area of the silicon that conducts heat more readily and therefore does not get as hot as the power-absorbing resistor termination. The difference in temperature between the junctions leads to a thermally generated voltage difference when the resistor termination is dissipating energy. The outputs of the two 100-ohm thermocouple pairs are summed. Ambient changes in temperature generate only slight thermal drift because both junctions are subject to the same ambient temperature. A more complete description of the thermocouple can be found in reference 5.

The thermocouple with its beam leads is slightly inductive and a short low-impedance line reduces the reflection coefficient to less than 0.1 at 50 GHz.

Performance

The reflection coefficient and calibration factor performance of the power mounts can be seen in Fig. 5, which plots the mean and the mean plus two standard deviations for a lot of HP Q8486As.

Acknowledgments

Ron Pratt conceived the idea for the calibration input port. Tony Schaffer was the designer of the excellent coax-to-waveguide transitions. Connie Chaing solved and wrote the radial transmission line programs and made a scale model to try the idea while at HP as a summer student.

References

1. S.B. Cohn, "Properties of Ridge Wave Guide," *Proceedings of the IRE*, August 1947, pp. 783-788.

2. S. Ramo, J.R. Whinnery, and T. Van Duzer, *Fields and Waves in Communication Electronics*, Second edition, John Wiley & Sons, 1984, pp. 460 and 463.

3. S.Y. Oh, "MOS Device and Process Design Using Computer Simulations," *Hewlett-Packard Journal*, Vol. 33, no. 10, October 1982.

4. K.C. Gupta, R. Garg, and I.J. Bahl, *Microstrip Lines and Slotlines*, Artech House Inc., Dedham MA, 1979, pp. 275-280.

5. W. Jackson, "A Thin-Film/Semiconductor Thermocouple for Microwave Power Measurements," *Hewlett-Packard Journal*, Vol. 26, no. 1, September 1974.

Adapting UNIX Logon Mechanisms to Automation Applications

Although originally intended for software development and document preparation, the utilities provided by the UNIX operating system can be adapted in various ways for use by novice operators in an automated environment.

by Marvin L. Watkins

THE CURRENT TRENDS toward computer-enriched environments and standardized computer systems invite the use of the UNIX® operating system in the service, office, and manufacturing automation domains. Applications such as telemarketing, data base management, and integrated manufacturing or CIM benefit greatly when more than one customer, user, or machine can be serviced simultaneously. The dedicated server and controller applications required in these domains are well-suited to the UNIX system's multitasking capability and tool-based architecture.

The UNIX system originated as an interactive, timesharing operating system for minicomputers. Originally oriented toward multiuser software development, it is not surprising that the UNIX boot/logon mechanisms do not provide exactly those services that dedicated applications might want. For example, typical dedicated application requirements include that a novice user with little computer knowledge be able to bring up a controller (i.e., a dedicated computer) from a power-off condition and access its control functions easily, while the general UNIX power-up and user logon facilities only allow a knowledgeable computer user to bring up the computer from a power-off condition and access a UNIX shell (i.e., a command line interpreter).

Special UNIX boot and logon programs have been created to handle the problems associated with dial-in lines, modem control, security, etc. The boot programs provide services that:

- Initialize system resources
- Create a multiuser environment.

The logon programs provide services that:

- Match an incoming data stream's unknown parameters (e.g., baud rate, parity, and stop bits) dynamically to establish a communications link
- Offer some, but not too much security against unauthorized access
- Prepare a customized personal working environment
- Provide common, system-wide logon services (e.g., maintenance and downtime notices, mail and news received notices, etc.)
- Finish in a working state in which any of an infinite

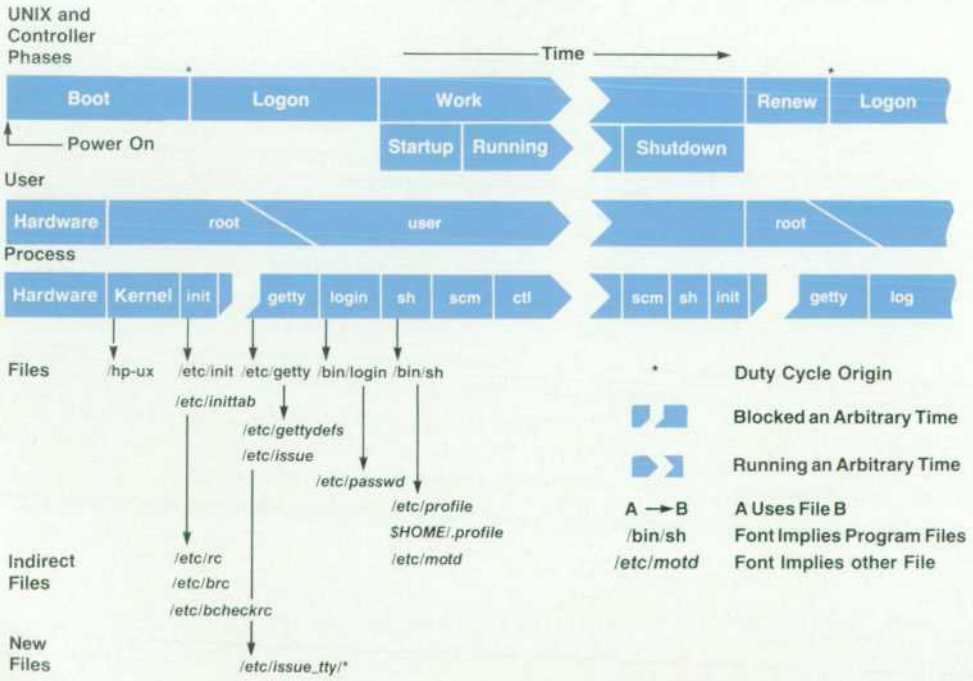


Fig. 1. *UNIX boot/logon mechanisms.*

variety of commands can be issued.

These services can be compared with those needed by a typical controller:

- Initialization of the system at power-up
- Use of dedicated, hard-wired terminals with fixed, well-known communications protocols when a user wants access to the system
- Some protection against accidental damage caused by naive access
- Common, system-wide logon notification services
- Simple, uniform, consistent interfaces for all users
- A small, fixed set of monitor and control operations.

In this article we present some adaptations to the UNIX operating system that make it better suited as a platform for dedicated automation and server applications. Our goals are to present some adaptations to the boot/logon mechanisms, demystify the UNIX system further,[1] and promote its tool-oriented philosophy.[2] Much of the narrative and all the examples in this article are taken from the manufacturing domain: CIM, data acquisition, and process control and monitoring. Despite this emphasis on manufacturing, the adaptations presented, and especially the principles and ideas underlying them, should be applicable to any computer running under the UNIX system and dedicated to a specific application.

In the next section, the UNIX boot phase and duty cycle are reviewed. The details are based on HP-UX Multiuser Revision 5.1.* The discussion's objective is to describe how events are related to one another.[3,4] References 5 through 9 should be consulted for exact details of the programs and files mentioned. Some terminology useful for understanding UNIX processes is also developed.

The potential adaptations are then presented in fairly general terms. Some automation design considerations follow. Here the focus is on adapting the UNIX boot/logon mechanisms for a system dedicated to a controller or server application (hereafter, referred to simply as a controller). Specific controller adaptation examples are presented and some simple observations close the article.

UNIX Boot/Logon Mechanisms

It is convenient to divide a typical system's operation into a boot phase and a number of duty cycle phases as shown in Fig. 1. These phases are defined by the nature of the processes executing within them. Table I provides additional details about these mechanisms. This article does not cover all possible boot/logon organizations. Many capabilities of the UNIX system are omitted to keep the discussion within reasonable bounds.

Two operating phases can be identified:

- A boot phase extends from power-on until the kernel is initialized and fully operational. The boot phase exhibits a single thread of control.
- A duty cycle phase begins after the kernel is fully operational. The duty cycle phase is characterized by multiple threads of control. Each thread courses through the following cycle:
  - A logon period from a user's first keystoke until the system commands for a special application program are accessible.
  - A work period after logon until the user exits the system.
  - A renew period while the kernel cleans up after the exiting user, reclaims system resources, and creates a new control thread to replace the expiring one.

Table I

Execution Initiation Points

Exec()'ing Process		Exec()'d Processes		
Process	File	Mode	Group Leader	Arguments Permitted
init	inittab	W,C*	Y	Y
init	rc	W	N	Y
getty	---	S	Y	---
login	passwd	S	Y	N
sh	profile	C	N	Y
sh	.profile	C	N	Y
sh	---	W,C**	N	Y

Key

S	Sequential execution
C	Concurrent execution
W	Suspended execution
Y	Yes
N	No

Notes

* init usually suspends itself while executing initialization programs. getty programs execute concurrently with init.

** The shell suspends itself while executing foreground keyboard commands, but executes concurrently for background commands.

Boot. The boot phase is composed of hardware, kernel, and init operations. The boot phase begins with the application of power. A ROM catches the power-up interrupt and begins executing hardware configuration and self-test programs. When these complete, a special bootstrap program is loaded from disc memory. The bootstrap program loads the kernel /hp_ux from the file system. The kernel configures and initializes itself, handcrafts the special process init (among others) and executes it (them). init reads through the file inittab line by line, performing the actions indicated therein. These actions usually fall into two categories: executing the contents of system initialization scripts such as rc, and executing instances of the program getty for each active terminal. When init finishes processing the file, it blocks and waits for a signal to reread it.

init performs the transition from a single-user, single-control-thread operation to a multiuser, multiple-control-thread operation. Each instance of getty spawned by init represents a unique, independent, concurrent control thread.

Logon. The logon phase is composed of getty, login, and some sh operations. getty processes match their invocation

*HP-UX is Hewlett-Packard's implementation of the UNIX operating system. It is primarily composed of AT&T's UNIX System V. Hewlett-Packard has added real-time and other custom extensions, and some features from 4.1 and 4.2 BSD. UNIX versions developed by the University of California at Berkeley. UNIX is a registered trademark of AT&T in the U.S.A. and other countries.

arguments from inittab with a line in gettydefs to select the proper port conditioning parameters and logon prompt for their respective ports. getty conditions a port, writes the contents of issue and the logon prompt to the port, and then reads the port's input buffer. The read operation causes getty to block until input becomes available. getty interprets input as an account name. Hence, getty blocks on an active terminal's port waiting for someone to begin logging on by typing an account name.

A getty process is awakened when the kernel places data into its port's input buffer. Normally, this data is an account name. getty runs login with this name as its principal argument. login searches the passwd file for the account name. If login finds the name, the line on which the name appears is interpreted. passwd contains fields for account name, password, user and group IDs, home directory, and shell. If the password field is empty, no password is requested from the prospective user, who then is automatically promoted to user status. If the password field is not empty, the prospective user must successfully enter the password before being promoted to user status. When login promotes a candidate to user status, the user is assigned the user and group IDs appearing on the account name line. These IDs determine the user's file access and system resource permissions while the user is logged on. login then changes the working directory to the account's home directory and runs the specified shell program.

Normally, the program executed is a shell such as sh. sh executes the instructions contained in the profile file on behalf of the newly logged-on user. The actions performed by interpreting profile vary from system to system. Typical actions are to notify the user of mail or the arrival of new news, set up common environment variables (e.g., PATH), and set shell trap handling for keyboard interrupts.

sh then searches for a file named .profile in the working directory (i.e., the account's home directory) and if the file is found, executes the instructions contained therein. In general, interpreting the contents of .profile creates a customized environment catering to a user's preferences.

Work. The work phase consists of most of sh's operations. After interpreting .profile, sh issues a prompt to its associated port and waits for input. Usually, input consists of commands and programs the user wants performed. A user disconnects from the system by terminating the logon shell.

Renew. The renew phase is composed of init's operations. init awakens when sh exits and creates a new getty process.

This completes one iteration of a typical UNIX system's duty cycle. The next events that occur are the beginning events of a new cycle—getty resets its port, outputs issue, and waits for a new candidate user to try to log on.

Duty Cycle. The UNIX system controls concurrent duty cycles with a convention called process group leadership. Processes spawned by init are process group leaders (other processes can also become process group leaders). init's process group leaders signal it when they terminate. (They also have other significant properties that distinguish them from common processes.) init rereads inittab when signaled about the death (i.e., termination) of one of its process group leaders. If the line in inittab associated with the process group leader specifies respawning, a new process is started to replace it. Usually getty processes are the only respawned processes in a UNIX system.

In the phases above, getty was spawned by init and so was a process group leader. login assumed process group leadership from getty and passed this on to the logon shell sh. Since the logon shell inherited getty's process group leadership, init was signaled that one of its process group leaders died when it exited. Thus, the UNIX system duty cycle:

*Logon-Work-Renew-*Logon-...

is implemented by:

*getty-login-sh-init-*getty-...

where the elements between occurrences of the * form one duty cycle.

Execution Initiation Points. The boot and duty cycle phases define seven standard points at which programs are or can be executed. These points, listed in Table I, define the universe of possible boot/logon adaptations. Five of the points have an associated ASCII file. Thus, special process-

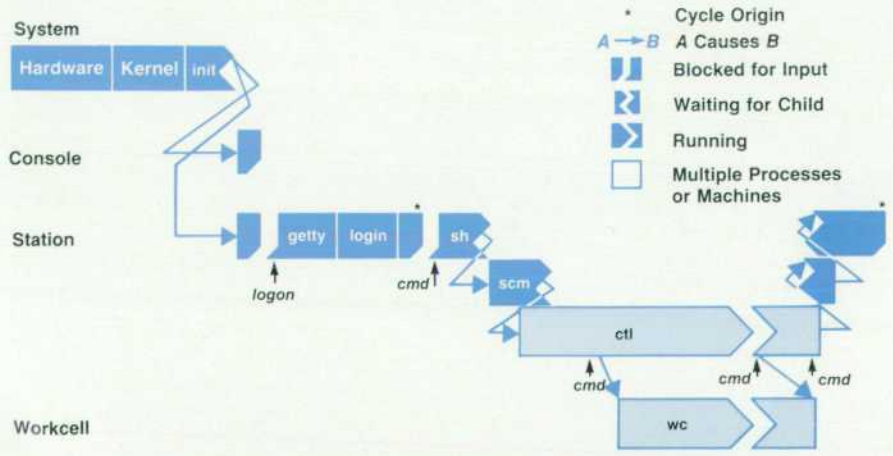


Fig. 2. *Conventional shell startup sequence. The long axis orders events in time. Distinct control points are listed vertically along the left side. The figure shows starting up, running, and shutting down a controller and graphically portrays these events' execution modes. Sequential execution is shown by succession within a row. Concurrent execution is shown by multiple rows. Suspended execution is shown by multiple rows with the suspended parent exhibiting a break where it is waiting. Finally, while the boot phase and init's processing are shown for context, complete details for them are omitted. Figs. 3 through 6, associated with later examples, are interpreted similarly.*

ing can be introduced at these points by merely editing those files. In general, the earlier the point, the more encompassing the effects. Table I is ordered with the earlier points near the top.

Execution Modes. UNIX kernel primitives fork(), exec(), and wait()[7] can be combined in different ways to provide execution modes with different properties. Three forms that play an important role are sequential, concurrent, and suspended execution modes.

Sequential execution occurs when a process executing on behalf of a program simply executes some program file using exec(). This causes the original process to start executing on behalf of the new program. Concurrent execution occurs when a process executing on behalf of a program first performs a fork() and the newly created process then executes some program file using exec(). This creates a new process, usually referred to as a child process to distinguish it from the original parent process, that executes on behalf of the new program. Suspended execution occurs when a process executing on behalf of a program first performs a fork(), the newly created child executes (using exec()) some program file while the parent waits (using wait()) for the child to complete execution. The child process executes on behalf of the new program. The parent process, still executing on behalf of the original program, stops execution until the child process completes. When the child terminates, the parent resumes running.

UNIX Boot/Logon Adaptations

There are, of course, many ways to adapt UNIX boot/logon mechanisms. The purpose of this section is to suggest some of the kinds of things that can be done and to discuss some of the problems that will be encountered if they are.

At least four kinds of adaptations are practical:

- Edit. UNIX boot/logon initialization and configuration files are simple ASCII files containing tables or scripts. The actions they govern can be adjusted with simple editing. Many of the ideas presented here introduce new functionality into the boot/logon mechanisms, but only involve editing existing ASCII files and adopting new conventions for their use. For example, by interpreting the contents of issue and passwd differently, menus can be created.
- Skip. Some boot/logon programs can be skipped entirely. Two possibilities in this category are that init could execute a program directly and login could execute a command other than a shell.
- Modify. The boot/logon programs themselves can be modified if source code is available. Two possibilities in this category are to enhance getty to display files assigned to each terminal and modify login to treat input as an action qualifier rather than as a password.
- Replace. In general, source code is not available. In this case, existing programs can be replaced with new ones. The above modifications can be accomplished by creating new, simpler programs to replace getty or login. Both the new and old getty versions can easily coexist in the same system by editing inittab accordingly. login is more problematic. Either the getty and login versions must be matched (e.g., xgetty exec()s xlogin) or only one login can exist.

In theory, new programs can be inserted between existing ones. In practice, these would be dubious adaptations. Any program executing before getty would have to condition the port before writing to it. Effectively, getty would be replaced. getty's connection to login is hardcoded. login could execute a program that executes sh, but since the shell is the most versatile program in the UNIX toolset, this would make little sense.

init Adaptations. init interprets the contents of inittab. Hence, inittab can be edited to initiate programs directly.

getty Adaptations. getty writes the contents of issue to its associated port before reading its buffer. issue normally contains some simple, constant message. However, it could contain text describing available commands and function key or touchscreen programming sequences, that is, a menu.

Since issue is sent to all active terminals, all of them would have the same menu. If access to source code is possible, a simple modification to getty will fix this situation. getty knows its assigned port's name, for example, /dev/tty17. Hence, getty can display files named, say /etc/issue_tty/tty17 on just port /dev/tty17 alone. Such a mechanism provides a simple, effective correspondence from specific terminals to special command sets available at them.

The text of the logon prompt is defined in gettydefs. Usually this prompt is login:. It can be changed to something like Please select a function key. The gettydefs logon prompt is output after the contents of issue.

login and passwd Adaptations. The passwd account field must match the first thing that a candidate user inputs. Usually, it identifies the user. It can be adapted, however,

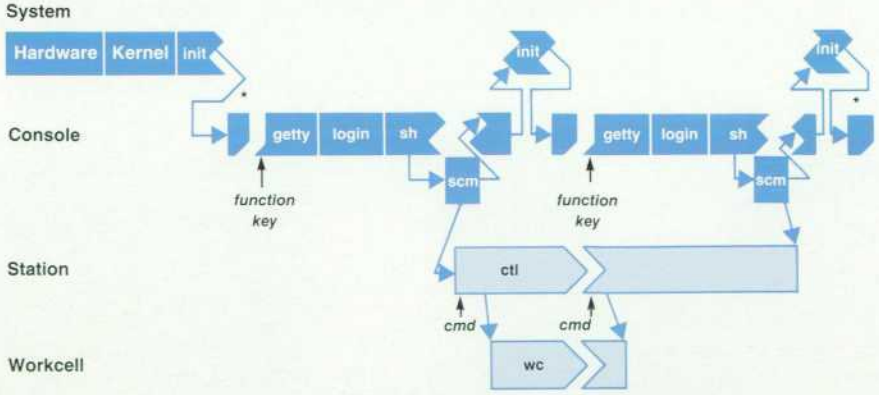


Fig. 3. *Console menu startup sequence.*

to identify an action the user wants performed or an object to act upon. A command can be given an account name (e.g., date). Then the command can be executed by simply typing the account's name in response to the logon prompt. The shell field should contain the command's pathname.

An object can be given an account name (e.g., workcell_X). A user's input could result in, say, a menu of operations on the object. Classes of users such as engineers, supervisors, technicians, and workers can be given account names. Entering the account name then could result in a menu of commands special to the class.

Usually, a password authenticates a user's right to access an account. In general, UNIX security mechanisms can be used for other purposes. That is, conventions can be established for interpreting passwd entries in interesting and unconventional ways. If no password is required, entering as little as one character can achieve logon and run one or more programs. When passwords are not used, account names can map directly to actions or objects.

An application's name could be repeated in passwd as many times as there are operations. Each operation then becomes a password. Typing an application followed by a "secret" request for an operation could be effectively used to perform that action. The shell field must be filled accordingly. If access to source code is possible, modifying login to echo its input and removing encryption would produce a cleaner design.

The home directory field of the passwd file contains the name of the working directory the user is placed in after logon. This can be any directory in the file system. Usually it contains a user's private files. However, it could be associated with a special program.

The shell field usually contains the pathname of a command line interpreter, or shell. However, it can contain any command's pathname. For example, workcellX, date, or some other program can be executed by login from passwd. Note that shell field commands cannot have arguments and that scripts are automatically interpreted with sh. login prepends a - to argument zero (e.g., argv[0] = -sh) to alert the shell that it is a login shell. This can break some commands.

The shell field has a special feature to provide very secure environments. If * is specified in an account's shell field, the specified home directory is used as the root of a new file system. That is, only commands and files appearing below the home directory are accessible after logging on. The login process is repeated again in this new directory. This can be exploited to produce a two-step access design. An issue menu could offer special selections for a supervisor, engineer, etc. in addition to controller operations. Selecting one of these classes, possibly with a class password, forces a prospective user to supply the user's personal account name and password. Since the second logon does not cycle back through getty, a menu cannot be provided for it without modifying login.

sh and .profile Adaptations. The login shell sh normally performs simple system-wide actions that are specified in profile. A menu could be displayed and terminal function keys programmed from profile. If a home directory contains a .profile script and the shell field contains /bin/sh, the login shell will interpret the contents of .profile before prompting the user for a command (csh and .login would work similarly). Thus, .profile can be used to run programs before the user sees a shell or to provide a restricted environment for the user.

The entire set of capabilities for an account can be provided within a .profile program. Construct .profile to contain a loop that displays a menu, programs the function keys, and prompts for an input. The user should only be able to perform menu functions. When a user selects some item, subsequent code within .profile parses the selection and takes appropriate action. Upon completion the menu would be redisplayed.

Environment Variable Adaptations. Several environment variables are automatically set by the logon mechanisms. PATH is an example. In particular, LOGNAME is set to the account name when a user logs on. Thus, it can be tested to select some action appropriate to a given logon account.

Environment variables also can be given in addition to an account name in response to the logon prompt. They must be typed after the account name. Environment variables defined in this way are passed through to the shell where they can be tested and some appropriate action taken.

Creating Menus. Commands and/or programs (e.g., a controller) can be executed without requiring a user to log on. Menus can be used to list and designate function keys to execute commands or run programs with as little effort as pressing a single key. issue is used as a menu display and function key programming mechanism. The menu's text is preceded by screen-blanking and cursor-homing sequences and followed by function key programming sequences. Each programming sequence has a mnemonic label to

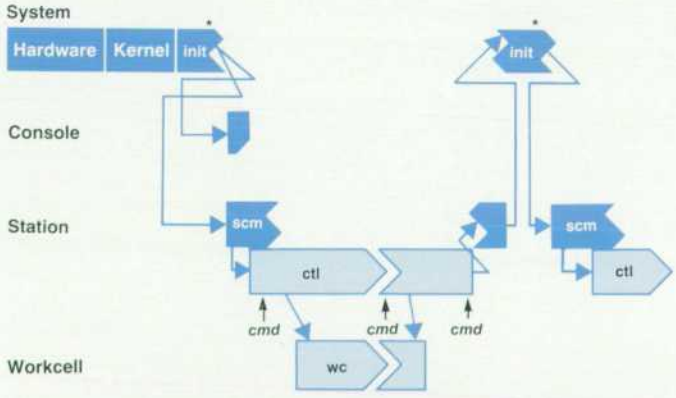


Fig. 4. init *sequential startup sequence.*

suggest its use and transmits a string that matches an account name in passwd.

passwd contains an account for every operation displayed on the issue menu. Accounts have the same user and group IDs, the same home directory, and the shell /bin/sh. The password fields are empty.

The home directory's .profile file contains code to select appropriate processing based on the value of the environment variable LOGNAME. LOGNAME is set to an account's name by login. Thus, processing corresponds to one of the menu's labels as indicated by the map:

label ▶ string ▶ account name ▶ fixed user ID ▶ user's home directory ▶ sh ▶ .profile ▶ $LOGNAME ▶ processing

Design Considerations

In this section, a series of design considerations and issues concerning controllers are raised. Possible resolutions using the adaptations outlined earlier are presented in the following examples, the related figures, and Table II, which relates design choices to system adaptations for each example. Thus, the examples do not specifically mention all of their respective properties.

Program versus Suite. Perhaps the most fundamental controller design consideration concerns its basic structure. A controller can be a single monolithic program or a program suite (or even something else). This choice determines many of a controller's important features.

In all but the simplest of systems, asynchronous, event-driven I/O to and from multiple machines or users must be handled. This suggests that a program suite is a good architecture for a controller. Events from several sources can be handled individually and at their own rate, each by a separate process. Such processes can and must be organized to work cooperatively.

In this case, controller startup involves executing many programs. Since they must work cooperatively, communications links must be configured and established.[1] In general, such services are best provided by a special program. Thus, this paper assumes (without loss of generality) that a controller ctl is composed of many concurrently running programs and that it is initiated by a startup and configuration module scm.

- **Example: Conventional Shell Startup.** This example uses the UNIX system's basic program initiation sequence. It is presented in detail to provide a baseline context with which the other examples can be compared. Fig. 2 shows this example's operational phases. The narrative details presented here should be compared with the figure's graphic details. Table II relates the example's design choices to the logon adaptations discussed earlier. No adaptations are required for this example since it is strictly conventional use.

  The conventional shell startup sequence begins with a worker logging onto an active terminal. After completing logging on, the shell blocks, waiting until the worker types some command.

  The worker types the startup and configuration module's name (scm) and arguments. The shell executes scm and suspends itself. scm concurrently executes all but one of the controller's (ctl) programs. scm executes the suite's last program and suspends itself waiting for the program to terminate.

  After the controller's operator interface comes up, the

Table II
Choices and Adaptations

Design Consideration	Conventional Shell Startup	Console Menu Startup	init Sequential Startup	Single-Program passwd Startup	Preload Station Startup
Program or Suite	Suite	Suite	Suite	Program	Suite
All-at-Once or Preload	All-at-Once	All-at-Once	All-at-Once	Not Applicable	Preload
Foreground or Background or init	Foreground	Background	init	Not Applicable	Foreground
Wait or Execute	Wait	Not Applicable	Wait	Not Applicable	Execute
One or Several Controllers	Don't Care	Several	Don't Care	One	One
Command Line or Menu	Command Line	Menu	Not Applicable	Command Line	Menu
Active or Inactive Terminal	Active	Inactive	Inactive	Active	Active
Dedicated or Shared Terminal	Don't Care	Dedicated	Dedicated	Dedicated	Don't Care
One-at-a-Time or Many	Don't Care	One-at-a-Time	One-at-a-Time	One-at-a-Time	One-at-a-Time
Operator Startup or Automatic	Operator	Operator	Automatic	Automatic	Automatic
Same or Different Terminal	Same	Different	Not Applicable	Not Applicable	Not Applicable
Password or None	Don't Care	None	Not Applicable	Password	None
File	**Contents**	**Contents**	**Contents**	**Contents**	**Contents**
inittab	Usual	Terminals Off	Terminals either not listed or off, scm listed respawn	Usual	Usual
rc	Usual	Usual	Usual	Usual	Spawns scm at boot for preloaded programs
gettydefs	Usual	Special Prompt	Not Applicable	Usual	Special Prompt
issue	Usual	Workcell banner and function key programming sequences	Not Applicable	Usual	Common Menu
issue_tty/*	---	---	---	---	Workcell banner and function key programming sequences
passwd Account	Usual	Operations for startup, reset, shutdown for each controller	Not Applicable	Controller Program	Operations for startup, reset, shutdown for each controller
Password	Usual	None	Not Applicable	Password	None
Home	Usual	Application's	Not Applicable	Application's	Application's
Shell	Usual	/bin/sh	Not Applicable	Controller Program	/bin/sh
profile	Usual	Usual	Not Applicable	Not Applicable	Usual
.profile	Usual	Like menu in "creating menus" subsection	Not Applicable	Not Applicable	Uses LOGNAME to determine workcell and function to perform. Executes scm, scm sequentially executes flag program

worker issues a command to start up the machinery. Manufacturing continues until a command is issued to stop the machinery. Then, another command is issued to shut down the controller.

The process on which scm is waiting terminates with controller shutdown. scm awakens and resumes running. It performs any necessary cleanup and then exits. The shell awakens, prompts the user for input, and then blocks until a new command is entered. At this point the cycle can begin anew or other work can be done.

Foreground versus Background versus init. A controller can run as a foreground or background program suite or init can run the suite directly. Foreground and init suites must arrange to prevent scm's process from exiting. Otherwise, in the foreground suite, two processes (ctl's user interface and sh) would end up taking input from the same terminal at the same time, which usually causes both processes to fail. In the init suite, scm's process must be prevented from exiting because otherwise, a process group leader would die, causing init to start up another copy of the suite.

In the previous example, the controller ran as a foreground process initiated by typing a command to the shell. The next example is of a background process initiated via a menu.

- **Example: Console Menu Startup.** Fig. 3 shows this example's operational phases. Table II lists this example's choices and adaptations.

  This sequence of events begins with a worker selecting the controller startup operation function key at the console. Pressing the key results, by the map in the "Creating Menus" subsection above, in scm starting a controller suite, ctl, in background (with respect to the console), and then exiting. Manufacturing is controlled at the controller's terminal until a command is issued to stop the machinery. Then the worker returns to the console and presses the function key that shuts down the controller. This results in scm terminating the controller suite.

Menus consist of mnemonically labeled function keys. They are updated by .profile as follows. If the label selected is associated with starting up workcell X, then the start X label is removed from the function keys and shutdown X and reset X labels are added. Opposite actions are performed for the shutdown operation. Resetting does not require updating the menu.

On a more technical note, scm or .profile must arrange to catch console keyboard signals, including hangup, so that ctl's processes cannot be terminated incorrectly by other console activity.

Wait versus Execute. scm's process can be kept from exiting in either of two ways: suspended execution, where it waits on one or more of the controller's programs, or sequential execution, where it can select one program to assume its process ID.

In the previous examples, scm ran as a foreground or background process. In the first example it executed concurrently and in the second example it used suspended execution. The next example is of an init process that runs sequentially.

- **Example: init Sequential Startup.** In this example (see Fig. 4 and Table II), assume scm is listed in inittab as a respawn process and that initdefault is its init_state. Then the boot phase spawns scm. In this case, scm is effectively a duty cycle control thread.

  scm concurrently executes ctl's programs, except for a special flag program, which it sequentially executes. When the controller shuts down, the flag process that assumed scm's process exits. init detects this and spawns another instance of scm.

Fig. 4 may be somewhat misleading in suggesting that this design is simple. Major problems stem from the fact that root is the user while init runs. Permissions and/or user and group IDs must be carefully adjusted to avoid compromising the system seriously. Since the logon processes are skipped, environment variables such as PATH must be dealt with explicitly. In effect, this design requires the controller to run continuously since shutting down automatically results in restarting scm.

If an init_state is dedicated to the controller, several other precautions must be taken. init's state should not be changed without first killing the controller's processes. Otherwise, at a state change init's process group leader (in this case the first process to open the station's port) will be killed automatically. Other processes are unaffected. Upon return to the controller's state a second copy of the controller will be created. Terminal (i.e., /dev/tty*) permissions are manipulated by getty as a security measure. Care must be taken to reset these permissions upon return to the controller's init_state.

Finally, synchronizing I/O is questionable. stdin, stdout, stderr, and /dev/tty are not defined at startup (no getty and no sh). If shutdown is an initstate change, then the controller's operator interface and getty can end up simultaneously accessing the station's terminal.

All-at-Once versus Preload. Programs for a multiprogram controller can be loaded (i.e., initiated or executed) all at

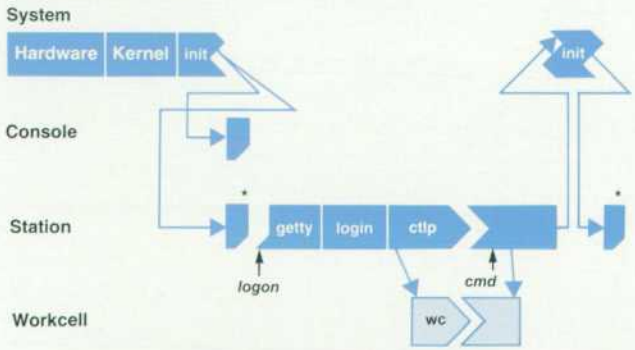


Fig. 5. *Single-program* passwd *startup sequence.*

once at controller startup or most of them can be preloaded. In the second case, startup involves only a single program. In general, the first approach is simpler, the second is faster.

One versus Several Controllers. Several controllers can run on a computer simultaneously. If so, user interface issues become much more complex. For example, there can be one terminal per controller, more terminals than controllers, or more controllers than terminals. In addition, scm must provide for starting up several controllers, each consisting of many programs.

The number of controllers per computer affects what users can do at a given terminal. A controller can have a specific terminal dedicated to it or any controller can be run from any terminal. More than one controller can be run simultaneously from a single terminal. For example, one screen could control several controllers or each controller could appear in its own window on the screen.

Active versus Inactive. A controller's operator station can be associated with an active terminal (i.e., one that also can be used for logging on) or an inactive one (i.e., a terminal that is activated by a program). In general, terminal-per-controller, dedicated, and/or inactive designs limit a system's flexibility, but simplify its design and construction.

Command Line versus Menu. An operator can be required to type command names and arguments or can be provided menus, function keys, a mouse, and/or a touch screen.

Operator versus Automatic Startup. An operator can be required to start up a controller and any associated machinery interactively, or the system can hide these steps by doing them automatically. Various strategies can hide some or all of the logon and startup phases. Controller startup and machinery startup can be at the same terminal or at different terminals. Placing both at the same terminal tends to unify these control levels. Conversely, using different terminals tends to separate the control classes.

Hidden phases and/or unified control levels conceal operational details. This concealment may simplify a worker's job when events are well-behaved. However, when things go wrong, concealed operational details make problem solving much more difficult.

Password Security versus None. Controller security can range from none at all to providing multiple-password, brick-wall regimes. A user can gain access to a controller's functions by simply walking up to it, typing only a logon account name, typing a logon account name and a password, or traversing several levels of account names and passwords.

Simplicity versus Complexity. The last two examples contrast the simplest and the most complex of the reasonable approaches for using UNIX mechanisms for controller applications.

- **Example: Single-Program passwd Startup.** The noteworthy features of this example (see Fig. 5 and Table II) are that the controller is a single monolithic program, ctlp, and that it is listed in the passwd file and executed by login instead of a shell.

In large part, this example's apparent simplicity is derived from these choices. Complexity that is manifest in other examples is transferred into ctlp's internal design. The system programmer does the work instead of building on UNIX tools.

- **Example: Preload Station Startup.** In this last example (see Fig. 6 and Table II), all programs except a special flag program are concurrently executed via scm from an entry in rc. A modified getty process displays individualized controller menus on each terminal that serves as a control station.

  In this example, scm sequentially executes the flag program. The flag process notifies the controller when it begins. The controller outputs the control screen and starts up the machinery without worker action. Manufacturing begins and continues until a shutdown is issued. The controller processes, including the flag process, are notified of the shutdown. The flag process exits, causing sh to exit. init is signaled and respawns getty. getty resets its port and outputs the original menu.

Conclusions

None of the adaptations discussed above is best. Most of them trade complexity in adapting the UNIX system's boot/logon mechanisms for simplicity in the controller. The advantages of using UNIX facilities lies in their combinatorial versatility and power and their pre-existence. The disadvantage lies in their timesharing orientation.

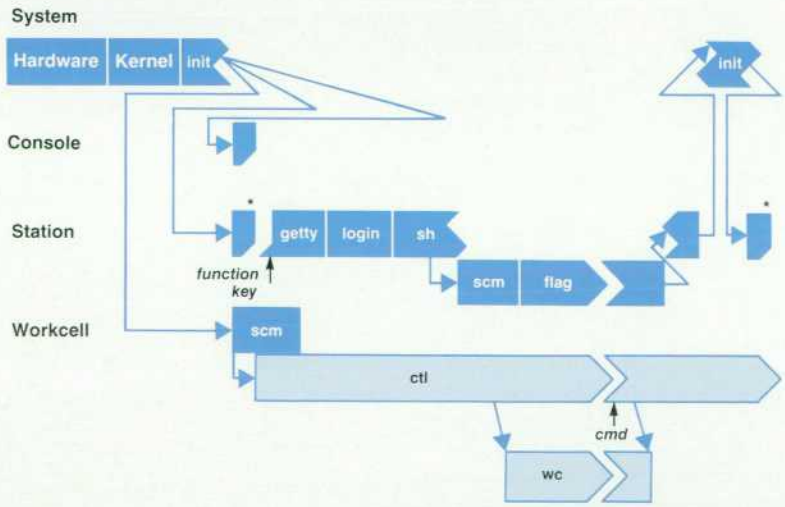


Fig. 6. *Preload station startup sequence.*

Acknowledgments

I would like to thank Gary Modrell and Kyle Polychronis for their valuable comments and the generous contribution of their time in reviewing this paper. I would also like to thank Jim Gruneisen and the software engineers of the Surface Mount Development Center for the opportunities and insights they provided that led directly to the ideas presented above.

References

1. M.L. Watkins, "Software Architecture and the UNIX Operating System: An Introduction to Interprocess Communication," *Hewlett-Packard Journal*, Vol. 38, no. 6, June 1987, pp. 26-32.
2. B.W. Kernighan and R. Pike, *The UNIX Programming Environment*, Prentice-Hall, 1984.
3. M.J. Bach, *The Design of the UNIX Operating System*, Prentice-Hall, 1986.
4. M. Rochkind, *Advanced UNIX Programming*, Prentice-Hall, 1985.
5. *HP-UX Reference, Vol. 1A: Section 1*, Hewlett-Packard Company, 1985.
6. *HP-UX Reference, Vol. 1B: Section 1*, Hewlett-Packard Company, 1985.
7. *HP-UX Reference, Vol. 2: Sections 1M and 2*, Hewlett-Packard Company, 1985.
8. *HP-UX Reference, Vol. 3: Sections 4 and 5*, Hewlett-Packard Company, 1985.
9. *HP-UX System Administrator Manual*, Hewlett-Packard Company, 1985.

CORRECTION

In the February 1988 issue, the authors of the article "Waveform Recorder Software Design" on pages 12-13 should have been listed as John Fenwick, Member of the Technical Staff, Information Networks Division, and John Ketchum, Project Leader, Santa Clara Division.

A Virtual User Simulation Utility

The vuser *utility makes it possible to simulate one or several users on a system. It is a useful tool for all types of testing, particularly interactive testing.* vuser *runs under the HP-UX operating system on HP 9000 Series 800 and 300 Computers.*

By Kjell A. Olsson and Mark Bergman

TESTING ACTIVITY is a major part of software development. The task is particularly acute if the testing requires interactive communication between the tester and the product. It is usually necessary to have a tester sit down and start the testing by hand. This person has to be very familiar with the application being tested and has to know what looks correct and incorrect when running a test. This requires many hours of high concentration. If a problem is found, the application must be fixed, and then the entire test has to be redone.

vuser is a software utility created to do interactive testing without a human tester. It is for internal Hewlett-Packard use and is not available as a product. To use vuser, the tester writes a vuser script that executes exactly the same test the tester would have performed by hand. vuser follows the script to do the interactive testing and capture the test results for verification. The test can be repeated anytime, over and over again. vuser can also easily reproduce any problems found in a product.

vuser allows a user to send, read, and compare characters (bytes) over a communication channel. Usually this channel is a regular RS-232-C communication link. On one end of the channel is a computer with the HP-UX operating system running vuser. On the other end of this logical channel is the application system with which vuser communicates. The system on which vuser resides is called the host, and the application system is called the system under test. vuser provides a programmer with the ability to describe the interaction over the channel using primitive operations. It then executes the script written by the programmer to cause specific tasks to be done on the system under test. vuser does not require that the byte stream or sequence of characters coming from the system under test have any particular format. Since instruments, printers, and other devices that exchange information with a computer also communicate using byte streams, vuser can be used to exercise any device that can be connected to an HP-UX system by some physical communication channel. In other words, vuser's name, which stands for "virtual user," describes only a subset of its functionality.

vuser is currently being used in HP sites in Australia, Japan, Germany, England, and in many places across the U.S.A.

Running vuser

vuser runs on the Series 800 and 300 computers of the HP 9000 product line, under the HP-UX operating system. It can communicate over a port from the host machine to another, or over a pty back to the host machine. A pty is a pseudoterminal developed for LAN communications. The port connection is usually RS-232-C from one port multiplexer into another port multiplexer. One of the connectors on such a line must be reversed like a terminal's connector. This way, the host computer looks like a terminal connected to the system under test. It is also acceptable to connect one port to another on the same machine. Figs. 1, 2, and 3 show different vuser systems.

The port on the host system will be connected in software to a tty in /dev. This should be already set up by the system administrator of the host system. Also, if there is a getty on the host's tty, it needs to be turned off, and there must be something running on the system-under-test's port to receive the incoming byte stream. (getty is the HP-UX program that allows a person to log onto a system.)

When a pty is used, only the program that is to run on the pty needs to be specified. This program is usually sh (Bourne shell) or csh (C shell). Tests are then run in a local shell. For more information about sh and csh, consult the *HP-UX Standard Specifications Manual.*

vuser is run like a normal HP-UX program. A script can be precompiled to save time. Interactions during a vuser run appear on the tester's terminal screen. When vuser ends, an HP-UX prompt appears. A tester can run as many vusers as desired at one time, but because of HP-UX limitations

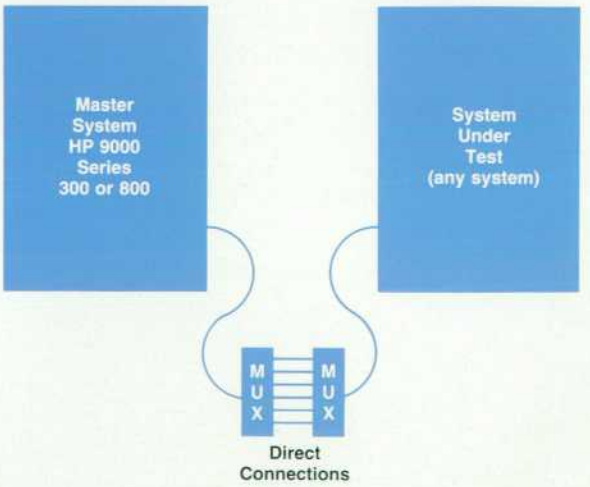


Fig. 1. *Basic* vuser *system.*

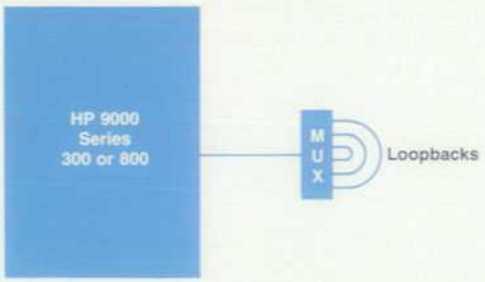


Fig. 2. *Loopback system.*

on the number of processes allowed a normal user, only two or three vusers should be run as a normal user. To run more vusers at the same time, the tester should be logged on with root permissions.

How vuser Works

vuser works on a simple communication theory: send something to a system under test, and expect something back. All transactions in a vuser script are done this way. There is a send statement and an expect statement. The send statement sends a byte stream to the system under test. An expect statement matches the expected response with the return from the system under test. An expect statement is set up like a switch statement in C, but instead of cases, there are selects. The select statement contains a regular expression to match with the incoming stream. A simple type of regular expression is a pattern of bytes. For example:

```
select "a.*b"
```

finds the shortest string that starts with an a and ends with a b, such as acb.

A basic exchange could look like this:

```
send "echo Hi There\r"

expect select "Hi There"
endexp
```

The \r is a carriage return.

Any information that comes back from an application is stored in a lookahead buffer. This is the buffer that is looked at by the select statements. All of the select statements in an expect are processed at the same time, and the first pattern that matches gets to execute its statements. When a pattern is matched from the lookahead buffer, that part of the buffer is thrown away, and any subsequent matching occurs from this point. A word of caution here. A select statement will match the first pattern that fits, not necessarily the one it might be expected to match. For example, if an HP-UX prompt is being matched, and the application sends back a string that looks exactly like the prompt, then the select will match that string. The result might be that a send is executed that should not have been executed yet.

Making a Test

The first thing a person doing interactive testing must realize is that the test is done on a real application. Many unexpected things can happen while an interactive application is being used. For example, in the HP-UX system, when a carriage return is hit while a prompt is on a terminal screen, another prompt would be expected to return from the computer. Usually, a prompt returns as expected. But sometimes, it may take a while for the prompt to appear, other characters from a background job may appear first, or the prompt could be lost somewhere by the computer or between the computer and the terminal. A tester would normally just hit carriage return again to get a new prompt before continuing.

vuser scripts should be treated the same way as real interactive testing and should be set up just like a live testing session. There should be a discussion between the test and the application. All the normal exchanges in this discussion are the meat of a script. The ability to handle the unexpected as well as the expected lies at the heart of writing a powerful test script. Writing a script can be a very complex task. Testing a system that works exactly as desired is easy. Testing a system that does exactly what it is told to do is not. Alas, most code and machines are imperfect. So, a script should be written to accommodate any problems as well as the normal interaction with the system to be tested.

timeouts and sleeps

It often happens that a system will fail to respond for a period of time. When this happens, a tester will normally hit the **RETURN** key or the **BREAK** key until the application responds again. The timeout statement was created to handle this condition.

A timeout is used like a default statement in a switch statement in C. A timeout is the last selection-type statement in an expect statement. It can take two parameters. The first parameter is the time in seconds that a script will stay in

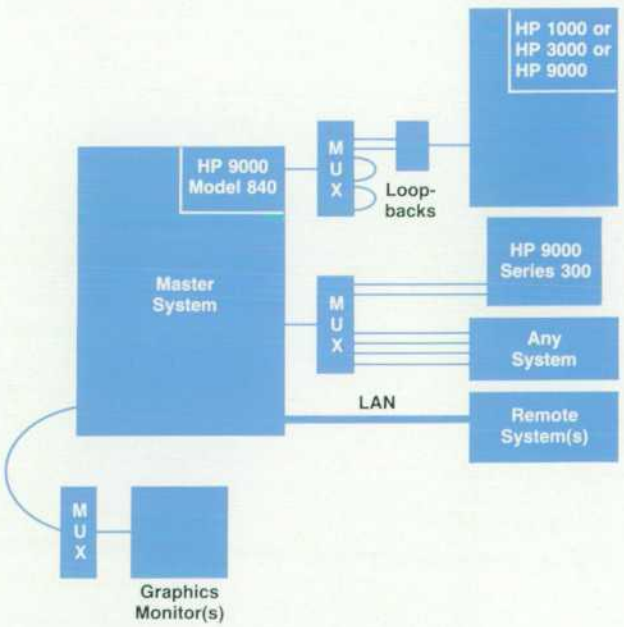


Fig. 3. *Full* vuser *system.*

an expect statement before selecting is stopped. If none of the expect statement's select statements is matched before this time runs out, a timeout occurs, and the statements associated with the timeout are executed. The second parameter is a number specifying the maximum number of times this timeout can occur. If the number of timeouts is greater than the second parameter, then the script ends, and an appropriate message is printed. This parameter is used to stop tests that are in serious trouble. If a timeout is not specified in an expect statement, the script will stay in the expect until one of its selects is matched.

Following is an example of a script that logs onto an HP-UX system. This example also uses labels and gotos.

```
send "\r"
login:                          /* label identifying this
                                   expect */
expect
   select "login:"              /* matched login */
      send "username \r         /* send login name */
   select "Password:"           /* saw "Password:" */
      send "\r"                 /* send RETURN*/
      goto login                /* look for login again */
   timeout 10   5               /* didn't see either */
      send "\r"                 /* send RETURN */
      goto login                /* look for login again */
endexp
password:
expect
   "Password:"                  /* matched Password */
      send "mypassword\r"       /* send secret password */
   select "login:"              /* hmm, shouldn't be here! */
      send "username\r"         /* oh well! */
      goto password             /* now we are back in sync */
   timeout 10 5
      send "\r"                 /* send return */
      goto login                /* try again */
endexp
```

Sometimes there needs to be a wait between two actions, or the speed of the sending of byte streams needs to be slowed down. The sleep time statement causes a script to wait time hundredths of a second before continuing. sleeps are used to set up timing in a script. Also, they can simulate a person's typing speed or response time.

A tester should be wary of just using sends and sleeps in a script. Most systems that are tested do not run the same every time. A script whose timing is correct for one test run might fail unexpectedly on another test run. The best way to be sure a test is running correctly is by using the basic send-expect communication pairings and use sleeps only where necessary. One other caution about sleeps is this example:

```
expect
   select "a.*b"
      send "more stuff\r"
      sleep 20
endexp
```

This sleep may cause a problem for the next expect. This is because the script might not leave the last expect before a reply comes from the system under test.

Macros

When a complicated exchange needs to be done many times in a script, the exchange should be made into a macro. A macro can be built out of any part of a vuser script and used in the script where necessary. vuser macros use the capabilities of the C preprocessor. For more information on the C preprocessor and the building of macros, look up cpp(1), in the *HP-UX Standard Specifications Manual*.

Finishing Up

vuser scripts are programs, and as such, usually take several iterations to become solid. If a test fails, the results will show exactly how the test failed. vuser does not do post-test checking. There are other programs that help in test verification, including grep(1) and diff(1) from the HP-UX utilities.

vuser sends all the output from its run to stdout, the standard output file of the HP-UX system. A tester can also have vuser leave a log of the test. A logfile is created in parallel with the output going to stdout. The logfile contains beginning and ending time stamps, elapsed time, the port that is being worked on, the type of transaction, the number of bytes in the transaction, and the byte stream of the transaction. The time stamps are accurate to 100 microseconds on the HP 9000 Model 840. A logfile is stored in a non-ASCII byte format. This format can be translated into ASCII by using the program vtoa.

Commands

This section presents all of the vuser commands and their syntax. Regular expression syntax is explained in the following section. vuser run string options are listed in the box on the next page.

Commands can be separated by newlines or semicolons. See cpp(1) for a description of cpp directives.

```
distribute normal mean deviation [min [max]]
distribute binomial mean deviation [min [max]]
distribute off
```

distribute specifies the distribution of time waited between each pair of characters sent to the output stream. The DELAY symbol used in a sleep statement is also evaluated to a new value consistent with the current distribution, each time it is encountered. distribute's first argument is the type of distribution. off is the default type, and turns the distribution feature off, causing output to the output stream to occur at full speed and DELAY to evaluate to zero. normal sets up a table-driven normal distribution. binomial sets up a binomial distribution. This implementation has more overhead and less long-term accuracy than the normal distribution, but is compatible with tepe.

The parameters for the normal and binomial distributions are in hundredths of a second. Mean is the average delay between characters. deviation is the standard deviation from the mean. min is the minimum delay between characters. This is optional, but required to specify max. The default min value is mean − (3 × deviation), but not less than zero. max is the maximum delay between characters. The

vuser Run String Options

Only one flag may follow each −. Parameters, when present, must follow the flag exactly as shown below, either after a space or with no intervening space.

script	The name of a file containing the vuser commands. It must be specified. An uncompiled script filename is expected to end in .v; if it does, a compiled version will be left in a file of the same name without the .v tail.
−Dname=def	Define name to the preprocessor, as if by a #define directive. If no =def is given, name is defined as 1. Usable only when compiling a source file; see cpp(1) for details.
−E	enq/ack compatible operation: whenever an enq is received, an ack is sent. Nothing else is changed.
−F	Flush. Inhibits buffering of output to stdout and to the logfiles. When this option is given, output is written immediately, at the expense of additional overhead.
−Idir	Include. Change the algorithm used by the preprocessor for finding include files to also search in directory dir. Usable only when compiling a source file; see cpp(1).
−N	This causes the source script, once compiled, to be decompiled and sent to stdout. A vuser session is not run. The decompiled script is recompilable. This option is useful if the source code for a compiled script has been lost, and is also good for debugging, since it shows the code vuser actually executes. For debugging, see also vub(1).
−Uname	Undef. Remove any initial definition of name in the preprocessor. Usable only when compiling a source file; see cpp(1).
−V token=str	All occurrences of token immediately surrounded by single, double, or grave (back) quotes will be replaced by str. This substitution is dynamic, not precompiled as is the −D option. It is usable only when compiling a source file or when token is also used in a substitute statement in the script.
−c	Compile the script, but do not run it. The source file must have a .v suffix. The compiled code is left in a file of the same name, but without the suffix.
−l logfile	Logfile. When selected, time stamped binary log records are written to logfile.O and logfile.I. Then, running vtoa will create a readable ASCII logfile. See vtoa(1). The logfile format is defined in vlog(4).
−m	Lock vuser into memory, so it cannot be swapped out.
−p cmd	pty. Specifies that a pseudoport is to be used on the local system, rather than a normal port (see the −t option). The particular pseudoport is automatically selected from those available, but the full pathname of the program seen at the other end of the pty (e.g., /bin/csh) must be specified here as cmd.
−t port	tty. Specifies the full pathname of the local system port used by vuser, e.g., /dev/tty04. It is required unless a pseudoport is used (see the −p option.
−w	wait. Tells vuser to wait for a SIGINT (see signal (2)) before starting. It compiles the script and pauses just before it is executed.

default is mean + (3 × deviation).

```
expect
  select "reg-exp"
    statement
    . . .
  [select "reg-exp"
    statement
    . . .
  . . .
  [timeout secs [count]
    statement
    . . .                ]
endexp
```

The expect statement is a case statement. The action of the expect statement is to monitor the input stream and branch according to its select and timeout statements. The statements associated with the first select (or the optional timeout) that is satisfied are executed, then control is passed to the next statement after the endexp. The contents of the input buffer are flushed once a select is matched or a timeout is executed.

The select statement matches on its regular expression reg-exp (see the section on regular expressions, next page). A timeout is selected if none of the selects in the expect statement is satisfied before the time secs has expired. The count parameter can be given to specify the maximum number of times this timeout case can be executed during execution of the vuser script. If not specified, count is effectively infinite. This parameter sets a counter with its particular timeout statement. Once set, this counter is never reset, but is decremented each time the timeout is executed. Once the counter reaches zero, execution of the whole script is aborted.

A timeout is no longer active after one of the expect statement's selects is matched, or once the expect statement is done. An expect statement is not allowed among a timeout case's associated statements, but a goto to another expect is acceptable.

goto label

The single argument of goto is the name of a label. The next statement executed will be the one immediately after the label.

imode char
imode line length
imode line "chars"

imode sets the format of output to the logfiles. Specifically, it sets the criteria for breaking the output into separate logfile records. char mode causes each character sent or received by vuser to be logged into a separate record. The parameter for line mode is either a maximum record length or a string of end-of-record characters. If the string is given, the occurrence of any one of them is sufficient to force the end of the current record. The default format is imode line.

label:

A label names a location in the script to be used as the target of a goto.

log "message"

The message within double quotes is put into the logfile when the log statement is executed, if the -l command-line option was given.

repeat count
    statement
    . . .
done

The single argument of repeat is a count of the number of times the statement(s) between it and the following done are to be executed.

rtprio priority

rtprio executes the HP-UX command that allows the user to set a real-time priority of up to 127. The process can take over the machine and run in real time for performance measurements.

send BREAK
send `command`
send 'file'
send "message" [disttype mean deviation [min [max]]]

send writes to the output stream any characters within double quotes, the contents of a file specified within single quotes, the results (stdout) from execution of an HP-UX command presented within grave (back) quotes, or a communications break if BREAK is specified.

Output to the output stream is done at the full speed vuser and the output port can handle unless a distribute statement has been executed, in which case the timing between characters is as specified by the last distribute statement executed. This can be overridden for a single send statement by providing the optional disttype parameters, which are the same as used in the distribute statement.

shell "command"

The argument of shell, within quotes, is sent to the HP-UX system to be executed. The output of the command is sent via the output stream to the system under test.

sleep DELAY [disttype mean deviation [min [max]]]
sleep time

The argument of sleep is either an integer constant indicating hundredths of a second to delay before further action, or DELAY. DELAY evaluates to a time consistent with the timing distribution specified by the last distribute statement executed. This can be overridden for a single sleep DELAY statement by providing the optional disttype parameters, which are the same as used in the distribute statement.

stty "arguments"

stty mimics the HP-UX stty(1) command. It acts on the output port connected to the system under test. Within double quotes, it takes the same arguments as the HP-UX command, and can be used to set character size, baud rate, and anything else settable from the HP-UX system. XON/XOFF is the default setting for both transmit and receive flow control. The statement stty should be used to turn off transmit flow control for communications with MPE systems. See stty(1) in the HP-UX manual.

substitute token "replacement"

Like cpp's #define statement, substitute causes token, when it occurs subsequently in the script, to be replaced by replacement. However, these substitutions are done in real time rather than precompiled, and apply only when token is immediately surrounded by single, double, or grave (back) quotes in the script. For clarity, it is recommended that each token begin with a #.

Substitutions are particularly valuable because they can be changed from the vuser command line. The statement substitute token enables use of the -V option with token.

Regular Expressions

reg-exp (used in select, above) represents a regular expression of the form used by ed, vi, and grep to specify strings to be selected from a bunch of text. select statements in vuser use these reg-exps—a subset of those available in ed—to specify strings to be selected from the input stream. A string is matched to a reg-exp from left to right. Except as noted below, each character of a string that matches a reg-exp must match the next character in the expression. For example, since . matches any character, the expressions s.x, s.*x, and s.* will match the string six; but the expressions sx, sax, and s..x will not.

The reg-exp notation is as follows:

- . A period matches any single character.
- \ Backslash escapes the special meaning of the following character.
- + The plus sign means "one or more of the preceding character." It may not appear as the last character of a reg-exp used with a select unless its special meaning is escaped.
- * The asterisk means "zero or more of the preceding character." It also may not appear as the last character of a reg-exp used with a select unless its special meaning is escaped.
- [] Any character within the brackets is considered a possible match for the next character in the input stream. Exceptions to this are - (see following), \, and], which must be escaped.

– Within brackets, the minus sign is a range specifier. A range of ASCII characters can be specified by placing the lower bounding character before the minus sign and the upper bounding one after it. For example, all numbers can be specified by [0–9], and all letters by [a–zA–Z] (Note: [c–a] is backwards and incorrect, and [A–z] includes some special characters). However, if the minus sign is the first character inside the brackets or if it is outside the brackets, it is interpreted literally.

CATS

vuser is a part of CATS, an internal HP system that includes many tools supporting vuser and many other aspects of testing. As mentioned before, there is vtoa, the logfile translator. vtoa takes vuser logfiles and produces a readable output. The information from this output is organized by time. All the transactions that occur during a vuser session are recorded in this logfile output. There is also vub. vub will compile a vuser script and expand out all the macros in the script. The result is the script vuser actually sees. vub is used mostly for debugging scripts. rcap and ucap are two programs that capture keystrokes as a tester actually works with a computer system. This allows a tester to run an application by hand. While the person is working on the computer, the entered keystrokes and the time between each keystroke are stored by rcap or ucap. Then, when the person is done with the test, rcap or ucap will create a vuser script file. When this new vuser script file is executed, it will exactly reproduce what this tester just finished, right down to the typing speed. expect statements still need to be put into the scripts made by rcap and ucap. This is so that any timing problems of interacting with the system under test are taken care of.

Large test runs can produce much information. This information usually needs to be processed intelligently to capture any important facts produced by the test. This can be be done using the Data Reduction Filter (DRF), which is currently being developed as part of CATS. DRF is actually another interpreter like vuser, with its own powerful language.

Scripts written in the DRF language do the data reduction on the test output. DRF scripts are written knowing how the data is stored and exactly what a tester wants to see, or not see, from the test results. This way, unimportant information can be thrown away, and the rest of the information can be compared to see if the the test was successful or not.

Acknowledgments

Acknowledgments go to the writers and instigators of vuser: Don Steiny, the original author of vuser, Randy Menna, Dave Decot, Ron Gremban, and Soren Stammers for his input and his assistance in defining the DRF language. Also, a special thanks to Don Barnett and his staff and our many CATS customers around the world.

An HP-UX Kernel Load and Measurement System

This system runs on HP Precision Architecture computers under the HP-UX operating system. It can be used to generate and measure different types of HP-UX kernel activities.

by Kjell A. Olsson and Grace T. Yee

THE KERNEL LOAD AND MEASUREMENT system has been developed to test the HP-UX operating system on HP Precision Architecture computers under various levels of stress, and at the same time, to measure and control that stress.

The system is designed to be used together with any type of tests or test packages (kernel, commands, subsystem, or application test package). The system makes sure that the specified load or measurement is generated and leaves the testing responsibilities to the test package.

The load generating portion of the system is a feedback controlled package that allows a user to specify a desired load on a scale between 0 and 100%. For each load area, a specific program is responsible for maintaining the specified load. A user can add a load program to any areas measured.

The measurement portion of the system is used by all of the load programs and by on-line and off-line display programs. The on-line display program displays the current load using bars or meters on bit-mapped displays while the off-line programs display measurements on hard-copy line diagrams or tabular reports.

The Kernel Load and Measurement System is for internal HP use and is not available as a product.

Objectives

The Kernel Load and Measurement system was designed with the following goals in mind:

- Allow independent load control of key kernel areas
- Provide real-time response to changes in load requirements
- Provide self-adjusting capability to respond to external load disturbances
- Provide one common utility to measure stress loads for all kernel parameters
- Provide on-line real-time monitoring capability
- Provide off-line display capability
- Provide a local, transparent distributed environment
- Flexible and extendable design to allow for future enhancements.

System Overview

The Kernel Load and Measurement System consists of a shared memory area, which serves as a central communication area between different components in the system, and the following system components: load measurement, load generation, and display programs.

The load measurement component consists of a program called Snap, which extracts load measurements from the kernel and stores the information in the shared memory. The HP-UX kernel has been instrumented in many areas to keep track of different types of load information used by Snap.

The load generation component is started by the load daemon, which first starts Snap and then starts individual load programs to generate the expected load. The expected load levels can be specified and changed by users interactively from a terminal or automatically from different types of load and test selectors. Each load program uses the data generated by Snap and is responsible for generating and maintaining a specific system load.

The display programs display the data collected by Snap on screens for on-line monitoring or on hard-copy line diagrams or tabular reports for off-line reporting and analysis. The data can also be stored in a file for temporary storage. This file can later be fed to the screen display programs, which can display the prerecorded data, or to the diagram and tabular report generating programs. The screen display programs can display the data in slow motion, fast forward, and reverse.

Key Data Structure

To accomplish the objectives of shortening response time and minimizing system resource utilization, we have used the UNIX® System V shared memory feature to store the critical communication information. This provides an easy access mechanism to central data for all the system modules.

There are nine key components in the shared memory for each load and measurement area:

- Address Label. Label to use when addressing this load or measurement area.
- Display Label. Used as a description label when displaying a measurement on screen or paper.
- Max Value. Maximum system load value (e.g., for a file table load, this contains the maximum number of entries in the table).
- Actual Value. Actual system load value (e.g., for a file

The HP-UX operating system is Hewlett-Packard's version of the UNIX System V operating system. UNIX is a registered trademark of AT&T in the U.S.A. and other countries.

table load, this contains the actual number of entries in use in the table).

- Expected Value. Expected or desired system load value in percentage of the max value (e.g., for a file table load, this contains the percentage of the file table that we want to fill).
- Damped Value. Moving average of the actual value. The damped value is useful for both load control and displaying the overall trend of load variations over time when there are frequent fluctuations in the actual value.
- Damping Factor. Value indicating the degree of damping to be applied to the actual value (0 = no damping, 1 = max).
- Regulating Factors. Factors used to control the PID regulator. The three fields control the weight of the proportional, integral, and differential sections of the PID regulator.
- Load Prog Parameters. Auxiliary parameters for debugging and informational purposes.

Fig. 1 shows an example of the key data structure.

Load Measurement

The program called Snap is responsible for taking snapshots of the actual loads from the kernel. Users can start the measurement system by running Snap, optionally passing to it the run-time priority and the path name of a startup file.

When Snap is started, it locks itself in memory and runs at a high real-time priority. This is to ensure that the load measurements are always collected, especially when the system is heavily loaded.

During startup, Snap checks for the existence of the shared memory. If it does not exist, Snap allocates the shared memory and then initializes address labels, display labels, max values, expected values, damping factors, and regulating factors based on the information from the startup file. It also stores the name of the startup file in the shared memory. The startup file is also used by the on-line display programs and the load programs.

Snap then goes into an infinite loop doing the following:

- Extracts the actual system load from the kernel
- Calculates the damped values
- Updates the actual values and damped values in the shared memory
- Sleeps a specified length of time.

Snap calculates the damped values as a moving average of the actual values based on the following formula:

$$DV(0) = AV$$

$$DV(n) = DV(n-1) * d + AV * (1-d)$$

where DV is the damped value, AV is the actual value, and d is the damping factor ($0 \leq d \leq 1$). If $d = 0$, there is no damping. If $d = 1$, there is maximum damping and DV will never change.

As mentioned earlier, damped values are useful for displaying the overall trend of load variations over time when there are frequent fluctuations in the actual values.

For each update of the shared memory, Snap updates the counter in the communications area of the shared memory. The counter is used by the load programs and display programs to find out if the shared memory has been updated by Snap since they last accessed the shared memory.

The frequency with which Snap updates the shared memory is a parameter in the shared memory and can be modified dynamically by the user.

Load Generation

The user can specify initial values for the shared memory and run-time parameters for the load programs in a startup definition file. The following is a list of information that can be given for each load and/or measurement area.

- Address Label. Identifies a specific load area in the shared memory. The label is used to address the specific load and/or measurement area.
- Display Label. Label used by the display programs.
- Max Value. Maximum value of the load area.
- Damping Factor. Damping factor of the load area.
- P. Proportional regulating factor of the load area.
- I. Integral regulating factor of the load area.
- D. Differential regulating factor of the load area.
- Priority. Priority of the load program. This can be a numeric value specifying the real-time priority or T for timesharing.
- Run String. Full path name of the load program followed by any run-time parameters to be passed to the load program.
- Accelerator File. File used by the load program specifying the relationship between the load and the values of the different parameters used when controlling the load.
- Working Directory. The directory the specified load program will be running under.

All but the first four entries are optional if the area is measurement-only.

The full path name of the startup file (default: .loadrc) can be passed as a parameter to the load daemon and Snap. It will be stored in the shared memory by Snap. This removes the requirement to have a specific file in a certain location. Users can use a shared memory display utility to find out the name of the startup file for modifications to the file.

Load Control. The load level for each load area is controlled primarily through the setting of expected values in the shared memory. The expected values do not have to be set

Key Data Structure

Regulating Factors

Label	Disp Label	Max	Actual	Exp	Dampv	Dmpf	P	I	D
proc_ld	Proc Table	256	110	40	112	0.5	0.0	0.0	0.0
cntxt_ld	Cont Swtch	1000	250	30	225	0.5	0.6	0.2	0.2
fault_ld	Page Faults	500	92	20	76	0.7	0.8	0.0	0.2
inode_ld	Inode Table	512	256	50	250	0.0	0.0	0.0	0.0
dread_ld	Disk read	550000	118000	20	112000	0.6	0.4	0.1	0.4

Fig. 1. *The key data structure in shared memory contains the parameters specified for each load and measurement area.*

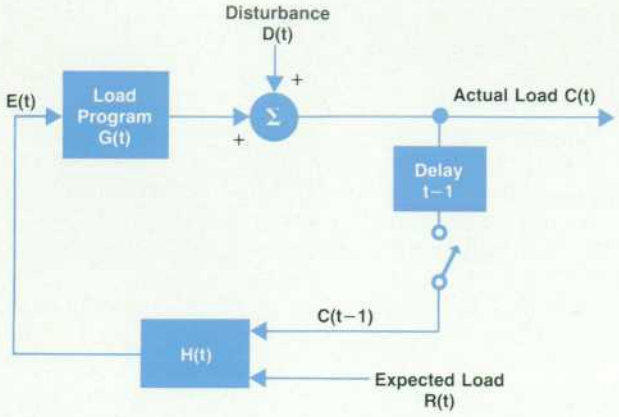


Fig. 2. *A load program depicted as a single-loop sampled feedback control system. See text for definitions of symbols.*

up before the load system is started. They can be set up or altered at any time while the load system is in operation, and each load program will automatically adjust to the new load level.

There are several ways to set or change the expected values in the shared memory. They can be specified by users interactively, using shared memory update utilities. They can be specified by a test script if there is a load environment that the test expects to run in. They can be selected and specified by a test selector controlling test and load activities on one or several systems in a network. In this case, one of the systems will be the master controller. From the controller, the user can start up test and load programs on the rest of the systems in the network and set or alter the expected loads dynamically. The load is usually set randomly or set to vary over time using different mathematical functions such as cosine or triangle waves. This setup method is extremely useful when testing networks and distributed systems.

Load Daemon

The load daemon is responsible for starting up the load system. It reads the startup file and starts the Snap program, passing to it the name of the startup file. It waits for Snap to allocate and initialize the shared memory, and then checks the expected values in the shared memory. For each load area with a nonzero expected value, the load daemon will start up a load program based on the run information in the startup file. Afterwards, it goes to sleep and wakes up periodically to scan the expected values. If an expected value has changed from zero to nonzero or vice versa, it will start up or terminate the corresponding load program. The frequency with which the load daemon checks the expected values is a parameter stored in the shared memory and can be modified dynamically by the user.

Load Programs

Each load program is implemented as a single-loop or multiloop sampled feedback control system. That means that each load program reads the current load from the system (i.e., the shared memory area updated by Snap), compares it with an expected load (i.e., the expected value in the shared memory), and tries to readjust the load to bring it closer to the expected value.

A single-loop system only compares the actual and expected values. A multiloop system also looks at how the actual value is changing (away from or toward the expected value) and the length of time the actual value has been different from the expected value.

The measurements and comparisons are done in discrete time. The actual values in the shared memory are only updated by Snap once each time interval (tunable parameter), and each load program can only read the shared memory between load generating duties. Fig. 2 depicts a load program as a single-loop sampled feedback control system, and Fig. 3 depicts a multiloop system.

In Fig. 2, R(t) is the expected value or reference value, and E(t) is the input to the load generating portion of the program. E(t) describes how well the program is doing in comparison to the expected load R(t). G(t) is the load generating portion of the program. It is up to the load program to convert E(t) into an increase or decrease of the load. D(t) represents disturbances put on the system by other programs. C(t) is the actual load on the system. The next time Snap runs, this value will be stored as the actual value in the shared memory. (t − 1) is a time delay of one Snap update time interval. H(t) is a function using the expected value R(t) and the actual load C(t − 1) to provide feedback to the load program.

Two routines or regulators have been written to implement the models of Figs. 2 and 3. They handle everything in the models except the load generating part, G(t), which is the responsibility of each load program. The two regulators differ in the H(t) section and the nature of the output E(t). One regulator (Fig. 2) is designed for static load and returns an absolute value (the difference between the actual and expected load) while the other (Fig. 3) is designed for a dynamic load and returns a relative value based on proportional, integral, and differential values of the difference between the actual and expected loads.

Each load program is responsible for attaining and maintaining the expected load for a specific load area. It calls the regulator routine which provides feedback on how

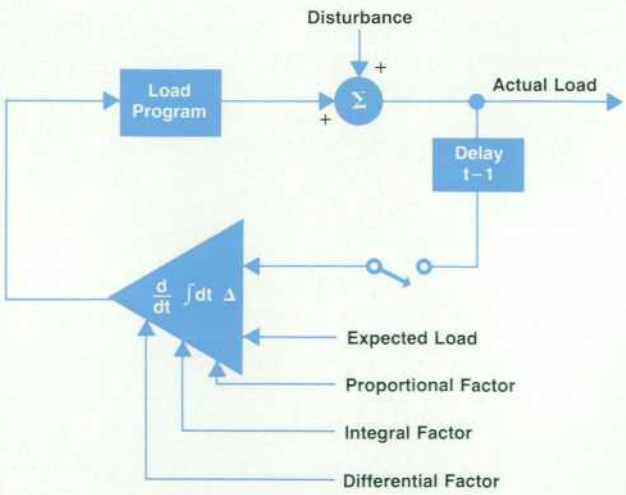


Fig. 3. *A dynamic load program implemented as a proportional-integral-differential (PID) control system.*

much the load needs to be increased or decreased to reduce the deviation of the actual load from the expected load to zero. The load program does the appropriate amount of work to achieve the expected load and then calls the regulator again to check the progress.

If the load program cannot achieve the expected load running at 100%, it will fork and let its child be responsible for achieving the expected load level while it is still running at 100%. The forking process is repeated until the target is reached. The reverse is also true. If the expected load has been reduced or an external disturbance, such as a test program running in parallel, causes the actual load to exceed the expected load, the load program will terminate its youngest child to reduce the load. The termination process is repeated until the target is reached. Note that the load program's role is to guarantee a minimum load. If an external disturbance generates a load higher than the expected load, the load program will remain dormant. The resultant load will still be higher than the expected load.

There are two types of load programs, static load and dynamic load. They differ primarily in the regulator they use and the mechanism for achieving the expected load.

Static Load. A static load, once it is generated, will remain stable. No work is needed to maintain the load. Examples of static loads are filling up system tables and using up memory areas. Static loads typically have a well-defined measure of the deviation of the actual load from the expected load. There is also a well-defined unit of work that needs to be done to achieve the expected load.

The regulator for a static load is very simple. The return value from the static regulator is calculated as follows:

Return (Static Regulator)
= Max Value * Expected Value/100 − Actual Value.

Note that the expected value is expressed as a percentage of the max value, so it needs to be converted to an absolute value.

Dynamic Load. A dynamic load, once it is generated, requires continuous input from the load program to maintain the load level. Examples of dynamic loads are I/O activities, paging activities, and swapping activities. Unlike static load, there typically is no well-defined formula of how much work is needed to achieve the expected load.

Because we need a general-purpose design that can be used to generate any dynamic load, we have implemented the dynamic load programs as multiloop sampled feedback control systems using a PID regulator. A PID regulator is a very general regulator that returns a relative value to the calling program, telling it how well it is doing in comparison to the expected value. The regulator calculates its return value based on the proportional distance between the actual and expected values, the time integral of this distance, and the time derivative of this distance. These three parameters are all multiplied by factors stored in the regulating factors area of shared memory for each load program

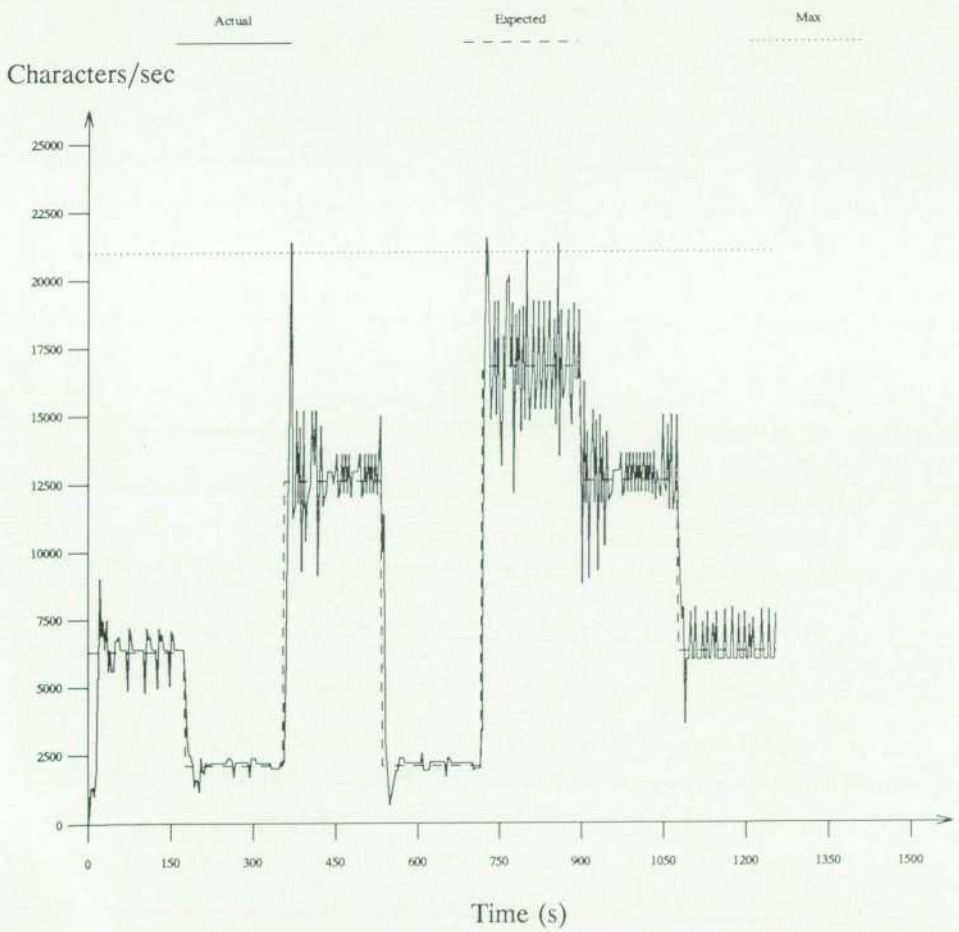


Fig. 4. *Diagram generated by the off-line display system showing a dynamic load program controlling the load on the* tty *driver. This load program uses the damped value to calculate the feedback. Parameters set for the program are: damping factor = 0.4, P = 0.4, I = 0.2, D = 0.4.*

before they are added to get the returned value. This makes the regulator very tunable for each load area.

Fig. 3 depicts a dynamic load program using a PID regulator. The return value from the PID regulator is mapped into an accelerator file to determine the amount of work needed to reach the expected load. Each entry in the file describes the amount of work needed to generate a specific load level. All the entries in the file are sorted in order of increasing load.

For example, a dynamic load program to generate a disc-read load accomplishes the work by reading in data of various buffer sizes and sleeping for a length of time after each read. Each combination of buffer size and sleep time results in a unique disc-read load. The entries in the accelerator file for this load program would consist of sleep time and buffer size.

The contents of the accelerator file are empirical data collected from measuring the load generated by varying the amount of work put out by the load program. The greater the number of entries in the file and the more linear the

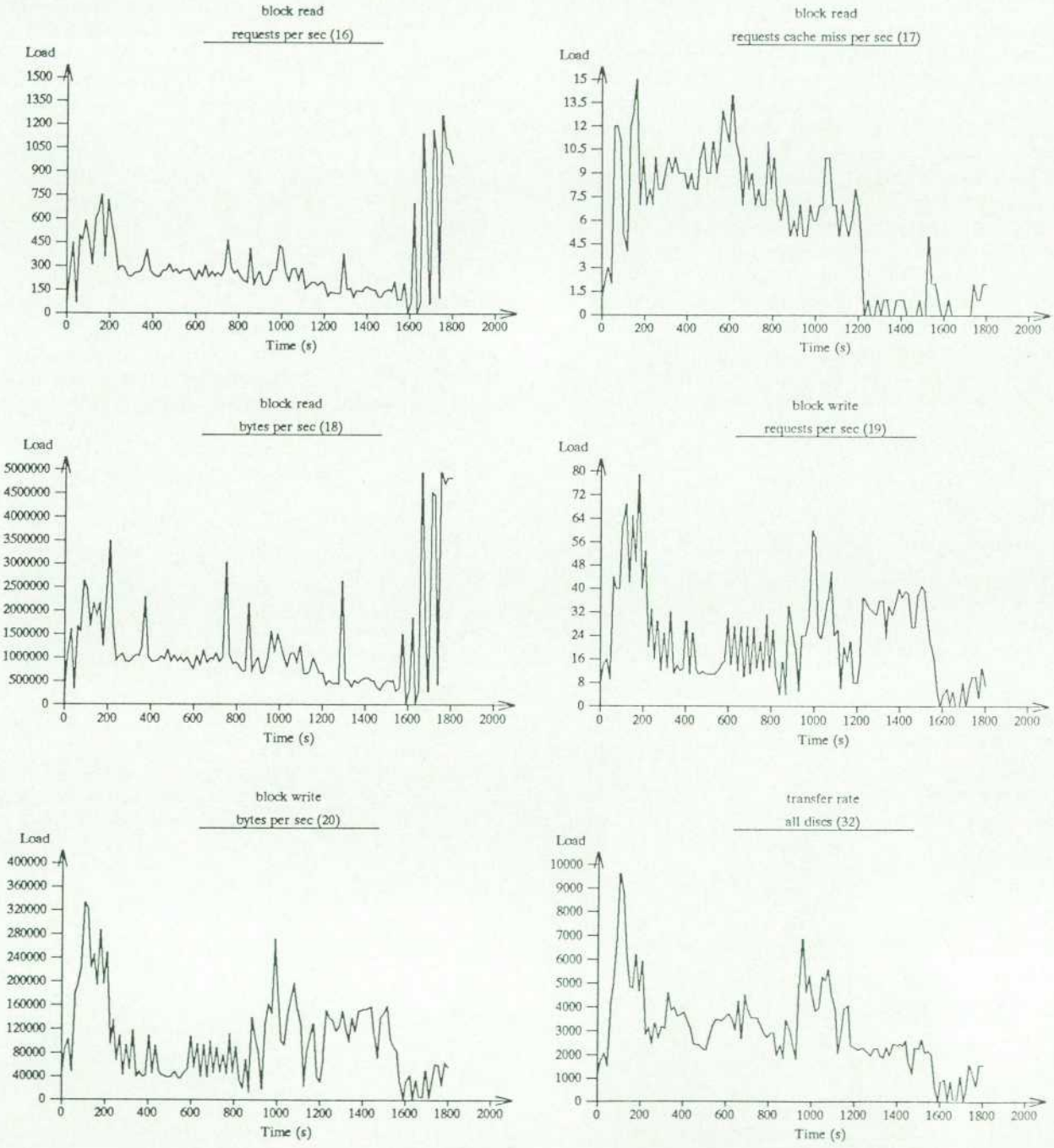


Fig. 5. *For a disc test program generating disc I/O activity on multiple disc drives, data from six load areas in shared memory was extracted, converted, and plotted on an HP LaserJet printer.*

increase in the load per entry, the easier it will be for the load program to achieve the expected load.

To be able to achieve fine enough granularity in the accelerator file, a special sleep routine accepting very short sleep time has been developed. This was done by using some of the new HP-UX real-time extension system calls, which provide very fine granularity. The developed routine, Nap, accepts sleep time down to one hundredth of a second.

Instability. Every time a feedback control system is used there is a possibility of creating instability in the system. This can easily happen if the regulator is not tuned to control the load properly or if the amplifying part (the load generating part) responds too much to the output of the regulator.

As an example, if the system responds so heavily on each mismatch between the actual and expected values that after the adjustment, the actual value overshoots and ends up farther from the expected value, the two values will continue to diverge until some system limitations are reached.

This can be avoided by testing and tuning each load program before it is added to the test package. The tuning is done by changing the regulating factors in the shared memory as described in a previous section.

This instability, however, may turn out to be useful in the reliability testing effort for the HP-UX kernel. If all load programs are tuned to be unstable, this creates an extremely difficult environment for the kernel to handle. We do not know today if it can be used in our reliability testing, but we are planning to experiment and see how well we can use this feature of the load system.

Fig. 4 shows a sample diagram of a dynamic load program maintaining the expected load level over time. The load program is generating terminal I/O load in units of characters per second. The solid line shows the actual load generated. The dashed line shows the expected load specified by the user and the dotted line shows the maximum load that can be generated.

Display System

The display system consists of data acquisition programs, data control programs, and data display programs. The display system is a distributed system that allows measurements from several systems to be displayed on one or several screens or diagrams at the same time. The data displayed can be on-line data or data previously stored in files. It is also possible to display recorded data in a file together with on-line data.

Data Acquisition. The data acquisition programs read the data generated by Snap from the shared memory or from a file with old measurement data. The data is sent together with a time stamp and system name over the network to a data control program.

Data Control. The data control programs read the data from one or several data acquisition programs via the network and send it to the display or file recording programs. The data control programs reside on the same system as the display or recording programs.

Data Display. There are three main types of data display programs: screen display programs, diagram generating programs, and programs displaying data in tabular form. They can display on-line or previously recorded data (or a mixture) from one or from several systems at the same time.

Screen Display Program. The screen display program permits quick comparison of data elements for interactive adjustment of the expected load levels. Data for different load areas or for the same load areas from different computer systems can be grouped meaningfully. In this way, the side effects of one adjustment can be seen in relation to other load areas. Two different display programs have been developed, one for standard terminals and one for bit-mapped displays.

For display on standard terminals, a cursor-based version has been developed. It is a menu-driven tool that displays the actual and expected values as a horizontal bar, with the actual values displayed using spaces in inverse video and the expected values displayed using the character | overlaying the actual values. The user can specify the load areas, step through the data forward and backward by line or page, or jump to a given element. The exact value of all data elements of the key data structure can be seen by switching to text mode. On any HP graphics terminal, the user can see the actual and expected values represented by an analog meter with two pointers. The default action is to fit one page on the screen, but one can select a different number of devices, which will be sized to fit the screen.

```
Measurement taken on system hpisqis
Snap ran every 10 second
Measurement started at Sat Sep 12 14:12:09 1987
Measurement stopped at Sun Sep 13 12:50:56 1987

Label          Maxval    Avg%    Cov%     0%  1-20% 21-40% 41-60% 61-80% 81-100%
-------------------------------------------------------------------------------
Proc Table        256    49.2     1.1    0.0    0.0   46.5   29.9   17.6    6.0
Cont Swtch       1000    25.9     1.1    0.0   42.4   45.6   10.2    1.8    0.0
Page Faults       500    18.6     1.5   17.2   45.6   24.6   11.0    1.0    0.6
Inode Table       512    42.0     1.0    0.0    0.0   29.0   71.0    0.0    0.0
Disk Read      550000    63.3     2.8    0.0   20.0   15.6   18.4   13.9   32.1

Legend: Label:  Address label
        Maxval: Max value for this label
        Avg   : Average value
        Cov   : Coefficient of variation
        0%    : % of entries at 0%
        1-20% : % of entries between 1 - 20%
```

Fig. 6. *Typical output of the tabular writing program.*

For test runs requiring more flexibility, a new tool was added to the package. This tool and the data acquisition program are interpreters of a simple command language and can change their behavior interactively. A menu-driven interface controls their configuration and synchronization.

The screen display program is a tool based on the HP Starbase graphics package. It accepts commands for the layout of one or more screens and converts data in the range from 0 to 999 into pointer movements relative to a scale. It follows an inquire-configure-add-delete-modify metaphor for hierarchical control of the layout by screen, window, and display unit elements. The screen element determines on which terminals to display the data, so the data can be displayed on several graphics terminals of different types. A screen can have several windows. The window element groups the display units. The window itself and any part of it can be enlarged or shrunk and moved. Display units consist of one or more of the same device type. The user can choose from an analog meter, a vertical bar, and a horizontal bar showing variations over time. Each device type consists of at least a base, a label drawing area, a label, a scale, a pointer drawing area, and a number of pointers. The label will be supplied via the data stream. The scales can be configured to their beginning and ending values and the number of ticks. The pointers can be configured to their number, colors, sizes, and shapes.

Data read is distributed over the display elements and their associated pointers in the order specified.

Diagram Generating Program. The purpose of the diagram program is to provide a permanent record of the data generated during a test run. The diagram generator processes columnar ASCII data, producing output suitable for the pic preprocessor. The output can be previewed on bit-mapped screens using the X-Windows package and then printed on any supported laser or HP LaserJet printer.

The user can control the header layout of each page and can place up to six diagrams on a page. Each diagram consists of a title, a horizontal axis, and a vertical axis and can be sized. Both axes' starting and ending values and number of ticks can be specified explicitly or generated automatically. There are three pens, each distinguished by a different line type (solid, dotted, dashed), and each pen can be given a label. If requested, for each pen, the minimum, maximum, and average values and the standard deviation will be determined.

Fig. 5 shows a sample diagram displaying data for six load measurement areas collected during a thirty-minute run of a disc I/O test.

Tabular Writing Program. Data reduction in conjunction with statistical methods is used to spot problem areas of the units under test quickly. Columnar data is processed using user-specified formulas and then formatted. The results are previewed on a standard terminal and then printed out. Fig. 6 shows a sample printout.

Conclusion

The Kernel Load and Measurement System allows us to develop a systematic approach to testing. For each test in our test package, we are compiling a profile of the stress load generated by the test in various load areas. In testing future releases of the HP-UX kernel, we will specify load levels that we will generate with the load system for each test cycle. We will also log the stress load generated by the test package as a whole during a test run for post-run analysis.

We feel that this system can be used as a load generating and load measurement tool when testing and measuring not only the kernel but also subsystems and applications running on the HP-UX system.

Acknowledgments

Acknowledgments go to all of the contributors to the Kernel Load and Measurement System. Jonathan Bozarth, Stanley Tsu, and Anders Wernblom are on the development team for the load system. William Horrix, Soren Stammers, and Mark Bergman contributed to the measurement system. Special thanks to Randy Menna for the meter display graphics library.

Process Measures to Improve R&D Scheduling Accuracy

Improvement is possible if scheduling is regarded as a process subject to continuous measurement.

by Richard M. LeVitt

PRODUCT DEVELOPMENT TEAMS at Hewlett-Packard typically commit to a project completion date at the conclusion of requirements definition, just before implementation. Meeting the commitment date has been a perennial challenge, particularly for software engineering teams. Last year, the Roseville Networks Division R&D laboratory began a major campaign to improve the accuracy of project schedules. Key to the effort is a view of scheduling as an ongoing process subject to continuous, objective measurement.

Roseville Networks Division is a member of HP's Information Networks Group, which supplies networking products for technical and commercial computer systems. At any time, approximately two dozen R&D projects are in progress at RND. Each project may suffer one or more unanticipated setbacks during the course of development. Although the particular events affecting a single project cannot be predicted, the cumulative effect of all sources of project delay can be measured and used to improve the accuracy of future schedule estimates. Since conventional measures of scheduling accuracy, such as DeMarco's estimation quality factor, or EQF,[1] are not suitable for real-time measurement of a set of project schedules, we developed two new process measurements that provide sensitive and timely indexes of project progress.

One of these measures is a modified form of DeMarco's EQF. By recording schedule adjustments on a month-by-month basis, we are able to make an accurate running estimate of the aggregate EQF of all lab projects. Another measure is project progress rate, which provides an ongoing prediction of the average actual project durations normalized by the average of the estimates. These metrics provide rapid feedback to schedule estimators on the accuracy of their project plans. Since we began the measurements a year ago, the effective (projected) EQF of the RND lab has improved by a factor of three. Our average lab-phase schedule duration overrun has been reduced from 70% to 20%.

This paper describes the derivation of these metrics and their application in the Roseville Networks Division lab.

Process Perspectives

Like other HP entities, RND uses formal software and hardware life cycles to define the internal process steps required for new product development. The most important milestones of the process are the investigation-to-lab (I-L) checkpoint and the manufacturing release (MR) checkpoint.

I-L marks the transition from the requirements definition and project planning phase to the implementation (lab) phase. At this checkpoint, the project manager is expected to have prepared a detailed project plan with dates for the intermediate milestones and for the MR milestone. This is the time at which the formal commitment is made by the project team to deliver on a particular date.

MR is the official end of the development effort, when all lab engineering and beta test work are complete. The product is suitable for customer shipments at this time.

Of the approximately two dozen projects active at any given time at RND, somewhat less than half are in the investigation phase and the remainder are in the lab phase. This is a large enough number that the projects can be considered collectively from a process perspective. Process measurements of scheduling accuracy can be applied and process improvements identified that benefit the entire organization.

The central issue in R&D for delivering on commitments is the match between actual lab-phase durations and I-L estimates of duration. Historically, the ratio of actual project durations to the I-L estimates has been about 1.7 at RND. Our process objective is to improve the ratio to 1.1 or better.

Process Measurements

Techniques in common use within HP for project track-

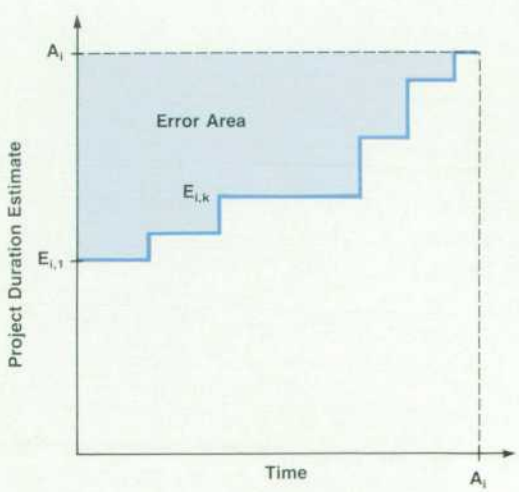


Fig. 1. *Estimates of a project's duration versus time. A_i is the actual duration.*

ing work well when applied to individual projects. A Brunner diagram,* for example, gives a good picture of time and effort expenditures in comparison with estimates as a project progresses. But it is difficult to combine information from Brunner diagrams to get an overall sense of how well an organization is managing projects.

Likewise, DeMarco's EQF has been used to evaluate the quality of schedule and effort estimates for individual projects. DeMarco suggests that project EQFs be averaged to obtain an overall value to represent an organization. Unfortunately, the average can be made useless by the presence of one well-planned project with very high EQF. EQF has a second drawback as a process measurement. The metric is normally calculated upon project completion, so it does not provide rapid feedback to aid improvement efforts while projects are in progress.

These considerations prompted the Roseville Network Division to develop new metrics to monitor the scheduling performance of the lab. For the past year, we have applied the process EQF (PEQF) and project progress rate measures to all active lab-phase projects with good results. Although our focus has been on tracking project calendar time, the metrics could be applied equally well to tracking engineering effort.

A feature of the RND lab environment has made data collection for the metrics relatively simple. Each month, all project managers are required to enter schedule updates in a central data base for status reporting purposes. The data base currently contains information covering over four years of lab history. Data from the status reports is entered once a month into a spreadsheet for the production of the metric graphs.

Process EQF

Process EQF (PEQF) is a modification of DeMarco's estimation quality factor specifically developed for application to a set of ongoing projects rather than to individual projects. PEQF provides a sensitive real-time index of scheduling accuracy that is free of the potentially misleading effects of EQF averaging. DeMarco defines EQF as a ratio of areas on a chart of estimates versus time maintained for a project. Referring to Fig. 1, let $E_{i,1}$ be the first or I-L estimate of, for example, L-phase project duration for project i. $E_{i,k}$ is the estimate for the kth interval (of a total n equal intervals) during the project. A_i is the corresponding actual duration recorded for project i on completion.

EQF for project i is then:

$$EQF_i = \frac{\text{total area}}{\text{error area}} = \frac{A_i^2}{\frac{A_i}{n}\sum_{k=1}^{n}|(A_i - E_{i,k})|}.$$

Consider the EQF for scheduling of a project i that experiences a constant rate of slippage. $E_{i,1}$ and A_i are the estimated and actual lab-phase durations, respectively. The history of this special case is diagrammed in Fig. 2.

The average rate of slip $\bar{d}_i$ is just the slope of a line drawn from $(0,E_{i,1})$ to (A_i, A_i). This line forms a right triangle with the vertical axis, and it is clear that the error area can be represented by the triangle area. Therefore,

$$EQF_i = \frac{\text{total area}}{\text{error area}} = \frac{A_i^2}{\frac{1}{2}A_i|A_i - E_{i,1}|}.$$

Substituting the slope $\bar{d}_i = \frac{A_i - E_{i,1}}{A_i}$ in the above equation we get

$$EQF_i = \frac{A_i^2}{\frac{1}{2}A_i|(\bar{d}_i \times A_i)|} = \frac{2}{|\bar{d}_i|}.$$

Note that when the slip rate is constant we do not need to know the initial estimate $E_{i,1}$ or the actual duration A_i to calculate EQF.

Definition of PEQF

Consider now a set of projects collectively slipping at

*Named after its HP originator.

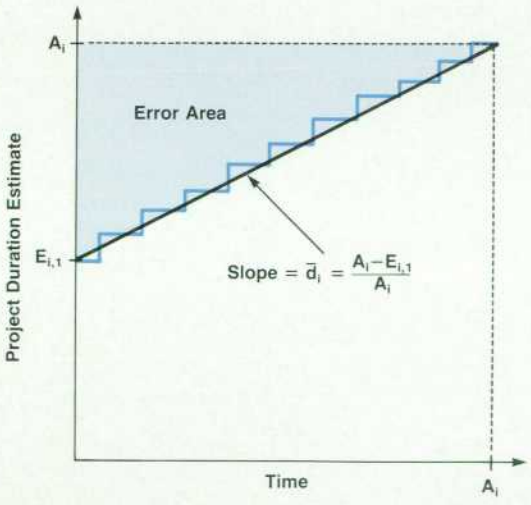


Fig. 2. *Duration estimates for a project with a constant rate of slippage.*

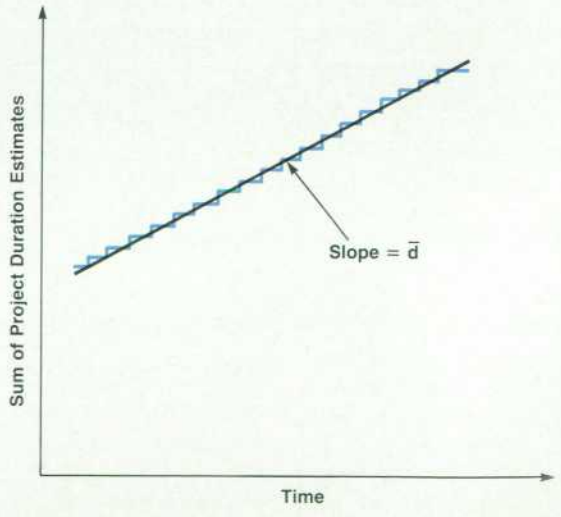


Fig. 3. *A set of projects slipping at an average rate $\bar{d}$, which is approximately constant.*

an average rate $\bar{d}$, where the average rate of slip is approximately constant. This situation is diagrammed in Fig. 3.

In analogy with the case described above, we define PEQF as

$$\text{PEQF} = \frac{2}{|\bar{d}|},$$

$$\text{where } \bar{d} = \frac{\text{cumulative slip for all projects in the set}}{\text{cumulative elapsed project time}}.$$

Fig. 4 illustrates the general case in which the average rate of slip varies slowly with time. A projection of process EQF can be made at each calendar interval k using the average slip rate at k. Thus,

$$\text{PEQF}_k = \frac{2}{|\bar{d}_k|},$$

$$\text{where } \bar{d}_k = \frac{\text{sum of project slip times in interval k}}{\text{sum of project elapsed times in interval k}}.$$

As shown in Fig. 4, the projection is made with a tangent line to the curve at k. Observe that the tangent line forms a right triangle within the box in the diagram. It can be seen that the PEQF projection has a geometric interpretation as a ratio of areas, which is similar to the original definition of EQF.

PEQF Results

Fig. 5 is a plot of actual values of cumulative slip versus cumulative elapsed project time for L-phase projects in the Roseville Networks Division lab. The curve includes nearly four years' project history at RND and is indeed slowly varying like the curve of Fig. 4. The interval k in this plot is one month.

Fig. 6 is a plot of historical PEQF_k values for the RND lab. For this diagram the interval has been increased to 3 months to reduce the effect of random fluctuations in the timing of schedule adjustments. This chart has been available to lab management each month since May 1986. Before that, the average PEQF was about 4.0. Since then PEQF has improved substantially, with one recent peak value exceeding 22.

Project Progress Rate

Although PEQF is useful, it, like the original EQF, is an abstract number having no simple or direct connection to schedule performance in management's terms. Management is less concerned with a ratio of areas than with the idea of completing a project on the commitment date.

RND has found it instructive to watch the ratio of the actual durations of projects to the I-L estimates of duration. A two-month slip represents a much larger scheduling error for a four-month project than for an 18-month project. Normalizing the actual durations by the estimates allows projects of differing lengths to be compared directly on a percentage basis.

This ratio is also helpful to evaluate the scheduling accuracy of a set of projects. Fig. 7 is a scattergram of estimated project durations versus actual durations. The slope of a best-fit straight line through the origin and the data points is given by

$$m = \frac{\sum A_i}{\sum E_i}$$

where A_i is the actual duration of the ith project and E_i is the estimate. (Note to statisticians: Although this is not a least squares line, it is an unbiased estimator. The variance is somewhat larger than with least squares, but a test on RND project data shows the difference to be insignificant.)

Unfortunately, m is only available after a number of projects have been completed and hence is slow to respond to process improvements. What is needed is a real-time predictor of m that is sensitive to process changes as they occur. Project progress rate is just such a predictor.

The estimator of m we require can be developed as follows:

$$m = \frac{\sum A_i}{\sum E_i} = \frac{\sum A_i}{\sum (A_i - \Delta_i)} = \frac{1}{1 - [\sum \Delta_i / \sum A_i]}$$

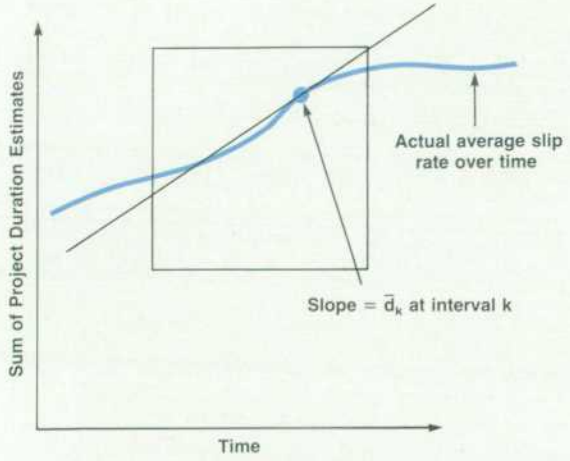


Fig. 4. *The general case of a slowly varying average slip rate.*

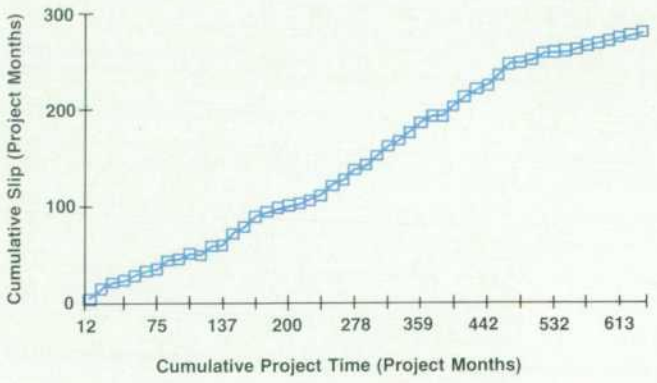


Fig. 5. *Actual values of cumulative slip versus cumulative elapsed project time for lab-phase projects at RND. Interval is one month.*

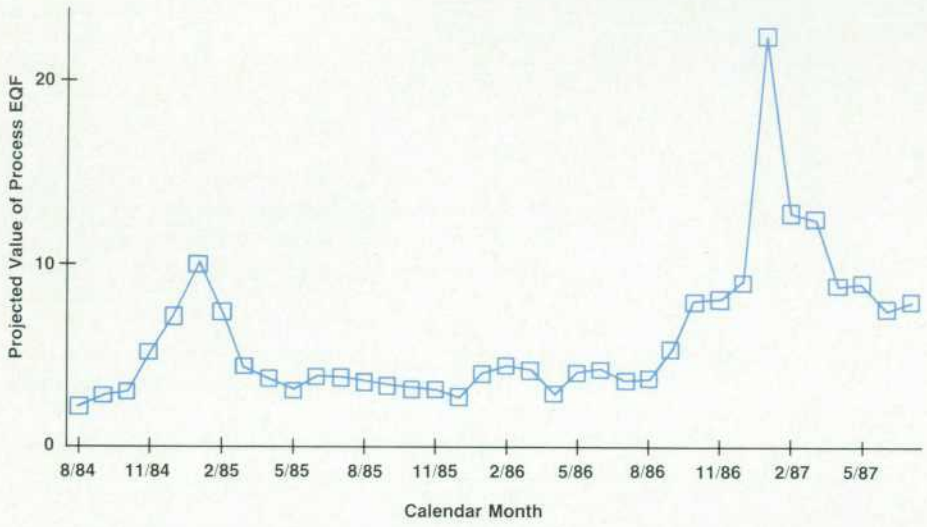


Fig. 6. *Process estimation quality factor values for the RND lab. Interval is three months.*

where Δ_i is the total slip of project i: $\Delta_i = A_i - E_i$.

To estimate m at time interval k we need an estimate of $\sum\Delta_i / \sum A_i$ at k. Let

$$\overline{d}_k = \frac{\sum_i \Delta_{i,k}}{\sum_i A_{i,k}} = \frac{\text{sum of project slip times in interval k}}{\text{sum of project elapsed times in interval k}}$$

Substituting $\overline{d}_k$ in the expression for m above, we define the project progress rate at interval k as

$$r_k = 1 - \overline{d}_k \approx 1/m.$$

Project progress rate is inversely related to m rather than directly for psychological reasons. We prefer a metric that increases when things get better. "Perfect" on this scale is a value of 1.0 sustained over time. Lower project progress rate values correspond to higher projected values of m and hence longer average schedule overruns.

Project Progress Rate Results

An m value of 1.7 was calculated in mid-1986 using RND historical data from 21 completed projects. For comparison, the project progress rate metric was calculated for the same set of projects. The two results agree remarkably well; the discrepancy is just 1%. This is a strong empirical confirmation of the relationship between overall scheduling accuracy (expressed as a ratio of actual and estimated durations) and the project progress rate metric. The metric has increased through the year and now forecasts an m value of about 1.2 for currently active L-phase projects. If this performance is sustained, RND will deliver the next generation of products with an average of 20% or less schedule overrun.

Fig. 8 is a chart of the project progress rate metric as it is used at RND. The chart is updated monthly to show I-phase projects (reflecting slips of the promised I-L dates), L-phase projects (reflecting slips in the MR dates), and the combination of the two. To reduce fluctuations, the interval size for metric calculation is three months. A lab with a smaller number of active projects may need to select a longer interval to ensure a clean and readable graph.

Even with three-month intervals, there is noise in the RND graph. Fig. 9 shows the L-phase data using a 12-month interval. This chart reveals a long-term trend of declining scheduling accuracy which was not apparent in the previous graph. A dramatic reversal in this trend began at the time we made project progress rate a key management tool in the effort to improve scheduling accuracy in R&D.

Conclusions

This paper has outlined the derivation and application of two process metrics that monitor overall scheduling accuracy in an R&D organization. The two metrics are quite similar; they both report average scheduling performance across many projects in a timely manner. The similarity has a deeper origin, however. Both the process EQF and the project progress rate measures are based on the average slip rate, $\overline{d}_k$. Thus the choice between PEQF and project progress rate is primarily a matter of style. PEQF is recommended for organizations that have internalized EQF as a

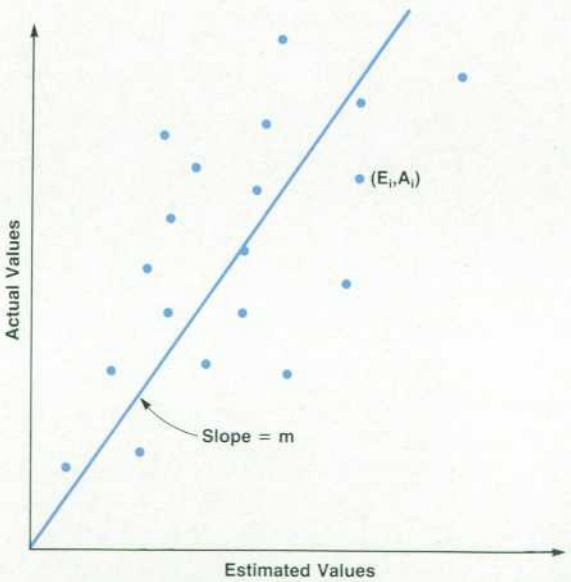


Fig. 7. *Scattergram of estimated versus actual project durations with a best-fit straight line through the origin.*

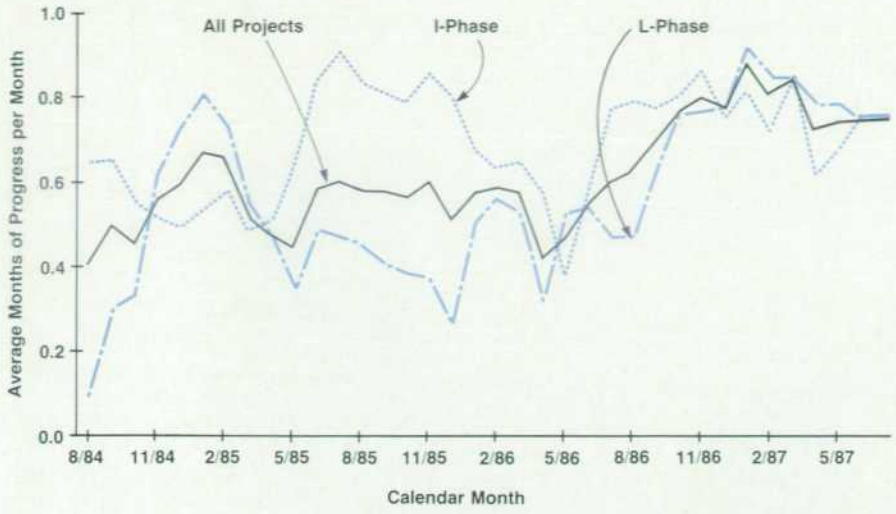


Fig. 8. *Project progress rate for the RND lab. Interval is three months.*

project estimation measure. Like EQF, PEQF tends to magnify small differences as scheduling perfection is approached. Values nearing infinity are theoretically possible. By contrast, project progress rate has a more uniform scale with values approaching 1.0 as accuracy improves. Project progess rate has the additional advantage of forecasting the average percentage schedule overrun for the organization.

Project progress rate is the primary schedule management metric at RND. The metric is quite scalable, and could in principle be used at a division lab level, at a group level, or across an entire corporation.

Although the topic of this paper is the metrics themselves, the discussion would be incomplete without a mention of the actions that have brought about substantial scheduling improvements at RND. The metrics have made scheduling accuracy much more visible to management. Management attention and emphasis on accuracy have caused project planners to devote more effort and care to scheduling, with beneficial results. Project managers have received training in scheduling and estimation. A personal-computer-based project management program has been introduced and is now in use in the RND lab. To ensure that project managers benefit from each other's experience, we have begun to conduct peer reviews of proposed schedules. A task force of project managers has been formed to discuss common causes of schedule error and identify corrective measures.

Although our overall schedule accuracy has improved, we feel that further gains are desirable. We have begun to understand that the significant sources of delay include unforeseen tasks, changing project objectives, coordination with outside groups, and shortages of tools. Our next challenge is to reduce or eliminate these sources of delay and promote overall gains in productivity. The measures of scheduling accuracy described here will provide rapid feedback on the effectiveness of these improvements as they are made.

Acknowledgments

The author gratefully acknowledges the assistance of Gary Harmon and Rita Lasick in the preparation of this paper.

Reference

1. T. DeMarco, *Controlling Software Projects*, Yourdon Press, 1982.

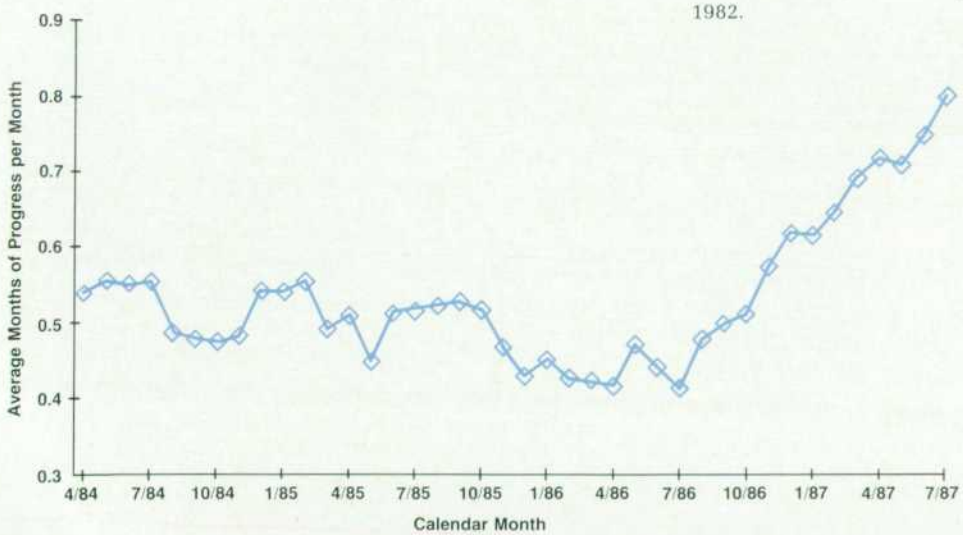


Fig. 9. *L-phase project progress rate using a 12-month interval.*

Authors

April 1988

6 MM-Wave Sources/Instruments

John R. Regazzi

John Regazzi studied electrical engineering at Rutgers (BSEE 1976) and Lehigh University (MSEE 1979). He then joined HP and has worked on plug-ins for the HP 8350 Sweep Oscillator Family and the HP 8553x Millimeter-Wave Source Modules. He is interested in production testing of active microwave components and computers and outside of work has served the past two years as a board member and fund raiser for the Santa Rosa Little League. Born in Madison, New Jersey, John now lives in Santa Rosa, California with his wife and two children.

Mohamed M. Sayed

A native of Cairo, Egypt, Mohamed Sayed attended Cairo University, earning a BSEE degree in 1964 and an MSEE degree in 1968. He continued his electrical engineering studies at Johns Hopkins University and was awarded a doctorate degree in 1972. After some postdoctoral work on solar cells at the University of Delaware, he joined HP's Stanford Park Division in 1973. Currently project manager for the HP 8349B Microwave Amplifier and the HP 8355x and HP 85100V/W Source Modules, Mohamed also has contributed to the development of the HP 70907A External Mixer Interface Module, the HP 71300A Millimeter Spectrum Analyzer, the HP 5350A Microwave Frequency Counter, the HP 5356A/B/C Frequency Converter Heads, and the HP 5355A Frequency Converter. He has written 15 articles related to his work and is a member of the IEEE. Outside of HP, he is active in the YMCA's child/parent program, coaches soccer, and enjoys traveling with his family. He is married, has two children, and lives in Santa Rosa, California.

12 MM-Wave Vector Network Analysis

Robert G. Dildine

Bob Dildine has been with HP since 1973. Before coming to HP, he worked as a field engineer for Bechtel Corporation and served four years as a lieutenant in the U.S. Navy Civil Engineer Corps. Bob worked on the development of the HP 8672 Microwave Synthesizer and the HP 8510 Network Analyzer and managed projects for the HP 8510 Test Sets and the HP 85102 IF and detectors for the HP 8510. He has written one previous article for the HP Journal and his work has resulted in two patents related to the synchronous detector and IF system in the HP 8510. Bob earned a BS degree at San Diego State College in 1967 and an MS degree at the University of California at Berkeley in 1973, both in electrical engineering. He is married, has a daughter, and lives in Santa Rosa, California. Outside of work, he enjoys amateur radio (W6SFH), particularly building equipment for the VHF and microwave bands.

James D. Grace

Born in Cameron, Missouri, Jim Grace served four years in the U.S. Navy before joining HP Associates in 1966. He initially worked in production engineering for switch components. He helped develop the HP 8408A Network Analyzer and contributed to the verification and other software for the HP 8510A/B systems. Jim is married, has two children, lives in Santa Rosa, California, and enjoys military air shows and rebuilding old cars.

18 MM-Wave Source Modules

Robert D. Albin

Dale Albin began work at HP in 1977 as a device test engineer for microwave transistors. Since then he has supported the fabrication of GaAs FETs, designed power amplifiers and integrated multipliers for millimeter-wave sources, and managed the millimeter-wave source module project. He currently is a project manager concerned with the development of optical fiber instrumentation. He is the coauthor of an HP Symposium paper on millimeter-wave sources, and his work has resulted in two pending patents related to millimeter-wave multipliers and finline technology. Dale was born in Dallas, Texas and studied electrical engineering at the University of Texas at Arlington (BSEE 1977) and Stanford University (MSEE 1980). He now lives in Santa Rosa, California and has many interests, including running, skiing, weight training, bowhunting, restoring antique cars, reading, and private aviation.

26 High-Power Microwave Source

Alan R. Bloom

Al Bloom received a BA degree in physics from Wesleyan University in 1972 and an MS degree from Rensselaer Polytechnic Institute in 1976. After a few years designing two-way radios for R.L. Drake Company, he came to HP in 1979. Interested in analog/digital design, he contributed to the development of the HP 83550A RF Plug-In. Al was born in Gettysburg, Pennsylvania, and now lives in Santa Rosa, California where he is building a solar heating system for his home and enjoys amateur radio (N1AL).

Ronald T. Yamada

Ron Yamada joined HP in 1985 after receiving a BS degree in electrical engineering and computer science in 1984 from the University of California at Berkeley. Currently working on firmware development for scalar network analyzers, he contributed to the firmware for the HP 83550A RF Plug-In and the HP 8350 Sweep Oscillator family. Born in Los Angeles, California, he now lives in Santa Rosa, California and enjoys golf, playing guitar, and watching movies.

Kenneth A. Richter

A native of Brooklyn, New York, Ken Richter studied electrical engineering at Drexel University (BSEE 1967) and designed electronic systems for satellites for a few years before joining HP in 1973. Currently a project manager in the Network Measurements Division, he has contributed to the HP 8757 Scalar Analyzer family, the HP 83550 RF Plug-In, and the HP 8753 RF Vector Analyzer family. Ken has written several articles on devices and circuit design, one for a previous issue of the HP Journal, and his work has resulted in one patent related to ac/dc detection using the HP 8757, and three patent applications. He lives in Santa Rosa, California, is married, and has three active sons. Ken enjoys coaching youth athletics, driving race cars, creative cooking, and long walks.

Roger R. Graeber

Roger Graeber worked as a technician for Varian Associates for seven years before joining HP in 1972. He earned an AA degree in electronics from the College of Marin in 1965 and a BS degree in electrical engineering from the University of California at Davis in 1982. Roger was a test technician for the HP 8500A System Console, HP 8542B Network Analyzer, and HP 8580A Spectrum Analyzer. He worked on the HP 436A Power Meter, the HP 8662A Signal Generator, and most recently, the HP 8349B Microwave Amplifier and the HP 83550A RF Plug-In. Roger also has been system supervisor for the HP 8542B and a service engineer for the HP 8510 System. Born in Oakland, California, he now lives some miles away in Sebastopol with his wife, son, and daughter. Outside of work, Roger is an avid scale model train enthusiast, from N gauge up to 1.5-inch scale, which is big enough to ride on. He also spends a fair amount of time helping his wife landscape their home and restoring old 1956 DeSoto automobiles (he has ten, four that run).

Andrew N. Smith

Interested in electromagnetism and low-cost designs, Andy Smith is a graduate of California Polytechnic State University at San Luis Obispo (BSME 1982). With HP since 1982, he worked on the mechanical production of the HP 83550A RF Plug-In and is currently working on parts for an HP scalar network analyzer system. Born in Oakland, California, he now lives not too far away in Forestville. Andy enjoys playing softball and general socializing when not involved in seemingly never-ending home improvements and constant car maintenance.

31 MM-Wave Detectors

Herbert L. Upham

An R&D project manager for scalar network analyzer accessories in the Network Measurements Division, Herb Upham has been with HP since 1972. He has contributed to the development of various directional couplers and a fiber-optic power sensor and the HP 11664D Detector, worked on the production of the HP 8505 Analyzer, and most recently, has project responsibility for accessories for HP's scalar network analyzer. He wrote an article on the HP 11664D for an earlier HP Journal issue, has a pending patent on a waveguide millimeter-wave detector, and is a member of the IEEE. Outside of work, Herb serves as a lieutenant in the U.S. Coast Guard Reserve, is the donations manager for the Sonoma County Junior Symphony, enjoys skiing and amateur radio (WD6FMG), and teaches flying (he owns a Cessna 182). Born in Red Bluff, California, he studied electrical engineering at the University of California at Berkeley (BSEE 1972) and Stanford University (MSEE 1976). He now lives in Sebastopol, California with his wife, son, and daughter.

35 MM-Wave Power Sensors

Lee H. Colby

After serving four years in the U.S. Air Force, Lee Colby joined HP in 1961 as a technician in the standards laboratory. After that he worked in production engineering for a few years and then came to the R&D lab where he designed the HP 8485A and HP 8486A Power Sensors and contributed to the development of the HP 8900C/D Peak Power Meter and the HP 8970B and HP 8971B Noise Measuring Systems. He is a member of the IEEE and in 1982 earned a BSEE degree from San Jose State University. Born in Seattle, Washington, he now lives in Sunnyvale, California. Lee is married and has a grown son. His outside interests include fly fishing, camping, backpacking, and amateur radio (WB6AMN). He also supports a volunteer fish hatchery and helps arrange for scholarships for fishery management students at Humboldt State University.

39 UNIX Logon Adaptations

Marvin L. Watkins

A California native, Marv Watkins holds a BA degree in psychology and an MS degree in mathematics from San Jose State University (1975 and 1978). He also studied computer science at Ohio State University and received his MS in 1979. With HP since 1986, his major contribution has been a UNIX-system-based workcell controller for a surface-mount manufacturing line. His professional experience before coming to HP was varied. Among other accomplishments, he developed data base managers and a graphics spooler and verified real-time software and firmware for an electronic flight instruments system. He also served in the U.S. Army. He's author or coauthor of two articles on software testing and computer displays and an earlier article for the HP Journal on the UNIX IPC system. Born in Santa Barbara, Marv is now a resident of Los Gatos. In addition to his interest in software engineering, he enjoys jazz, science fiction, house plants, and exploring the backcountry.

48 Virtual User Utility

Kjell A. Olsson

Author's biography appears elsewhere in this section.

Mark Bergman

Mark Bergman first joined HP as a student in 1981. He became a full-time employee in 1983, after receiving his BS degree in electrical engineering and computer science from the University of California at Berkeley. Since 1986, he's been a consultant to HP's Information Technology Group, concerned with various aspects of vuser, the virtual user simulation utility. He's a member of the ACM and is interested professionally in graphics, humanizing computer interfaces, system integration, and compilers. Mark was born in Fort Worth, Texas. A jazz dancer, he's preparing for a new show that opens in June. He also enjoys woodworking and studying physics.

54 HP-UX Kernel Load

Kjell A. Olsson

Kjell Olsson was born in Sweden and graduated from Upsala University with a Master's degree in physics and computer science in 1979. His first involvement with HP was his master project, which he did at HP in Stockholm in 1978. He joined HP's Data Systems Division in 1980 as a software engineer, returning to Stockholm as a systems engineer a year later. In 1984 he rejoined the Data Systems Division, testing HP-UX on HP Precision Architecture. Moving to the Information Software Division as project manager for HP-UX kernel reliability and tools, he was involved in the development of the HP-UX Kernel Load and Measurement System and the vuser utility. He's the author of several papers on test automation, tools, and measurement. Away from HP, Kjell is a skier and a racquetball player and likes to cook, specializing in Swedish, French, and American cuisine.

Grace T. Yee

Grace Yee is a project manager with HP's Information Software Division. She joined HP in 1980 after receiving her BS degree in computer science from the University of California at Berkeley. She has developed financial and manufacturing software both as an engineer and as a project manager, and is currently a project manager at the HP-UX Test Technology Center. She is married, has a daughter, and enjoys jogging and cross-country skiing.

61 R&D Scheduling Accuracy

Richard M. LeVitt

As quality assurance manager at HP's Roseville Networks Division, Dick LeVitt's responsibilities include developing tools and methods for more effective R&D. Before assuming his present position, he served in other HP divisions as a reliability engineering manager and as project manager for the HP 2393A and 2397A Graphics Terminals. A native of Placerville, California, he received his BSEE degree in 1969 from the University of California at Berkeley. Before joining HP in 1981, he was an R&D project manager and section manager with Bently Nevada Corporation, developing data acquisition and control systems. He's a member of the AAAS and his work has resulted in one patent on an analog flow meter. Dick is married and has a major avocation in fine-art photography, which he has studied with Ansel Adams, Jerry Uelsmann, Olivia Parker, and others.

69 Arbitrary Waveform Synthesizer

Roland Hassun

Author's biography appears elsewhere in this section.

Albert W. Kovalick

Author's biography appears elsewhere in this section.

78 125-MHz 12-Bit ADC

Wilfredo T. Sagun

Willy Sagun is a specialist in analog circuit design at HP's Stanford Park Division. He received his BSEL degree from California Polytechnic State University at San Luis Obispo in 1976 and joined HP's Delcon Division (now Colorado Telecommunications Divi-

sion) the same year. In 1981 he received his MSEE degree from the University of California at Berkeley. Willy has contributed to the design of the HP 4935A Transmission Impairment Measuring Set and the HP 8770A Arbitrary Waveform Synthesizer. Born in Manila, he's a skier, a ballet fan, and a sailboat racer (J/24s). He also confesses to eating "massive" amounts of chocolate.

Thomas Hornak

Tom Hornak graduated from the Bratislava Slovak Technical University, Czechoslovakia, in 1947, receiving a Dipl. Ing. in electrical engineering. In 1966 he earned a PhD degree from the Czech Technical University in Prague. From 1947 to 1961 he worked at the Tesla Corporation's Radiotechnical Research Laboratory, and between 1961 and 1968 he was head advisor for electronics R&D at the Computer Research Institute in Prague. In 1968 he joined HP Laboratories and now heads the High-Speed Electronics Department of HP Laboratories. His department is responsible for R&D on high-speed data communication links, high-speed analog/digital interfaces, and electronic instrumentation using advanced Si and GaAs IC processes. Tom is a Fellow of the IEEE. He was a guest editor in 1978 and is presently associate editor of the IEEE Journal of Solid-State Circuits. From 1979 to 1981 he was chairman of the IEEE Solid-State Circuits and Technology Committee, and from 1978 to 1982 a member of the IEEE International Solid-State Circuits Conference program committee. He has authored 49 professional papers and his work has resulted in 36 patents. He is married, has a son, and enjoys reading, music, and hiking.

Gary L. Baldwin

As director of the High-Speed Devices Laboratory of HP Laboratories, Gary Baldwin manages the research effort in III-V materials, devices, and ICs. Previously, as project manager and department head, he led design projects for NMOS and GaAs 12-bit digital-to-analog converters, including the first prototype of the HQMOS DAC for the HP 8770A Arbitrary Waveform Synthesizer. A native of El Centro, California, he studied electrical engineering at the University of California at Berkeley, receiving BSEE, MSEE, and PhD degrees in 1965, 1966, and 1970. Before joining HP Laboratories in 1978, he was a member of the technical staff of Bell Telephone Laboratories for eight years. He has authored 15 papers on precision ICs and his work has resulted in six patents. He is a Fellow of the IEEE and serves as president of the IEEE Solid-State Circuits Council. He has two sons and is an avid woodworker and cabinetmaker.

Fred H. Ives

Fred Ives is project manager for the HP 8904A Multifunction Synthesizer at HP's Spokane Division. Previously, he was project manager for the DAC/sampler IC and microcircuit for the HP 8770A Arbitrary Waveform Synthesizer. A specialist in digital synthesis and RF, analog, and phase-locked loop design, he received his BSEE and MSEE degrees from the Massachusetts Institute of Technology and joined HP's Stanford Park Division in 1972. He has also been responsible for high-frequency phase-locked loop design for the HP 8662A Synthesized Signal Generator, and his work has resulted in two patents. Fred was born in Margaretville, New York. He is married and enjoys boating, outdoor activities, and landscaping.

86 Arbitrary Waveform Applications

Albert W. Kovalick

Al Kovalick is a project leader with HP's Stanford Park Division, specializing in digital synthesis and digital signal processing. A native of San Francisco, he received his BSEE degree from California State University at San Jose in 1972 and his MSEE degree from the University of California at Berkeley in 1974. With HP since 1974, he has contributed to the hardware design of printing calculators and to the firmware and power supply design of the HP 8662A Synthesized Signal Generator. More recently, he contributed to the architecture, digital, and software design of the HP 8770A Arbitrary Waveform Synthesizer. His work has resulted in six patents in the areas of printer technology and digital synthesis architectures. Al is married (his wife has also been an HP Journal author) and is active in studying and teaching Bible topics. He enjoys sailing, reading, travel, and oriental studies.

Roland Hassun

A project manager with HP's Stanford Park Division, Rolly Hassun has been with HP since 1961. He has worked on the HP 8660A Synthesized Signal Generator and other products, and served as project manager for the HP 8662A Synthesized Signal Generator and the HP 8770A Arbitrary Waveform Synthesizer. Specializing in signal generation and processing for radar and magnetic recording, he has authored ten papers on noise and frequency synthesis, and his work has resulted in two patents on digital frequency synthesis. He's a member of the IEEE. Rolly received his BSEE degree from the Politecnico di Milano in 1960 and his MSEE in 1963 from California State University at San Jose. He is married, enjoys tennis, reading, and history, and collects 15th century Spanish ballads.

94 Waveform Generation Language

Rafael F. Miranda

Rafael Miranda joined HP's Stanford Park Division in 1984 after 12 years in hardware and software design at the NASA Ames Research Center. He has designed HP-IB firmware for the HP 8980A Vector Analyzer and worked on the graphics, math, and waveform generation portions of the Waveform Generation Language. Born in San Juan, Puerto Rico, he received his BSEE degree in 1973 from Pratt Institute and his MSEE in 1976 from Stanford University. He's the author of a 1986 IEEE paper on reading aids for the blind. Rafael is married, has two children, and enjoys racquetball, running, bicycling, and other outdoor activities.

Peter A. Thysell

A software development engineer with HP's Stanford Park Division, Pete Thysell has served as product marketing engineer for the HP 11776A Waveform Generation Software and the HP 8770A Arbitrary Waveform Synthesizer and has developed applications software for the 11776A/8770A system. He's a member of the IEEE and a native of Walnut Creek, California. After serving in the U.S. Coast Guard for four years, he attended California State University at Sacramento and received a BS degree in electrical and electronic engineering in 1983. He joined HP the same year. Pete is married and has one child. His hobbies are woodworking and electronics.

Derrick T. Kikuchi

Derrick Kikuchi is a specialist in digital signal processing and real-time software design with HP's Stanford Park Division. Raised in Santa Barbara, California, he attended the University of California at Santa Barbara, graduating in 1978 with a BS degree in electrical engineering and computer science. In 1981 he received his MSEE degree from Stanford University. With HP since 1978, he has designed software and firmware for the HP 8903A Audio Analyzer, the HP 8955A RF Test System, and the HP 8673 Signal Generator and served as project manager for HP 8980A Vector Analyzer HP-IB firmware and the HP 11776A Waveform Generation Software. Derrick plays digital music synthesizers, likes backpacking, and volunteers his time to provide emotional support for people dealing with death and dying.

An Arbitrary Waveform Synthesizer for DC to 50 MHz

Precision, flexibility, and repeatability of signals are ensured by a digital architecture. Two or more synthesizers can be synchronized to provide several sources of complex signals with an identical time reference.

by Roland Hassun and Albert W. Kovalick

ANALOG SIGNAL SYNTHESIZERS have been used for years in communications, radar, and general-purpose testing. The output of these synthesizers is usually a carrier with simple amplitude, frequency, or phase modulation. This is adequate for many applications, but more complex waveforms are needed for others. These more complex waveforms are difficult or impossible to generate with currently available commercial equipment. This has led to the development of the HP 8770A Arbitrary Waveform Synthesizer. The HP 8770A has a digital architecture and an analog output of ±1 volt into 50 ohms, from dc to 50 MHz.

An example of a complex waveform that is easily synthesized using the HP 8770A is a simulated radar return pulse with all the distortions, overshoots, and noise that occur in actual operation. Another example of an HP 8770A application is as a calibration source for magnetic media certifiers. A test signal with missing bits, extra bits, or amplitude modulation can be precisely provided. For TV applications, the HP 8770A can be used to add distortion or extra modulation to TV test signals, or to try new scrambling and transmission formats. Many other applications have been demonstrated.

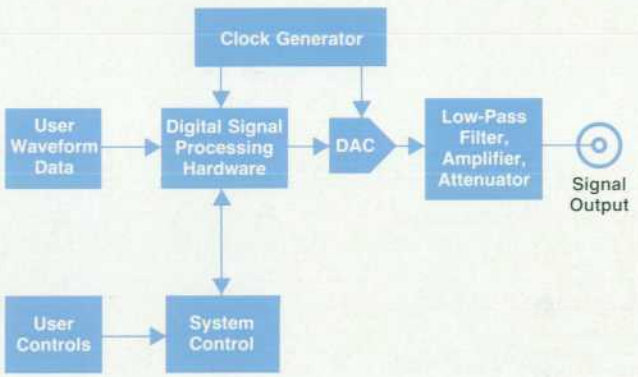


Fig. 1. *Generic architecture of a waveform synthesizer that has a digital architecture and an analog output.*

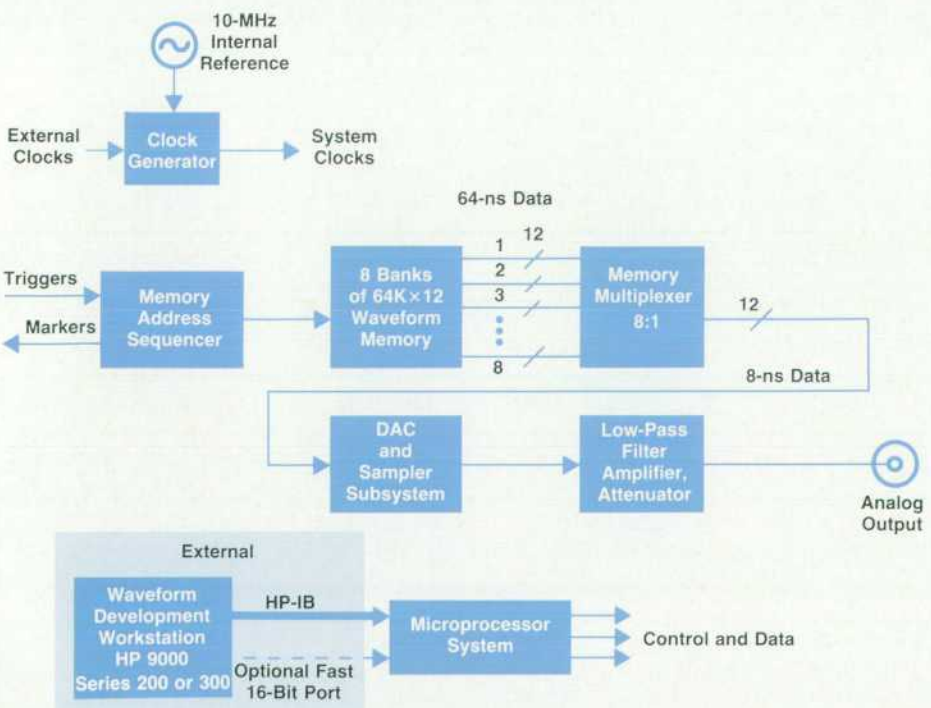


Fig. 2. *HP 8770A architecture. The desired waveforms are created using the HP 11776A Waveform Generation Language (WGL) with an HP 9000 Series 200 or 300 computer workstation.*

Internal to the HP 8770A, digital signal processing hardware provides a well-defined stream of words to a digital-to-analog converter (DAC). The DAC output is filtered and amplified to provide the desired output signal. Fig. 1 shows a generic architecture of such a synthesizer.

An example of a product based on such an architecture is the compact disc (CD) player. The CD provides the data defining the waveforms (music). The digital processing hardware reads and massages the data. The DAC (16 bits in a commercial player), and a low-pass filter convert the sampled data to analog values. Even the listener with the golden ear can't tell that the music was once in digital form.

The advantages derived from the architecture chosen for the HP 8770A are the precision, flexibility, and repeatability inherent in a digital process. For example, complex modulation formats like simultaneous AM, FM, and PM, or QAM, MSK (minimum-shift keying), or AM stereo are relatively easy to produce. Also, precision control over amplitude (±0.002 dB), and phase (±0.01 degree) are provided. Generally, any waveform can be generated, provided it is of finite length and band-limited. This flexibility enables almost unlimited applications of the HP 8770A. (Two applications are discussed in the article on page 86.)

Although many analog synthesizers have a frequency range greater than dc to 50 MHz, few actually use all the available bandwidth for any given application. Generally, a carrier frequency is chosen and modulated. The bandwidth is usually less than 50 MHz, although it may be centered around a much higher frequency. If the output of the HP 8770A is upconverted, then the 50-MHz bandwidth is usable at any desired center frequency.

HP 8770A Architecture

Fig. 2 shows the block diagram of the HP 8770A. An external computer workstation (HP 9000 Series 200 or 300), running the HP Waveform Generation Language (WGL), creates the desired final waveform. This waveform is quantized into 4096 amplitude sample points and downloaded, via the HP-IB (IEEE 488/IEC 625), to the HP 8770A memory system. The memory is organized into 512K 12-bit words and has an equivalent access time of 8 ns. The RAM now contains an image of the desired output waveform as represented by amplitude samples uniformly spaced in time. The waveform memory is composed of 8 groups of relatively slow 64K-by-12-bit CMOS RAMs. Using ECL multiplexing, the groups are converted into a 12-bit, 8-ns data stream that sources the DAC.

Memory sequencing extends the effective waveform RAM size by repeating often-used wave patterns (see "Address Sequencer," page 72). To create a complete NTSC horizontal color bar test pattern waveform, we used only 25K of waveform memory and sequenced through the RAM to form the final output. The final wave was stored in RAM as selected wave fields. A compression factor of 150:1 was achieved over a brute-force waveform lookup method.

The key to the spectral purity and high bandwidth achieved by the HP 8770A is the DAC subsystem. It must convert a 12-bit sample point every 8 ns to the appropriate analog value.

All DACs of conventional design have an inherent output glitch whenever the input data changes. The glitch is usually different for each data input value. This results in a spectrum salted with unwanted signals. Some of them may be only 25 dB below the desired signal.

Our solution was to design a gallium arsenide (GaAs) sampler that "deglitches" the unwanted portion of the DAC output (see "A 125-MHz 12-Bit Digital-to-Analog Converter System," page 78). The glitch occurs at the DAC output every 8 ns, and has a duration less than 4 ns. The GaAs sampler immediately follows the DAC, and removes the unwanted glitch during the first 4 ns of the sample period. Once the glitch is removed, the sampler output settles to the final amplitude value during the remaining 4 ns of the sample period.

The sampler output is ideally the desired analog wave sampled and half-held. Fig. 3 shows such an example. Its frequency spectrum is shown in Fig. 4. Half-hold sampling has a frequency response that follows a $[\sin(\pi f/2f_{clock}]/(\pi f/2f_{clock})$ envelope. A major advantage of half-hold sam-

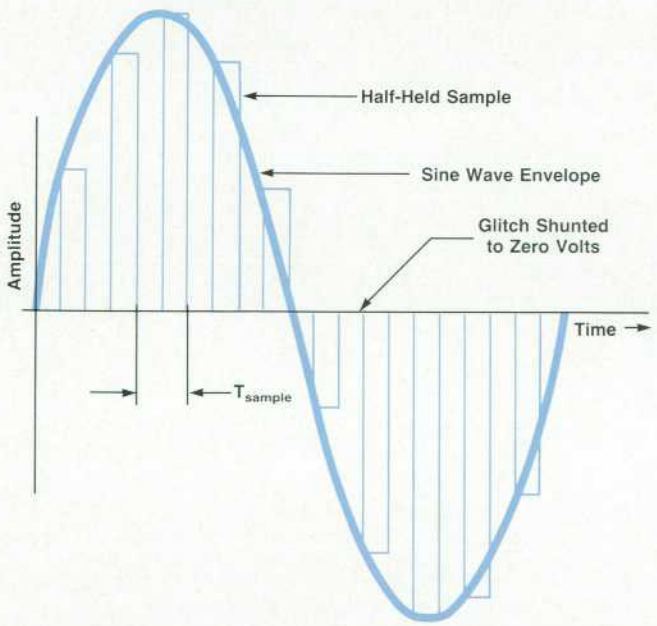


Fig. 3. *Output of desired analog wave sampled and half-held. The inherent output glitch from the DAC output is removed by the GaAs sampler during the first 4 ns of the 8 ns sampling period.*

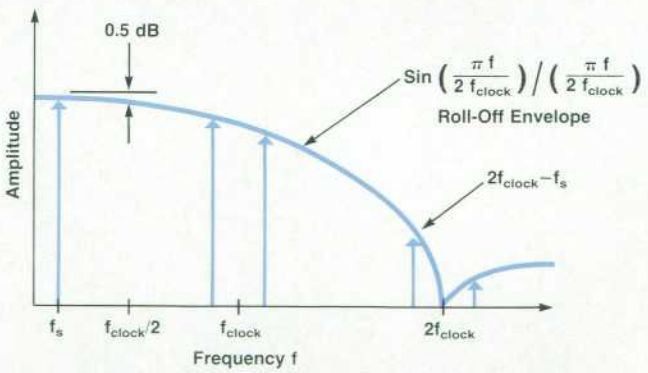


Fig. 4. *Frequency spectrum of sampled wave shown in Fig. 3.*

pling over full-hold sampling can be seen in the passband roll-off characteristics. For half-hold, the passband roll-off is only down 0.5 dB at $f_{clock}/2$. For normal full-hold, it is down 3.92 dB at the same frequency. This slower passband droop lessens the burden of flattening the passband through a peaking circuit.

If the original signal frequency f_s is less than $f_{clock}/2$, the original signal can theoretically be recovered without any error except for a scale factor adjustment. Recovery is accomplished by simply filtering the sampler output with a low-pass filter. Viewing the reconstruction from the time domain, the low-pass filter acts as a perfect interpolator, connecting the samples together to form a smooth analog signal.

Another important component in the block diagram is the clock generator. Since the output analog signal is derived directly from the clock, it is essential that the clock generator output is not contaminated by any impurities such as jitter or nonharmonic signals. This assures a clean output spectrum. The clock frequency is 125 MHz, and the clock signal is locked to a 10-MHz internal reference (see "Sampling Clock Requirements," page 76).

The synthesizer is controlled by a Motorola 68000 microprocessor system (Fig. 2). More than 90 HP-IB commands allow a convenient user interface. All commands conform to the IEEE 488.2 format specification. The front panel has no keys except for the power switch. No combination of keys was found that could meet all our needs. Therefore, the user interface was put outside the HP 8770A into the WGL workstation.

The final components of the block diagram are the markers and triggers. Four markers are provided. Three are used to identify specific time points in the waveform memory. The other marker is user-programmable, and can mark any time point. Also present are two trigger inputs. One allows for advancing the memory sequencer to the next waveform. The other is used to start the memory lookup process, and to start two or more HP 8770As in a synchronized configuration.

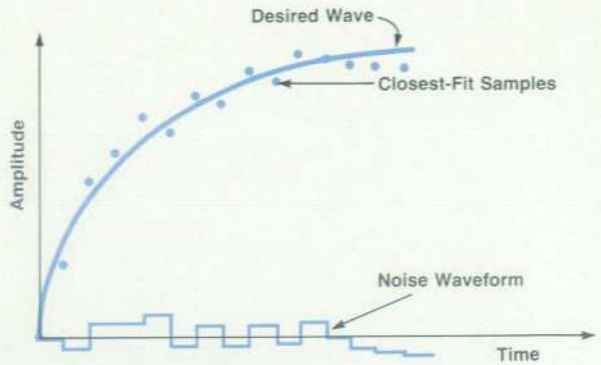


Fig. 5. *DAC quantization noise.*

Design Trade-Offs

It is a formidable task to steer through a 512K memory with a stop sign every 8 ns. A brute-force solution to the implementation could be designed using ECL logic, but this would be impractical. Instrument size and cost would increase. To overcome this objection, the data flow is partitioned into high-speed and low-speed paths. This enables the use of a CMOS waveform memory, a TTL memory sequencer, an ECL memory bank multiplexer, an NMOS DAC, a GaAs sampler, and TTL logic where needed.

The biggest trade-off in designing a digital synthesizer is the output bandwidth versus the number of bits in the DAC. These two parameters are always in opposition. For example, the 12-bit DAC must settle to 16 times the amplitude accuracy of an 8-bit DAC, but has 24 dB more dynamic range. Since one of our goals was to design and build a synthesizer with a dynamic range greater than 65 dB and a 50-MHz bandwidth, we chose a 12-bit DAC.

Another reason to choose a 12-bit system is the reduced quantization noise. Whenever the sampled signal is loaded into the HP 8770A it must be quantized to 12-bit resolution. The operation of quantizing causes a noise component to appear, appropriately called quantization noise. This noise has a maximum level of ±½ LSB in amplitude. Fig. 5 shows the desired wave, the closest-fit quantized wave, and the

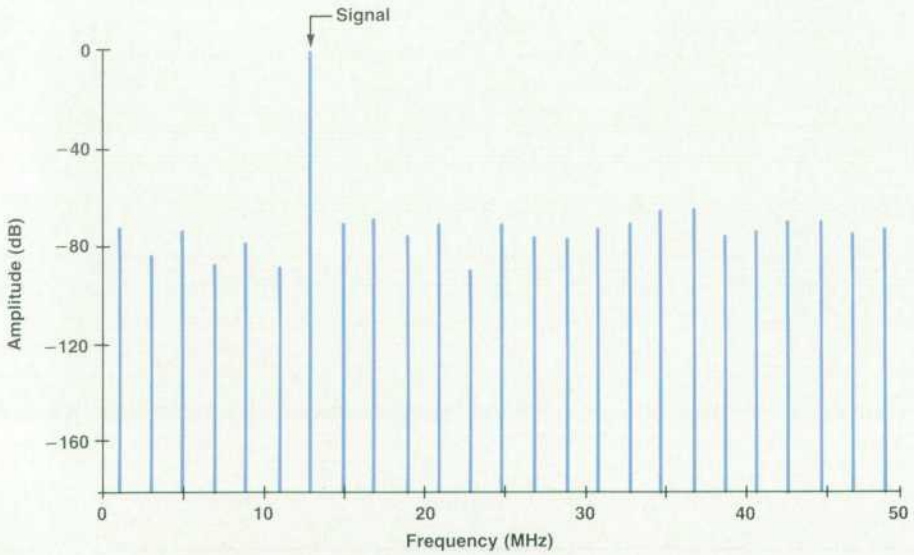


Fig. 6. *Signal-to-noise ratio for a signal quantized to 8-bit accuracy. The total noise power of all the noise lines is 53 dB below the desired signal.*

Address Sequencer

Many times it is desired to create unusually shaped waveforms made up of smaller periodic waveforms. An example is a television scan pattern containing horizontal and vertical sync pulses. This type of waveform can be created by storing data representing the desired waveforms in RAM, and then feeding this data into a high-speed DAC. By repeating a series of incrementing addresses over and over again, going through one group of addresses several times, then another set of addresses, and perhaps a third and a fourth, many unusual and useful waveforms can be created.

The addressing tasks in the HP 8770A are accomplished by the address sequencer. The sequencer is a complex state machine that has the capability of generating addresses to look up RAM data in an order that corresponds to the time-varying amplitude of a desired waveform. It will automatically look up the packet information (see Glossary) for each successive packet,

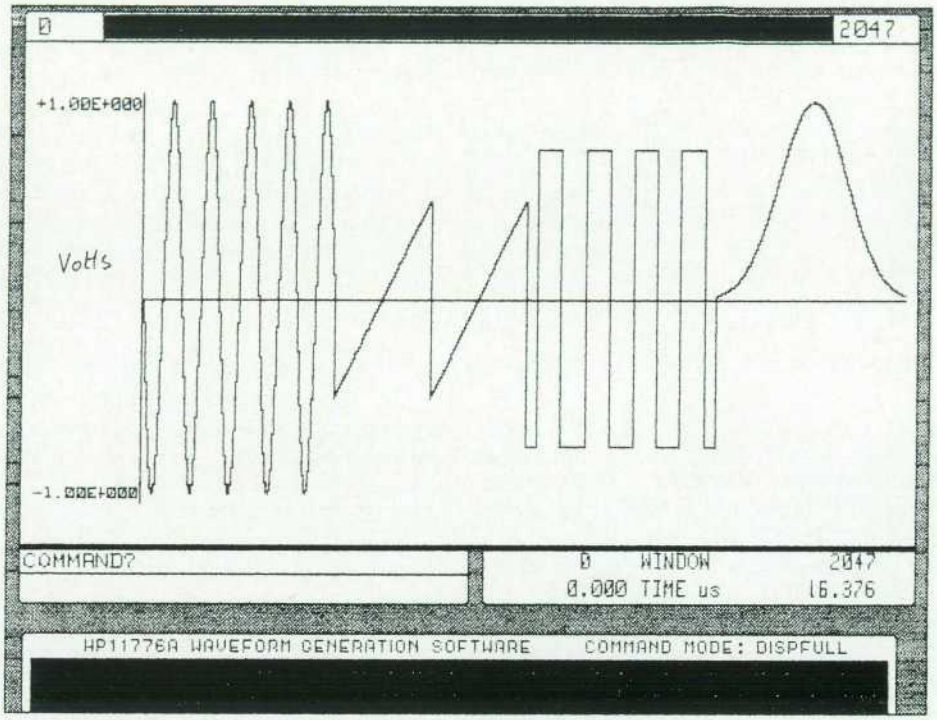


Fig. 1. *Waveform generated by programmed sequencing through groups of data representing the desired waveform. The data is stored in RAM. The address sequencer generates addresses to look up data in an order that corresponds to the time-varying amplitude of the desired waveform.*

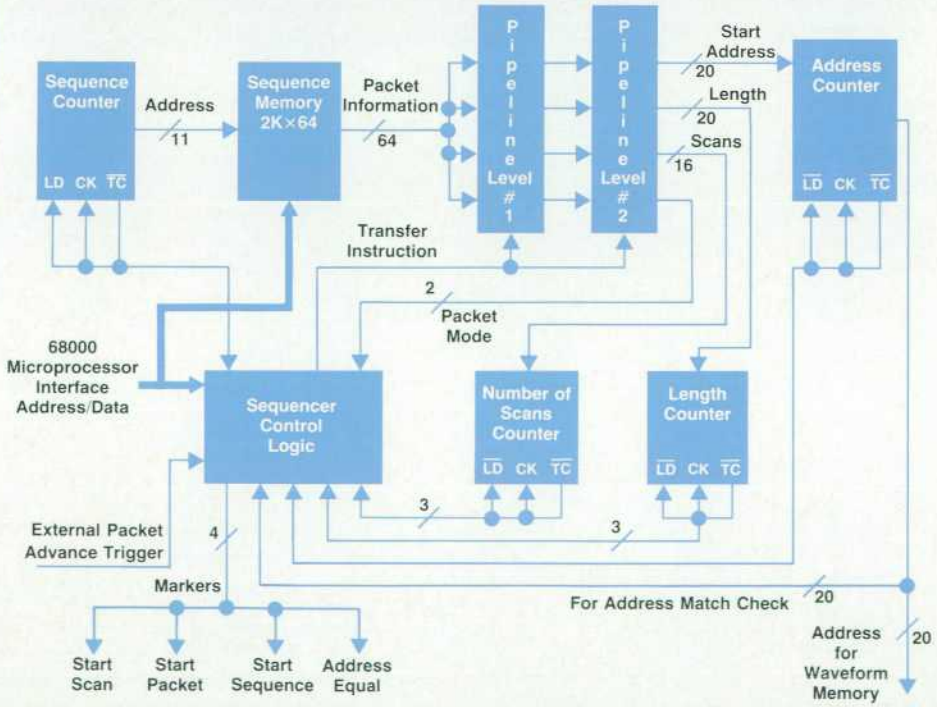


Fig. 2. *Address sequencer block diagram.*

and begin to work on it without losing a single cycle of the 64-ns (15.625-MHz) clock. Fig. 1 shows an example of a waveform generated from the data stored in RAM. Fig. 2 is the sequencer block diagram.

Sequencer operation begins after the desired data is loaded into the waveform memory. The sequence memory is then loaded with the packet information for the sequence to be run. Using a pipeline register in the sequencer that is two levels deep, storage of 64 bits of packet information for the first and second packets is forced to occur under microprocessor control. The sequencer then starts running at full speed. When the sequencer logic senses that the currently running packet is on its last scan (scan counter = 0), and its third-to-last address (length counter = 0), it is ready to transfer the next packet information to the output of the pipeline registers at the input to the scan, length, and address counters. This is done on the next clock (length counter = −1). On the following cycle all three counters are loaded with the new packet information, effectively (length counter = −2) just before the load. The sequence control logic starts to read new packet information out of the sequencer RAM and load it into the first level of the pipeline. Because this new load cycle takes several clock cycles, the minimum number of clock cycles per packet is 43. This means that the number of scans multiplied by the packet length must be greater than 43 for any given packet. Finally, after the specified number of packets in the sequence have been cycled, the sequencer returns to the first packet. This process continues until the sequencer is stopped.

As a convenience to the user, four TTL markers are externally available. These represent scan start, packet start, sequence start, and address equal. The address for the address equal marker is set by the user before running the sequencer. Each marker produces a pulse when its condition is met. Advancing to a new packet can also be triggered by the microprocessor or an external TTL trigger.

With its large waveform memory accessing ability, high clock rate, and on-the-fly packet information loading, the sequencer is useful for many complex signal generation applications.

Matt Klein
Development Engineer
Stanford Park Division

Glossary of Address Sequencer Terms

Waveform RAM. 512K of high-speed RAM in eight groups of 64K 12-bit words.
Scan. A pass through a group of addresses in waveform RAM.
Length. Number of waveform RAM addresses representing a scan (max. = 512K).
Start Address. Beginning waveform RAM address (0 to 512K − 1).
Number of Scans. The number of times to go through the group of waveform RAM addresses specified by the length address and start address (max. = 64K).
Packet. The combination of number of scans, length of address, and start address required to create a desired waveform from data stored in RAM memory. Each packet has a defined number of scans, length, and start address (max. number of packets = 2K).
Packet Mode. The way a packet is exited. This is either by exhausting the desired number of scans in internal mode, being triggered from an external TTL signal, or being caused to advance via an HP-IB command. These three modes are INT, EXT, and BUS respectively.
Sequence Memory. 2K 64-bit words of RAM that have instructions for the number of scans, start address, length, and packet mode for each packet.
Sequence. A group of 1 to 2048 packets.

resultant noise waveform. Assuming that the noise wave has a uniform probability distribution function (any value between ±½ LSB is equally likely), the signal-to-quantization noise ratio can be calculated. For an N-bit DAC, the signal-to-quantization noise ratio is 6N + 1.8 dB. If a low-pass filter is used to remove half the noise power, the ratio is 6N + 4.8 dB.

For comparison, let's look at the signal-to-noise ratio for 8-bit and 12-bit DAC systems. The desired signal is a sine wave at about 13 MHz. In Fig. 6 the signal is shown quantized to 8-bit accuracy, and in Fig. 7 it is quantized to 12 bits. Both figures show a comb of lines below the desired signal. For the 8-bit case, the total power in all the noise lines is 53 dB below the signal. For the 12-bit system, the total power is 77 dB below the signal. It is important to note that the sum of all the noise lines is the noise power. No one line provides a large contribution to the power. The noise lines are closely spaced and wideband. Since the undesirable noise power their sum produces is greatly reduced by using a 12-bit DAC system, a 12-bit DAC system was chosen for the HP 8770A.

The signal-to-noise ratio discussed here is the ideal case. In practice, additional noise is always present in the form of nonharmonically related signals. These unwanted signals are products arising from GaAs sampler mixer terms, and from some glitch energy skirting the sampler.

Amplitude Resolution

The 12-bit resolution of the DAC provides 4096 different amplitude states that the signal can occupy at any 8-ns sampling point. The effect of the low-pass filter after the DAC is to connect all these points smoothly. Consequently, worst-case amplitude resolution is 1/4096 of the full-scale value of the signal, or about 0.024%. In the case of the HP 8770A, the full-scale value is 2V peak-to-peak at the output into a 50Ω load. This gives a quantization level of 0.5 millivolt. Both these numbers are doubled when operating into a high-impedance load.

The stability and relative accuracy of the states offer unusual amplitude modulation performance. Accuracy, speed, depth, and linearity of the modulation are exceptional compared to analog implementations. Amplitude modulation linearity better than 0.1% is achievable.

Phase Resolution

Creating a signal by providing samples every 8 ns might lead one to believe that timing resolution is also 8 ns. In fact, the 12-bit resolution in amplitude allows the creation of linear ramps separated by a single DAC state. This corresponds to a shift in time with a resolution of (V_{max}/4096/slope of ramp). Thus, a ramp with a repetition rate of 1 μs can be placed in time with a resolution of (V_{max}/4096)/($V_{max}/10^{-6}$) = 244 ps. Similarly, sine

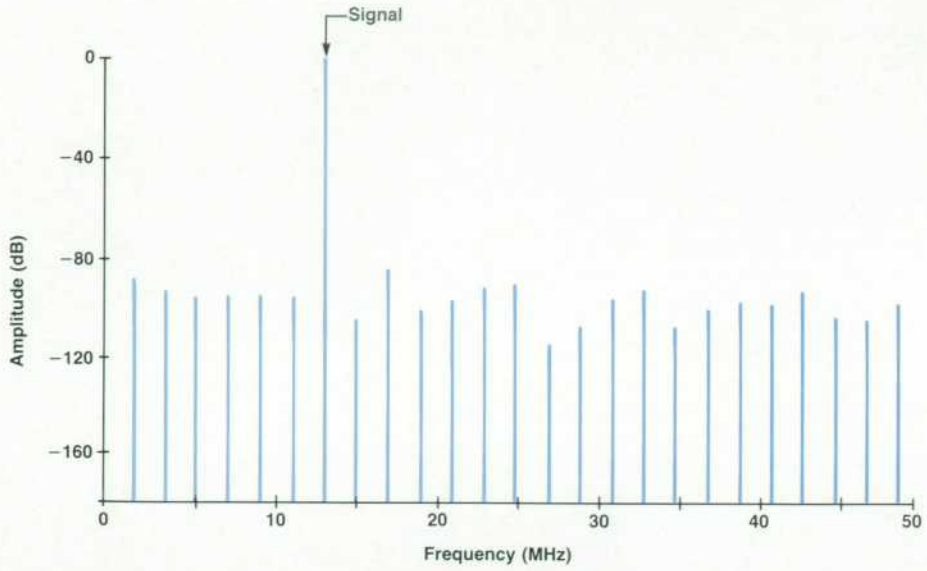


Fig. 7. *Signal-to-noise ratio for a signal quantized to 12-bit accuracy. The total noise power of all the noise lines is 77 dB below the desired signal (24 dB lower than when signal is quantized to 8-bit accuracy).*

waves can be shifted in phase by an amount $\pi/4096$ radians.

Fig. 8 shows the spectrum of a 10-MHz carrier modulated sinusoidally in phase at a 125-kHz rate with a peak deviation of 1 milliradian. Note that the sidebands are 66 dB below the carrier, as predicted by a Bessel function expansion for the spectral lines.

Frequency Resolution

The frequency of a single tone (sine wave) is determined by the total number of points Q used to construct the tone and the total number of cycles P in that period. Both P and Q are integers.

The frequency of the tone, f, is given by P/QT, where T is the sampling interval. T is also the reciprocal of the sampling frequency. The frequency resolution is:

$$\frac{df}{f} = \left(\frac{\partial f}{\partial P}dP + \frac{\partial f}{\partial Q}dQ\right)\frac{1}{f}$$

If Q is kept constant (i.e., $dQ = 0$), then since $dP_{min} = 1$, $df_{min} = 1/QT$. For $Q = 10{,}000$ and $T = 8$ ns, $df_{min} = 12.5$ kHz. If $P = 1 =$ constant (i.e., $dP = 0$), and Q is increased from 10,000 to 10,008, the frequency change df is about 10 Hz for a frequency tone of 12.5 kHz.

Harmonic Distortion

Fig. 9 is a plot of the harmonic distortion of the HP 8770A as a function of frequency. The low harmonic distortion

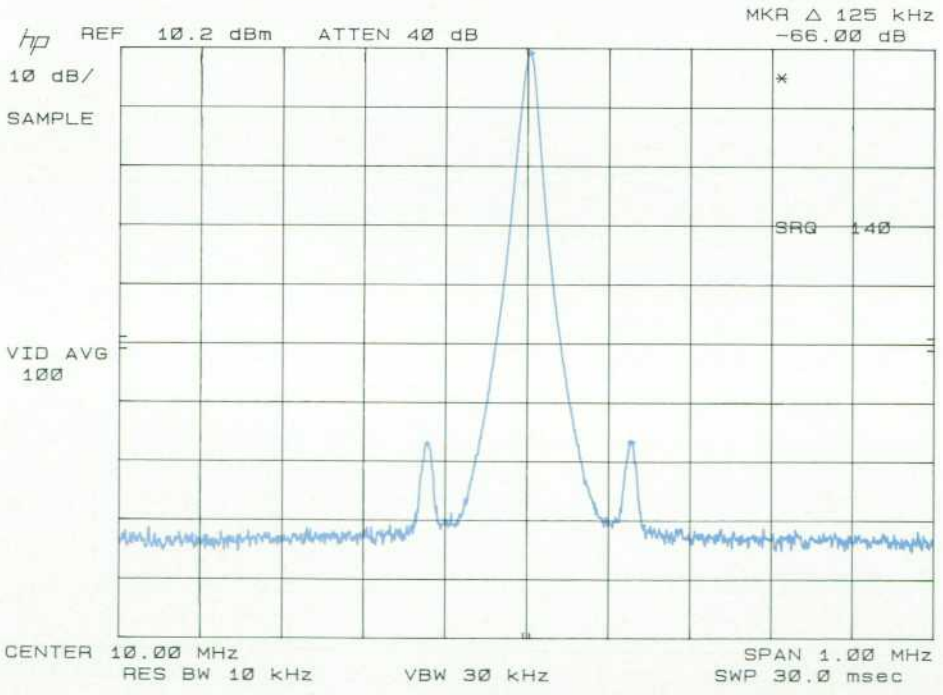


Fig. 8. *Spectrum of a 10-MHz carrier modulated sinusoidally in phase at a 125-kHz rate with a peak deviation of 1 milliradian. The sidebands are 66 dB below the carrier.*

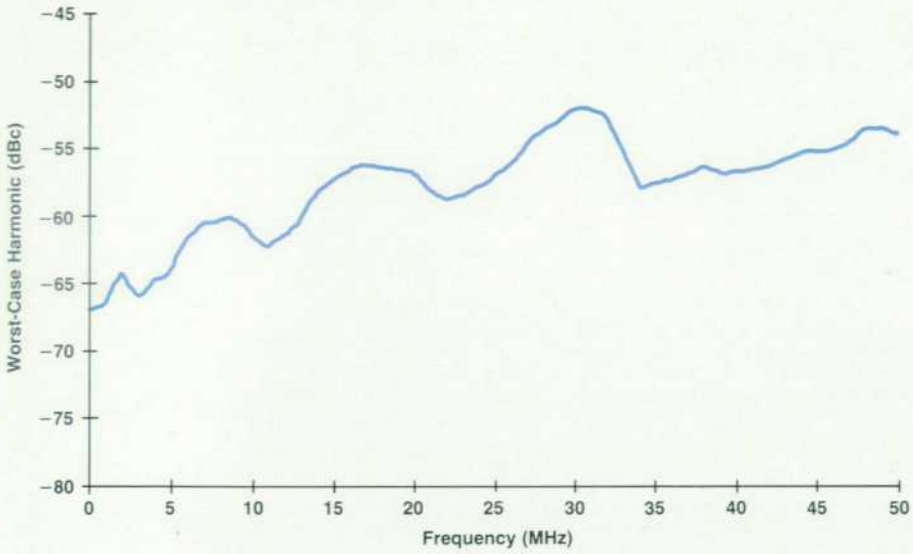


Fig. 9. *Harmonic distortion of the HP 8770A as a function of frequency.*

indicated is a measure of the accuracy with which a signal operating in a broadband environment can be generated.

The evaluation of wideband high-precision analog-to-digital converters, waveform recorders, and digital oscilloscopes benefits from signals generated with low harmonic distortion. Fig. 9 shows that the HP 8770A is capable of characterizing a 20-MHz, 9-bit analog-to-digital converter. Missing-bit detectors used in disc certifiers can be calibrated to better than 1%.

Spurious Sidebands on Sine Waves

Fig. 10 shows spurious sidebands on a sinusoidal output from the HP 8770A incremented in 1-MHz steps to 50 MHz, while searching for any spurious sidebands. The highest spurious level observed at each frequency is shown in dB.

Spurious sidebands set a limit to the dynamic range of the signal being created unless they happen to fall outside the bandwidth of the receiving equipment. For example, the emulation of multiple radar targets with the HP 8770A is limited to a dynamic range of about 65 dB by spurious signals. From a practical standpoint this is acceptable, since other factors present more restrictive limitations.

It is important to keep in mind that many digital-to-analog conversion techniques can give rise to spurious sidebands no better than 30 dB below the carrier. The spurious sidebands shown in Fig. 10 for the HP 8770A range from about 57 dB below the carrier to about 69 dB below the carrier.

Two-Tone Intermodulation Distortion

Two-tone intermodulation distortion is a measure of the accuracy with which a signal operating in a narrow band can be generated. It represents the degree of odd-order nonlinearities in a network. Two-tone intermodulation distortion is often used to measure the ability of radar and communication receivers to operate error-free in the presence of large interfering signals.

Fig. 11 shows the two-tone intermodulation distortion of the HP 8770A. The two plots show the level of the undesired sidebands referred to the two tones. Each tone is +4 dBm at the output of the HP 8770A.

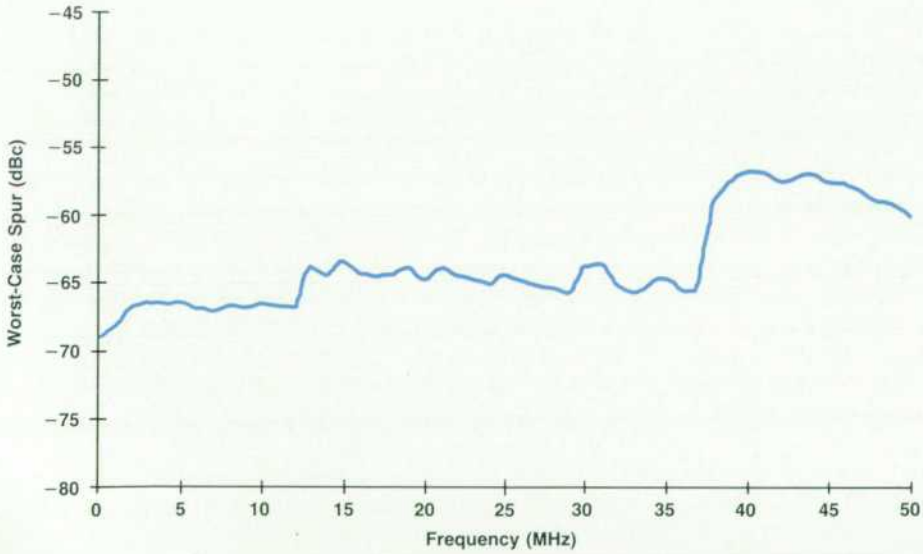


Fig. 10. *Spurious sidebands on a sinusoidal output observed while incrementing the frequency in 1-MHz steps. Spurious sidebands are well below the levels achieved with other digital-to-analog conversion techniques.*

Sampling Clock Requirements

The need for a clean sampling-clock spectrum can be seen by modeling the DAC and sampler as shown in Fig.1. Although the sampler is actually a switch, it functions as a multiplier or mixer. The multiplication of the DAC output by the sampling clock in the time domain is equivalent to the convolution of the sampling clock spectrum with the ideal deglitched DAC output spectrum.

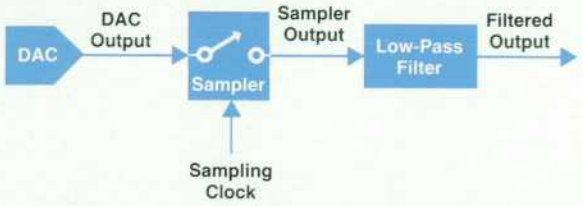


Fig. 1. *Model of DAC and sampler.*

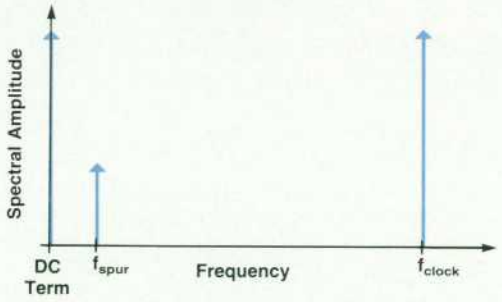


Fig. 2. *Example of undesired spurious sidebands from sampling clock.*

Assume that the sampling clock has some undesired spurious sidebands as shown in Fig. 2, and that the DAC is generating the samples of an ideal DAC waveform without glitching and settling. When this ideal DAC waveform is convolved with the sampling-clock spectrum, the low-pass-filtered spectrum shown in Fig. 3 results. Although the DAC output was ideal, undesired sidebands of the sampling clock appear around the carrier inside the passband. This indicates that the sampling clock must be kept at least as clean as the desired output spectrum.

Clock Distribution

The spectral purity of the sampling clock is maintained by the clock distribution method shown in Fig. 4. The clock generators produce clocks for the memory and sequencer at one eighth the sampling rate. These TTL clocks are a potential source of undesired clock/8-related sidebands. The sampling clock is kept free of these sidebands by the isolation buffer. Although distributing all the clocks from a single clock generator would produce lower skew between the sampling clock and the rest of the system, it would not produce the required spectral purity.

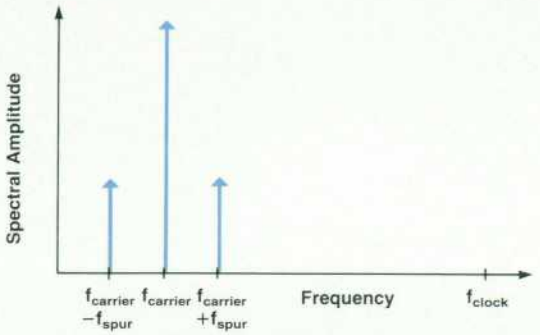


Fig. 3. *Low-pass-filtered spectrum that results when an ideal DAC waveform is convolved with the sampling-clock spectrum shown in Fig. 2. Although the DAC output was ideal, undesired sidebands of the sampling clock still appear around the carrier.*

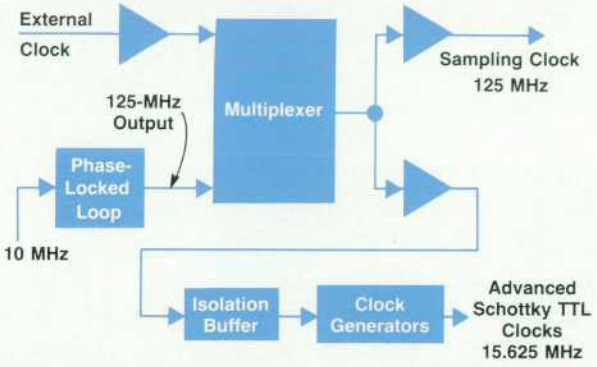


Fig. 4. *Clock distribution method to maintain spectral purity. The isolation buffer keeps the sampling clock free of any undesired sidebands from the TTL clocks.*

Douglas A. Larson
Development Engineer
Stanford Park Division

Amplitude Flatness and Phase Linearity

These two characteristics influence transient response, carrier modulation accuracy, and signal purity. They are of special importance in data communications and digital magnetic recording. These applications require very low bit error rates. Departure from amplitude flatness or phase linearity causes intersymbol interference. This leads to uncertainty in recovery of the data. In magnetic recording, simulated read signals are strongly affected by passband phase nonlinearity.

HP 8770A amplitude response is flat within ±0.65 dB (total broadband flatness including the attenuator). Phase linearity is within ±5 degrees.

Versatility

The depth of the main memory and the power of the sequencer affect the versatility of the HP 8770A waveform synthesizer. The purpose of the sequencer is to create relatively long waveforms with the available memory. These waveforms can be of unusual shape and complexity (see "Address Sequencer," page 72). Applications such as emulating radar emission signals and disc-drive servo signals make use of this capability.

Memory sequencing in the HP 8770A allows up to 2048 different packets. This translates to 1024 pulses of different shape, width, or height to form a complex waveform. Each specific pulse can be repeated within one packet up to

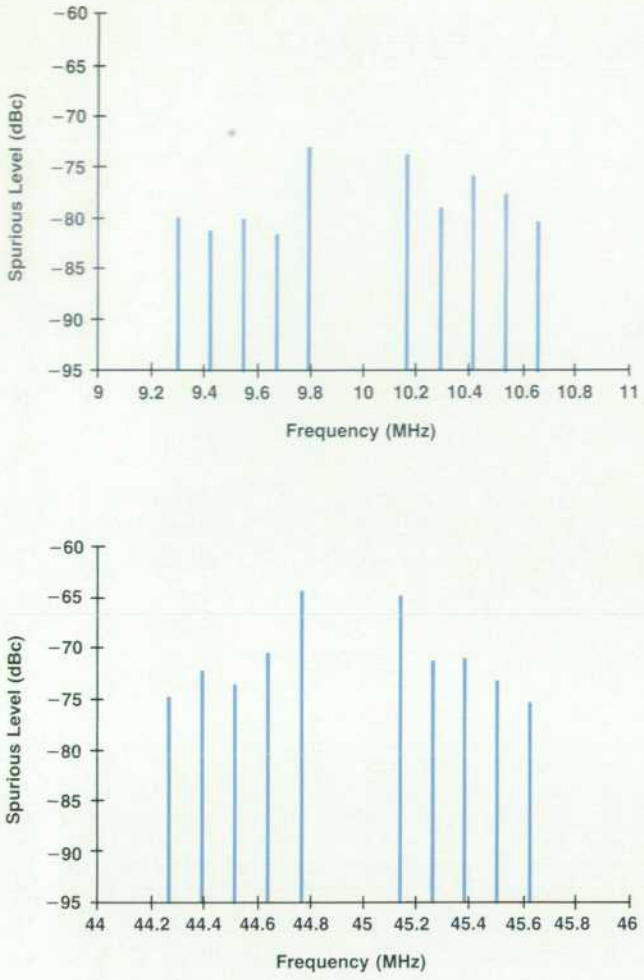


Fig. 11. *Two-tone intermodulation distortion. The two plots show the level of the undesired sidebands referred to two tones. Each tone is +4 dBm at the output of the HP 8770A. In the top plot, the tones were at 9.9 and 10.1 MHz. In the bottom plot they were at 44.9 and 45.1 MHz.*

65,536 times.

An elaborate profile for the repetition interval of a pulse can be defined to allow for staggered or even jittered repetition intervals. Similarly, typical waveforms have been created describing the servo signals encountered by servo circuits in disc drives as the read head slews across many tracks.

Synchronization

A very important feature of the HP 8770A is the ability to synchronize the operation of two or more units. Several sources of complex signals with an identical time reference can be realized. Time reference uniformity of 20 picoseconds or better can be achieved. This is especially useful in representing the in-phase (I) and quadrature (Q) components of signals used in radar and communication systems.

With two HP 8770As and an HP 8780A Vector Signal Generator,[1] wideband (100-MHz) modulation signals at carrier frequencies ranging from 10 to 3000 MHz can be created. This feature can be used to emulate the signals in radar transmitter sections or the reflected returns from single or multiple targets.

Acknowledgments

We would like to thank the following people for their contributions to the HP 8770A. Kudos to Mike Murphy for his contributions to the software and hardware. Thanks to Bob DeVries and Julie McDonald for the product design and to Chris Bennett, Greg Lowitz, Bruce Anderson, Robbie Armitano, Matt Klein, Ray Lew, Chris Muir, and Bob Stoltz for their production support. A tip of the hat also to Mike Pan for his hardware contribution. Marketing support was provided by Donn Mulder, Dave Martinez, Omid Kordestani and Pete Thysell. Finally, thanks to Ron Kmetovicz for unfailing support and encouragement.

Reference

1. D.R. Gildea and D.R. Chambers, "Vector Modulation in a Signal Generator," *Hewlett-Packard Journal*, Vol. 38, no. 11, December 1987, pp. 25-29.

A 125-MHz 12-Bit Digital-to-Analog Converter System

Advanced IC DAC technology and a system design approach were needed to achieve the performance of the HP 8770A Arbitrary Waveform Synthesizer.

by Wilfredo T. Sagun, Fred H. Ives, Gary L. Baldwin, and Thomas Hornak

THE FINAL HIGH-SPEED CONVERSION of digital data to the analog output signal in the HP 8770A Arbitrary Waveform Synthesizer is done in a custom ceramic microcircuit followed by a filter and amplifiers. The underlying design goal for these circuits was to achieve precision and repeatable waveform synthesis within a 50-MHz bandwidth. The analog system architecture consists of a 125-MHz, 12-bit, NMOS digital-to-analog converter (DAC), a gallium arsenide sampler, a differential-to-single-ended converter, a low-pass filter, and output amplifiers (Fig. 1).

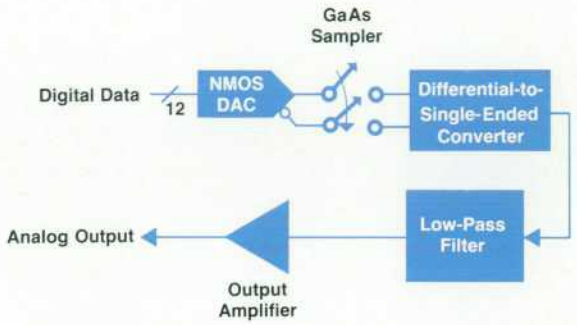


Fig. 1. *Basic elements of the output system of the HP 8770A Arbitrary Waveform Synthesizer.*

Accuracy Evaluation

An extensive amount of simulation was done on the proposed system. Analysis revealed that a sampled system with a 12-bit word size would provide sufficient accuracy for generating complex waveforms. The projected signal-to-error-signal ratio is better than 78 dB from dc to the Nyquist frequency.

The performance bounds of the ideal DAC were predicted using probabilistic models and direct computer simulations. A substantial portion of the work involved isolating and determining the distortion mechanisms of the DAC and sampler circuitry. The static linearity of the DAC was easily achieved by incorporating an R-2R ladder current-source circuit that is accurate to better than ¼ LSB. The

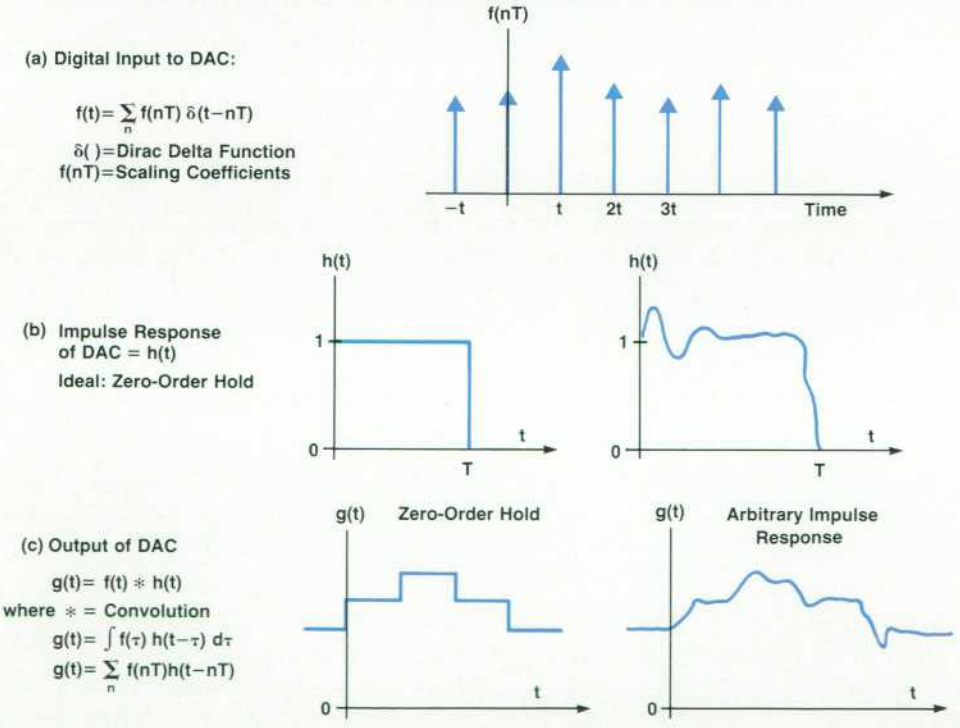


Fig. 2. *Mathematical model of a linear digital-to-analog converter (DAC) with no bit truncation.*

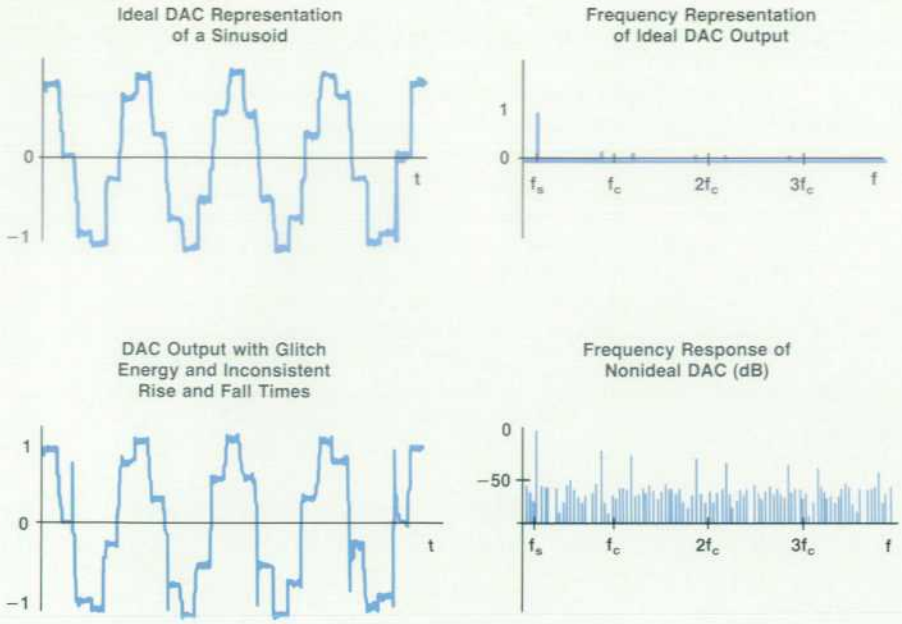


Fig. 3. *Dynamic nonlinearities in a DAC.*

accuracy is maintained by the NMOS and GaAs FET circuitry since there is very little gate leakage in the switching of the current source circuitry. In the system evaluation, a large amount of attention was paid to elimination of the dynamic nonlinearities.

Dynamic nonlinearities of a DAC arise from the inconsistency of its impulse response from one sampling period to another. For the mathematical model of a linear DAC, the digital input is expressed as a sequence of Dirac delta functions with weight coefficients (Fig. 2a). The value of the binary input at time nT is the area under the delta function. The ideal response of the DAC is a zero-order-hold function (Fig. 2b). However, this is not necessary as long as the impulse response is consistent. The output g(t) of an ideal DAC is the convolution of the digital input, f(t), with the DAC's impulse response, h(t) (Fig. 2c). This produces a series of delayed DAC impulse responses that are scaled by the value of the binary input at that time and added together. It is important that the impulse response be consistent in its shape from sample to sample.

In the actual DAC circuitry different combinations of transistors connected to the summing node are turned on and off corresponding to different digital inputs. Each combination creates a capacitive load that depends on the output state and the transistors' operating points. In addition, the ideal elimination of signal skewing between current switches can only be approached within certain practical limits. Any residual skewing causes some currents to be summed in the node before others are turned off, thereby creating glitches (Fig. 3). The resultant state dependent glitches and rise and fall time variations produce distortion. Measurement of a DAC system without additional output processing typically shows a 45-dB signal-to-distortion ratio with sine waves at a 125-MHz sampling frequency.

A return-to-zero GaAs sampler greatly reduces the DAC dynamic nonlinearities. Careful placement of the sampler circuitry directly after the DAC with a minimum amount of packaging interconnection parasitics allows the DAC to settle within 4 ns. The GaAs circuitry uses a 4-ns window to extract the settled portion of the DAC waveform. Close placement of the two circuits is necessary so that input/output impedance mismatch does not cause multiple reflections of the nonlinear transients of the DAC output into the sampling window.

The sampler circuitry also suffers from dynamic nonlinearities, but to a lesser degree. The subnanosecond switching speed of the GaAs circuit minimizes the nonlinearities in the transitions. Not only does the differential nature of the DAC/sampler circuitry provide high-speed performance, but also the distortion produced is largely common mode. A differential-to-single-ended converter (balun) after the sampler has a common mode rejection

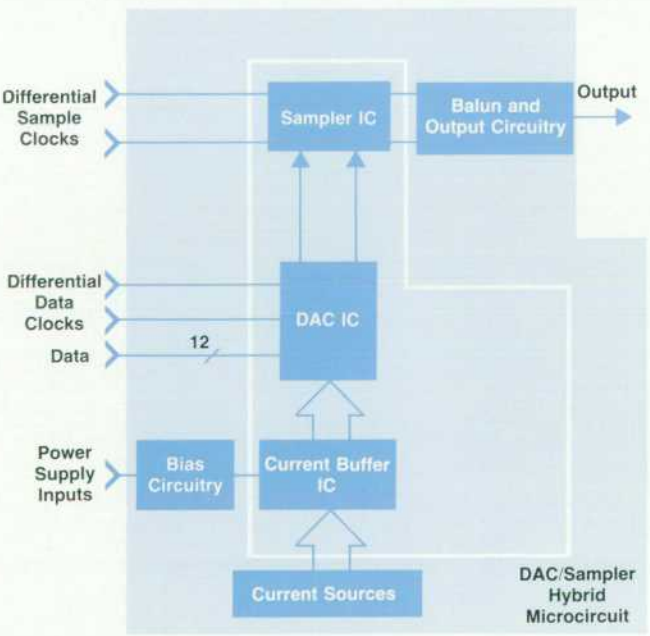


Fig. 4. *Block diagram of the HP 8770A DAC/sampler board.*

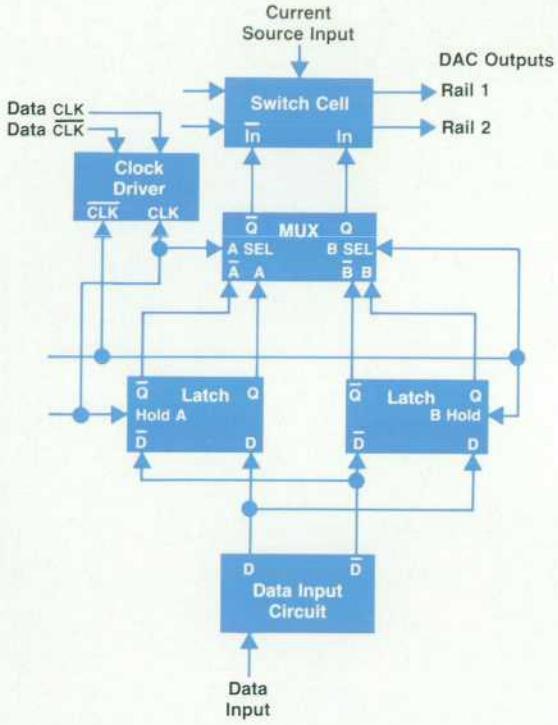


Fig. 5. *Dual data latches with multiplexer for one switch cell.*

characteristic that typically reduces the distortion level from −55 dB to −65 dB.

DAC/Sampler Microcircuit Design Implementation

To obtain high-speed performance with linearity, a set of complementary integrated circuits had to be designed. The DAC and its associated current buffer are fabricated in HQMOS[1] and the output sampler switch is in GaAs. These parts are mounted in very close proximity along with 16 other chip resistors, monoblock capacitors, and metal-insulator-semiconductor chip capacitors in a custom ceramic pin-grid array package.

Fig. 4 is a block diagram of the DAC/sampler system. The sampler IC, DAC IC, and buffer IC are monolithic parts. The current sources, external bias circuitry, and balun and output circuitry are contained on the supporting printed circuit board.

The HQMOS DAC and buffer ICs perform two major tasks: data synchronization and high-speed current steering of external current sources. The DAC converts twelve bits of digital data running at a data rate of 125 MHz. The data and the differential data clock inputs are at ECL levels at the input to the DAC IC to minimize coupling into the switch circuits inside the DAC. The input circuits convert these levels to 5V levels for the synchronization circuitry in the IC. Data synchronization is accomplished with twelve differential dual-latch multiplexers. An on-board clock driver is used to synchronize the switching of all twelve data latches with a data clock running at half the data rate of 125 MHz, or 62.5 MHz. Both the rising and falling edges of the data clock are used to synchronize data

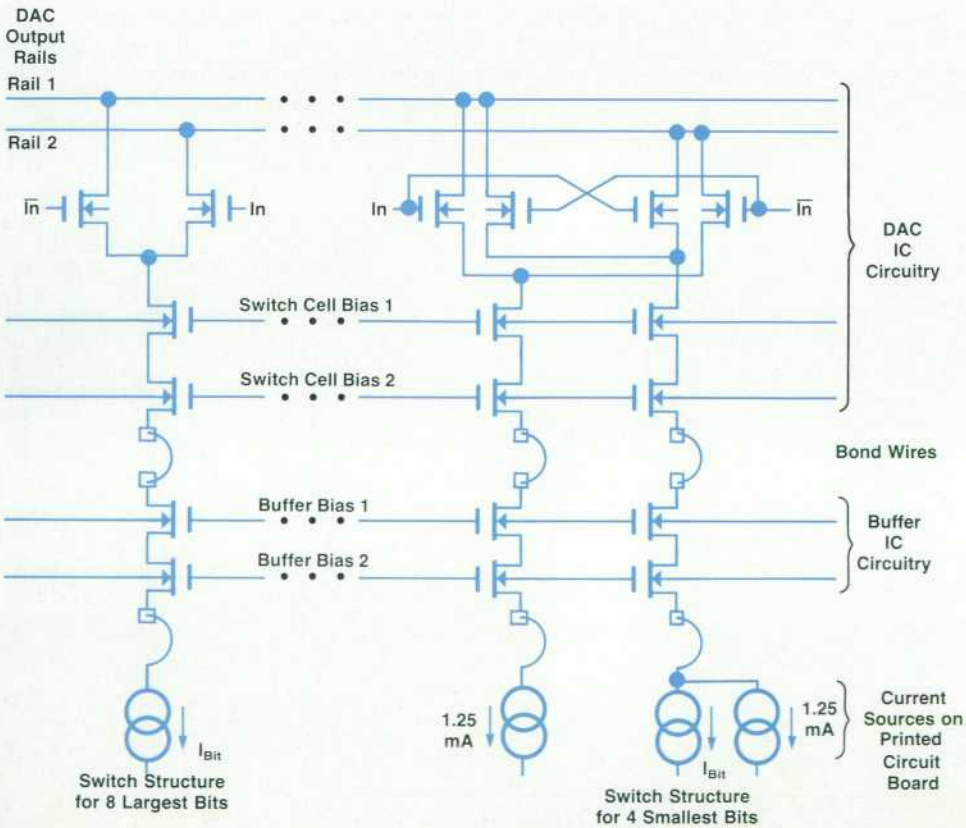


Fig. 6. *DAC current switch structures with buffer FET IC.*

in the DAC IC. This removes the glitches that would normally accompany master/slave flip-flop synchronization with a full-speed clock. As shown in the schematic diagram of a single bit cell, Fig. 5, the DAC's dual-latch multiplexing uses a pair of D latches to synchronize the data on both edges of the 62.5-MHz clock alternately. This allows the glitches associated with the logic swings to be moved away from the "quiet time" after the data is latched, keeping noise coupling low when the DAC is sampled. Two latches are used for each bit to allow one latch to settle with new data while the other is selected with the previous data. Selection is done by a multiplexer on the opposite phase of the clock. Thus, the data clock phases can be used both to latch new data and to select the previous data to synchronize the switching of the current switch cells in the DAC.

The twelve data synchronization circuits drive the twelve switch cells that steer external currents onto two complementary output current rails. The switch cells are sized appropriately for the current being switched. With an MSB bit current of 10 mA, the LSB bit current will be 4.9 μA. The lower DAC bit currents are too small for fast switching, so the last four bits were made differential by adding 1.25 mA to the bit current and switching it against an identical 1.25 mA current without the bit current added. This technique is treated in more detail later in this paper.

To increase the switch settling speed even more, the capacitances of the external current sources are isolated by the current buffer IC. Fig. 6 is a diagram of the current switch cell with the current buffer FETs.

Also included on the DAC and current buffer ICs are on-chip biasing circuits and static protection circuitry. Static protection is customized for each pin dependent on its speed and settling requirements.

The sampler performs the function of selecting the DAC output after it has settled to reject spurious noise pickup and nonlinear DAC slewing. This circuit is fabricated in HP's 1-μm GaAs RF IC process to achieve the speed and linearity needed. The sampler operates by taking a 4.0-ns sample of the HQMOS DAC after it is settled and quiet. The effects of the nonlinear settling of the DAC on the analog output are thus reduced, and the nonlinearity of the sampler becomes the dominant contributor to the output distortion and spurious content.

On the sampler IC is a three-stage ac-coupled differential amplifier used to convert the sample clock inputs from ECL to GaAs levels. Outputs from the last differential stage feed a dc-coupled level-shifting buffer. This buffer drives the differential sampling switch. The differential sampling switch is a balanced structure that steers the current outputs from the two DAC output rails through the sampler switch cells either to the differential analog outputs or to the sampler null output termination. All parts of the GaAs sampler IC are balanced to give maximum rejection of spurious signals and noise. Even the sampling switch cell is balanced by having an identical dummy switch cell to balance out the clock feedthrough from the gate-to-drain capacitance of the switch cell FETs.

The GaAs sampler IC contains 43 FETs, 31 diodes, 14 capacitors, and 29 resistors. Capacitors are used for ac coupling and for circuit bypassing. Diodes are used for static protection, and resistive dividers are used for bias setting.

Fig. 7, a photo of the DAC/sampler hybrid, shows the three ICs. The small IC on the left is the GaAs sampler, the large IC toward the top is the HQMOS current buffer, and the large IC just below it is the HQMOS DAC. Also shown are the chip capacitors and resistors used to bias, terminate, and bypass the ICs inside the package. The high performance of this system mandated the mounting of these parts in close proximity to maintain stability and reject pickup of the data, data clocks, and sample clocks by the differential outputs of the hybrid. The outputs from this hybrid are subtracted by a balanced transformer (balun) for conversion to a single-ended signal and further system noise rejection before they get amplified by the output chain on the DAC/sampler printed circuit board.

This 12-bit system runs at a 125-MHz data rate, producing

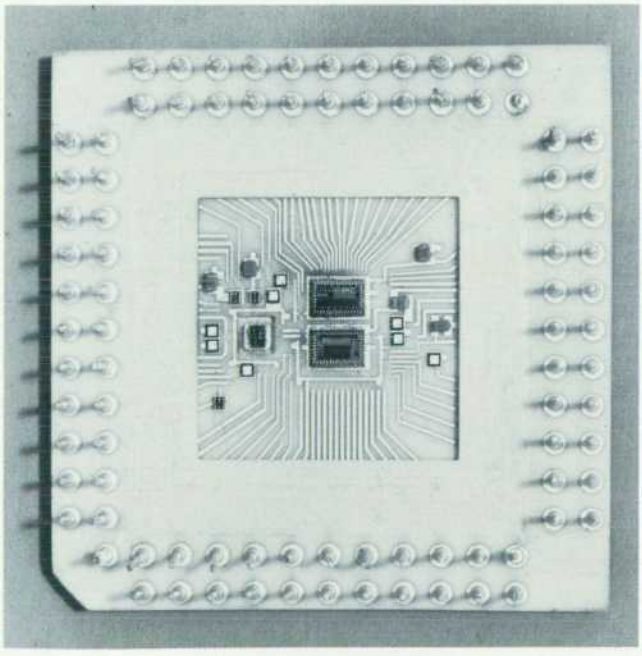


Fig. 7. *DAC/sampler hybrid microcircuit.*

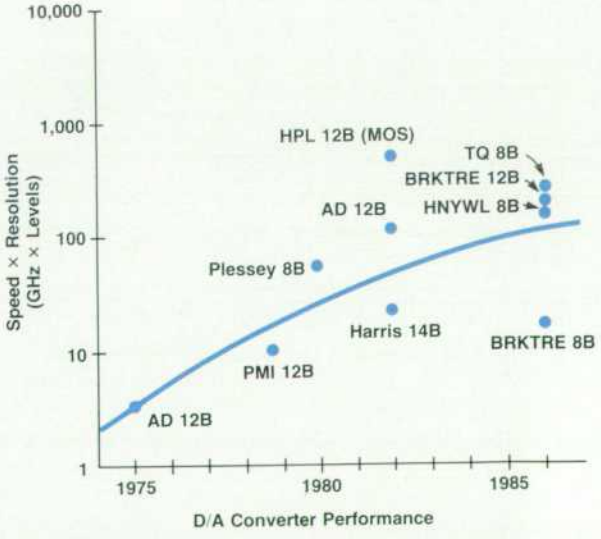


Fig. 8. *Historical trend in commercial DACs.*

a 4.0-ns differential sample with enough linearity to maintain distortion products typically 65 dB below the carrier at the input to the analog postprocessor, filter, and output amplifier section.

DAC IC Initial Development

In the early planning stages of the arbitrary waveform synthesizer, it became clear that the inadequacies of commercially available digital-to-analog converters would have a serious effect on the planned performance of the instrument. Tracing the historical trend in commercial DACs, as indicated in Fig. 8 (with more recent data now added), offered little encouragement that HP's needs would be met. To synthesize sinusoidal frequencies up to 50 MHz with spurious content no more than the design goal of −70 dBc, a very advanced IC technology and significant design improvement would be required.

Silicon bipolar technologies have excellent matching properties and have been used to implement DACs with resolution and linearity well above the 12 bits required in this application. However, two limitations of these technologies make it very difficult to achieve high DAC speed simultaneously with high DAC precision. First, device matching and current gain degrade significantly when minimum geometries are used to increase transistor switching speeds. Second, the simple current switches that must be employed to achieve the specified DAC speed are not compatible with 12-bit low-frequency precision, owing to insufficiently high current gain.

MOS technologies, which do not suffer from dc-current losses through gate conduction, offer excellent frequency response as channel lengths approach 1 to 2 μm, but MOS device matching was, and is, inadequate to support 12-bit linearity simultaneously on the same chip with the same devices that are used to achieve high switching speeds.

The compromise solution was to employ MOS technology for switching precision binary-weighted currents generated off-chip.[2] We focus our comments here on the MOS IC that performs the current switching function.

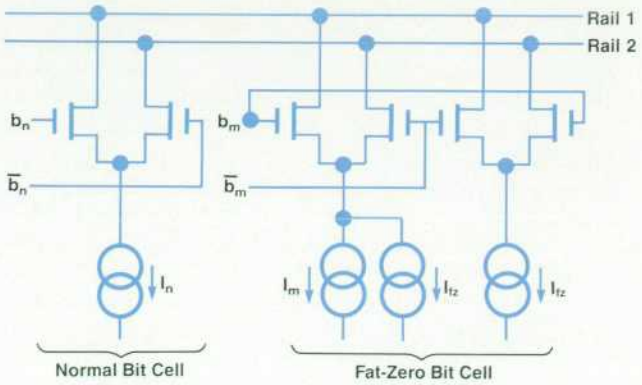


Fig. 9. *Normal and fat-zero bit cells.*

The DAC IC uses precision current switches that are scaled in size to match the scaled dc binary-weighted currents. However, reducing the current-switching transistors' dimensions to achieve proper scaling cannot be carried out beyond a certain minimum size imposed by the MOS process. The use of transistors with the smallest size allowed by the design rules also degrades device matching and results in unacceptably long settling times. The general solution to this problem employed in this DAC is the use of so-called "fat-zero" currents.[3] These dc currents are typically much larger than their respective bit currents. They allow the use of transistors that are sufficiently large to ensure that circuit and device parasitic capacitances are charged and discharged sufficiently rapidly during the switching of current in the bit cells. Fat-zero currents are accurately matched in pairs and are added to the bit currents as shown in Fig. 9. The four least-significant bits in

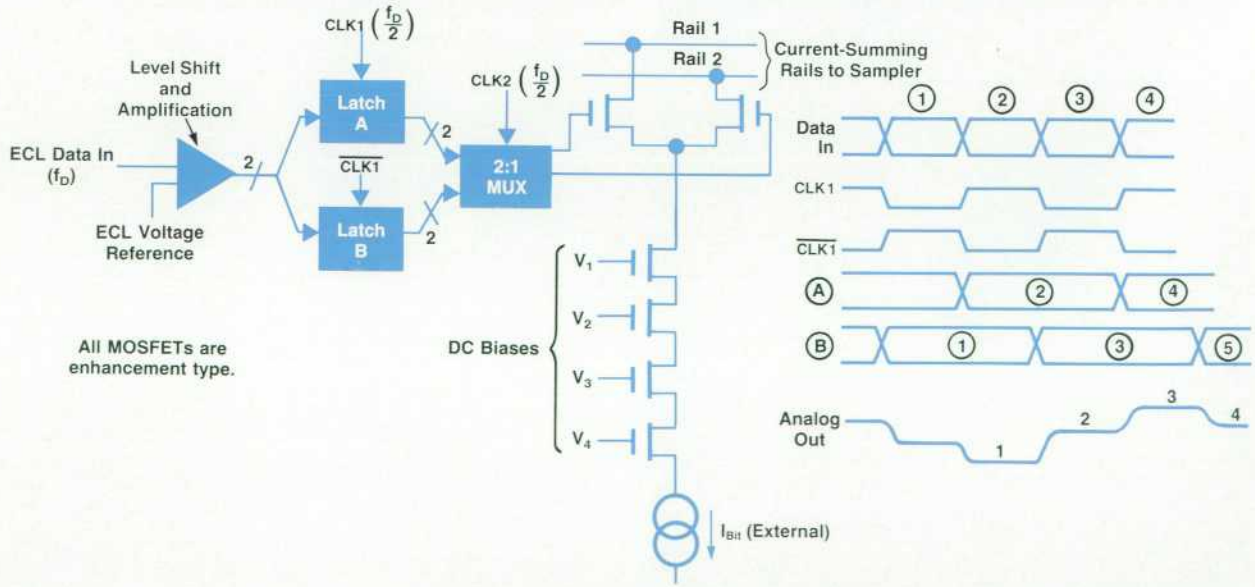


Fig. 10. *Diagram and timing waveforms for a bit cell and one-period stretch circuit. The one-period stretch minimizes the effects of timing skew.*

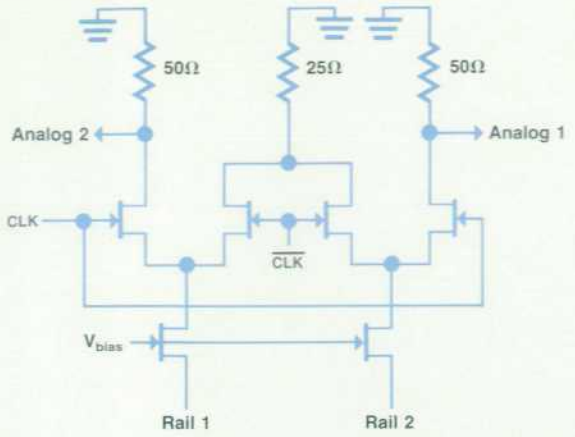


Fig. 11. *GaAs sampler with differential analog outputs.*

the DAC use fat-zero circuits like the one shown in Figs. 6 and 9.

Each bit cell in the IC contains a dc-level translator and amplifier that are used to level-shift and amplify the ECL digital inputs to the levels required by the MOS circuits. After the data is amplified, it is stretched by one data period, as shown in the bit cell block diagram in Fig. 10. The one-period stretch is used to minimize the effects of timing skew in the incoming data and to ensure that all current switches operate at the same time. The use of two latches in parallel and the subsequent 2-to-1 multiplexing prevents any transition, owing either to externally applied clock signals or to internal DAC circuitry, from occurring in the chip during the time when the analog output is settling. In a more conventional approach that uses, for example, a D-type master/slave flip-flop to retime the data without splitting its path, a midperiod clock transition would occur on the chip and would feed through to the analog output. In some small but significant way such a transition disturbs the settling process and contributes to spurs in the sine wave produced by the DAC.

The operation of the DAC bit cell is summarized with the help of the timing diagrams shown in Fig. 10. All of the MOS circuits use conventional enhancement/depletion MOS design methods.

The analog output of the DAC exhibits transitions that die out or settle in a nonlinear fashion. One approach to minimizing the effects of these nonlinearities is to use only the settled part of the analog output and to discard the nonlinear transitions. The success of this approach depends strongly on the ability to realize a discard or deglitching sampler whose switching speed is much faster than that of the MOS DAC. Such a circuit is shown in the schematic of Fig. 11. Depletion-mode GaAs transistors are used to steer the current from Rail 1 and Rail 2 to the 50Ω loads.[4] The transition times (10 to 90%) of the sampler are typically less than 100 ps, much less than those of the MOS switches.

Analog Postprocessors: The Filter

The low-pass filter is the product of a lot of computer simulation about its effect on signals generated by a digital synthesizer. A low-pass filter removes the unncessary signal energy beyond the Nyquist frequency. It also relaxes the requirements on the subsequent amplifiers to process signals of higher frequencies without distorting. An ideal low-pass filter with an inverse sin(x)/x magnitude compensation in the passband is the optimal filter selection for a general waveform synthesizer. Aperture correction is necessary because of the sampler's output characteristics. The ideal low-pass filter does not distort any signal with spectral energy components that are totally confined in its passband region. Therefore, strict accuracy is preserved from dc to the Nyquist frequency. There is a disadvantage for waveforms such as a square wave created in the time domain whose rise time occurs in one sample period. Its

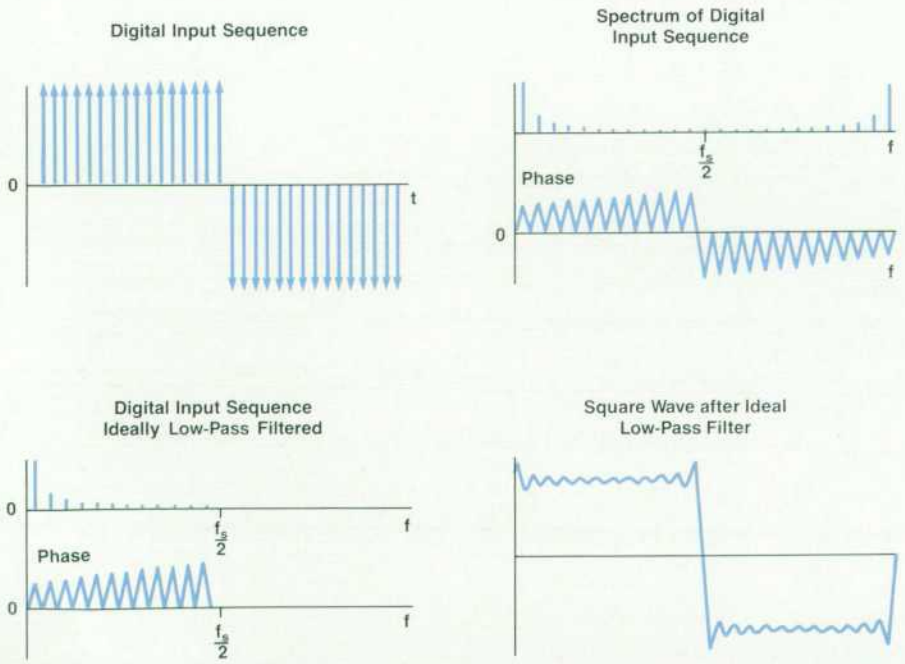


Fig. 12. *Effect of an ideal low-pass filter.*

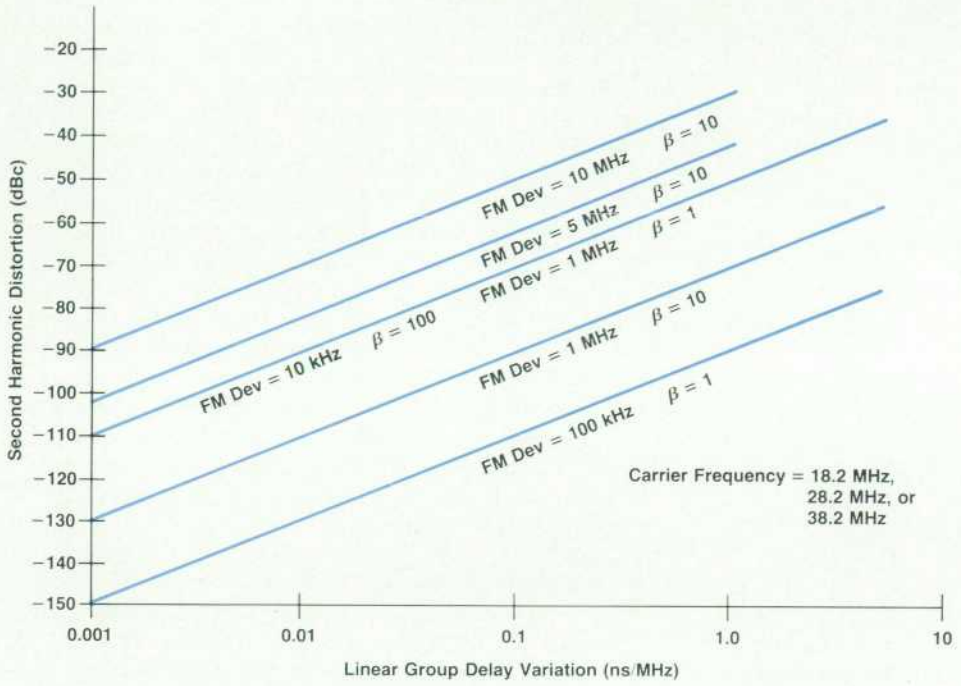


Fig. 13. *Second-harmonic distortion of a demodulated FM signal with a sinusoidal modulation source after distortion by an ideal low-pass filter with group delay variation. β is the modulation index ($\beta = \Delta f/f_m$).*

frequency components extend beyond the Nyquist rate, causing aliasing. An ideal low-pass filter truncates the spectrum at half the sampling frequency and the resultant waveform has ringing (Fig. 12). The ringing can be minimized by prefiltering the waveform using the computer software to smooth out the waveform. A Bessel-type low-pass filter could easily be chosen over the ideal low-pass filter to improve the appearance of the waveform, but the magnitude distortion of signals in the passband region can be as high as 3 to 10 dB, depending on how much attenuation one requires in the stopband region to eliminate the clock and aliasing components. The time waveform looks correct, but the spectrum is very inaccurate, and signal accuracy is measured by its spectral accuracy within the specified signal bandwidth (dc to 50 MHz for the HP 8770A).

Given that an ideal low-pass filter could not be implemented, the instrument's signal bandwidth is limited to 50 MHz and allows a filter transition region of 12.5 MHz. The instrument user can provide additional specialized filters to modify the time response of the waveform if the dc-to-50-MHz signal bandwidths are exceeded.

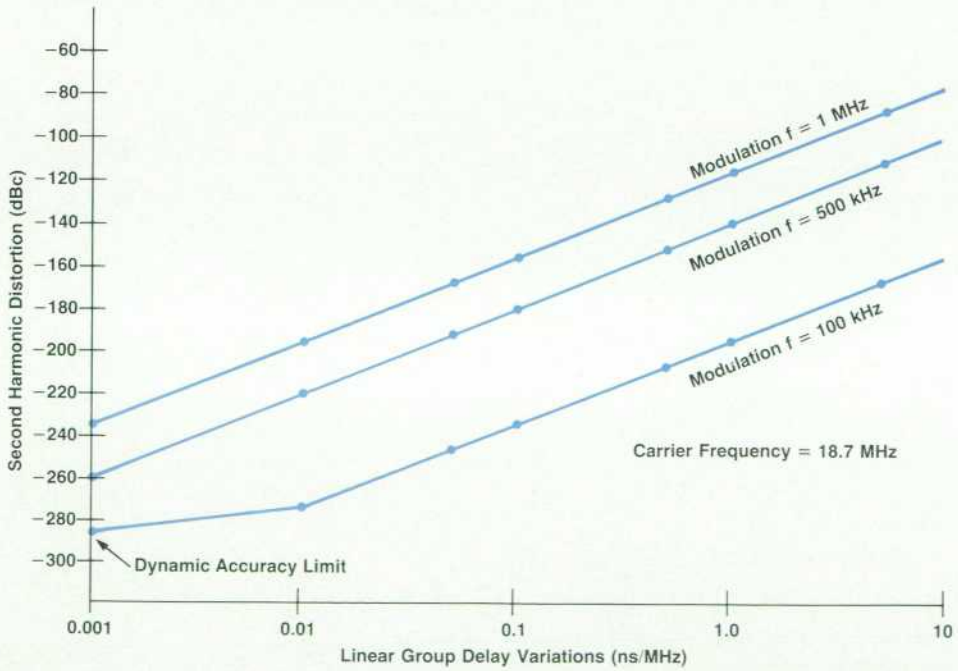


Fig. 14. *Second-harmonic distortion of a sinusoidal modulating signal in an AM system (50% AM) caused by linear group delay variation.*

Computer simulations of the filter's nonlinear phase effects on FM and AM signals provides an acceptable upper limit of a constant group delay variation of 0.1 ns/MHz. This generally translates to a maximum of four degrees peak-to-peak nonlinear phase variation for the filter phase response. Fig. 13 shows the second-harmonic distortion of a demodulated FM signal with a sinusoidal modulation source when distorted by an ideal low-pass filter with a certain amount of group delay variation. The case of an FM deviation of 10 MHz and a modulation index β of 10 covers the whole 50-MHz signal bandwidth and is considered to be the worst-case example. The filter is designed for better than −50-dB demodulated FM signal distortion. Simulations on 50% AM signals showed much less sensitivity to group delay variations for sinusoidal modulation frequencies as high as 1 MHz (Fig. 14). The Waveform Generation Language (WGL) for the instrument allows compensation for magnitude and phase errors to within 0.4 dB p-p and 2 degrees p-p, respectively.

Summary

A good balance is achieved between the performance of the DAC/sampler, filter, and amplifier. Particular attention to component linearity during the design provides delivery of an accurate signal with distortion typically between −55 dB and −70 dB. The magnitude and phase are accurate within 1.3 dB and 7 degrees, respectively, and instrument-to-instrument performance numbers are tightly grouped.

Acknowledgments

The key to the success of the HP 8770A project was the design of a 125-MHz, 12-bit digital-to-analog converter. The initial research and development were done at HP Laboratories. Stanford Park Division designed the prototype system that generated high-speed ECL signals to evaluate the DAC. The design and implementation of the DAC IC, buffer IC, sampler IC, and hybrid microcircuit assembly were done at the Spokane Division, along with the development of the systems and the processes for the construction and verification of the completed systems. The analog post-processors were developed and systems integration was done at the Stanford Park Division. Robert Hallgren did the DAC IC and buffer IC design. Particular thanks should be given to Mike McNamee, Brad Anderson, Alan Hedge, Larry Wright, Ben Helmso, Frank Yeck, Dick Waite, Scott Smith, and George Conrad from Spokane Division, Val Peterson, Andy Teetzel, and Rory Van Tuyl from the Microwave Technology Division, Larry Chesler and Doug Bartlett from the Loveland Technology Center, and K.C. Hsieh, Tom Knotts, and Horng-Sen Fu from HP Laboratories. Thanks also to Ray Fried for his guidance and to Steve Holdaway for his support of this project.

References

1. H.S. Fu, et al, "HQMOS: A High-Performance NMOS Technology," *Hewlett-Packard Journal*, Vol. 33, no. 10, October 1982.
2. E.C. Kwong, G.L. Baldwin, and T. Hornak, "A Frequency-Ratio-Based 12-Bit MOS Precision Primary Current Source," *IEEE Journal of Solid-State Circuits*, Vol. SC-19, no. 6, December 1984, pp. 1029-1037.
3. T. Hornak and G.L. Baldwin, *Digital to Analog Converters Having Supplementary Currents to Enhance Low Current Switching Speed*, U.S. Patent 4,405,916.
4. G.L. Baldwin, R.B. Hallgren, T. Hornak, and F.H. Ives, *Electronic Sample Switch*, U.S. Patent 4,639,619.

Arbitrary Waveform Synthesizer Applications in Magnetic Recording and Radar

by Albert W. Kovalick and Roland Hassun

THE HP 8770A ARBITRARY WAVEFORM SYNTHESIZER is well-suited for generating signals to test read channels and servo channels in disc drives. It can also be used to calibrate disc media certifiers. This article will explore these applications and the use of signal simulation in the field of analog radar system design.

Magnetic Recording

Fig. 1 shows a typical read channel block diagram as used in contemporary disc drives. All the signals up to the decoder are analog. During the development engineering process each block needs to be tested for design integrity. Once the design moves to production, read/servo channels need to be tested with a variety of input signals that verify board operation.

In disc drives, the data bits are encoded as magnetic field transitions on a rotating magnetic surface. The read/write head senses the magnetic field and outputs a pulse for each field change. The HP 8770A is capable of emulating the read signals for many useful applications. With the availability of such precision signals, the head-disc assembly is no longer needed as a source of signals in several instances. One instrument can generate all the required signals without additional support equipment.

Defining the Signal

Many different types of signals are needed for testing drive circuitry. In general, there are two methods for obtaining the desired waveforms. One is to prerecord all desired waveforms using an actual drive and a waveform recorder, load the data into the HP 8770A, and play back the signals as required. The record/playback method is good for situations that require the exact real-life signal. It has disadvantages, however. For one, the brute-force playback time is limited to 4 ms (125-MHz clock, 512K memory), although with some waveform editing and the use of the sequencer, a 16.66-ms (1 disc revolution) signal period can be achieved. Another disadvantage with the method is that any waveform modification would likely require that the user go through the record process for each change. Also, adding margins to test signals is not easy. In all, the record/playback method has its place and may be used successfully, given its limitations.

The second method for obtaining waveforms is computation based on waveform modeling. In this case, an isolated read head transition is considered the fundamental building block to build all other signals. For the traditional ferrite head, a Lorentzian or modified Lorentzian pulse shape (Fig. 2) can be used as the fundamental isolated signal. For thin-film heads, the Lorentzian is a poor approximation to the actual pulse. In this case, a captured isolated pulse can be digitized and used as the fundamental pulse (Fig. 3).

Once the fundamental pulse is obtained, it is used to construct long, complex signals with all manner of amplitude, phase, and frequency changes as needed. The main advantage for using computed signals over recorded signals can be seen in margin testing. With a computed test signal, almost every conceivable signal anomaly can be generated to test performance margins.

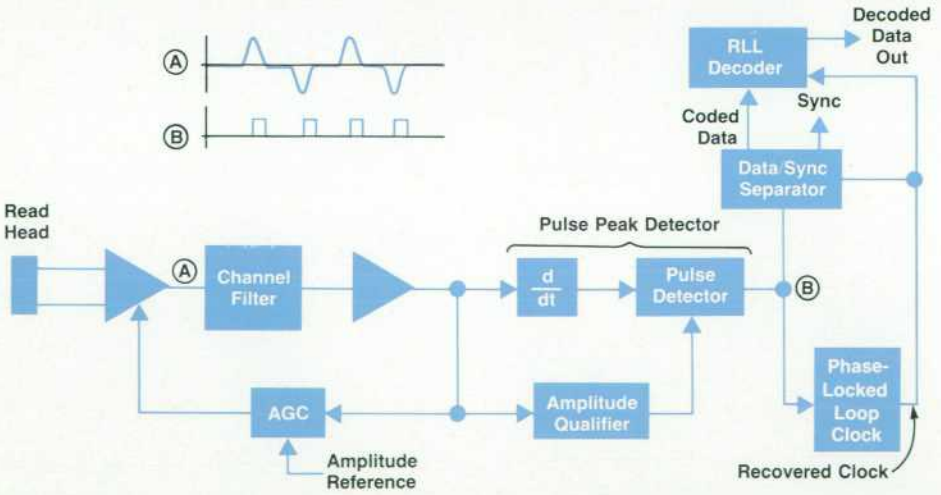


Fig. 1. *Block diagram of a typical disc drive read channel.*

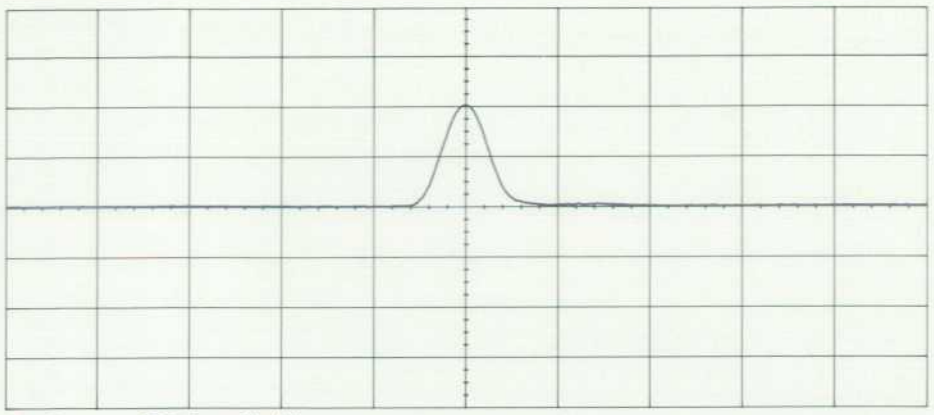


Fig. 2. *Monopulse from a ferrite head.*

Testing the Read Channel

Let's consider a few examples of read channel waveforms. All were built using the HP Waveform Generation Language (WGL, see page 94). For long waveforms whose duration exceeds 4 ms, the memory sequencer was used. Generating waves that last hundreds of milliseconds is relatively easy.

Consider, again, Fig. 1. The channel filter is usually a low-pass filter cascaded with a phase equalizer. The filter characteristics can be tested using the synthesizer, a digitizing oscilloscope, and a software Fourier transform algorithm. In this case, the synthesizer generates a special signal that is passed through the filter. The oscilloscope captures the filter output and the result is converted into a frequency-domain representation using the Fourier transform algorithm. The frequency-domain representation contains the filter's amplitude and phase characteristics.

A discussion of coding is beyond the scope of this article. However, it should be mentioned that run length limited (RLL) codes are prevalent in today's disc drives. Using RLL codes has enabled drive designers to pack data very efficiently onto the platter surface. As an example, Fig. 4 shows an RLL(2,7) code sequence that yields more high-frequency spectral content than other RLL(2,7) sequences. Here the

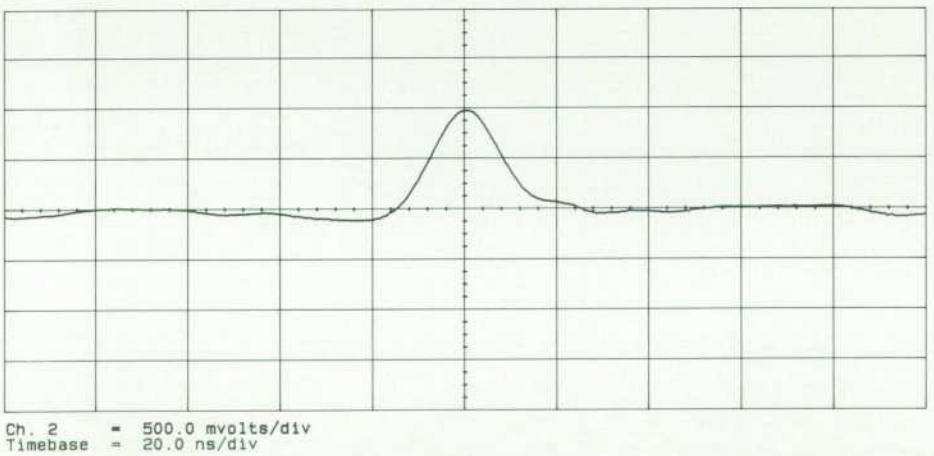


Fig. 3. *Monopulse from a thin-film head.*

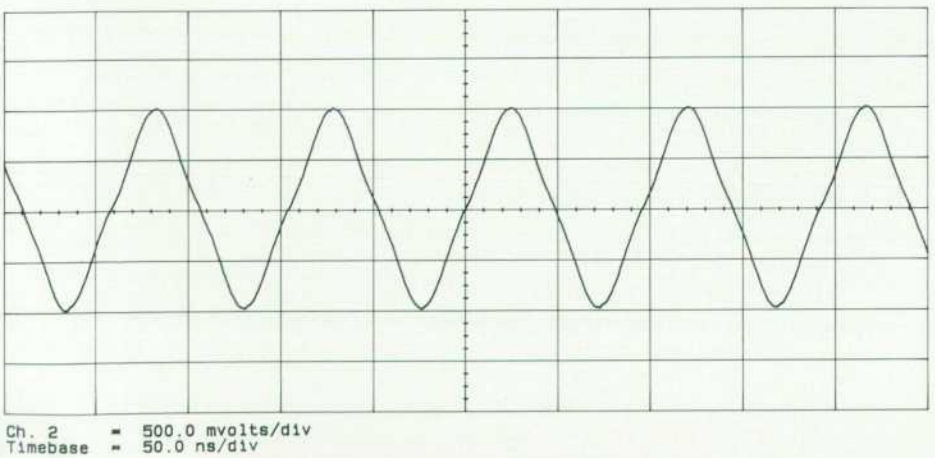


Fig. 4. *Maximum-frequency (2F) coded data (100100100...).*

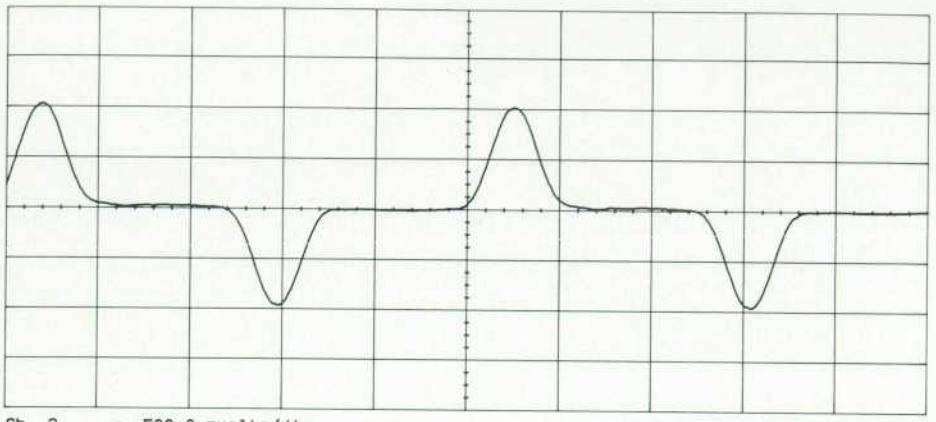


Fig. 5. *Minimum-frequency (0.75F) coded data (100000001000000010...).*

spacing between magnetic transitions is a minimum of two code cell positions—hence the 2 in the RLL(2,7) identifier. Fig. 5 shows an encoded stream that has much more energy at lower frequencies. Here the maximum spacing between transitions is seven code cell positions. In fact, seven code cell positions between transitions is a maximum for the RLL(2,7) code. Part of the beauty of RLL coding is that transitions cannot be too close or too far apart. This means that transitions cannot be too closely packed. This may seem contrary to the dictum "Waste not, want not." However, transition density is intelligently traded for transition position information.

These sequences, and others like them, are also useful in testing the tolerance levels of the phase-locked loop. The loop oscillator output is the recovered clock source. The loop must stay locked to the desired frequency and not drift even though the input data pattern varies wildly. Initially, the loop is locked to a sync sequence. A thorough test of the channel filter and the phase-locked loop would require a sync frame. This frame consists of a series of

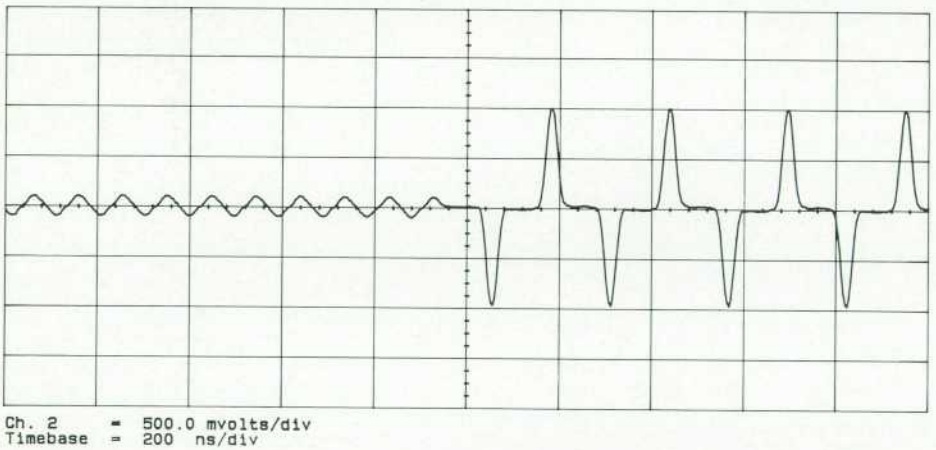


Fig. 6. *AGC test waveform.*

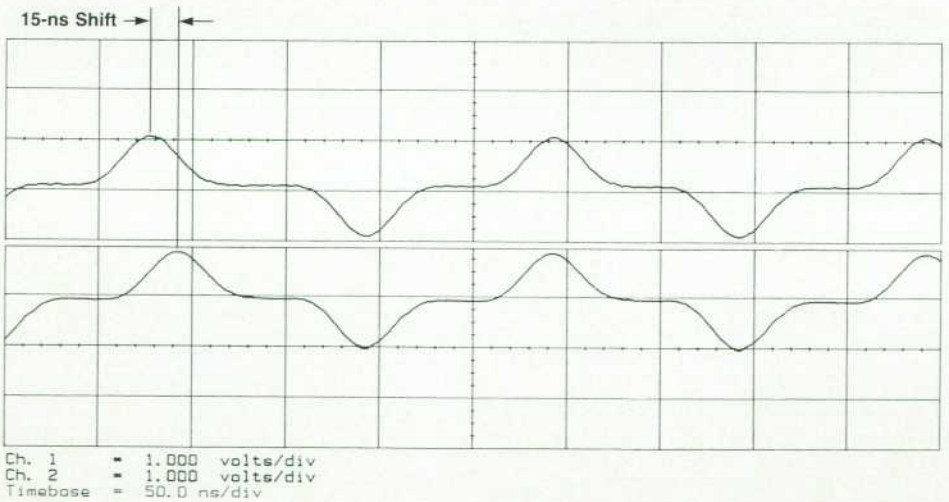


Fig. 7. *Bit shift test waveforms.*

pulses that the loop locks onto. It is usually located in the preamble that precedes the data block. For margin testing, the sync frame can be designed with aberrations on the individual pulses.

The AGC block in Fig. 1 also needs to be tested. Three important AGC parameters can be tested with the code sequence shown in Fig. 6: amplitude dynamic range, attack time, and immunity to frequency deviations. Here the amplitude changes by a factor of 10:1 during an abrupt step. Again, one of the HP 8770A markers can be used to mark the amplitude change and to trigger a scope for a measurement of the AGC response.

It Does Windows, Too

Another area of interest is window margin (also called phase margin). The window in question is the width of the code cell detection window. It is the window in which a transition may be detected with a high certainty. Ideally, the transition will occur in the center of the window. The loop clock is used to identify the center of the detection span. All physical systems, however, have bit shift, or movement of the detected bit off the ideal center. If too much shift occurs, a read error will occur.

Although a complete window margin test needs the head-disc assembly, the HP 8770A can assist in testing the read channel for bit shift susceptibility. A transition sequence with one bit shifted off the ideal center can be used as the test signal. Fig. 7 shows an unshifted bit pattern at the bottom and a shifted bit pattern at the top with the leftmost pulse shifted 15 ns off center. The shifted bit can be moved with subnanosecond resolution until an error occurs. The amount of shift tolerated before an error occurs is a measure of the integrity of the detection process.

Testing the Servo Channel

Disc drives also have a servo channel to process the servo

Fig. 8. *Two sync pulses followed by four di-bits. The amplitudes of the di-bits are related to the off-track alignment error. The di-bits are used to center the read head on the desired track.*

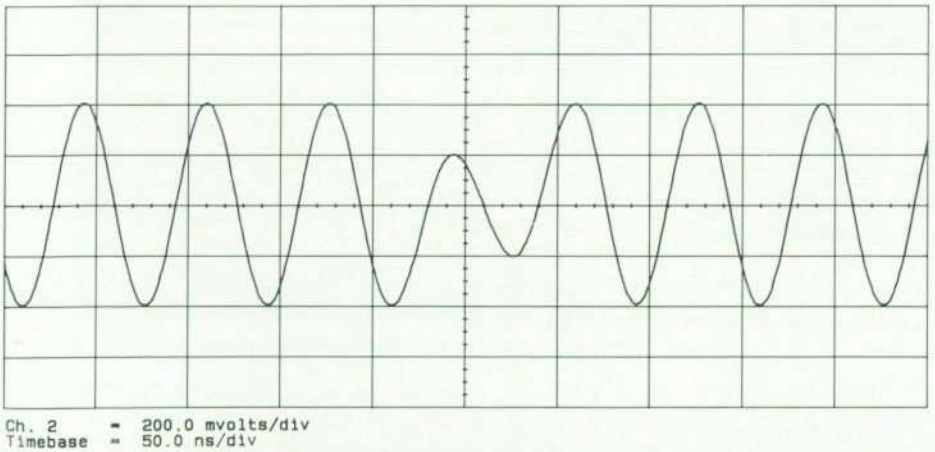


Fig. 9. *Two missing bits at 50% of normal amplitude.*

head signal. In its simplest form, the servo system positions the read head over the desired read track. The testing needs of the servo channel are similar to those of the read channel. Typically, complex patterns with signal anomalies are needed. Fig. 8 shows the industry standard quad di-bit pattern preceded by two sync pulses. It is relatively easy to generate this pattern with the HP 8770A, along with other long, complex, high-bandwidth patterns that simulate the head crossing tracks.

Disc Media Certifiers

Manufacturers of disc media and drive manufacturers who buy disc media need a way to test the quality of the magnetic surfaces. There are established specifications in the industry for testing the media, and several manufacturers supply media certifiers. Many tests are performed before a disc surface passes a quality check. Two tests that are usually needed to qualify a surface are the missing and extra bit tests. Fig. 9 shows a portion of a 16.66-ms code bit sequence with two transitions reduced by 50%. The sequence is at a transition rate of 30 MTPS (million transitions per second). This signal has about 500,000 transitions per period (16.66 ms).

The missing bit signal can be input to the certifier hardware to verify its integrity over a range of reduction levels and a range of missing bits per revolution. Certifiers can be programmed to detect missing bits at a user-defined level. A good test of the certifier's ability to detect such missing bits is to send it a succession of signals, each with bit reduction levels near the desired setting. In practice, the certifier will mark some bits as missing and some as not missing. For example, assume the certifier missing bit threshold is set to 50%. Now, send it several signals, each having a different, but known, bit reduction between 45% and 55%. By noting the certifier's response to each signal, its accuracy can be determined. Also of interest are the extra bit test, positive and negative modulation tests, peak shift test, overwrite test, and AGC test. Signals for all of these can be produced by the HP 8770A.

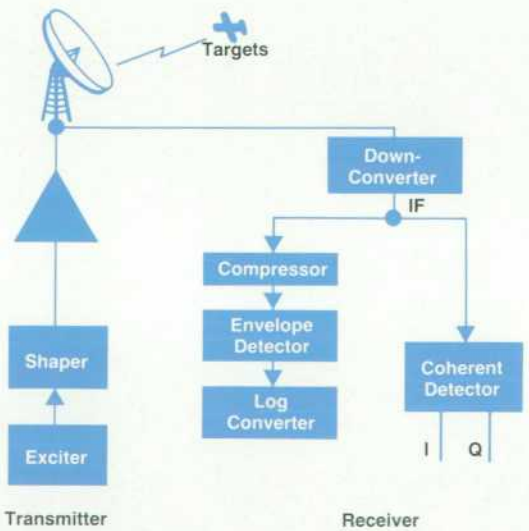


Fig. 10. *Chirp radar generic block diagram.*

Search Radar Simulation

Consider the design of a system as complex as a chirp radar used for surveillance. The generic block diagram, Fig. 10, consists of a transmitter section and a receiver section. Each requires special consideration when it comes to the selection of design parameters that are compatible with the mission of this particular radar. HP 8770A signal simulation can be used to define and optimize the architecture.

The simplified transmitter section contains an exciter which is responsible for creating the linear chirp function at some intermediate frequency. In general, it is undesirable to use waveforms with sharp edges. Therefore, a network capable of shaping the envelope of the pulse is included. The shaped waveform can be upconverted to the desired RF frequency by means of a fixed local oscillator and a mixer. The final stage in the transmitter section is a power amplifier. It is possible to simulate all of these modules, including the effects of saturation or distortion in the power amplifier.

Receiver

The first stage in the receiver is a down-converter whose function is counter to that of the upconverter in the transmitter. Any anomalies in this section can also be modeled through the use of WGL. Following the down-converter are two parallel paths. One is the range detector subsystem and the other is the coherent detector. The latter has two outputs, I and Q, which are generally fed to digital signal processors. The processors perform a number of algorithms relating to the mission of the radar. In particular, the extraction of Doppler shift relating to target radial velocity is of special importance, as is clutter cancellation.

Simulation Program

An interactive chirp radar simulation program was written using WGL. It consists of a number of modules that can be run separately or concatenated. Each module roughly corresponds to the function of a hardware module. Thus, module PULSECHIRP performs the function of the

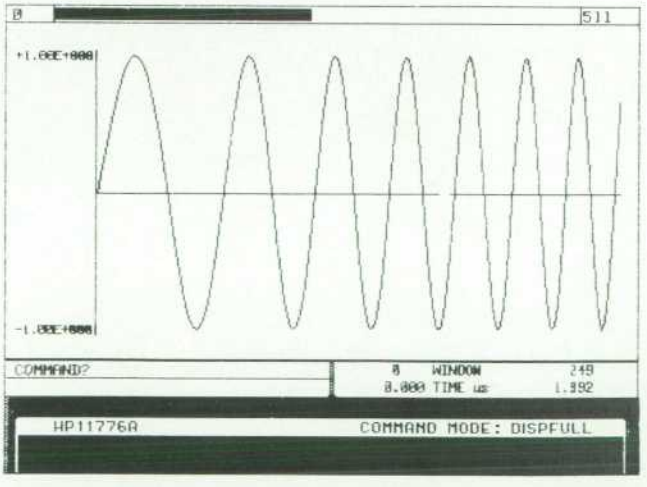


Fig. 11. *A Waveform Generation Language (WGL) display of a linear chirp from 10 MHz to 12 MHz in a 2-μs interval.*

exciter. Module SHAPING is related to the shaper, COMPRESS to the compressor, IQDETECT to the coherent detector, and so on.

Chirp Parameters

The module PULSECHIRP asks for the parameters of the linear chirp and returns the chirp wave as a function of time. The required parameters are the pulse repetition interval, the pulse width, the frequency deviation, and the start frequency.

The chirp is computed from a definition of the phase profile. A linear frequency function with time implies a quadratic phase function over the same period of time. The starting value of phase is zero and the end value corresponds in radians to the number of cycles to be accumulated during the period of the pulse times 2π. For example, if you have a pulse width of 2 μs and you wish to have a linear frequency deviation of 2 MHz, the peak phase deviation is $2\pi \times 4$ during the 2-μs interval.

The WGL program looks like this:

```
0 249 WINDOW          {Time interval of 2 μs}
RAMP 1 + 2 / SQ       {Normalized quadratic profile}
PI* 2 * 4 *           {Set peak to 8π radians}
STORE A               {Store array in register A for}
                      {future use}
RAMP 1 + 2 / 20 *     {Linear phase profile for a 10-MHz}
                      {carrier}
A +                   {Add chirp profile}
SIN ?                 {Take the sine and display. This is}
                      {a chirp from 10 MHz to 12 MHz in}
                      {2 microseconds}
```

The pulse created by the above program can then be repeated at the desired rate. Fig. 11 shows a plot of the waveform within that pulse.

Shaper

The next module, SHAPING, provides three options for functions by which to multiply the pulse generated by the exciter. One of the options is no shaping at all. The others are Gaussian and raised cosine. Of course, any other function can be written into this module to be used for shaping.

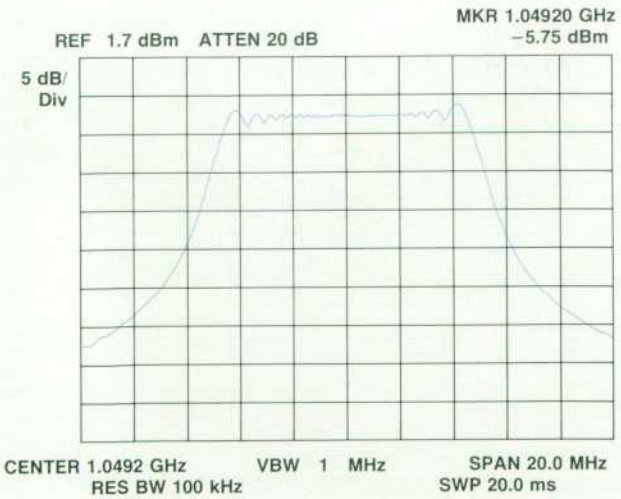


Fig. 12. *Spectrum analyzer display of a 10-μs, 10-MHz chirp about 1.0492 GHz with a 32.768-μs pulse repetition interval, obtained by upconversion.*

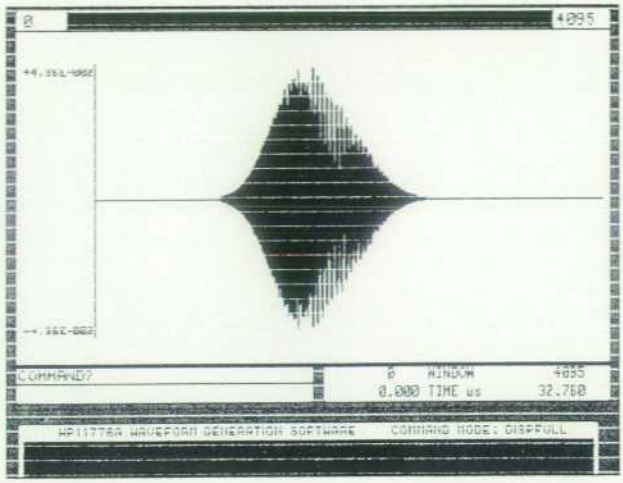


Fig. 13. *A WGL display of a two-target return for a Gaussian-shaped 10-MHz, 10-μs, 32.768-μs-PRI chirp.*

Matched Filter

Matched filtering is the ideal detection scheme for pulse-type signals used in radar. It maximizes the signal-to-noise ratio when the noise is Gaussian (a common occurrence). A matched filter is matched to the signal being detected. It has a transfer function equal in magnitude to the signal in the frequency domain and a phase function that is the negative of the phase-versus-frequency profile of the signal. If the signal being detected is the same as the reference signal, the resulting spectrum is the square of the magnitude of the reference signal with the phase being zero at all frequencies. The corresponding time function is the autocorrelation function of the reference signal. This whole process is described in the WGL routine called COMPRESS

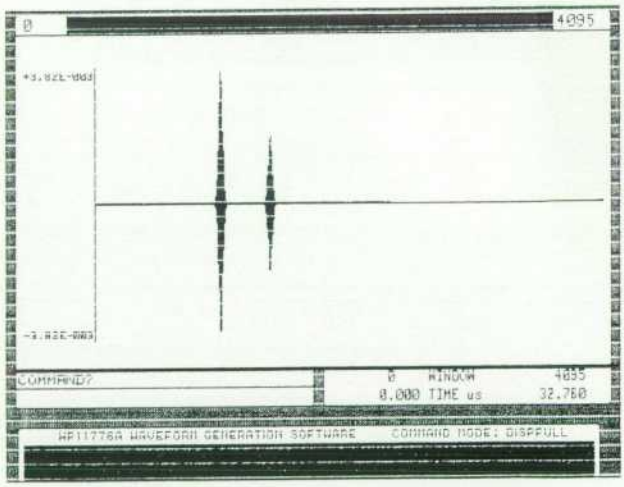


Fig. 14. *WGL display of the compressor output for an input like Fig. 13.*

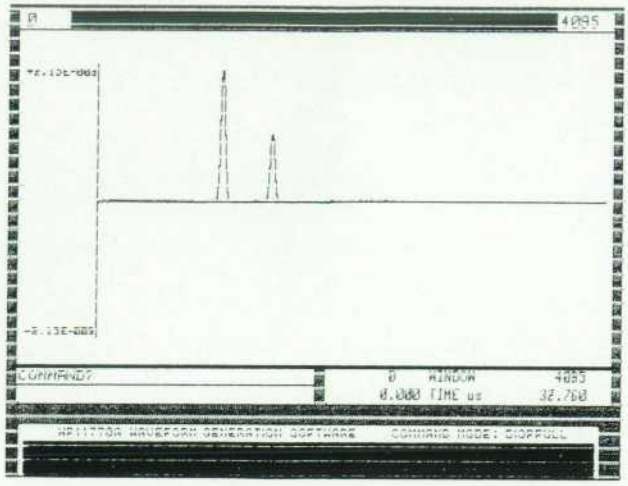


Fig. 15. *WGL display of the envelope detector output for an input like Fig. 14.*

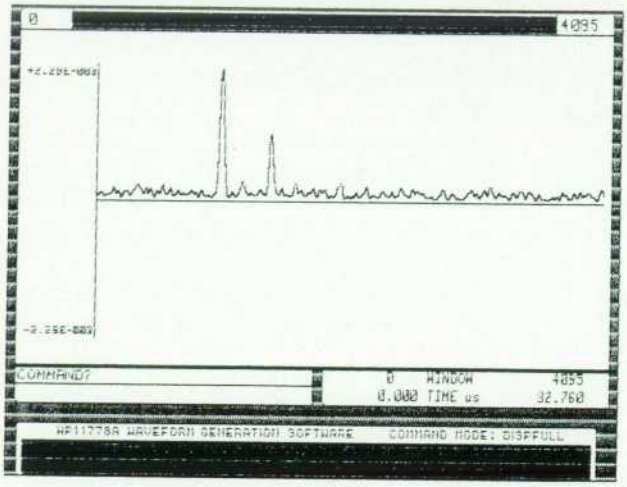


Fig. 17. *WGL display of the envelope detector output for the waveform of Fig. 16.*

very conveniently, since Fourier transform and inverse transform commands are available. A reference amplitude and phase for the radar signal are stored in two registers. The transmitted signal is attenuated and delayed according to the distance of the targets and then converted to the frequency domain, where the magnitude is multiplied by the reference magnitude and the phase subtracted from the reference phase. The result is converted to the time domain and represents the output of the pulse compressor or matched filter in the block diagram.

It is possible to add noise or various types of interference or jamming before the matched filter and observe the effect on the output. This is also the point where the range resolution of the radar can be evaluated. Multiple targets at various spacings can be hypothesized and the output of the matched filter tested for target discrimination. The testing can be as straightforward as a visual inspection of the graphical output of the matched filter. The actual signal can always be created by downloading the data to the HP 8770A and giving it the GO command.

Figs. 12 through 20 illustrate some of the concepts that are mentioned in this paper.

The envelope detector and the logarithmic converter are combined into one routine called ENVDETECT. Envelope detection is accomplished by taking the absolute value of the waveform at the output of the matched filter and then low-pass filtering. Logarithmic conversion is accomplished by a single command in WGL. The advantage of the logarithmic display is the compression of the dynamic range so that signals of widely different magnitudes are readily observable.

Synchronous Detection

Some modern radars will have provisions for synchronous detection of the output of the matched filter into I and Q channels. Each of these channels is digitized and the data processed to partition the target returns into range bins and velocity bins. A range bin has a width approximately equal to the range resolution of the radar. This is

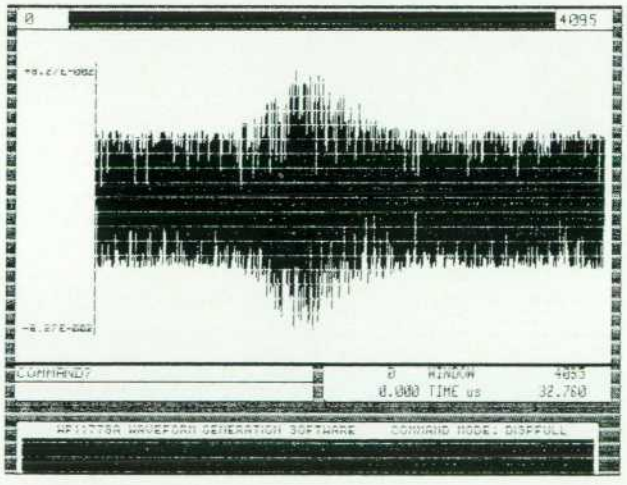


Fig. 16. *WGL display of the waveform of Fig. 13 plus noise.*

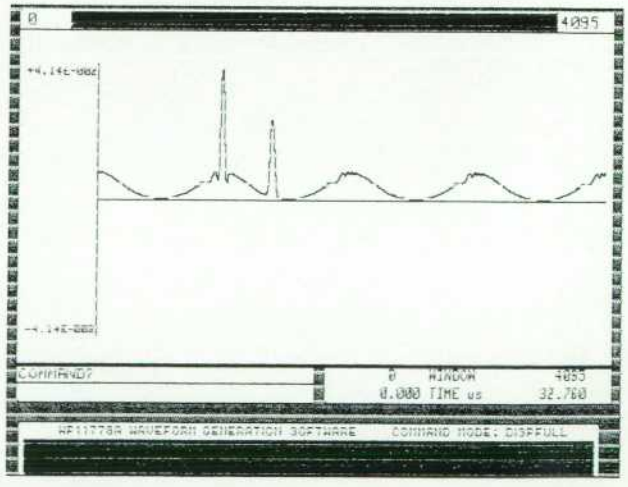


Fig. 18. *WGL display of the envelope detector output for a two-target return subjected to wideband sweep jamming.*

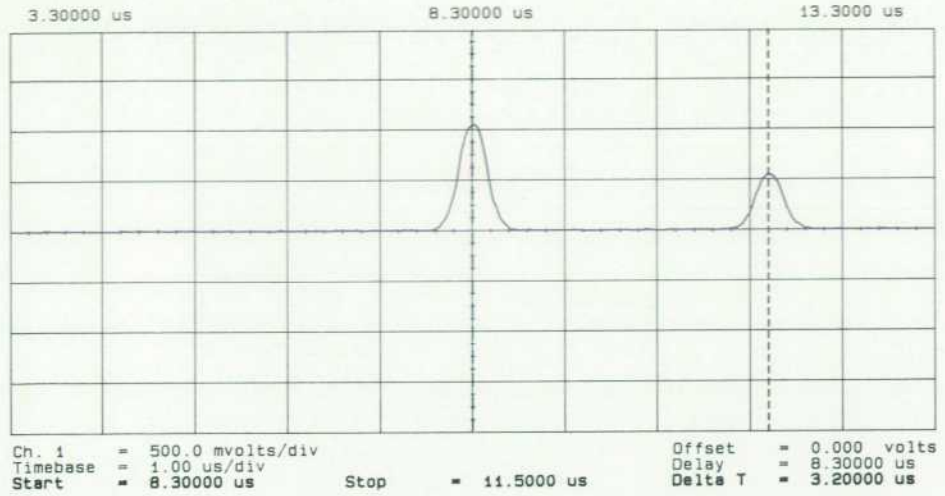


Fig. 19. *Plot of an oscilloscope display of the envelope detector output for the waveform of Fig. 13. The programmed target separation was 3.2 μs.*

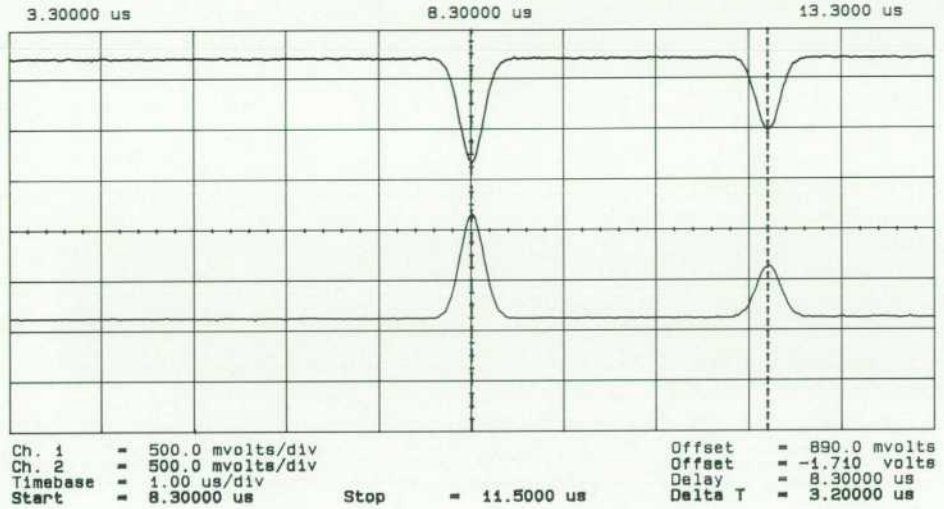


Fig. 20. *Plot of an oscilloscope display of the I and Q outputs of a coherent detector for the two targets of Fig. 13.*

related to the bandwidth of the chirp as follows:

$$R_{res} \propto c/\Delta f$$

where c is the velocity of light and R_{res} is the minimum distance between targets for which they can be distinguished as separate targets.

The signal in each range bin is processed through a discrete Fourier transform which places it in one of several Doppler bins. These bins are separated in frequency by an amount equal to the inverse of the observation period. Thus, each target has specific range and velocity or frequency bins. The frequency bins can be positive or negative depending on whether the target is moving towards or away from the radar.

As can be imagined, the signal processing, which is done mostly by digital means, is very complex. The algorithms have to be selected with great care and need to be verified very thoroughly. This is where signal simulation with the HP 8770A is especially useful. A number of scenarios can be defined for the purpose of testing the validity of the algorithms and the hardware in specific circumstances. These could be tracking a moving target in the presence of another target, or tracking in the presence of various countermeasure techniques.

Effects of target fluctuations can be modeled according to established theory and simulated by the HP 8770A. Signal conditions capable of determining the probability of detection for a search radar together with the false alarm rate for a specific target at a specific range can be created. Scan-to-scan or pulse-to-pulse target fluctuation scenarios can be computed and applied to the receiver under test.

A Waveform Generation Language for Arbitrary Waveform Synthesis

Easier to use than conventional programming languages, WGL is the primary front-panel interface for the HP 8770A.

by Derrick T. Kikuchi, Rafael F. Miranda, and Peter A. Thysell

THE WAVEFORM GENERATION LANGUAGE (WGL) was developed by Hewlett-Packard as a software tool for waveform design and analysis. Running on HP 9000 Series 200 and 300 Computers, it is the primary front-panel interface used to control the HP 8770A Arbitrary Waveform Synthesizer. Depending on whether you are a beginning user, a programmer, or an applications engineer, WGL can be approached in different ways.

The Beginning User's View

From the beginning user's viewpoint, WGL is a reverse Polish notation (RPN) calculator that can operate on entire waveforms in the same way that normal calculators operate on single numbers. Waveforms can be easily created using single-command operations. In all, there are over 100 commands at the user's disposal, but only the most demanding waveforms require the use of more than a few of them.

Waveforms are created by performing these operations on three basic building blocks: ramps, constants, and noise. For instance, a one-cycle sine wave can be created by the following steps:

```
(1)  RAMP      {Create a ramp from −1 to 1.}
(2)  PI*       {Ramp between −π and +π.}
(3)  SIN       {Sine between −π and +π.}
```

Line 1 initializes an array of elements with linearly increasing values from −1 to +1. Line 2 multiplies all element values by π resulting in array element values from $-\pi$ to $+\pi$. Line 3 takes the sine of all array element values.

The advantage of using RPN is that intermediate steps of waveform development are visible. The above sequence created y = sin(x). We can continue to operate on this waveform:

```
(4)  SQ        {Yields y = sin²(x).}
(5)  COS       {Yields y = cos(sin²(x)).}
```

Step by step, we can immediately see the effects of each operation. If we make a mistake, we can back up a step by simply typing UNDO.

The waveforms that can be created by WGL are for the most part limited only by the user's imagination. Waveforms can be designed mathematically and/or graphically in both the frequency and time domains. This means that a waveform created in the time domain can be shaped in the frequency domain with a digital filter and then generated with the HP 8770A. In the frequency domain, spectral components are easily created, thereby allowing the user to generate extremely complex signals.

The Programmer's View

From the programmer's viewpoint, WGL is a threaded interpretive language (TIL). Threaded interpreted languages accept source commands and execute them directly, rather than having to be compiled. TILs have the additional advantage that new commands can be created in terms of old commands. The new commands can in turn be used to create even higher-level commands.

For example, a new command called SINEWAVE could be created by simply typing:

```
(6)  DEFINE SINEWAVE
(7)  RAMP PI* SIN
(8)  END
```

This new command can now be used just like any other WGL command.

The flexibility for the user to expand and change the WGL language easily and the speed advantage that comes from efficient execution make TILs very attractive. But many TILs have a number of major limitations: simplistic data stacks that do not protect the user from putting too much data in or taking too much data out, very limited data types and a need for many duplicated commands to handle each data type, hard-to-read and clumsy program control constructs, and the inability to have commands that can accept input both interactively and programmatically.

WGL differs from other reverse Polish notation TILs (like Forth and Forth variants) in several ways. One is the existence of a universal data stack capable of handling any of the WGL data types: waveforms, numbers, and strings. The effect of an operation varies depending on the data on the stack so that addition (+) results in a single number when adding numbers but results in a waveform when adding waveforms. Thus, WGL can get away with having much fewer unique commands. Also, the data stack cannot be overflowed or underflowed, since it is managed like the stack in other RPN calculators.

Another WGL difference is that it keeps a program stack that is separate from the data stack. This eliminates accidental corruption of program flow information related to incorrect data manipulation. WGL also differs from RPN TILs in that program control structures aren't reversed. This eliminates hard-to-read constructs like:

```
IF    ELSE_STMT1    ELSE_STMT2    ELSE  THEN_STMT1
THEN_STMT2    THEN
```

In WGL, the construct reads:

```
IFTRUE    (THEN_STMT1    THEN_STMT2)
IFFALSE   (ELSE_STMT1    ELSE_STMT2)
```

A final major difference about WGL is that some of its commands can either interactively prompt for input or get their values from the data stack. For example, the SAVEWAVE command without any parameters will prompt the user for a filename. However, "filename" $ SAVEWAVE will cause SAVEWAVE to take its filename from the stack. The dollar sign ($) indicates that the next command requiring interactive input should instead get its information from the "script" currently on the data stack.

For the programmer, WGL is a highly structured language that has the flexibility and ease of use of a threaded interpreted language while incorporating philosophies that overcome deficiencies that TILs normally have. This leads to easily developed programs.

The Application Engineer's View

Finally, we look at WGL from the application engineer's view. WGL makes application development based on digital signal synthesis easy because of its fast prototyping capabilities. Fast prototyping is possible because WGL is a general-purpose tool that manages all of the low-level tasks associated with array manipulation. Combined with a rich command set and the ability to operate in both the frequency and the time domain, this makes WGL a very powerful tool.

Of course, it is always possible for experienced programmers to implement various test signals in the language of their choice. Typically, the more complex a test signal is, the more complex the program to generate it. If the signal cannot easily be represented with an equation, or if several variations of the signal are needed, the programmer's job becomes much more difficult. The high-level array manipulation commands and graphic interface of WGL make it easy to prototype a wide range of signals.

Let's prototype a signal for an application that requires damped sine waves to simulate the effects of electromagnetic pulses (EMP). The signal is represented by the following equation:

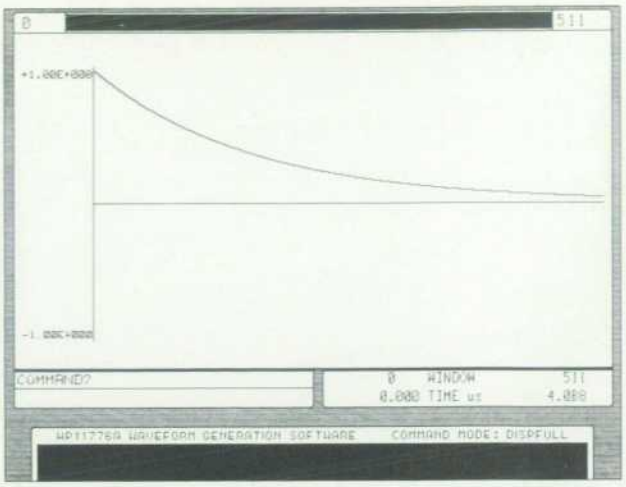
$$V(t) = e^{(-\omega t/2Q)}\cos(\omega t)\Big|_0^t$$

where ω is 2π times the sine wave frequency and Q controls the decay rate. The signal needed has the following parameters:

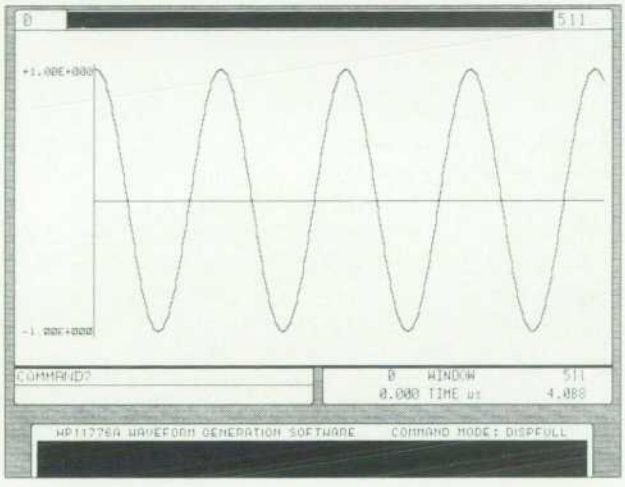
$f = 1$ MHz
$Q = 4$
$0 \leq t \leq 4.096\ \mu s$

By substituting these values into the equation for V(t), we can obtain:

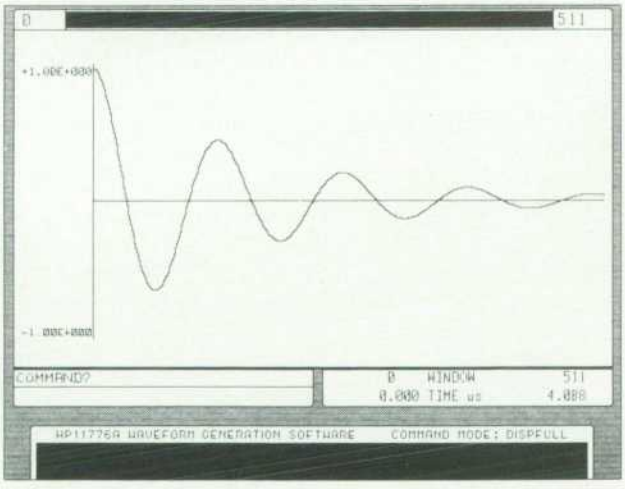
$$V(t) = e^{x_1}\Big|_{x_1 = 0}^{x_1 = -1.024\pi} \cos(x_2)\Big|_{x_2 = 0}^{x_2 = 8.192\pi}$$



(a)



(b)



(c)

Fig. 1. *Output produced by the WGL program example in the text of this article. (a) Output of line 2. (b) Output of line 4. (c) Output of line 5.*

This signal can be implemented in WGL with the following commands:

```
(1)  4.096E-6   SECS    CTX          {Set array size}
(2)  RAMP 1+    2/    -1.024 PI** EXP ?
                                     {Build decay envelope}
(3)  STORE A                         {Save decay envelope}
(4)  RAMP 1+    2/    8.192  PI** COS ?
                                     {Build 1-MHz sine wave}
(5)  A* ?                            {Damped sine wave}
(6)  DOWNLOAD GO                     {Generate actual signal}
```

Line 1 converts 4.096 μs into the appropriate number of array elements. CTX uses this number to establish the working array size.

Line 2 produces the exponential decay envelope (see Fig. 1a). RAMP 1+ 2/ fills the array elements with linearly increasing values from 0 to 1. The array elements of line 2 are then multiplied by -1.024π. EXP uses all the element values as exponents for e, resulting in the decay envelope.

Line 3 stores the decay envelope in waveform storage register A.

Line 4 produces the 1-MHz sine wave (Fig. 1b).

Line 5 multiplies the sine wave by the decay envelope, as in the original equation, resulting in the damped sine wave (Fig. 1c).

Line 6 sends the damped sine wave to the HP 8770A Arbitrary Waveform Synthesizer, where the waveform is generated.

In this application, the signal, a damped sine wave, is represented by a fairly simple equation. Why then couldn't we write a BASIC, Pascal, or C program to implement the equation? We could have. But let's look at the things we did not have to do or even worry about by prototyping in WGL:

- We didn't have to dimension any arrays.
- We didn't have to initialize any arrays.
- We didn't create a FOR-NEXT loop to process the equation for $0 \leq t \leq 4.096$ μs.
- We didn't have to increment the array element pointer to coincide with the increment of t.
- We didn't have to scale the resulting data to the values required by the digital signal synthesis hardware.
- We didn't have to write a graphics driver to see if the signal looked like the one we wanted.
- We didn't have to debug the program(s).
- We didn't have to write an instrument driver for the HP 8770A.

The Waveform Generation Language (WGL), therefore, is a tool easily approachable by a novice user and yet powerful enough to meet the programming and fast prototyping needs of today's sophisticated real-life signal requirements.

Hewlett-Packard Company, 3200 Hillview Avenue, Palo Alto, California 94304

HEWLETT-PACKARD JOURNAL

April 1988 Volume 39 • Number 2

Technical Information from the Laboratories of Hewlett-Packard Company
Hewlett-Packard Company, 3200 Hillview Avenue
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
P.O. Box 529, Startbaan 16
1180 AM Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

Bulk Rate
U.S. Postage
Paid
Hewlett-Packard
Company

5953-8568