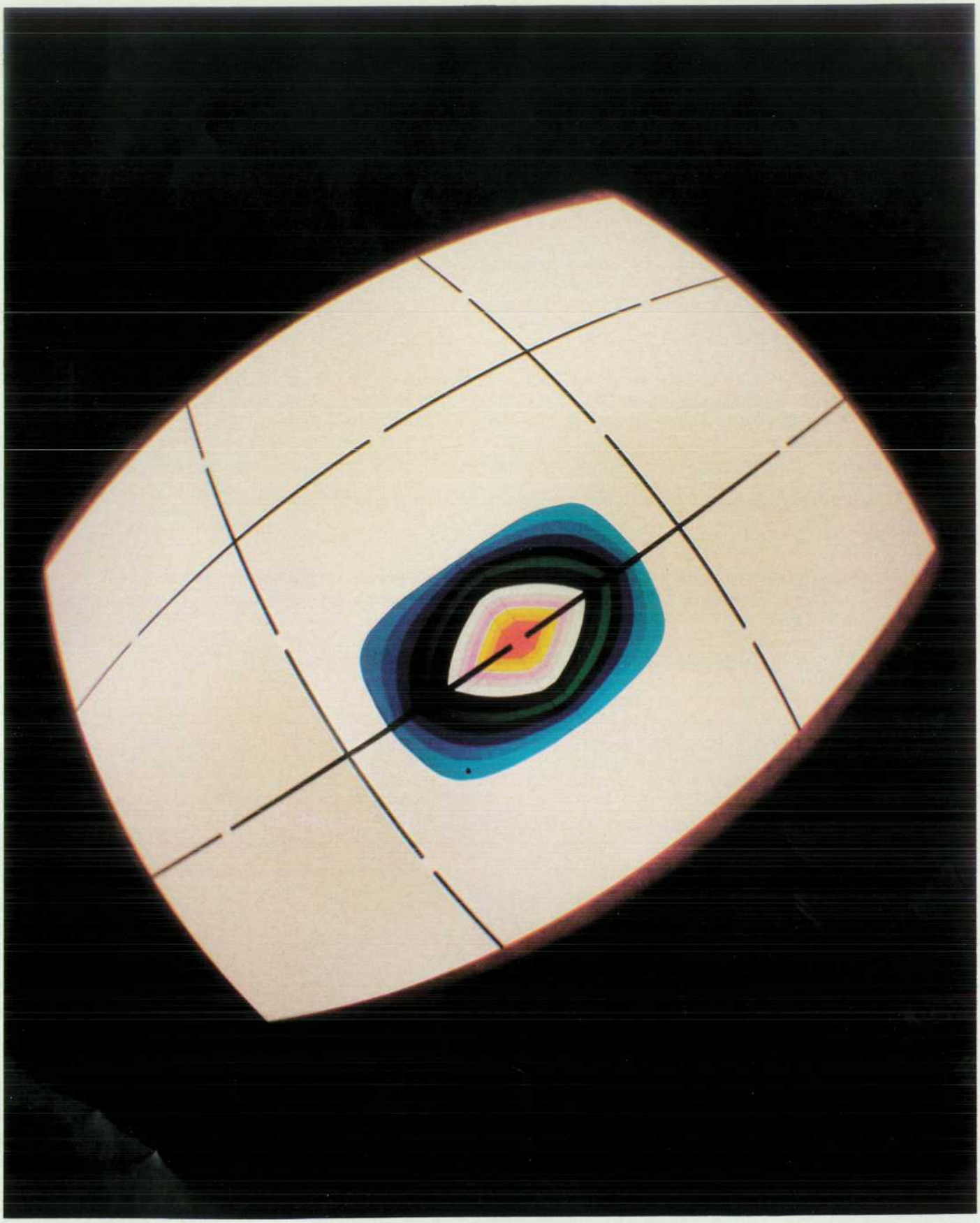
HEWLETT·PACKARD
JOURNAL

APRIL 1987



HEWLETT-PACKARD JOURNAL

April 1987 Volume 38 • Number 4

Articles

Research Reports

Departments

In this Issue

Like other instruments, signal generators are getting more and more built-in intelligence. One result is that some are beginning to offer more than the usual sine, square, triangle, and pulse waveforms. Arbitrary waveform generators that will reproduce whatever waveform you program into them are now commercially available. This means that testing can be more realistic, using signals that simulate all of the conditions a device is likely to encounter in normal operation. The HP 8175A Digital Signal Generator offers arbitrary waveform generation as an optional capability of a digital data generator. Depending on how you configure it, it can produce 24-channel parallel data patterns or two-channel serial data for testing digital hardware, two arbitrary analog waveforms for testing analog circuits, or a combination of digital and analog waveforms for testing such things as analog-to-digital and digital-to-analog converters. Its outputs can be independent or synchronized. In the articles on pages 4 and 12, the HP 8175A's designers tell us how it works. Noteworthy is their program-controlled looping scheme that makes a 1K-bit memory look much deeper. They use variable timing to conserve data memory, and they've been careful to make the arbitrary waveform generator easy to program—you can enter a series of levels or a mathematical formula, or point with a cursor on the display.

HP's Semiconductor Productivity Network (SPN) was an early approach to computer-integrated manufacturing for the semiconductor manufacturing industry. SPN consists of several software modules, each controlling or monitoring a particular aspect of the semiconductor manufacturing process. Recently, HP decided to withdraw from this business, to support only the current SPN users, and not to work on further enhancements. However, one development effort was allowed to proceed, so that current SPN users could have a complete system. The final SPN module, PL-10, is a master planning tool. In the article on page 21, PL-10's designers describe its operation and explain why semiconductor manufacturing needs a special planning system.

Small printed circuit boards are often manufactured several at a time on large boards called subpanels. At a certain point in the manufacturing process, a router path is cut around each board, leaving only small tabs holding the subpanel together. This, of course, reduces the rigidity of the panel. To see whether a partially routed panel would deflect enough during subsequent handling to affect the components and solder joints on it, HP Laboratories researchers used finite element analysis (cover photo). They report their results on page 29. Their conclusion? The panel deflection will not exceed the maximum allowable, so costly modifications of the manufacturing process are not needed.

Conventional reliability theory often describes hardware reliability in terms of a failure rate—the percentage of the units in service that can be expected to fail in a given period of time. For hardware, the rate is usually higher for very new or very old equipment, and lower in between. Application of the failure rate concept to software is in its early stages. In the report on page 35, two HP engineers tell how they have modified an existing software failure rate theory to include the testing process, so that the failure rate observed at the end of testing can be used to predict the in-service failure rate after release of the product.

-R. P. Dolan

What's Ahead

The development of ME Series 5/10, a mechanical engineering CAD product for HP's Design Center workstations, is the primary subject of the May issue. In addition, a short report discusses the potential use of ac power lines for data communications in buildings.

The HP Journal encourages technical discussion of the topics presented in recent articles and will publish letters expected to be of interest to our readers. Letters must be brief and are subject to editing. Letters should be addressed to: Editor, Hewlett-Packard Journal, 3200 Hillview Avenue, Palo Alto, CA 94304, U.S.A.

Digital Signal Generator Combines Digital and Analog Worlds

This new generator provides 24 parallel or two serial data channels, two arbitrary waveform analog channels, or a combination of digital and analog outputs.

by Uwe Neumann, Michael Vogt, Friedhelm Brilhaus, and Frank Husfeld

THE HP 8175A DIGITAL SIGNAL GENERATOR, Fig. 1, is capable of providing up to three different types of output stimuli depending on the selected configuration. It can be configured by simple keystroke as either a 24-channel parallel data generator, a two-channel serial data generator, or a two-channel arbitrarily programmable waveform generator. A combination of two types of output stimuli is also possible. For instance, a digital data sequence can be generated while the analog equivalent or a different analog function is generated by the analog part of the machine.

A large number of different applications can be supported by this type of stimulus capability. The HP 8175A can generate digital patterns for at-speed functional testing, simulation, and stimulation of digital hardware, and analog functions for medical applications (e.g., electrocardiographic and plethysmographic signals) and mechanical applications (e.g., ignition-point measurements in the automobile industry). Simultaneous generation of arbitrary waveforms and digital patterns bridges the gap between the analog and digital worlds, and is very useful for testing digital-to-analog and analog-to-digital converters, because the analog and the digital signals are available at different outputs of the same source. Add a comparator, and a go/no-go test can be performed. Testing of mixed ICs or circuits that include both analog and digital components is very easy, since it is possible to generate independent synchronous analog and digital data streams. For example, to test a programmable filter, the data patterns can control the filter's Q factor and center or corner frequency, while the analog output stimulates the IC.

Fig. 2 shows the HP 8175A system. Fig. 3 is a block diagram of the HP 8175A.

Parallel Data Generator

Data patterns can be generated on 24 channels, each 1K

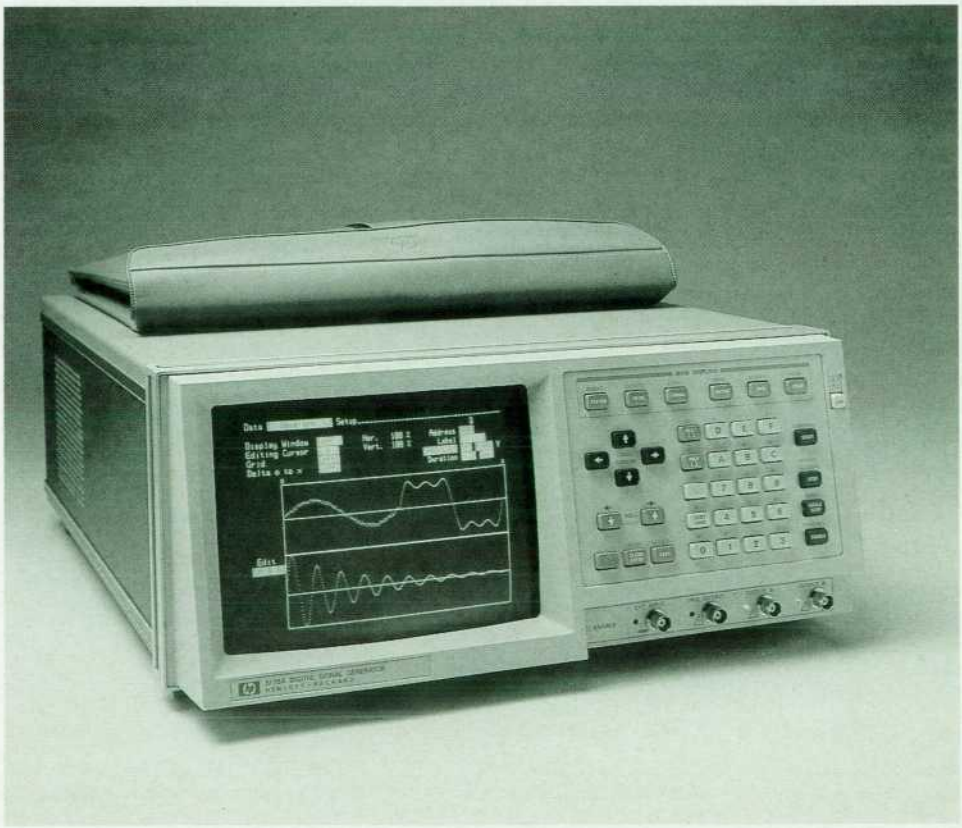


Fig. 1. *The HP 8175A Digital Signal Generator can be used as a 24-channel, 50-Mbit/s parallel data generator, a two-channel, 100-Mbit/s serial data generator, a two-channel arbitrarily programmable waveform generator (display shown), or a combination data and waveform generator.*

bits deep, at 50 Mbits/s. Two machines can work together in a master/slave configuration to generate up to 48 channels. To overcome the depth limitation, two enhancements help the customer minimize the number of memory locations used. One is virtual memory expansion—a pattern sequence that is to be used repeatedly needs to be stored only once in the machine and can then be included in the output data stream as often as required. The other is programmable pattern duration—each pattern consists of data and duration parameters, so sequences of fast bursts and long steady-state signals can be generated by using only a few memory locations.

Interaction with the device under test is facilitated by a trigger input, which can indicate start, stop, continue cycling, enable or disable outputs, or branch to other data cycles. Branching can be implemented, for example, when a DUT requires initialization routines. The trigger pins can be assigned to monitor logic states at various points of the DUT.

A fine-timing option provides a programmable delay on four channels. The delay can range from 20 ns to 40 ns with 100-ps resolution.

The HP 8175A Digital Signal Generator and an HP 163x Logic Analyzer form a versatile stimulus-response system for efficient digital testing.

Serial Data Generator

Internal serializing of eight parallel channels generates a data stream with 8K-bit depth and a bit rate of 100 Mbits/s on two channels, and a clock signal. The switch from parallel mode to serial configuration is done by a single keystroke. The same interaction features are available as for the parallel data generator.

Arbitrary Waveform Generator

If Option 002 is installed, the digital signal generator acts as an analog stimulus. The HP 8175A delivers an analog waveform on two fully synchronized channels with 10-bit amplitude resolution and up to 1024 data points. Virtual memory expansion and programmable pattern duration are also available to save internal memory locations, and the same DUT interaction capability is available as for the parallel data generator. The two channels can also be operated independently.

The maximum sample update rate of 50 MHz, the large amplitude range of up to 16V into 50Ω, and the minimum resolution of 0.2 mV meet the needs of a wide range of applications in electrical, medical, and mechanical environments. Other useful features include dc offset, settling times less than 20 ns, and synchronized parallel data channels if only one analog channel is used.

Different methods of entering the arbitrary waveform to be generated are possible. Waveforms can be generated by entry of a series of levels, by entry of a mathematical formula, or by pointing with the cursor on the CRT screen. More information on entering waveforms is given in the article on page 12.

Pods and Accessories

One trigger pod and three output pods are available to serve all interconnections to the device under test. The function of the HP 15461A ECL Pod, the HP 15462A TTL/CMOS Pod, or the fixed-level HP 15464A TTL Pod is to translate the internal signals of the HP 8175A into signals used by standard IC families like 10K, 10KH, or 100K ECL, 54/74 TTL, or 4000 CMOS. Each pod provides 8 of the 24 channels of the HP 8175A. The eight outputs of a pod can

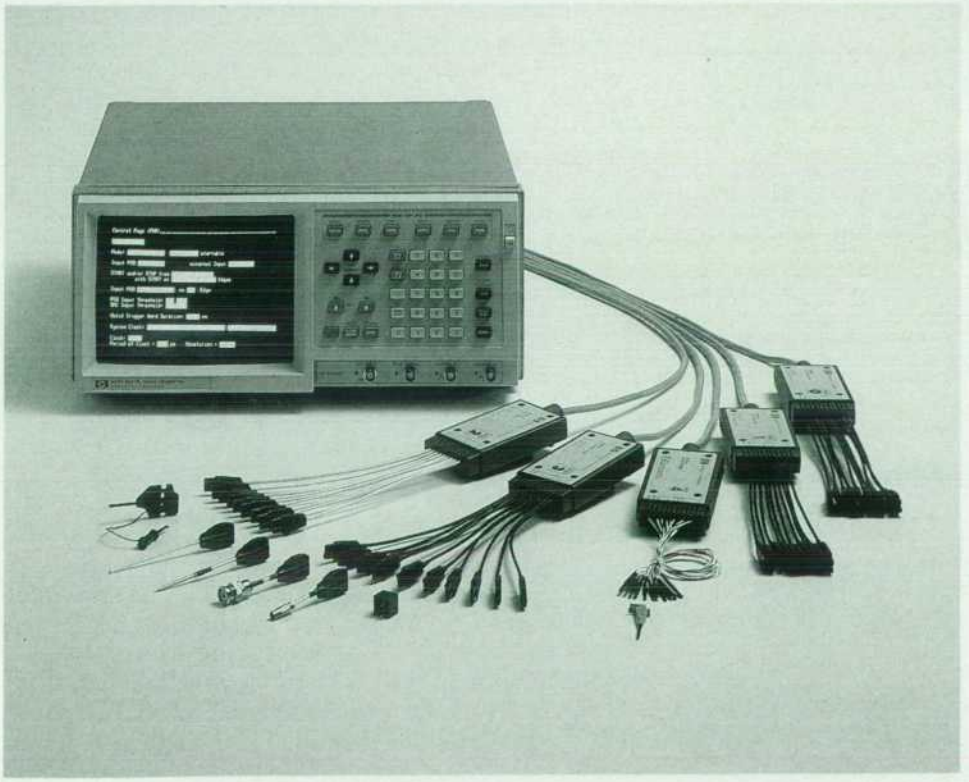


Fig. 2. *The HP 8175A system includes a trigger pod, three kinds of data or flag pods (ECL, TTL/CMOS, TTL), cables, connectors, and various probe accessories.*

be disabled, which for the TTL and TTL/CMOS pods means the outputs have a high impedance (tri-state outputs) and for the ECL pod means that the outputs are low to allow wired-OR connections in the device under test. In the TTL/CMOS pod the high level can be programmed by the HP 8175A from 2.4V to 9.9V. This provides flexibility for CMOS ICs, which operate at nominal supply voltages from 2V in low-power circuits up to 10V. An additional feature of the TTL/CMOS pod is direct control of the high level by the device under test. This requires that the external high-level control input of the pod be connected to the desired high-level voltage (Fig. 4).

The outputs of a pod are connected by coaxial cables to connectors of the type used on the HP 8182A Data Analyzer.[1] For this connector family a large number of accessories like plug-ins, crimps, and BNC or SMB plugs are available. The HP 8175A transmits its internal ECL signals via twisted-pair cables in the differential mode to the pods. Here, ECL differential line receivers amplify the signal either to ECL levels for the ECL and TTL/CMOS pods or to TTL levels for the fixed-level TTL pod. In the TTL pod, special TTL line drivers amplify the signal before it is connected to the outputs. In the TTL/CMOS pod, discrete output amplifiers are used to get the desired pulse performance at the outputs. All pod inputs and outputs are protected against electrostatic discharge (ESD), reverse voltage, and short circuits.

One of the design objectives was to have the pod housing as small as possible, because the space near the device under test is normally limited. Therefore, two techniques are used to miniaturize the pods. The ECL and TTL/CMOS pods are fabricated using ceramic thick-film technology. The fixed-level TTL pod, a low-cost version, is produced using surface mounted device technology (Fig. 5).

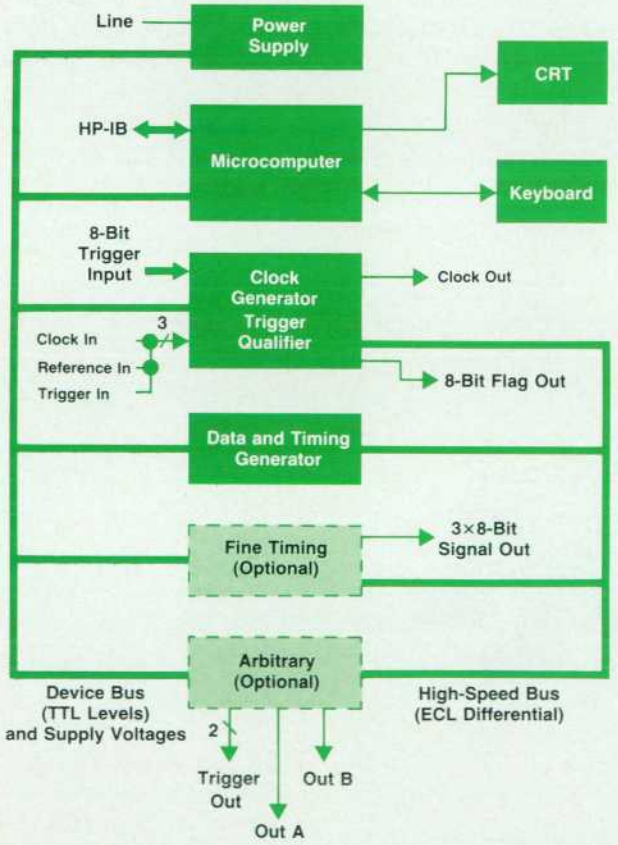


Fig. 3. *HP 8175A block diagram.*

User Interface

A major aspect of this project was the firmware development. A requirement was that the user interface be similar to that of the HP 163x logic analyzer family so that a customer who needs a stimulus/response system will find it easy to understand both machines.

A cursor-driven menu interface was selected. Any one of six major pages (System, Control, Timing, Output, Data, Program) can be selected by a single keystroke. The content of the main pages and their subpages depends on the chosen generator configuration, selected on the System page. The possibility of choosing different configurations depends on the hardware configuration.

The storage capability is also accessible via the System page. The nonvolatile memory can save two complete machine settings plus the current setup. Up to 255 different setups can be supported without an additional controller if an external disc drive is connected to the HP 8175A. Hard-copy output of data or screen settings to an external printer is also possible.

Implementation

As shown in Fig. 3, the HP 8175A Data Generator consists of a power supply, the microprocessor board, the clock board, which processes the input trigger signals and creates the clock signals for the whole machine, and the data board. The data board stores the patterns and generates the parallel data signals depending on the stored patterns, the pattern duration, and the programmed data stream. A buffer board drives the external pods, and if the fine timing option is installed, contains the delay lines for four channels. Also optional is the dual-channel arbitrary board, which provides the arbitrary analog waveform generation capability.

Interconnection between boards is the function of a motherboard, which provides the supply voltages, a TTL control bus for data interchange between the microprocessor and the other boards, and an ECL high-speed bus, which carries data and clock signals between the data board and the clock, buffer, and arbitrary boards. For easy servicing, a connector that can accept any board is provided on top of the motherboard. The internal self-test is capable of detecting almost any fault on any board and displaying an appropriate error code on the display. This helps minimize troubleshooting time and keep service costs low.

Data Board

The data board (Fig. 6) stores the information for the 24 channels and generates the timing of the data stream for the 24 channels. Behind each data channel is a high-speed ECL memory that has 1K-bit depth and a data access time of 10 ns. The timing generated by the data board is dependent on the clock frequency. With the internal 100-MHz clock, the shortest time interval is 20 ns and the longest is 9.99 s. The shortest increment is 10 ns. With an external

clock, the shortest interval is one clock period and the longest is 1,000,000 clock periods. The external clock input accepts clock frequencies from dc up to 100 MHz. For efficient use of the data memories, two special techniques are used to generate the data stream:

- Program-controlled looping of information
- Generation of variable timing.

In a data stream, certain patterns, synchronization information for example, are frequently repeated. The HP 8175A allows the user to create 256 information modules, each specifying a data pattern. The sequence in which these modules are used is controlled by the program.

Three different programs are available. The one to be used is selected by an external trigger event called the START, JUMP A, JUMP B vector. The START, JUMP A, JUMP B vector points to one of three entrance points of different programs (see Fig. 7). Another event called the CONT vector causes a program to be repeated continuously in the autocycle mode.

A vector, the START vector for example, represents an entrance point P0 and an endpoint Pn of a program. P0 through Pn are program sequences, and there can be up to 256 such sequences in the HP 8175A ($n \leq 255$). Each program sequence refers to an entrance point D0 and an endpoint Dm in the data memory, which is 1024 bits deep ($m \leq 1023$).

By this technique of program-controlled looping, the 1K bits of physical memory depth is increased to a much larger virtual memory depth.

Operation of the data board is as follows (see Fig. 6). The vector memory is addressed by the START, JUMP A, JUMP B or CONT vector received from the clock board. The program memory address counter reads the start and stop information of the program out of the vector memory and addresses the program memory, which contains the start and stop information for the data and timing memories. This information is loaded into the data memory address counter, which addresses the data and timing memories in parallel. From the timing memory, the pattern duration is loaded into the timing generator. At the end of each pattern, the timing generator increments the data memory address counter. The LATCH signal from the timing generator writes the pattern information from the data memory via the fine timing/buffer board to the outside world with precise timing. The increment signal of the data memory address counter fetches new data and timing information out of the data and timing memories. When the endpoint in the data memory address counter is reached, a new program sequence will be addressed, because the last data memory address counter increment also increments the program memory address counter. In single-cycle mode, the data stream stops when the last pattern of the last program sequence is completed. In autocycle mode the CONT vector will occur and start the program again.

Variable Timing

Because a normal data stream will rarely be a square wave, fixed timing of a data steam is inefficient. For example, a traditional data generator needs 18 memory locations for the data stream illustrated in Fig. 8. The HP 8175A, which has variable timing, needs only 8 memory locations for the same data stream. Fig. 9 is a diagram of the divider/timing generator that implements variable timing. The HP 8175A system clock is connected to the divider-counter blocks A and N. The divide-by-A counter has four constant division ratios: 1, 100, 10,000, and 1,000,000. These result

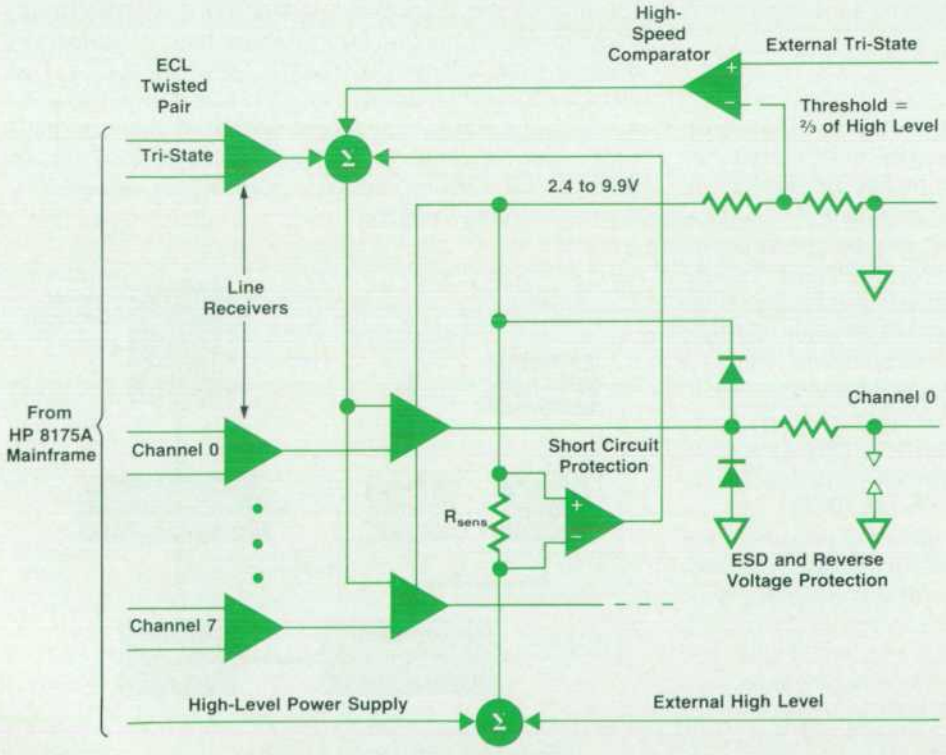


Fig. 4. *Diagram of the TTL/CMOS pod. This pod provides for control of its output high level by the user or by the device under test.*

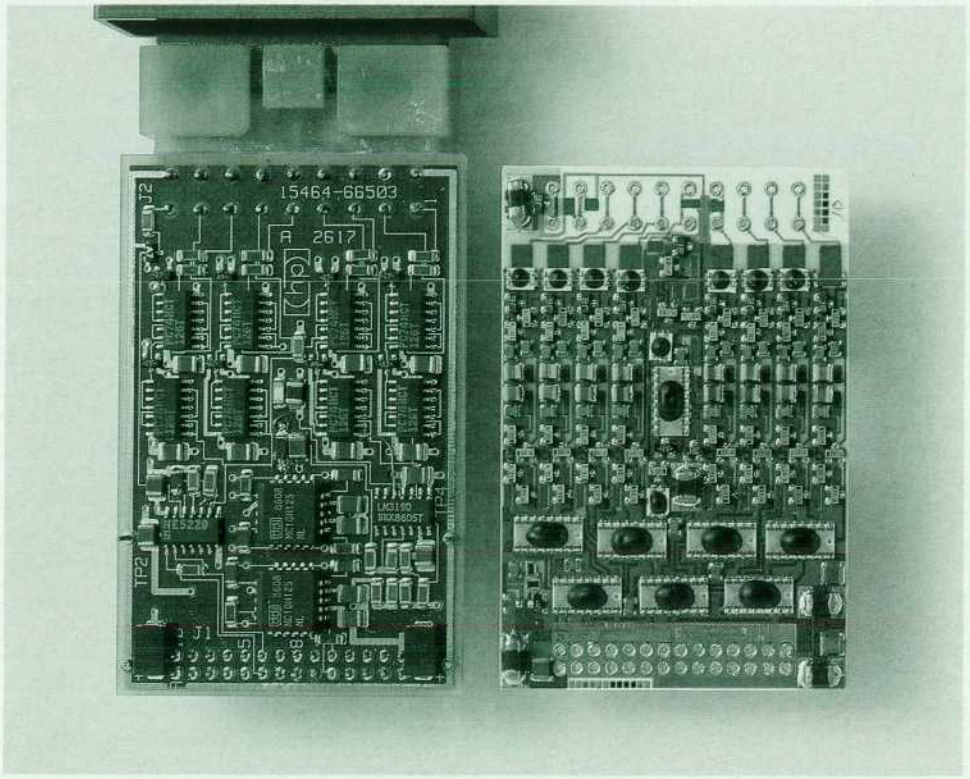


Fig. 5. *The TTL pod (left) uses surface mount technology. The TTL/CMOS pod (right) and the ECL pod use ceramic thick-film technology.*

in clock periods of 10 ns, 1 μs, 100 μs, and 10 ms, respectively, if the internal 100-MHz clock is used. The divide-by-N counter multiplies one of the basic periods by N, which is programmed by the timing RAM. The smallest value of N is 2 and the largest is 1023. The multiplexer, also controlled by the timing RAM, decides which basic period (value of A) is in use. This technique allows the user to program a 20-ns interval followed by a 9.99-s interval with the same memory consumption for both.

Clock Board

The internal system clock is a 100-MHz oscillator that has synchronous start/stop functions and an asynchronous function: the oscillator can be controlled by a phase-locked loop. Normally, if an oscillator is controlled by a phase-locked loop, it runs continously, and a start/stop signal can only gate its output. In a data generator like the HP 8175A, this could result in a timing deviation of up to one clock period (10 ns) in the output data patterns. In some applications when the generator must react to conditions in a DUT this might be a problem, so a synchronous start/stop oscillator that can be started and stopped in real time is required. In the HP 8175A the system clock can be switched between phase-locked operation and synchronous start-stop operation either from the front panel or via the HP-IB.

To ensure a high-performance start/stop oscillator we decided on a delay-line oscillator with a defined delay line and 100K ECL gates. The frequency of this oscillator type depends on the length of the delay line (nearly 4 ns fixed) and the propagation delay of the 100K ECL gate. This delay is 0.5 ns to 1.3 ns, which means that different gates can result in frequencies between 92.5 MHz and 108.5 MHz. Investigations proved that this was the case, so we included an adjustment circuit to provide an adjustment of −9 MHz to +11 MHz. This adjustment is done by a variable resistor at the input of the gate, as shown in Fig. 10.

Addition of the resistor influenced the time constant of the feedback signal at the gate input. Now the problem was that the oscillator could oscillate at its third harmonic (300 MHz in this case). To avoid this problem we decided on a flip-flop circuit that shuts itself off for 4 ns. The timing diagram and circuit in Fig. 11 show how it works. The original 100K ECL gate is shown as gate 1 in Fig. 11. Low signals at time t0 at inputs 3 and 4 of gate 2, which depend on the output of gate 1, set the output of gate 2 to high. This output is connected to input 2 of gate 1 and closes this gate for a wrong oscillator edge, which can occur at time t1 at input 1 of gate 1.

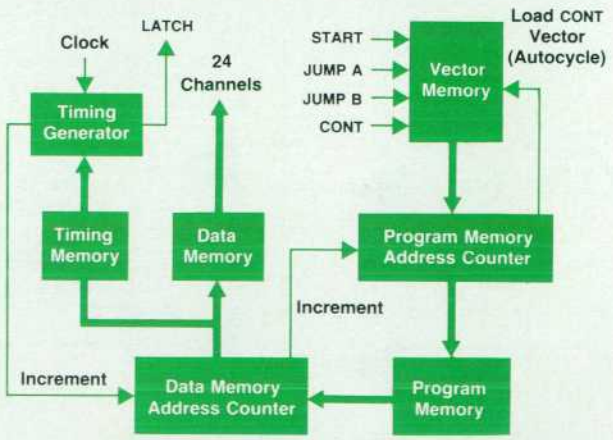


Fig. 6. *Data board block diagram.*

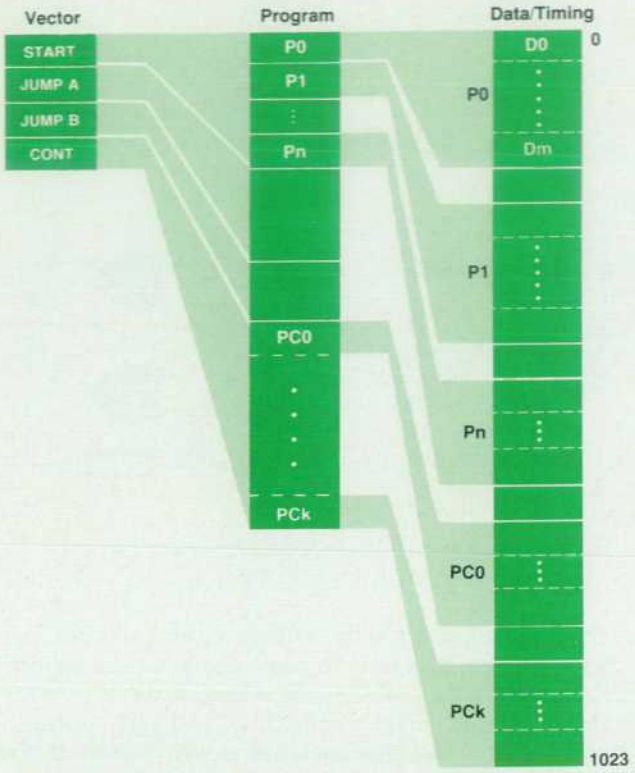


Fig. 7. *The* START, JUMP A, JUMP B *vector selects one of three programs. Each program can have up to 256 sequences, each of which occupies a section of the data memory, which is 1024 bits deep. This arrangement gives a virtual memory depth much greater than 1024 bits.*

The next step was to develop a phase-locked loop circuit. A standard phase-locked loop is used (Fig. 12), which supplies a voltage proportional to the phase error between the internal quartz reference (an external reference can also be used) and the divided clock signal. This voltage is fed to a varicap diode circuit (see Fig. 13) which influences the slew rate of the falling edge by varying the capacitive load of the gate output. It provides a controllable frequency range of ±2.4 MHz.

The diode circuit can also be switched to the output of

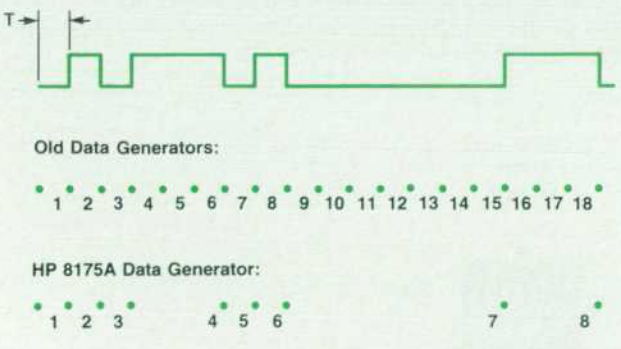


Fig. 8. *A traditional data generator needs 18 memory locations for the data stream illustrated. The HP 8175A has variable timing and needs only eight.*

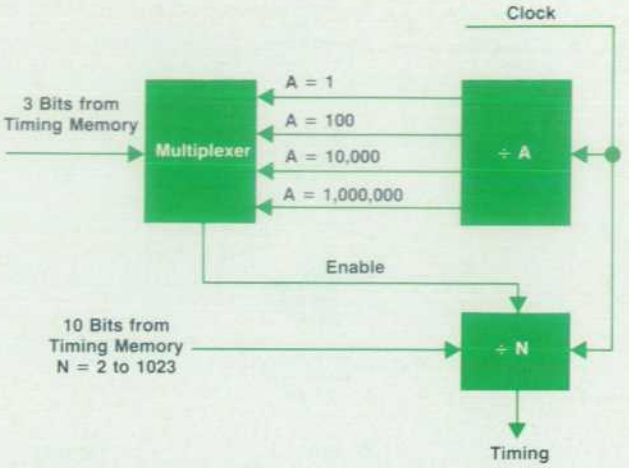


Fig. 9. *Variable timing is implemented by this divider/timing generator.*

a digital-to-analog converter. This provides the capability of recalibrating the oscillator in the synchronous function mode, when the phase-locked loop isn't locked.

Arbitrary Waveform Generator

The main function of the arbitrary board is the generation of two independent analog signals—the arbitrary waveforms. These are derived from the data and timing values entered by the user. The values are stored on the data board. Under control of the microprocessor, the arbitrary board generates the output signals for channels A and B and the two corresponding trigger outputs.

The arbitrary board consists of six functional areas, as shown in Fig. 14:

- Device bus interface. This is the interface to the microprocessor-produced control signals.
- Vernier control circuitry. This produces the amplitude range and offset control voltages.
- Trigger circuitry. This produces ECL or TTL trigger signals as required.
- Output A. This provides digital-to-analog conversion and power amplification. It buffers the channel A data from the data board and feeds it to the fast DAC and the

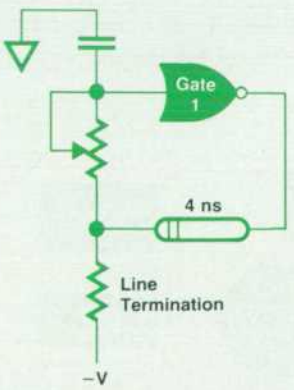


Fig. 10. *Delay-line oscillator has an adjustment for gate delay variations.*

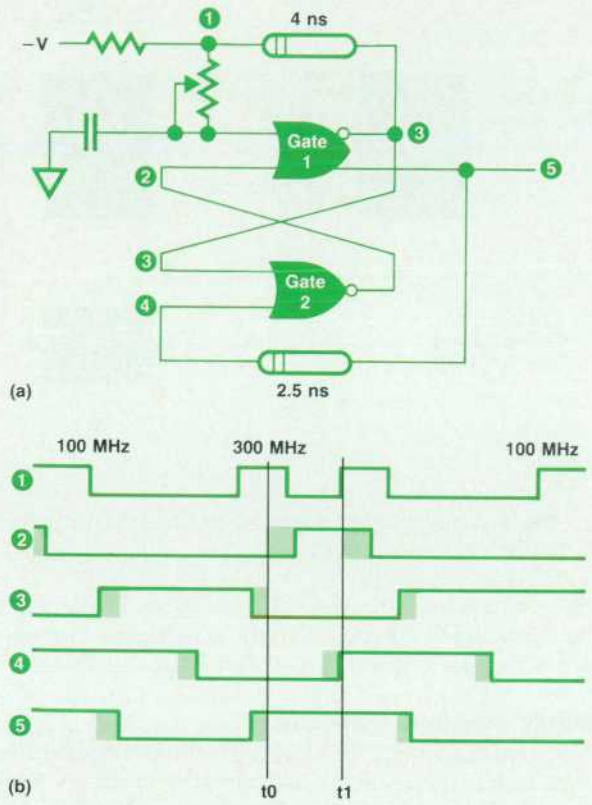


Fig. 11. *Flip-flop circuit prevents third-harmonic oscillations.*

power amplifier and attenuators for the channel A output.

- Output B. This provides the same functions as output A, but for the channel B output.
- Calibration circuitry. This performs an autocalibration of the amplitude and offset DACs each time the HP 8175A is powered up.

Fig. 15 is a more detailed block diagram of the arbitrary waveform generation circuitry. The binary data streams from the data board, ten bits wide for each channel, go through differential line receivers, which act as buffers, and a set of ECL latches to the multiplying ECL DAC, a Plessey SP9770B. The multiplying input of the DAC is used for coarse and fine control of the output amplitude. The amplitude factors are ×1, ×0.8, ×0.5, and ×0.4. These factors can be fine-tuned ±3.2%, ±4%, ±6.4%, and ±8%, respectively, for calibration purposes.

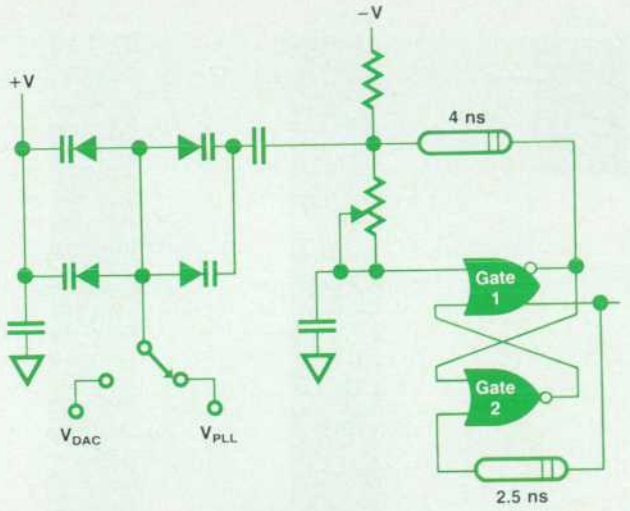


Fig. 13. *A varicap diode circuit provides a controllable frequency range of ±2.4 MHz at 100 MHz.*

The complementary outputs of the multiplying DAC can be interchanged by a relay to perform the normal/complement function, and are fed to a summing amplifier stage to obtain a single-ended signal. This signal passes through a switchable 1/5 attenuator to the power amplifier. The amplifier delivers up to ±8V into a 50Ω load or ±16V into an open circuit. Its design is derived from the HP 8116A and HP 8111A power amplifier designs.[2] A switchable 20-dB attenuator follows the power amplifier.

The amplitude range and offset controls are generated in four independent 12-bit AD7548 DACs. The amplitude range controls influence the multiplying inputs of the high-speed ECL DACs. The offset control DACs generate the voltages for the offset voltage inputs of the output power amplifiers.

All control signals for the normal/complement switches and the various attenuator relays are delivered from the device bus interface. Data for the amplitude range and offset control DACs also comes from the microprocessor via this interface. The DACs' control function requires two eight-bit addresses for each DAC.

Two of the 24 data channels of the data board can be programmed for trigger purposes. These two channels bypass the bus buffers on the arbitrary board and go instead through differential power amplifiers capable of delivering ECL and TTL levels into a 50Ω load. The trigger circuitry

(continued on page 12)

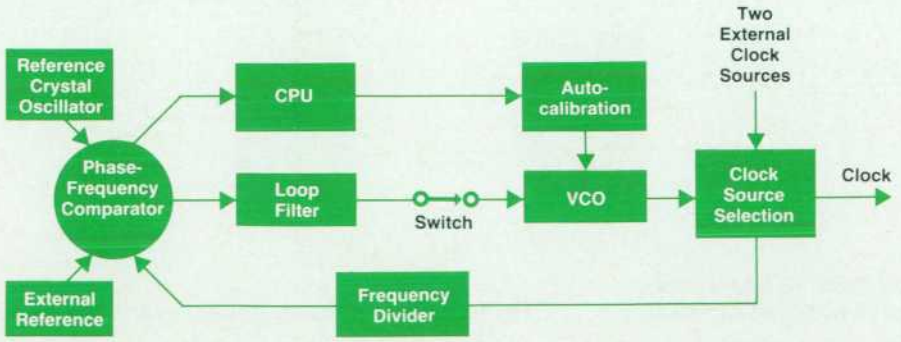


Fig. 12. *For phase-locked operation, the delay-line oscillator is embedded in a standard phase-locked loop.*

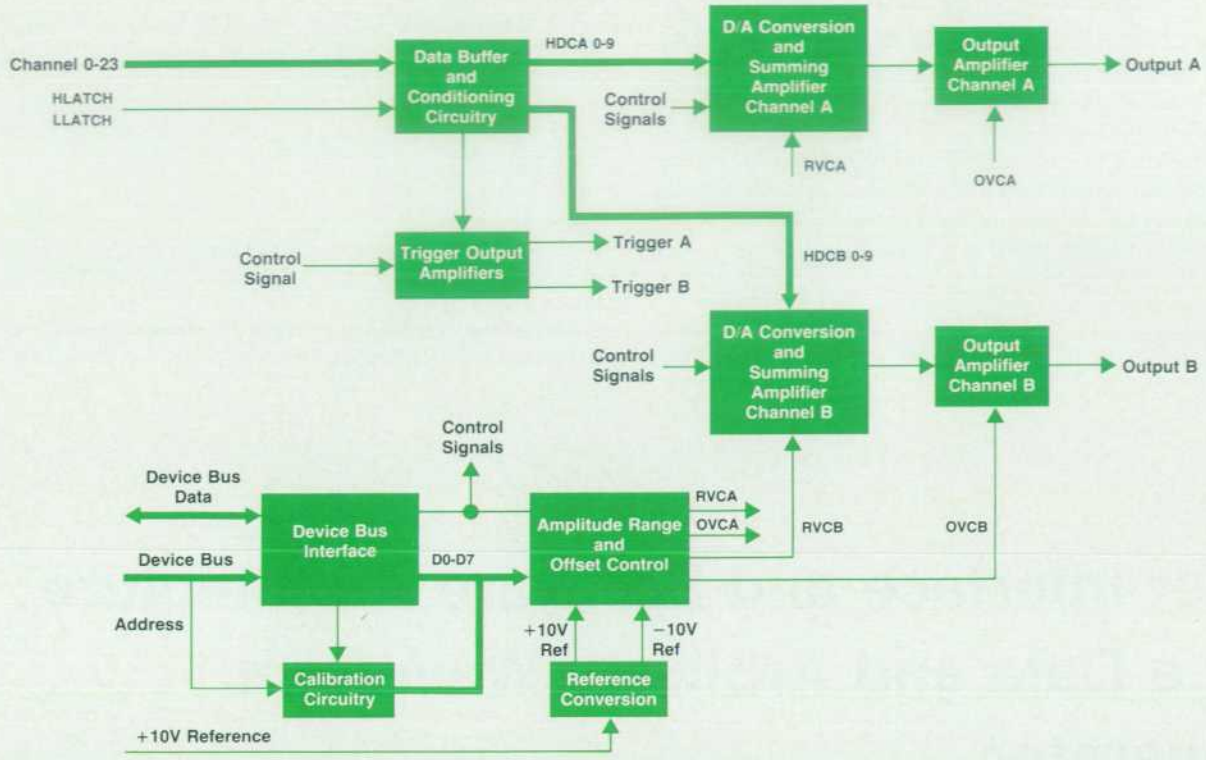


Fig. 14. *Arbitrary board block diagram.*

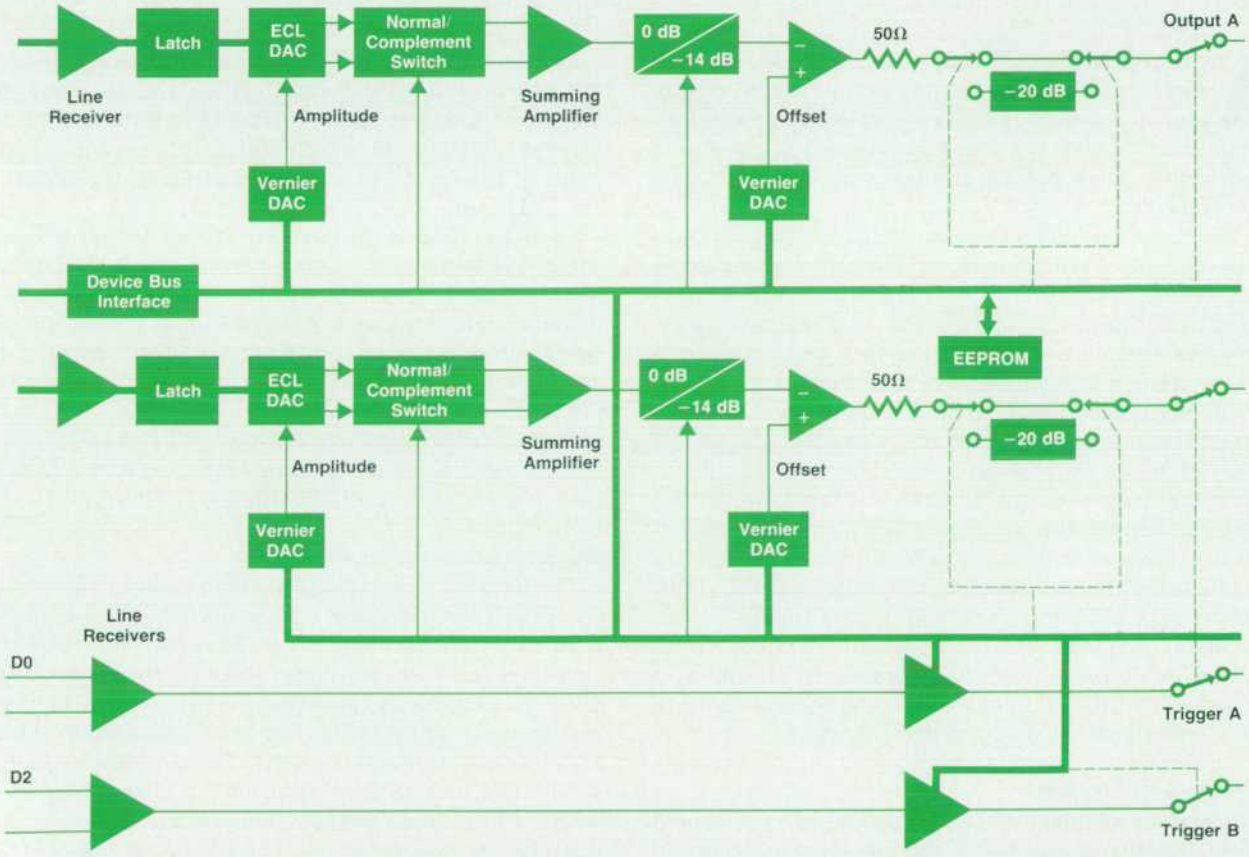


Fig. 15. *Arbitrary waveform generator functional block diagram.*

(continued from page 10)

is controlled by the microprocessor through the device bus interface.

The calibration circuitry consists essentially of an electrically erasable programmable read-only memory (EEPROM). During the adjustment process, which takes place only in production or field service, correction values for each amplitude range and voltage range are measured for 50Ω and open-circuit loads. These values are stored in the EEPROM. They are read under control of the microprocessor each time the board is powered up.

Acknowledgments

We want to thank Peter Aue and Georg Trappe, who defined the instrument. Kurt Kuebler, Helmut Rossner, Thomas Fischer, and Emmerich Mueller contributed to the software, hardware, and mechanical design. Thanks also to the HP 1630 design team at the Colorado Springs Division for their support.

References

1. D. Kible, et al, "High-Speed Data Analyzer Tests Threshold and Timing Parameters," *Hewlett-Packard Journal*, Vol. 34, no. 7, July 1983.
2. M. Fleischer, et al, "A New Family of Pulse and Pulse/Function Generators," *Hewlett-Packard Journal*, Vol. 34, no. 6, June 1983.

User Interface and Software Architecture for a Data and Arbitrary Waveform Generator

by Ulrich Hakenjos, Wolfgang Srok, and Rüdiger Kreiser

A COMPLEX, VERSATILE, AND POWERFUL instrument like the HP 8175A Data Generator requires a human interface that is easy to learn and comfortable to use. The HP 8175A human interface is based on a cursor-driven menu concept similar to that of the HP 1630 Logic Analyzer.

A 9-inch CRT provides the user with all necessary information like status, parameters, etc. All instrument settings (data, operation conditions, etc.) are distributed over six main display pages, or menus (Fig. 1). Each main page allows the user to make particular types of settings. To enter or change a parameter, the cursor must be moved into the highlighted field, where the current value is displayed. Then the value or condition can be changed via the keyboard.

A special feature of the HP 8175A is that the user can preset changes of settings or conditions, such as data patterns or cycling conditions, while the HP 8175A is running. The new conditions can then be transferred to the outputs at some convenient time. A powerful new capability is the creation of timing diagrams (data generator) and waveforms (optional arbitrary waveform generator) by means of a graphic editor that includes several interpolation capabilities, a pattern editor, or a calculator mode.

Software Architecture

The system architecture can be described as a layered model consisting of four layers, the first three software and the last hardware (Fig. 2). The top layer contains the human interface, which is the bridge between the outside world and the instrument and contains the various menus and the HP-IB interface. The next layer is a real-time multitasking operating system, which performs the internal system control. The lowest software layer contains the drivers for the hardware.

Each data field on the menus is assigned to a corresponding driver number. If a parameter has changed, the corresponding driver number will be stored in a FIFO buffer called the driver queue. A specific task of the operating system (the hardware task) is to read the content of this queue periodically. If the queue contains a driver number, the hardware task takes this number and calls the corresponding driver routine to change the hardware state.

The lowest layer, the hardware layer, consists of the microcomputer, clock, data, fine-timing, and arbitrary boards.

Memory and I/O Allocation

The memory size exceeds the direct addressing capability of the microprocessor, a Motorola 6809. This microprocessor can address 64K bytes. To access the HP 8175A system memory of 288K bytes, the addressing capabilities of the microprocessor are extended by means of a memory management unit (MMU). The MMU transforms a virtual address into a physical address. The physical memory is subdivided into 2K-byte segments. To allocate a physical segment to a virtual segment, the system needs the virtual page number and the physical address of the segment. The operating system supplies these numbers and controls the

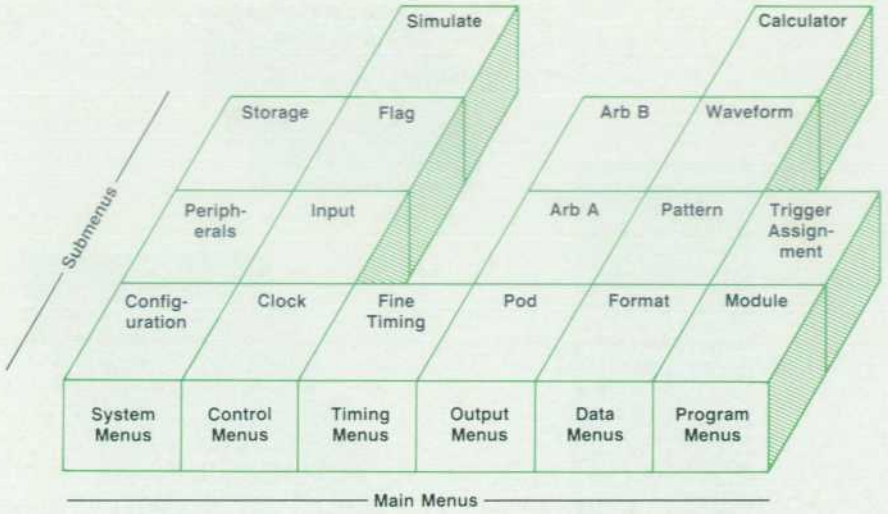


Fig. 1. *The user interface of the HP 8175A Data Generator distributes all instrument settings over six main display pages, or menus.*

operation of the MMU. Fig. 3 shows a typical memory configuration.

Editing Data

The HP 8175A provides comprehensive data editing support. Data can be edited in several modes, using either the pattern editor or the waveform editor. The arbitrary waveform generator provides two additional modes: a waveform calculator, which allows the creation of a waveform using a mathematical expression, and several interpolation functions, including linear interpolation and natural and periodic spline interpolation.

The pattern editor allows the entry of data in different formats including binary, octal, decimal, hexadecimal, and level (arbitrary waveform generator option), and provides a set of useful editor functions. An overview of the functions of the pattern editor is shown in Fig. 4.

Graphics

A significant feature of the HP 8175A is its graphics capability. Data set up using the pattern editing capabilities, the calculator, or the interpolation functions and stored in tabular form can be displayed graphically. A waveform displayed graphically is obviously far more meaningful than several pages of data.

The HP 8175A has a bit-mapped graphic capability with 512×256-pixel resolution. There are two graphic displays: one for the parallel/serial data generator configuration and one for the arbitrary waveform generator (see Figs. 5 and 6). For either case, numerical and graphical editing are possible.

On either graphic display, the values in the upper right numerical part of the display (address, label, duration, etc.) relate to the position of the currently active editing cursor. A window in which a specific part of a waveform is displayed can be defined (Fig. 6). The window can be scrolled. The active cursor remains within the window, so if the cursor tries to "escape" from the window, the window will also move.

By means of the editor, columns (addresses) on the display can be inserted or deleted.

Parallel/Serial Data Generator Graphics

The graphic display for the parallel/serial data generator is generated by means of a hardware formatter, which converts the information stored in memory to a form suitable

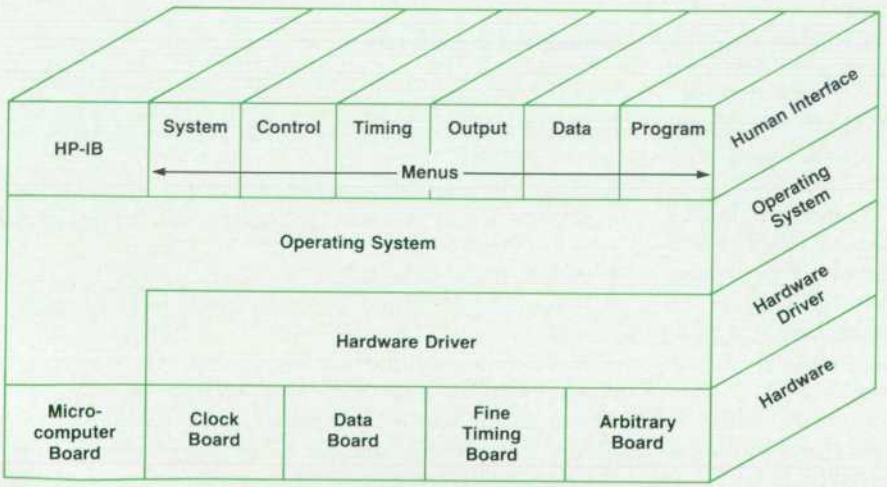


Fig. 2. *HP 8175A system architecture has four layers, three software and one hardware.*

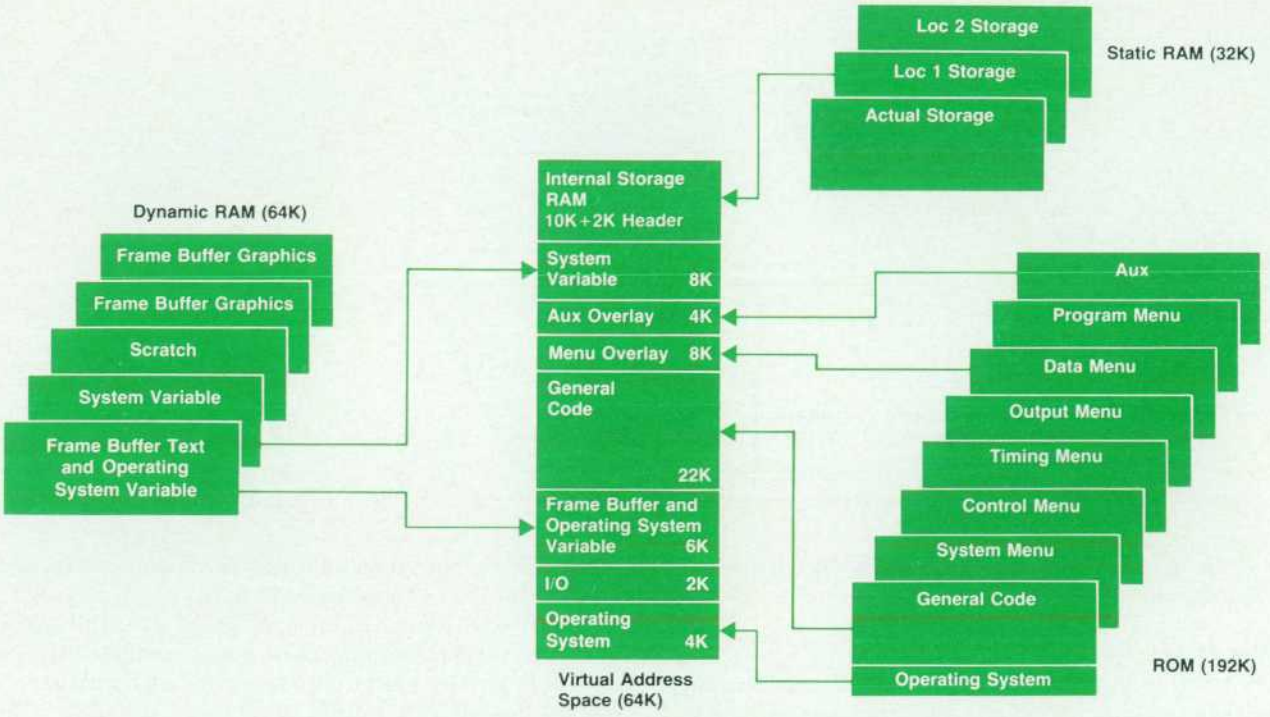


Fig. 3. *Typical HP 8175A internal memory allocations.*

for the CRT.

The required large dynamic range for time intervals, from 0.01 μs to 9.99 s for the internal clock and from 1 to 999,999 cycles/pattern for an external clock, led to the following implementation for the data generator graphic display. The screen is horizontally divided into ten vertical sectors. Time per division (or sector) can be set from 50 ns to 5 s in nine steps (Fig. 5). A small value expands the display, creating a small window. A large value compresses the display, creating a large window.

The position and width of the window and the cursor position within the window are displayed as bars on the lowest pixel row.

Up to 16 channels can be chosen for simultaneous display.

Arbitrary Waveform Generator Graphics

For displaying arbitrary waveforms, 1024 addresses with ten-bit resolution can be represented on a 512 × 256-pixel CRT. To ensure that no information is lost when 1024 addresses have to be accommodated in 512 pixel columns, each pixel column has two memory locations available for level information for an address pair.

The user can define a window as a subset of the full range to increase the resolution (zooming). For this purpose, in the upper right corner of the display, the full range of addresses and amplitudes is displayed. Within this full range, the user can define a window with width selectable from 100% down to 25% in three steps, and height selectable from 100% down to 3% in six steps (Fig. 6).

Two graphics cursors are available. Fast positioning of the active cursor and/or the window is possible by:

- Cursor swap
- Label quick search
- Scrolling
- Directly entering numerical addresses.

The physical screen can be divided vertically into as many as four logical screens (Fig. 7). The following modes are available:

- Only one curve
- Curve and associated trigger
- Two curves
- Two analog curves and their associated digital triggers.

Editing is possible in all of the previously described modes. However, the level increment depends on the chosen display mode. Other capabilities include voltage and time measurements between the two graphics cursors, and setting and deleting triggers.

Simulating Waveforms

In practice, it is not always possible or practical to describe a waveform mathematically. Instead, many waveforms need to be simulated (Fig. 8). Examples of such waveforms are:

- Medicine: heartbeats, nerve responses
- Material/electromechanical testing: disc drives, switching motors, relays
- Radar and sonar signals
- Waveforms for stimulating electrical/electronic components.

Waveforms are simulated with the HP 8175A by entering characteristic points and interpolating between them. Characteristic points can be graphically set and deleted in either of two ways, and can be interpolated by means of linear, normal, or periodic splines within any user-defina-

	Insert	Delete	Copy	Move	Modify	Function
HP 8175A Data Generator	Insert data line before pending line Data in inserted line is default set to zero	Delete pending data line	Copy data line from # to # → # Store data line from # to # → # (overwrites existing data) Copy data channel from # to #	Move data line from # to # → # Exchange data channel from # to #	Modify data and duration to new values from # to #	Increment or Decrement counter from # to # PRBS from # with shift register length x and feedback pattern
Arbitrary Waveform Option Option 002	Insert data line before pending line Data in inserted line is default set to data in pending line	Delete pending data line	Copy data line from # to # → # Store data line from # to # → # (overwrites existing data) Copy data from channel A to channel B from # to # → # Copy data from channel B to channel A from # to # → #	Move data line from # to # Exchange data from channel A with channel B	Modify -data -level -trigger -duration to new values from # to #	Interpolation -linear -natural spline -periodic spline from # to # and given characteristic points

Fig. 4. *Overview of the functions of the pattern editor.*

ble address range. Characteristic points can be deleted in blocks.

Interpolation

There are two principal methods of interpolating N characteristic points: by polynomials or by splines.[1]

When polynomials are used, N characteristic points are interpolated with a best-fit (N−1)th-order polynomial. From the N characteristic points, N equations are derived, from which the unknown coefficients of the polynomial can be determined.

Since, normally, the matrix of this set of equations has no zeros, the Gauss procedure can be used for solution. However, the computation time is proportional to the square of the number of unknown coefficients and can be

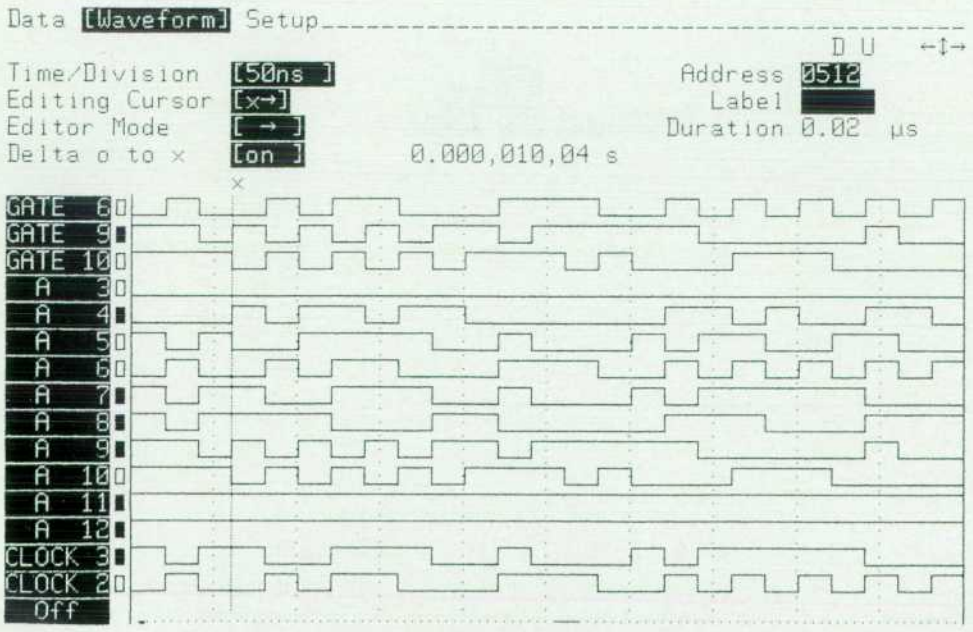


Fig. 5. *Graphic display for the data generator configuration.*

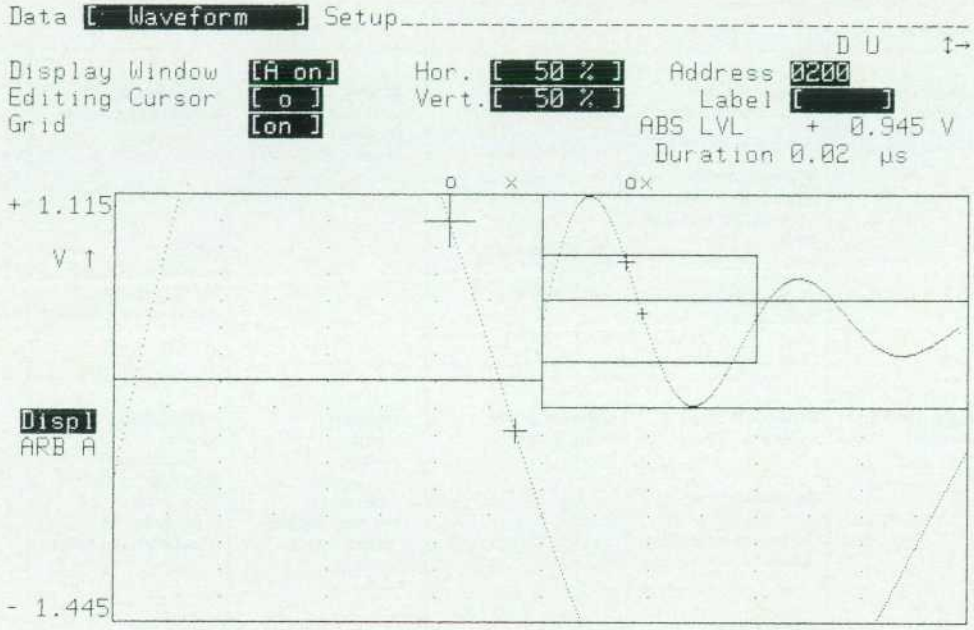


Fig. 6. *Graphic display for the arbitrary waveform generator configuration.*

very long. It is also possible to use the Horner formula, for which the computation time is linearly proportional to N. However, in cases where a large number of characteristic points are defined, this function oscillates at the interval margins.[2]

Splines

Because of these disadvantages of polynomial interpolation, the HP 8175A uses splines. The term spline can be traced back to the early days of shipbuilding, and relates to the way in which the hull curves were formed. With splines, the time required to compute the coefficients of a waveform is proportional to N, and the time required to determine an interpolated value is constant.

In spline interpolation, between each two consecutive characteristic points, third-order polynomial interpolation is done (Fig. 9). Thus, there are $(N-1)$ sectors, and $(N-1)$ third-order polynomials to be computed.

To determine the coefficients, four conditions must be satisfied. Two conditions are related to the requirement that the two characteristic points at the ends of a sector have to be elements of the resulting function. The other two conditions are related to the requirement that the slopes at the two margins of a sector must merge smoothly. That is, the first and second derivatives at the sector margins must be identical.

Additional conditions apply at the start and end of the interval to be interpolated. For example, for a periodic spline, the slopes at the start and the end of the interval have to be identical.[3,4]

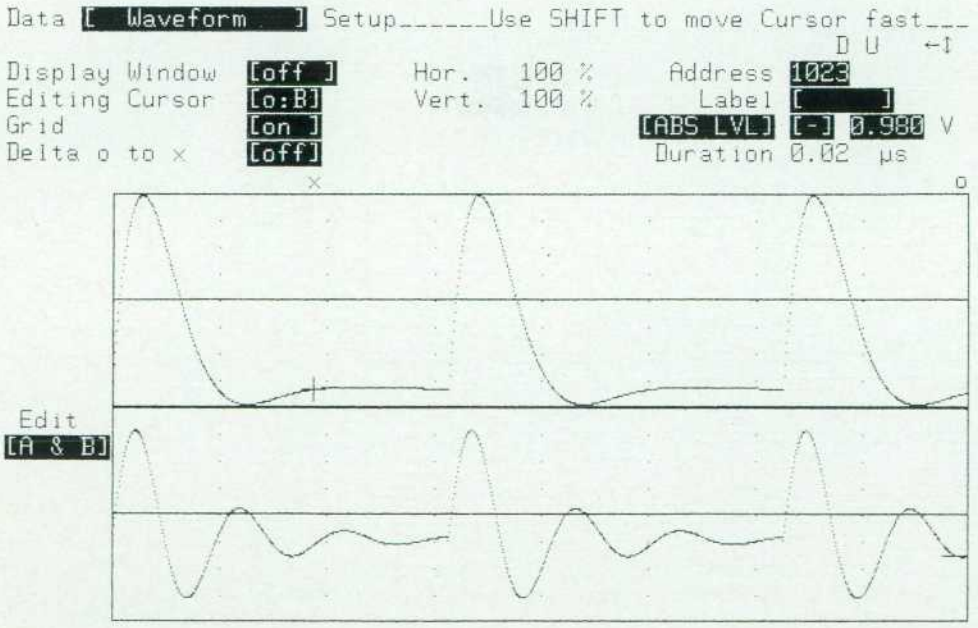


Fig. 7. *The physical screen can be divided vertically into as many as four logical screens.*

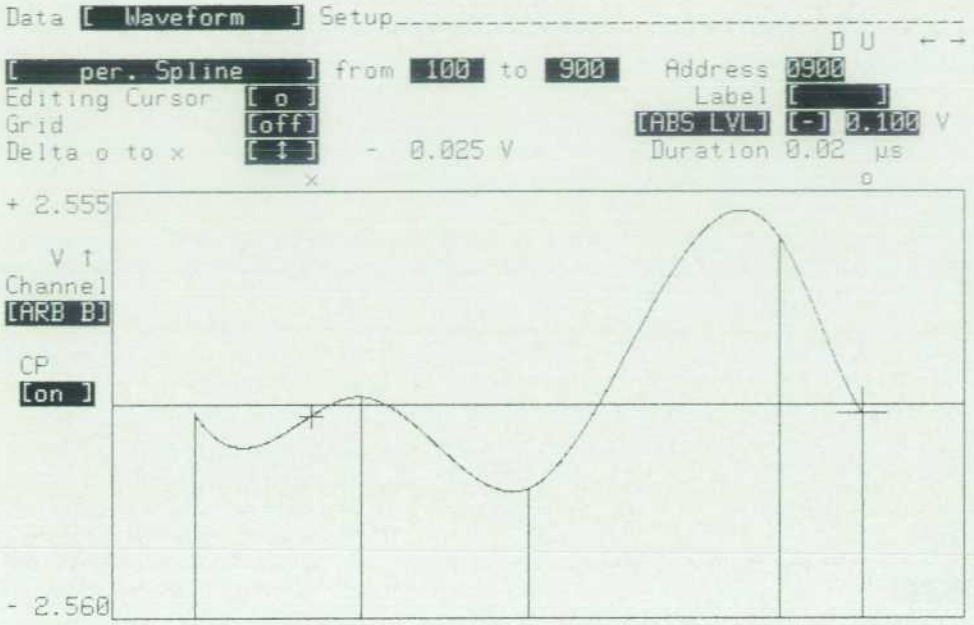


Fig. 8. *Waveforms that are impractical to describe mathematically can be simulated by entering characteristic points.*

For spline interpolation, the matrix of the set of equations is a cyclic tridiagonal matrix. Therefore, a simplified Gauss procedure can be used. The computation time is linearly proportional to N instead of N^2.

The user can estimate the trend of a natural spline function to be computed simply by imagining how a flexible ruler would be bent if it were confined within or supported by the characteristic points. There is no problem of oscillation at the sector margins.

The Calculator

The waveform calculator is a special capability of the arbitrary waveform generator option that enables the user to write and run programs to set up most complex mathematical waveforms by directly entering the formula in its written form. This can be done using only the HP 8175A keyboard; no separate computer is needed.

A comprehensive selection of standard statements and mathematical functions is available via softkeys. It is not necessary to type every character, so programs can be written very quickly.

The process that does the actual calculation and transfers the new amplitude and timing values to memory includes editing (setting up the function), setting the parameters, and finally, running the module. The following example shows how to set up and run a simple sinusoidal waveform.

Deriving the Mathematical Function

As already mentioned, using the calculator involves setting up the mathematical expression that represents the waveform. In practice, the first step is to derive this for the waveform. This is done as follows:

A sinusoidal waveform can be expressed as

$$y = \sin(\omega t) = \sin(2\pi f t) = \sin(2\pi f T_x)$$

where T_x is the instantaneous time. Within any amplitude range, waveform accuracy depends on the combination of maximum and minimum level values and the number of data points used. We will use 1000 data points for one cycle, which means that 1000 steps (level values) must be calculated (1024 steps are available but 1000 is an easier number to work with).

For a frequency of 10 kHz, the period is 100 μs. The timing window (the duration that the waveform will be calculated for) must be set to 100 μs and each incremental step of T_x must therefore be 0.1 μs. Calculator functions are set up as one or more modules, each of which consists of a header and an algorithm part. In this example the header defines the required time window of 100 μs and the incremental step duration of 0.1 μs. The algorithm part is the expression for the waveform. The complete module is shown in Fig. 10, where line 0 represents the header and line 1 the algorithm.

Syntax, Restrictions, and Limitations

The module syntax is similar to normal computer program statements. Fig. 11 shows the syntax diagrams. All

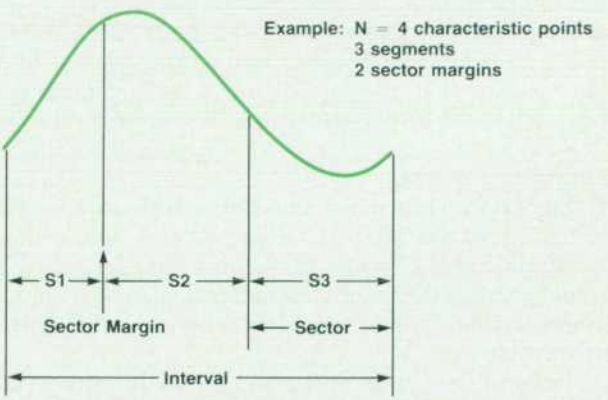


Fig. 9. *Third-order polynomial interpolation is done between characteristic points.*

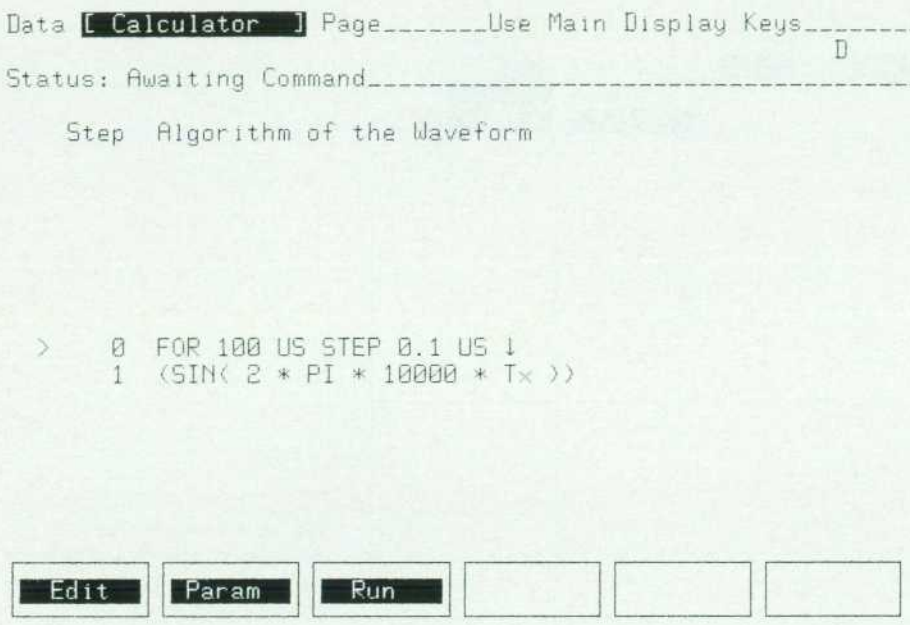


Fig. 10. *To define a waveform using the waveform calculator, functions are set up as one or more modules, each consisting of a header (line 0 here) and an algorithm (line 1).*

items enclosed by rounded envelopes are available directly via a corresponding softkey. Words enclosed by a rectangular box are names of items used in the statements.

The complete function can consist of 750 bytes. Each ASCII character, including spaces, requires one byte. Keywords such as FOR, STEP, etc. and standard functions such as SIN, LOG, etc. also require only one byte each. The step duration must be within the HP 8175A's timing (20 ns to 9.99 s) and resolution (10 ns to 0.01 s) capabilities. All calculations are done by means of a floating-point library with an internal 32-bit format. Therefore, the calculation time for the sinusoidal waveform is about 7 seconds, because most of the standard functions including SIN are derived from truncated series. The maximum relative error in these approximations is less than 0.00039% for the SIN function.

Setting the Parameters and Running the Module

After the function has been edited, it is compiled automatically by leaving the editor. If a syntax error is discovered, an appropriate error message will be displayed and the location of the error within the function will be indicated by the cursor. In such cases, any corrections required must be made before it will be possible to end the edit mode. If there is more than one error, another error message will be displayed after the first error is corrected. Only when all errors have been corrected is leaving the edit mode possible.

The next step is to define the relation between the edited function and the desired output voltage and arbitrary waveform channel. In Fig. 12, the first field defines which output channel the function is to be calculated for, and the second defines how to consider the levels, either with or without offset.

The amplitude range previously set on the output page, 5V for example, will be displayed. The required maximum and minimum levels can now be set. These are the amplitude values that are assigned to the maximum and minimum data values calculated from the function.

The Calculate from Address is the address from which the calculated data patterns will start.

The last step is to start the calculating process. The resulting amplitude and timing values are transferred to the HP 8175A's memory.

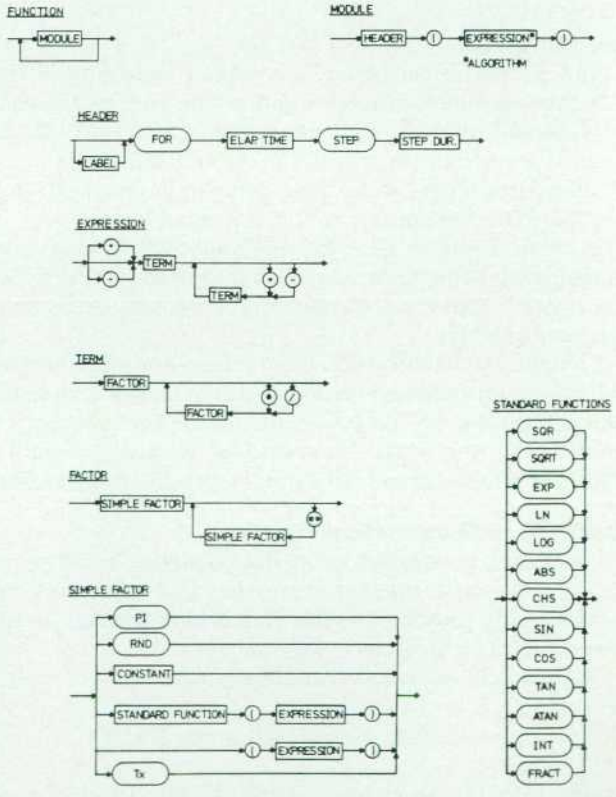


Fig. 11. *Waveform calculator syntax diagrams.*

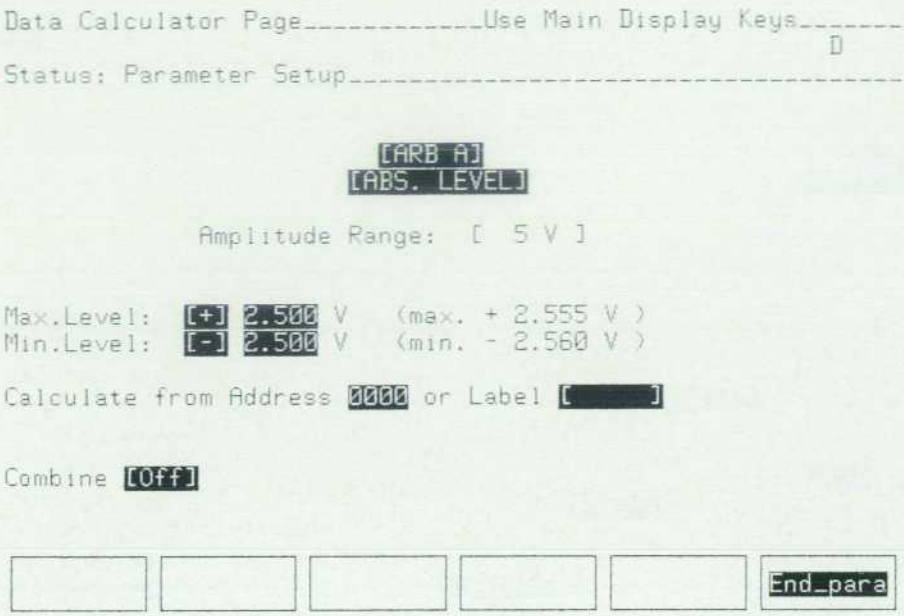


Fig. 12. *Defining output parameters for a waveform calculator function.*

Local Overlaying

Crosstalk simulation often requires noise, spikes, or other functions to be overlaid in a specific part of an existing waveform. The HP 8175A's Combine feature makes this possible and offers great flexibility. A function can be combined in a variety of ways with any part of an existing waveform. This is useful in cases where, for example, a waveform that is very difficult to describe mathematically and has therefore been set up graphically or by spline interpolation needs to have noise added to it. The noise waveform can be set up using the calculator and then combined (add, subtract, multiply, divide) with the original waveform. The signals can be combined in any ratio between +60 dB and −60 dB. An example illustrating the use of this capability is shown in Fig. 13.

For the sine wave example, the original sine wave module must be deleted and replaced with the noisy function. The Combine field of the parameter setup page must be set to On (Fig. 14). The Combine mode and the ratio of the two signals can then be set.

After calculation is completed, the HP 8175A's memory will contain the new waveform calculated from the old pattern and the new function (Fig. 15).

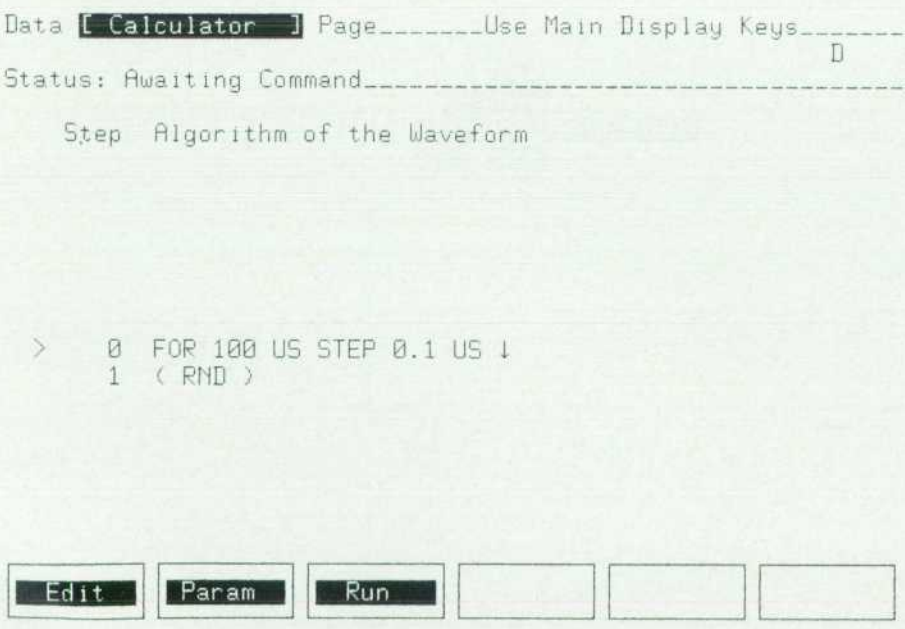


Fig. 13. *Definition of a random noise waveform to be added to a sine function.*

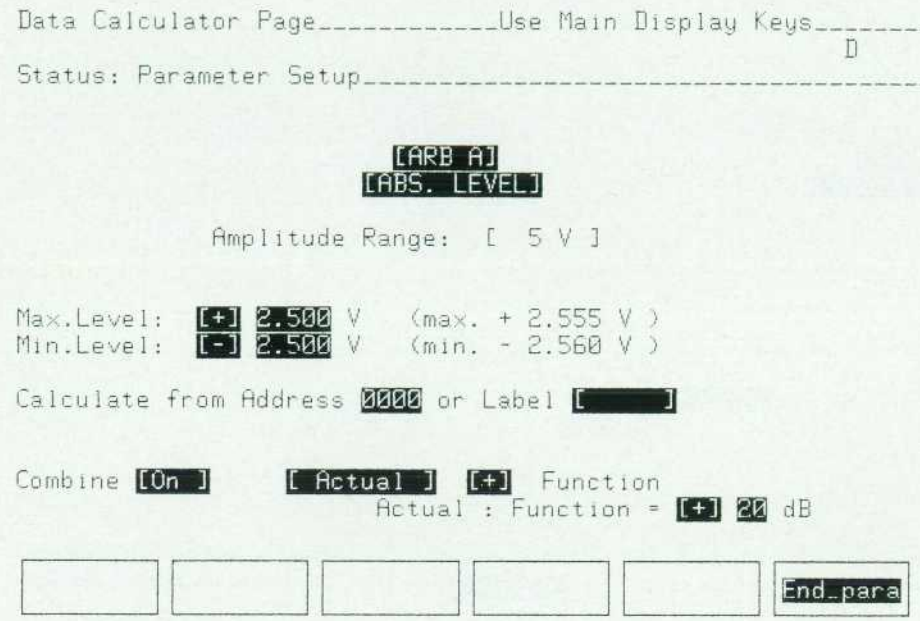


Fig. 14. *To combine functions, the* Combine *field is set to* On, *and the ratio of the two signals is specified.*

References

1. F.B. Hildebrand, *Methods of Applied Mathematics*, Prentice-Hall, 1965.
2. G. Alefeld and R.D. Grigorieff, *Fundamentals of Numerical Computation (Computer Oriented Numerical Analysis)*, Springer-Verlag, 1980.
3. J.H. Ahlberg, E.N. Nilson, and J.L. Walsh, *The Theory of Splines and their Application*, Academic Press, 1967.
4. C. Reinsch, "Smoothing by Spline Functions I/II," *Numerische Mathematik*, Vol. 10 (1967), pp. 177-183.

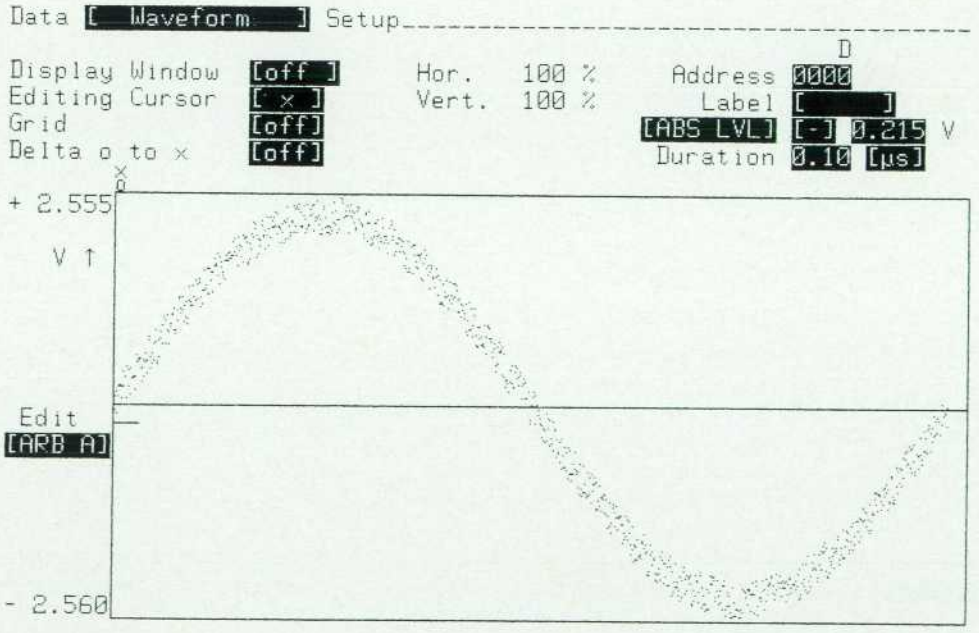


Fig. 15. *The result of combining noise and a sine wave.*

A Planning Solution for the Semiconductor Industry

Semiconductor device manufacturing has several situations that complicate normal production scheduling and medium-range planning. PL-10, part of HP's Semiconductor Productivity Network, was developed to deal with these peculiarities.

by Edward L. Wilson, Kelly A. Sznaider, and Clemen Jue

PL-10 IS A MASTER PLANNING TOOL developed by Hewlett-Packard for the semiconductor manufacturing industry. It addresses the task of medium-range production planning (up to 2 years) for a geographically distributed semiconductor division or company.

PL-10 helps the planner develop master production schedules for all major areas of a semiconductor company: wafer fab, sort, assembly, and test. The planner enters basic planning parameters—product demand forecasts, time-phased WIP (work-in-process) availability, product yields, and lead times—either manually or through batch files (see Fig. 1). PL-10 works backward from product demand to determine wafer requirements, which the planner can review and modify. It then projects forward from planned wafer starts to expected shipments. The planner can review rough-cut capacity use projections and associate component requirements with product demand. Repeated iterations of this process result in a workable companywide production plan.

PL-10 is currently in use in several HP semiconductor facilities and is being installed at several customer sites inside and outside the U.S.A. The release of the product unfortunately comes at a time when HP has decided to withdraw from the semiconductor CIM business. Despite this decision, development of PL-10 was deemed critical enough to continue with release plans, but the product will not be sold outside the current SPN user base and no further enhancements will be funded.

Semiconductor Manufacturing Planning

The semiconductor industry is a complex hybrid of batch and process manufacturing. While the basic job of scheduling production is the same in any industry, there are problems that make traditional MRP (material requirements planning) solutions difficult to implement:

No work orders. Semiconductor wafers are processed in lots which are sized according to the equipment they will be processed on. Typically they are not tracked by specific work order; rather, they are treated as a continuous flow of material. This creates problems for MRP systems that schedule work orders to meet end-item demand.

Inverse part structure. Semiconductor devices are grown from simple raw materials. Assembly (placing the semiconductor die into a ceramic or plastic package) occurs late in the manufacturing process. A single integrated circuit, or die, can be made into a variety of products depending on the circuit layout, the type of package used, and the tests administered. ROM chips have additional variations when a generic ROM wafer is programmed by a customized interconnection pattern defined in a final mask layer. This inverse part structure—simple bills of material with thousands of end items made from a few basic components—does not work well with traditional MRP. MRP helps planners answer the question, "What must I order to build the products I need?" The semiconductor planner must also answer the question, "What can I make with the material already in-process?"

Fluctuating yields. Semiconductor device yields can be unpredictable, especially for new processes. Normal MRP systems can handle the possibility of scrap, but may be unable to deal with situations such as:

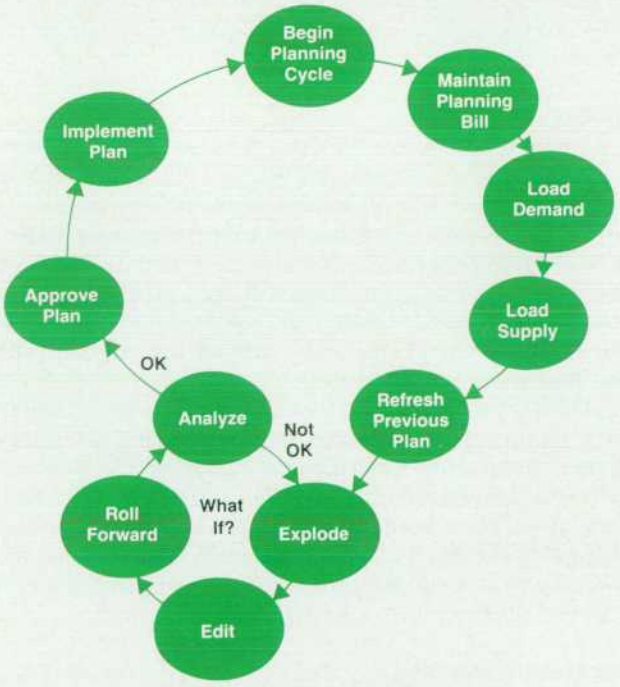


Fig. 1. *Planning cycle.*

- Wafer-to-die conversion. The number of good parts (die) on a wafer can vary widely. The planner must estimate the number of wafers needed to make a required number of parts. This is different from the standard MRP "Quantity Per" in which a truck, for example, always needs four wheels and cannot be built unless those wheels are available.
- Yield improvements. While yields fluctuate, they tend to improve over time. A planner who does not consider this gradual improvement will risk scheduling too many wafer starts in later periods.
- Binning. Not only does the yield of good material vary, but the very output of the process is not completely controlled. Finished circuits are subjected to a variety of electrical and environmental tests to determine the specifications under which they may be sold. The outcome of this binning process is subject to statistical distribution but is also influenced by the desired outcome of the process. That is, the set of tests done on the material depends on which grades of product are in demand. Material that fails a test may also be retested to a less-stringent specification (downgrading).

Worldwide planning. Nearly all semiconductor manufacturers do assembly in the Far East. Any planning system for the semiconductor industry must be able to schedule multiple sites at once. This involves allocating production between factories that have different delivery times and work schedules. To be fully functional, the planning system must be able to send and receive data to and from these factories without manual rekeying.

The semiconductor industry has gone through a radical shift in the last few years. When business was booming, manufacturers could sell everything they made. Accordingly, the emphasis was on improving yields and process throughput. With business currently declining, there is an increased emphasis on production planning—building the right product—and on matching capacity to uncertain demand.

History

As mentioned earlier, PL-10 is the last of the Semiconductor Productivity Network (SPN, see Fig. 2) products to be developed.[1,2] The precursor to PL-10 was created at HP's Corporate Manufacturing Information Systems (CMIS) to be a standard scheduling system for internal HP divisions that would eventually converge with HP's MM/3000 master scheduling package.[3] Instead, the software was used as the starting point for PL-10, with much of the original code and functionality carried over.

The first version of PL-10 was released in early 1985 to HP's Northwest IC Division and Loveland Technology Center, and one outside customer. Based on the feedback of these users, a substantial redesign was done. The second version was completed in August 1986, and included major changes to the data base, the production calendar, and the MRP Explode logic. New interface programs and the Roll Forward program were also added.

Integration with SPN

PL-10 extracts essential parametric and production data from IC-10, the SPN lot tracking system. The systems are loosely coupled by file interfaces.

Shop floor calendars, process yields, cycle times, part numbers, and part information such as cost and probe yields are examples of parametric data that can be extracted from IC-10. PL-10 obtains WIP status from a standard IC-10 extract file containing a snapshot of production—ending on-hand quantities for each process location. Using IC-10 process yields and cycle times, PL-10 forecasts these out to finished goods inventory to determine expected outs quantities and their expected due dates.

PL-10's Part Mapping function allows the planner to consolidate several production parts, spanning several IC-10 WIP and stockroom areas, into one planning part. Consolidation can be done on the basis of similarities in the part numbers; this is particularly useful for planning families of parts. The goal is to reduce the number of items that the planner has to forecast.

Since IC-10 and PL-10 are loosely coupled, either system can be installed first. The interface allows any inventory tracking system, such as MM/3000, to feed data into PL-10. PL-10 can thus be used to do production planning and scheduling for several tracking systems, either on the same computer and/or at remote computer sites. This flexible interface approach also applies to PL-10's customer order and forecast interface, which allows any order processing system to enter demand into PL-10. Forecasts can be uploaded from spreadsheet programs using the same interface.

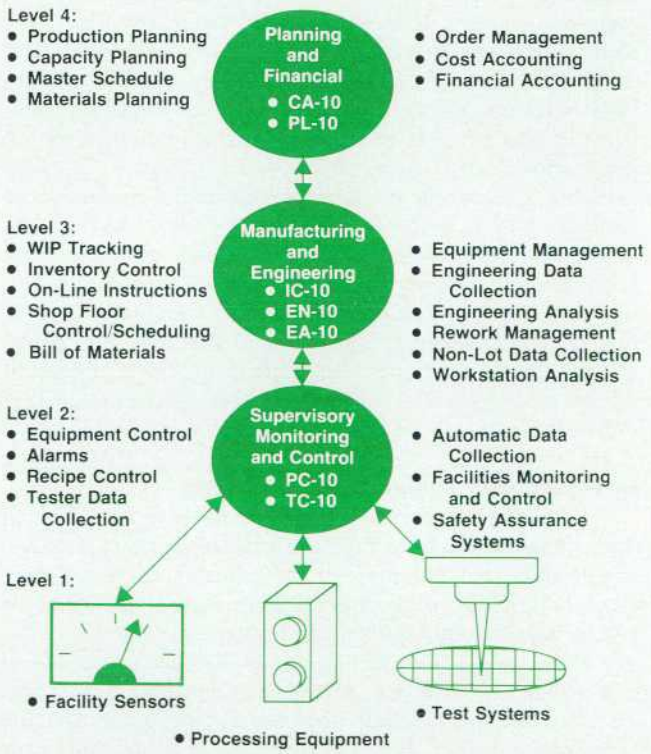


Fig. 2. *HP's Semiconductor Productivity Network is a collection of hardware and software products designed to aid microelectronic product fabrication personnel in controlling, monitoring, improving, and managing many of the levels of the complex manufacturing process.*

PL-10 as a Manufacturing Tool

Editing the Plan. PL-10 allows the user to manipulate the production plan on-line in real time, unlike most conventional MRP systems. The planner can perform simulations or make last-minute changes to the plan depending on the manufacturing environment. To plan more effectively, it is very important to be able to assess various tentative plans and schedules by varying some or all factors. This what-if analysis is one of PL-10's capabilities, called Edit Plan. The consequences of a proposed production plan can be determined before finalizing and implementing it.

Edit Plan can suggest the optimal production plan based on various factors, such as lead time, yield, work-in-process, inventory, and demand. One or all factors can be varied to view how the changes will affect the production plan. This provides flexibility for adjusting to the rapid changes in the manufacturing environment. The planner can be assured that the production plan is the best plan possible under the current manufacturing conditions.

Since the planner needs to access a large variety of data to create a reliable production plan, Edit Plan can access the information by three different windows (see Fig. 3 for one example). The windowing feature in Edit Plan is performed by using parent and child forms in VPLUS/3000. The result is a planning spreadsheet that can be manipulated by adjusting individual period values or by applying any of several precoded algorithms.

Edit Plan has another special feature called Fill Column. The Fill Column function allows the planner to simulate or modify arrays of data easily and quickly. This feature, used throughout the system, gives the user functionality similar to that provided by 1-2-3™ from Lotus™. Fill Column can set a range of period values to a constant, or adjust them by a constant or a percent increase or decrease. Within Fill Column is a feature called ramping, defined as adding the quantity entered to the existing quantity and storing the result in the next period on the horizon. This algorithm continues until all the specified periods are filled.

1-2-3 and Lotus are U.S. trademarks of Lotus Development Corp.

Since the Fill Column function is used in multiple programs, it was decided to include it as a Segmented Library procedure. This provides for easy maintenance and more quality in the PL-10 product.

Explode and Roll Forward. PL-10 has scheduling functions known as the Explode and Roll Forward transactions. Explode is similar to standard MRP explosion (with some extras). Roll Forward is a bottom-up MRP implosion.

The scheduling function is a process of maintaining the production plan to satisfy the total demands. A finished goods item, also referred to as the end item, is demanded by customers. This end-item demand generates requirements for intermediate parts. The intermediate parts generate requirements for raw materials and so forth. Explode in PL-10 calculates the total of each component in the bill of material required to manufacture the planned quantities of end items. Based on this, it will then suggest what will be made by date and quantity for each part. These plans are based on the part information, planning bill structure information, demand already entered, current inventory, and work-in-process.

Explode starts with the end item and works downward (see Fig. 4). A schedule based on top-down demand alone will not show when a schedule is impractical because Explode does not know the lead-time requirements of component parts. For example, if a customer orders product A to be shipped tomorrow, but it takes seven days to produce component B of product A, the earliest Product A can be shipped is one week from today. Explode would suggest

```
Edit Plan                                                    EDIT PLAN

Plan ID (B,T,or A) T     Part Number 100-001                    Area
Window  (P or S)   P     Plan Driver (D or R) D

Freeze Periods 0              Beg Backlog    0              Beg Inventory    3750

Fill Row        From    To       Quantity          or %Change        Ramp?

Period             1      2      3      4      5      6      7      8

  Total Demand  2000   5000   5000   5010   5000   5000   5000   5010
  Expectd Outs     0      0      0      0      0      0      0      0
I Desired Inv    100    100    100    100    100    100    100    100

D Des Per Inv   0.02   0.02   0.02   0.02   0.02   0.02   0.02   0.02
  Proj Per Inv  0.35   0.02   0.02   0.02   0.02   0.02   0.02   0.02

P Prod Plan      0   3350   5000   5010   5000   5000   5000   5010
  Ship Plan   2000   5000   5000   5010   5000   5000   5000   5010
  Proj Bklg      0      0      0      0      0      0      0      0
  Proj Inv    1750    100    100    100    100    100    100    100
  Proj WIP       0      0      0      0      0      0      0      0

  Previous Pln   0      0      0      0      0      0      0      0
  Change         0   3350   5000   5010   5000   5000   5000   5010
  %Change

 SHIFT KEYS | FILL ROW | RECALC PROJ VAL | SUGGEST PLAN | PREV PERIODS | NEXT PERIODS | GET PLAN | MAIN MENU
```

Fig. 3. *Edit Plan screen display.*

the factory begin producing Product A today because it has no way to adjust the parent's schedule for the component's lead time.

This problem is solved by using Roll Forward, a bottom-up approach, often referred as implosion (see Fig. 4). This function in PL-10 begins with the requirements (from Explode) for the lowest-level parts in the bill of material and calculates when these parts should be produced and how many. Once produced, they can be allocated to the next higher-level part to arrive at the parent part's dates and quantities. This continues until the dates and quantities are calculated for the end item. Roll Forward shows how feasible the schedule is, based on both work-in-process and planned starts (building the part from scratch).

Implosion logic quickly becomes unmanageable when many components are assembled into one parent, which is why standard MRP does not have this feature. Since semiconductor bills are short and simple, PL-10 handles Roll Forward for multiple components by having one part act as the driver. In the example shown in Fig. 4, only the die supply is rolled forward. Package requirements are calculated by Explode, but Roll Forward assumes that sufficient packages will always be available.

On the technical side, both of these programs use external files that are loaded into memory at the beginning, then read from and written to throughout the program. For the Explode program, the files are requirement files. These files state the demanded quantity and date. There is one requirement file for each level in the bill of material. The Roll Forward program reads the requirement files from Explode and generates supply files for the parent parts. If the planner is unsure of the results from either Explode or Roll Forward, these files can be used to analyze the production plan.

By using Edit Plan, Explode, and Roll Forward, the planner can make more intelligent decisions. Since the production plan is considered the company's game plan, it is very important for the plans to be realistic and reliable.

Binning. A common process in the semiconductor industry is binning: a part generates multiple byproduct parts based on sorting or testing. This can create excess inventory.

When Explode processes a part that bins into byproducts, the requirements are optimized by generating requirements that satisfy demand while minimizing the excess inventory.

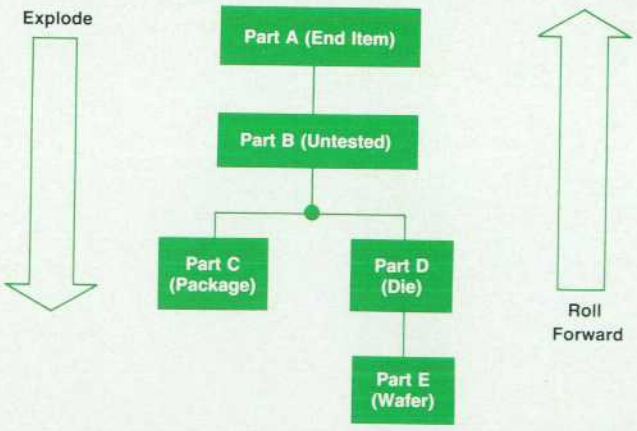


Fig. 4. *Explode and Roll Forward processes.*

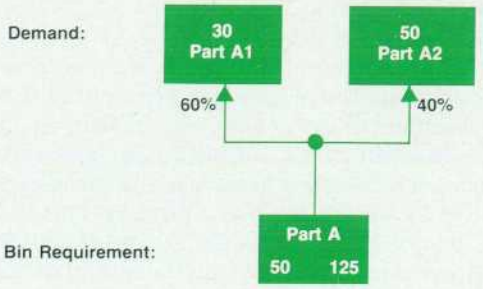


Fig. 5. *Binning example.*

For example, component Part A bins into two parts, Part A1 and Part A2, as shown in Fig. 5. If demand for Part A1 is 30 and demand for Part A2 is 50, then how much of Part A should be produced?

The first consideration is the expected bin-out percentages for Part A1 and Part A2. In this example, the bin percentages are 60% and 40% respectively. A traditional MRP system, using these percentages, would start 50 units of Part A to build Part A1, plus 125 units of Part A to build Part A2, for a total of 175 units started. At bin-out time, this would result in 105 units of Part A1 and 70 units of Part A2, clearly a waste of material.

PL-10 optimizes the plan by considering all the by-products of Part A together. In this example, 125 would be the minimum quantity of Part A to satisfy the demand. However, this still results in excess inventory of 45 units for Part A1. True optimization occurs when the entire planning horizon is considered. The excess inventory of one period may be used to meet the demand in another period, thus minimizing the need to produce more Part As than are necessary.

To illustrate this better, consider a planning horizon (Table I) that is three periods long. The maximum requirements come from Part A2 in periods 1 and 2 and Part A1 in period 3. Notice the excess inventory is kept to a minimum and all demand is satisfied with this method.

In this example, the 125 units planned for Part A in Period 1 satisfy the demand for both Part A1 and A2. They also create enough excess inventory (45 units of A1) to satisfy demand for A1 and A2 in Period 2 without additional starts. In Period 3 the only demand is for 15 units of A1 not covered by inventory, so only 25 Part A units are started.

Allocation Rules. Allocation of supply is performed by Roll Forward in PL-10. Allocation is needed when several parent parts (different package configurations, for example)

Table I
Three-Period Planning Horizon for Binned Parts

Part	Part A1			Part A2			Part A		
Period:	1	2	3	1	2	3	1	2	3
Demand:	30	30	30	50	0	0			
Bin Requirement:	50	50	50	125	0	0			
Cumulative Need:	50	100	150	125	125	125			
Max Cum Need:							125	125	150
Net Bin Need:							125	0	25
Excess Inventory:	45	15	0	0	0	10			

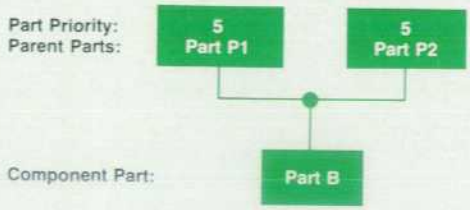


Fig. 6. *Allocation example for equal part priority.*

can be made from a single component, such as a die. It differs from binning in that the planner must decide how to allocate the available dice. Roll Forward suggests a default allocation using parameters set by the planner. The transaction will not allocate more than the amount requested for any part. Any excess inventory will remain as the part's inventory.

In the bill of material, many parent items can be structured to a single component. Likewise, a single parent item can be obtained from several sources of supply (components). The planner can set parameters in PL-10 to allocate these supplies effectively.

Each part contains a field called Priority. This signifies that the supply should be allocated first to the highest priority to meet all its demand. If priorities are the same, then the supply is allocated based on demand. If there is insufficient demand, the ratio of each part's demand to the total demand will be calculated and the supply allocated by that fraction. An example will illustrate this better. Fig. 6 and Table II show the results when the part priorities are the same.

Consider the same example, but with different priorities where Part P1 has a priority of 5 and Part P2 has a priority of 7. The results are given in Table III. Notice that Part P2's demand is satisfied before allocating to Part P1.

The allocation shows if the production plan generated by the explosion process is indeed workable. In the example above, the plan is considered impractical because of insufficient supply.

Resource and Capacity Planning. Semiconductor facilities are equipment and capital-intensive. Equipment is sophisticated and expensive, and capacity utilization is a very important factor in increasing productivity and decreasing costs and long cycle times. Equipment underutilization is no better than equipment overutilization.

A major feature of PL-10 is a vehicle to report utilization (over and/or under) of resources predicted by the production plan. Rough-estimate resource utilization reports show the planner any periods where there are potential capacity bottlenecks or excess idle time. With this information the planner can adjust the schedule by pulling up or pushing out production, or by reallocating resources.

Table II
Allocation with Same Priority

Part:	Part P1			Part P2			Part B		
Period:	1	2	3	1	2	3	1	2	3
Demand:	10	20	10	20	20	30			
Demand Ratio %:	33	50	25	67	50	75			
Supply:							15	30	40
Allocation:	5	15	10	10	15	30			

Table III
Allocation with Different Priority

Part	Part P1			Part P2		
Period:	1	2	3	1	2	3
⋮						
Allocation:	0	10	10	15	20	30

Resources in PL-10 can be defined for any critical item at a facility, such as overall process capacity, machine capacity, or labor. Resources can be tied to a shop calendar depicting the days the resource will or will not be available.

Resources have a certain amount of capacity and downtime, which are not usually evenly distributed across the planning horizon. To accommodate this situation, the user can adjust all planning parameters of a resource by time period.

The next step for the planner is to specify the production parts that will use each resource. PL-10 lets the planner map the following parameters to the part-resource pair:

1. Resource requirement for one unit of the part.
2. Resource lead time to define the number of working days before the part's completion date when the resource is required.
3. Resource yielding factor (%) to compensate for anticipated yield loss between the number of units scheduled and the number that will use the resource.

A part in a semiconductor device fabrication process can return several times to the same equipment or process step, such as the photomasking step for a wafer. To accommodate this situation, PL-10 can assign a part to the same resource for several steps in the manufacturing process.

PL-10 extracts production plan data and calculates the required resource use given the factors listed above. A summary report for a resource is given in Fig. 7.

Memory-Resident Shop Calendars. PL-10 makes intensive use of date calculations. Because the system schedules manufacturing areas having different work schedules, PL-10's date calculations must take into consideration multiple shop calendar parameters. Production Starts, Outs, and Supply are all associated with either a start date or a due date. All dates are represented in year-month-day format.

Date calculations must cover the complete planning horizon (up to 26 months), and must take into account the different calendars (up to 20) that can be defined by the user, the workday and holiday configuration for each individual calendar, and the lead times and lead time adjusts of the various planning parts and their associated part structures. Again, all dates are represented in year-month-day format.

The problem posed to the PL-10 team was how to access this massive calendar quickly and still keep batch run times to a minimum. PL-10's predecessor, designed to schedule only a single work area, stored the calendar in a flat file and read it into memory. The representation of the file was in year-month-day format where each digit was represented by 4 bits. Thus each calendar day was represented by 24

```
REPORT NUMBER  MPS4300 A 06 00                PL-10 EXAMPLE                          RUN  31 JUL 1986  10:12    PAGE      1
                                              Capacity Utilization Report
Cycle Date   860706

Resource:   BURN-IN                                          Source:   TRIAL PLAN
Description:   OVEN                                  Unit of Measure:   HR

Period    Period    Period                    Cum    Adjust   Percent   Percent Utilization Histogram
 Start  Capacity     Usage Deviation Deviation   Percent  Utilized   0        50       100       150       200       250       300
                                                                      |....|....|....|....|....|....|....|....|....|....|....|
                                                                      |                                                          |
860706       114         2      -112      -112      5600%       2%   |                                                          |
860713       114       146        32       -80       -22%     128%   |************************                                  |
860720       114        44       -70      -150       161%      38%   |********                                                  |
860727       114       157        43      -107       -27%     138%   |***************************                               |
860803       114        41       -74      -181       183%      36%   |*******                                                   |
860810       114        41       -74      -255       183%      36%   |*******                                                   |
860817       114        11      -103      -358       904%      10%   |**                                                        |
860824       114        16       -98      -456       613%      14%   |***                                                       |
                                                                      |                                                          |
                                                                      |....|....|....|....|....|....|....|....|....|....|....|
                                                                      0        50       100       150       200       250       300

End of Resource Utilization by Resource Report
```

Fig. 7. *Capacity utilization report.*

bits or 3 bytes. In this configuration only workdays were kept in the file. If a calendar year had 260 workdays, and the planning horizon spanned two years, then 1560 bytes (260×2×3) were needed to hold the table. If the user were able to define 20 different shop floor calendars, then a total of 31,200 bytes would have been needed to hold all the calendars in memory.

In building the current version of PL-10, the memory storage requirements for 20 different calendars using the old scheme would have been impractical. Another approach was taken to represent the calendars and still keep them resident in memory. For each different shop floor calendar, a Holiday/Workday mapping is kept for each calendar day in the planning horizon. Hence, two tables are needed, one for each calendar day in the planning horizon, and one for the Holiday/Workday indicator for the various shop floor calendars. A calendar day is represented by 16 bits instead of 24 bits. The new year-month-day format representation is: year—7 bits, month—4 bits, and day—5 bits.

With this new format, the memory requirement for 20 different shop floor calendars, each spanning 26 months, is (considering that a planning month in PL-10 can have up to six weeks or 42 days):

Calendar Table:

$$(26\ \text{months}) \times (42\ \text{days/month}) \times (2\ \text{bytes}) = 2184\ \text{bytes}.$$

Work Day Table:

$$(20\ \text{calendars}) \times (26\ \text{months}) \times (42\ \text{days})/(8\ \text{bits}) = 2730\ \text{bytes}.$$

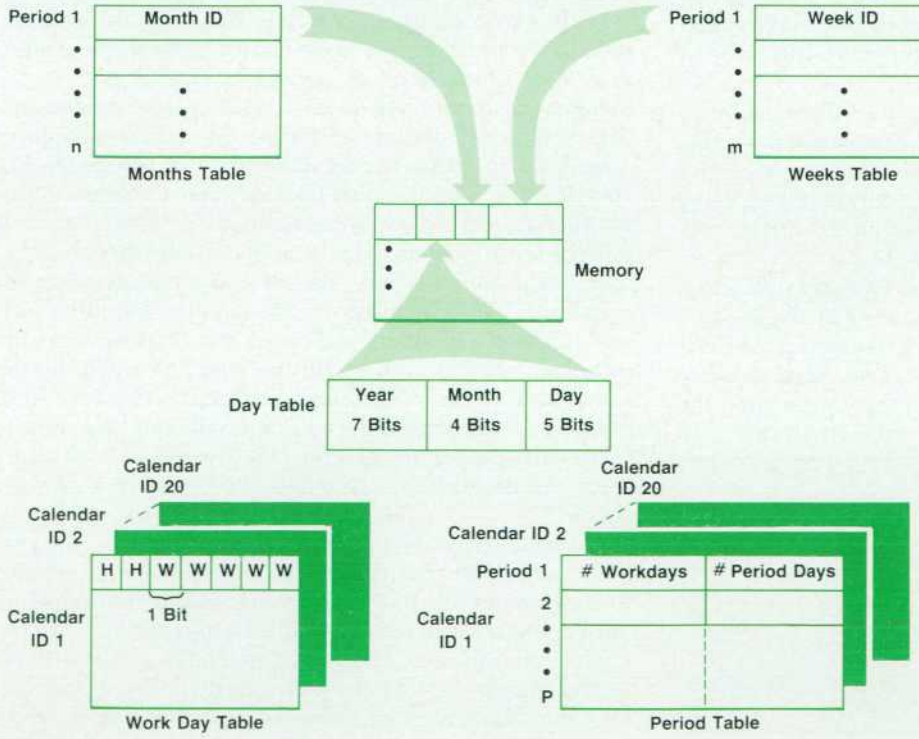


Fig. 8. *Organization of memory-resident calendar tables.*

Three other tables of smaller sizes are kept in memory to provide quick date calculations for month or week planning. The month and week tables have pointers into the day table to locate the starting date of each calendar month and week. The period table summarizes the workday table—it shows the total days and work days in each planning period. All five tables and their relationships are represented in Fig. 8.

Acknowledgments

We would like to thank the following people whose contributions made PL-10 a reality: the MPS 1.5 group at CMIS—Kevin Cooper, Andra Marynowski, Randall Fryer, Mark Wunderman, and Preston Carter—the marketing group—Julie Schoenfeld, Tom Coughlin, Monica Randall, and Patti Herbst—and lab members Ken Yao, Jan Grady, Scott Ingraham, and Pat Grever. Special thanks go to Mark Kinsman of the lab for his insight and help with many of the algorithms.

References

1. W.H. Higaki, "Remote Monitoring and Control of Semiconductor Processing," *Hewlett-Packard Journal*, Vol. 36, no. 7, July 1985, pp. 30-34.
2. R.I. White, "Automated Test Data Collection for IC Manufacturing," *Hewlett-Packard Journal*, Vol. 36, no. 8, August 1985, pp. 30-36.
3. N.C. Federman and R.M. Steiner, "An Interactive Material Planning and Control System for Manufacturing Companies," *Hewlett-Packard Journal*, Vol. 32, no. 4, April 1981, pp. 3-12.

Authors

April 1987

4 Data/Waveform Generator

Uwe Neumann

A project manager at the Böblingen Instruments Division, Uwe Neumann was responsible for the HP 8175A Digital Signal Generator and for the firmware for the HP 8116A Pulse/Function Generator. When he first came to HP in 1978 he worked on firmware for the HP 8161A Pulse Generator. His Diplom Ingenieur (FH) was awarded in 1978 by the Engineering School in Constance. Born in Zwickau, Uwe now lives in Gäufelden. He and his wife have two children. His leisure activities include walking, cross-country skiing, and travel.

Frank Husfeld

Frank Husfeld started at HP in 1978 as a microwave applications engineer and moved to the Böblingen Instruments Division in 1983. He designed the arbitrary function generator option for the HP 8175A Signal Generator and is currently working on a pulse generator. His work on differential operational amplifier drift has resulted in a patent application and he has written a number of papers on scientific calculators and microwave components. A native of Hamburg, he received his master's degree in RF engineering from the Technical University of Darmstadt. He and his wife and two daughters live in Ehningen. Frank keeps busy by working on the restoration of his home, which was built in 1780. He's also interested in amateur radio (DF7FT), celestial navigation, and renewable energy technologies.

Michael Vogt

With HP since 1982, Michael Vogt worked on the data board and the HP 15464A TTL Pod for the HP 8175A Signal Generator. He's now a production engineer in the Waldbronn Division. A native of Pforzheim, Baden-Württemberg, he received his Diplom Ingenieur from the Technical University of Karlsruhe in 1982. Michael is a private pilot and flies his own plane. He also enjoys swimming and working in his garden.

Friedhelm Brilhaus

An R&D hardware engineer, Friedhelm Brilhaus has been with HP since 1983. His work on the HP 8175A includes designing the clock board, the ECL pod, the fine timing board, and the arbitrary board. He is also named inventor on a patent for an impulse generator. Friedhelm was born in Greven, Westfalen, and attended the Engineering School in Münster. His Diplom Ingenieur was awarded in 1982. He lives in Gärtringen, Württemberg, is married, and has one child. His outside interests include soccer and astrophysics.

12 Data/Waveform Generator Software

Wolfgang Srok

Wolfgang Srok was born in Stuttgart and received his Diplom Ingenieur in electronic engineering and computer science from the University of Stuttgart. He has been with HP since 1984 and has contributed to the design of the disc driver and graphics for the HP 8175A and to the MATE option for the HP 8161A Pulse Generator. Outside of work Wolfgang enjoys swimming and jogging.

Rüdiger Kreiser

Rüdiger Kreiser studied computer science at the Engineering School in Constance, from which he received his Diplom Informatiker (FH) in 1984. He has been with HP since 1985 and has contributed to the dual arbitrary waveform generator (Opt. 002) for the HP 8175A. He has also worked on the MATE option for the HP 8160A and HP 8161A Pulse Generators. Rüdiger was born in Singen and now lives in Gärtringen. His leisure activities include skiing, soccer, and squash.

Ulrich Hakenjos

A project leader at the Böblingen Instruments Division, Ulrich Hakenjos has been with HP since 1983. He was project leader for the arbitrary waveform generator firmware for the HP 8175A and is now project leader for a function generator product. Before coming to HP he developed software for a consulting company. A native of Baden-Württemberg, he was born in Schwenningen and received his Diplom Ingenieur (FH) from the Engineering School in Furtwangen. He lives in Herrenberg and likes outdoor sports, natural history, and travel.

21 IC Manufacturing Planning

Clemen Jue

Born in Ventura, California, Clemen Jue studied computer science at California Polytechnic State University at San Luis Obispo (BS 1973) and at the University of California at Los Angeles (MS 1974). He developed software for real-time data acquisition systems at GTE Sylvania before coming to HP in 1977. He has worked on a number of software systems for IC design and manufacturing, including HP PL-10 and HP IC-10 and has also written HP QDM/1000 software. Married with three children, Clemen lives in Los Altos Hills, California. His outside interests include tennis, softball, and table tennis.

Edward L. Wilson

With HP since 1979, Ed Wilson has contributed to the development of several planning and management software modules. In addition to his work on the HP PL-10 planning module for the Semiconductor Productivity Network, he has worked on two order management systems, currently for HP Materials Management/3000. A California native, he was born in San Francisco and attended the Massachusetts Institute of Technology. His SB degree in humanities and sciences was awarded in 1970. After working for the Prudential Insurance Company he continued his education at the Wharton School, completing work for his MBA in 1979. Ed lives in Felton, California and enjoys running, swimming, and playing guitar.

Kelly A. Sznaider

With HP since 1982, Kelly Sznaider is a software engineer at the Manufacturing Productivity Division. In addition to her work on HP PL-10, she has developed software for manufacturing and management information systems. Kelly was born in Wabash, Indiana and received her BA degree in computer science, accounting, and management from Anderson College in 1982. Now a resident of San Jose, California, she's married and likes jogging and softball.

29 Panel Deflection

John H. Lau

With HP Laboratories since 1984, John Lau is a mechanical engineer who is working to improve surface-mount reliability. He holds a 1977 PhD degree in theoretical and applied mechanics from the University of Illinois and before coming to HP worked as a civil and nuclear engineer. He's a registered professional engineer in California and New York. John and his wife and daughter are residents of Palo Alto, California, and his wife also works for HP Laboratories. He and his family enjoy walking and bicycling.

George E. Barrett

A graduate of San Jose State University (BSME 1982), George Barrett came to HP in 1981. He has been a process engineer and is now a manufacturing engineer at the Sunnyvale Personal Computer Operation. He worked on the installation of the production line for the HP 150 Computer and is now working on equipment for a surface-mount assembly line. He also served for six years in the U.S. Navy. When he takes time out from renovating his home in Santa Clara, California, George enjoys art and outdoor sports.

35 Software Testing

Duane E. Wolting

Duane Wolting is a specialist in reliability, pattern recognition, and multivariate methods who worked at HP from 1978 to 1986. Born in Long Beach, California, he attended Purdue University (BS aeronautical engineering 1972) and the University of California at Davis (MS statistics 1986). He also worked at Hughes Aircraft Company as an R&D engineer for communication satellites. A member of the IEEE and the American Statistical Association, he has published three papers on statistical methods in engineering. Duane and his wife and two children live in Roseville, California in a home he and his wife designed and built. He likes backpacking, bicycling, and most outdoor sports.

H. Dean Drake

Born in Tulsa, Oklahoma, Dean Drake grew up in California and earned his BSEE degree from the University of California at Davis in 1969. After serving as a lieutenant in the U.S. Army, he worked on an automated missile testing system for the U.S. Navy. His assignments have been varied since he first came to HP in 1973. He developed diagnostic programs and extended memory array firmware for HP 21MX Computers and wrote a portion of the RTE Microprogramming Package for HP 1000 Computers. More recently he has been a quality assurance engineer for a number of terminals at the Roseville Terminals Division. He left HP in 1979 to work on a data acquisition and monitoring system for Bently Nevada Corporation and rejoined the company in 1984. He also completed work for an MS in computer science from California State University at Chico in 1983. Dean lives in Rocklin, California, is married, and has one son. He's active in youth soccer and enjoys jogging and backpacking.

A Study of Panel Deflection of Partially Routed Printed Circuit Boards

Finite element analysis was used to show that the stress and deflection of partially routed boards during handling will be within allowable limits.

by George E. Barrett and John H. Lau

TO REDUCE MACHINE SETUP TIME during the manufacture and assembly of printed circuit boards, small circuit boards are processed in subpanel form. A subpanel consists of one or more finished boards still attached to a supportive frame. To facilitate separation of the boards after assembly, a partial router path is cut around the perimeter of each board, so that the boards are held in place by connecting tabs.

While partial routing simplifies depanelization (the separation of boards in a subpanel), it also reduces panel rigidity, so that a panel supported on the sides, such as during board transport between processes, will bow more than an unrouted panel. When assembling surface mount boards, board deflection must not exceed 0.7% of the board length, since deflections greater than this will affect component planarity and the quality of solder joints (see Fig. 1).

Since the elimination of partial routing would require the development of a more sophisticated and costly depanelization process, a finite element analysis was used to determine the impact of routing paths on panel rigidity.

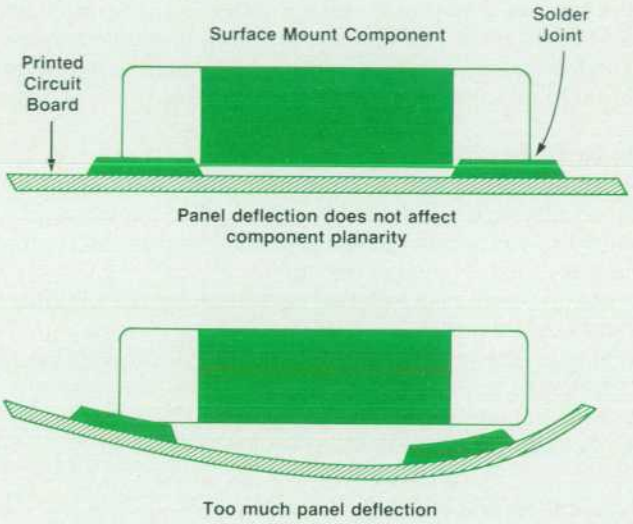


Fig. 1. *Panel deflection and component planarity.*

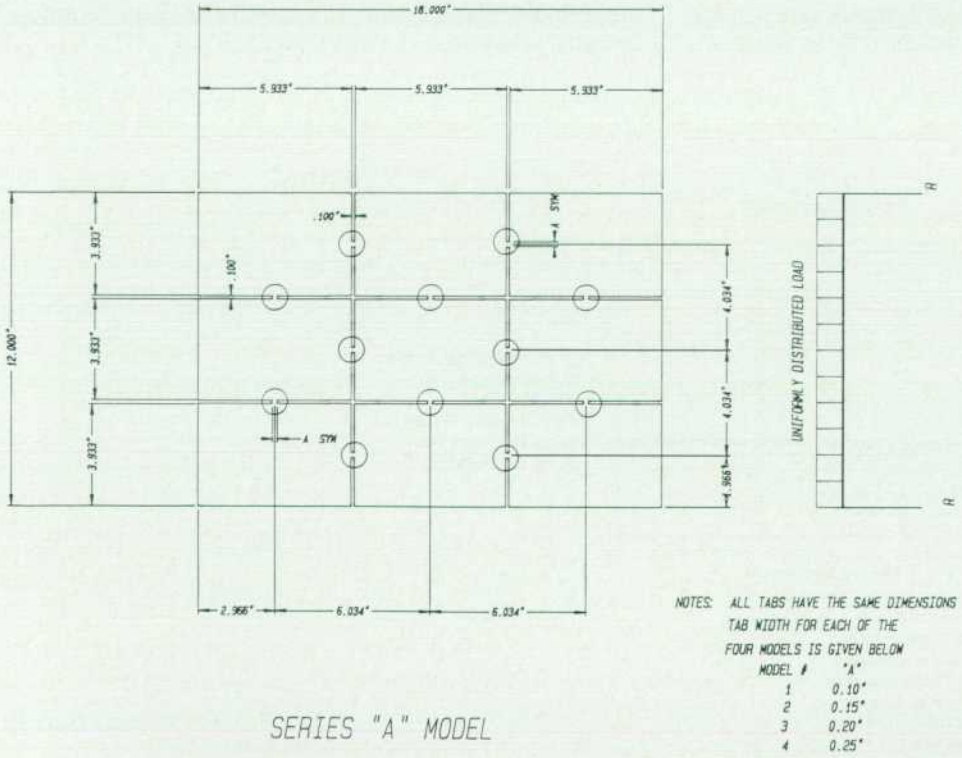


Fig. 2. *Model of a partially routed printed circuit board panel with a uniform load.*

The primary objective of this analysis was to characterize the deflection of a printed circuit board that had been partially routed (Fig. 2), and to compare panel deflection with the specification for panel flatness. Induced stresses were also characterized, to verify that material failure did not occur.

Why Use Finite Element Analysis?

The finite element method was selected as a tool for this study because no closed-form solution for the partially routed panel was available. In addition, available software allowed flexibility in modeling and provided high-resolution output both graphically and in tabular form. To verify the accuracy of the finite element method, a finite element analysis was used to solve for the deflection of an unrouted panel. The results of this analysis were compared to the closed-form solution (see box, page 33), and the two were found to be in excellent agreement.

Acceptance Criteria

Noting the properties of the material to be used and industry standards for the assembly of surface mount boards, the following acceptance criteria were established for the partially routed panel under study:

- Maximum allowable deflection of each board is less than 0.7% of the length.
- Maximum strength in tension equals 35 to 80 ksi (1 ksi = 1000 psi).[1]
- Maximum strength in compression equals 35 to 85 ksi.[1]
- Maximum strength in shear equals 14 to 25 ksi.[1]

Application and Problem Definition

The surface mount assembly line, being developed jointly by the Hewlett-Packard Personal Office Computer Division, Manufacturing Research Center, and Entry Systems Operation, will incorporate stations for solder-paste application, component loading, solder reflow, wave solder, board cleaning, inspection, and depanelization. Panels will be transferred from station to station by a board handling system. While boards are being assembled in each of the process steps they will be supported from the bottom so they will not deflect. However, while boards are being transferred by the board handling system they are supported only along two opposite edges. As a result, boards will bow and components may dislodge or skew in the unhardened solder paste.

Since board deflection will only occur during board transport, the displacement boundary conditions are set to simulate this process. As can be seen in Fig. 3, displacement boundary conditions are set along the leading edge (stress-free boundary), the axis of symmetry in the X direction, the axis of symmetry in the Y direction, and one of the supported edges. The displacement boundary conditions for each edge are as shown in Fig. 3.

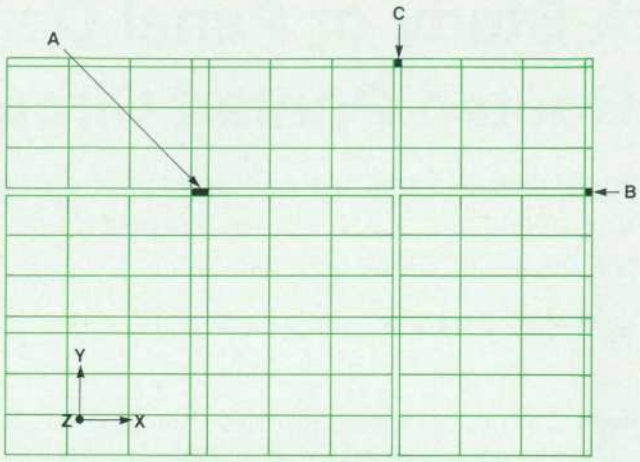


Fig. 4. *Finite element model for the partially routed panel. Elements A, B, and C refer to Figs. 8 and 9.*

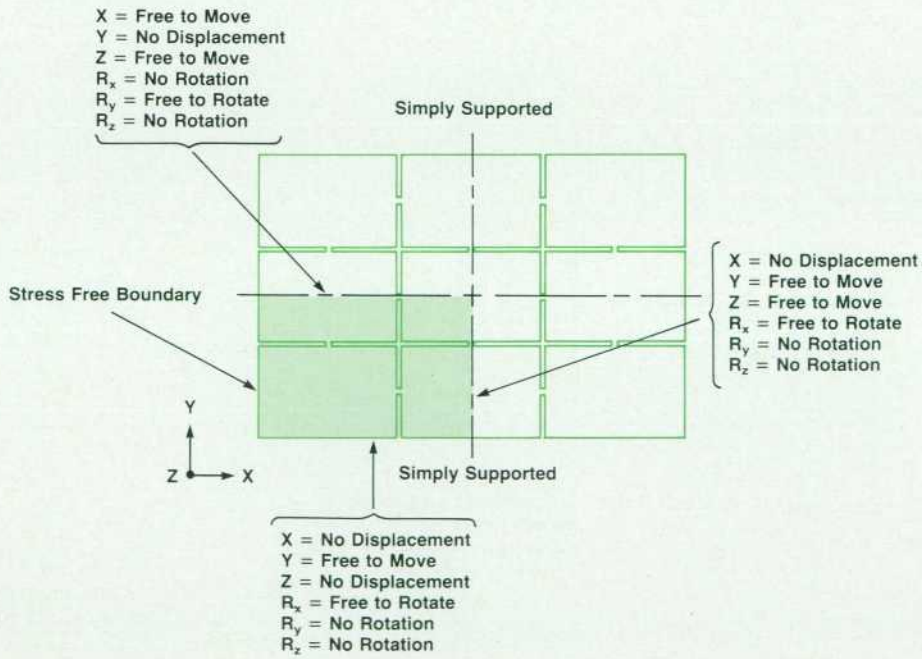


Fig. 3. *Displacement boundary conditions.*

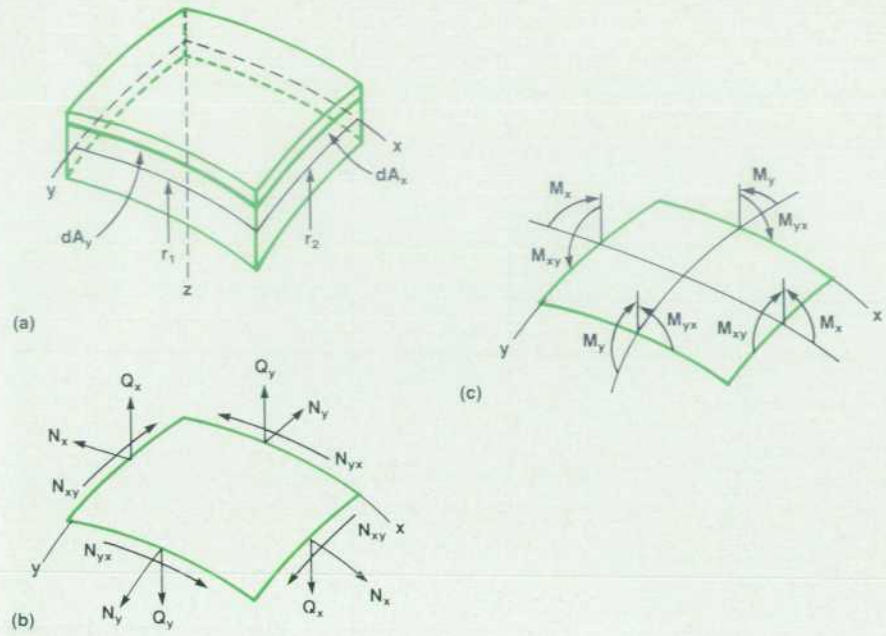


Fig. 5. *Forces and moments acting on a thick-shell element.*

The boundary value problem is further defined by noting the load density (w = 0.013 psi) and material properties (Young's modulus = 1,600,000 psi, Poisson's ratio = 0.028). The load density was established by measuring the load density of an assembled board currently being manufactured, and applying the same load density to the model. Material properties were taken from reference 2.

Structure Description

A partially routed panel was created for this study, and four different connecting tab dimensions were considered. These dimensions were set to present a worst-case scenario, while still adhering to current manufacturing practices and projected product design. Fig. 2 gives the dimensions of the design. It can be noted that this design has one tab connecting adjoining boards and the cross-sectional area connecting adjoining boards is identical. Since the structure is symmetric about the X and Y axes, a finite element model was created for only one quarter of the panel (shaded area in Fig. 3). Fig. 4 shows the model. When creating the finite element model a thick shell element was used. The shell element was used because a thick plate element was not available on the software used, and because a shell will assume the form of a flat plate as the radii go to infinity. A thick element was used instead of a thin one to capture the effects of transverse shear through the thickness.

The thick shell element is shown in Fig. 5. Fig. 5a shows the incremental cross-sectional area and the radii of curvature in the X and Y planes. Fig. 5b shows the tensile (or compressive) stresses as well as the in-plane and transverse shear. Fig. 5c gives the bending and twisting moments in each plane.

Results

The analysis showed that the maximum deflection for the entire subpanel was between 0.61 inch and 0.42 inch depending on the tab width used. While these values exceeded the allowable limit, individual deflection profiles for the boards in the subpanel were significantly less and within the allowed limits (as measured across each board separately). This is seen in the deflection plot shown in Fig. 6. In Fig. 6, the undeformed mesh is shown in the background and the deformed mesh in the foreground. While the deformed mesh shows significant subpanel displacement, it also shows a discontinuity in the slope of the subpanel. This discontinuity occurs where the connecting tabs hold the boards together. The curvature of the boards on either side of the connecting tabs appears to be insignificant. Noting that the acceptance criteria are for board deflection rather than subpanel deflection, this design meets the criteria for deflection.

The stress contours are shown in Figs. 7 and 8. These are the distribution of the Von Mises stresses in the panel. (The Von Mises stresses are used because they take into consideration all the stresses.) The contours show that

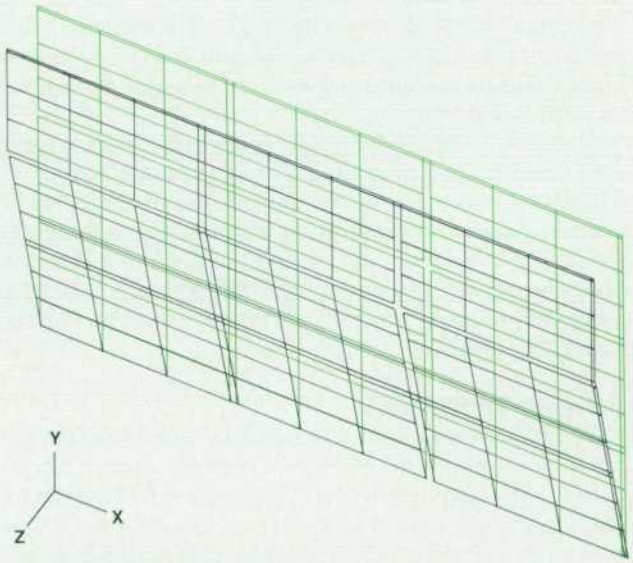


Fig. 6. *Deflection of the panel.*

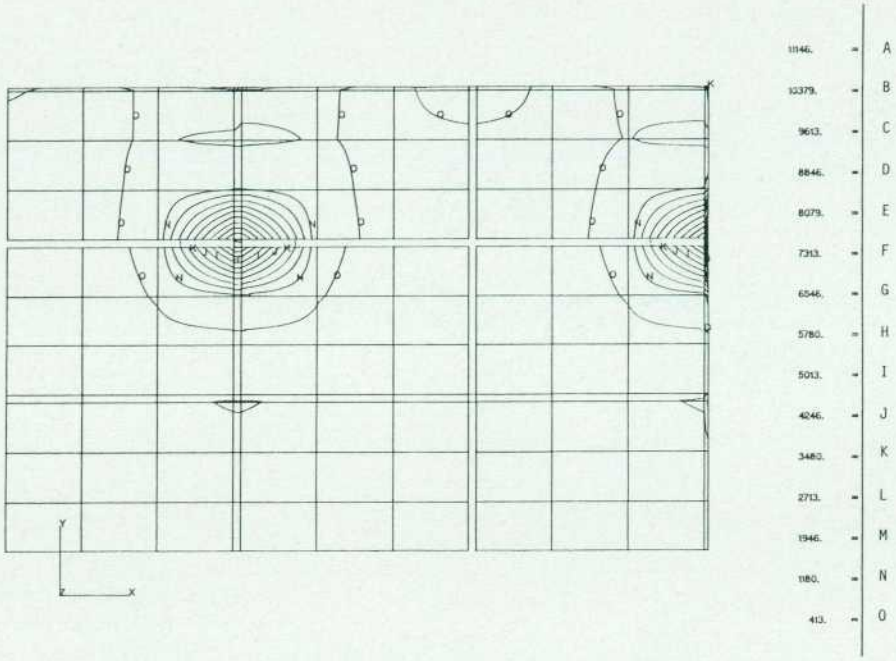


Fig. 7. *Von Mises stress distribution in the panel.*

stresses reach a maximum at the connecting tabs. This is reasonable since the connecting tabs must support the same load as portions of the board having a larger cross-sectional area.

The stresses for the tabs are given in Fig. 9. In this table, λ represents the tab width for that iteration of the analysis, σ_x and σ_y are the tensile/compressive stresses, and τ represents the in-plane shear. Elements are listed to correspond with the respective element plots, Fig. 4. A review of Fig. 9 shows that as the tab width is increased the induced stresses are reduced. It can also be seen that the induced stresses are less than the maximum allowable.

Conclusions and Summary

The results of the finite element analysis indicated that subpanels with the prescribed dimensions could be partially routed without causing mechanical failure or compromising solder joint quality. It also showed that while partial routing will increase overall panel deflection, boards within the subpanel will not be deformed beyond the allowable limit.

Since the designs used in this study represented a worst-case scenario, partial routing of surface mount boards was declared a safe practice. As a result, the need to develop a sophisticated depanelization workcell was eliminated. Benefits included savings of one hundred fifty thousand to two hundred fifty thousand U.S. dollars for capital equipment, and reduced process development time by approximately six engineer months.

Acknowledgments

The authors would like to thank Ron Lloyd and Don Rice for their support and comments on this work. They are particularly indebted to Chao Chung-Huei for his effective help.

References

1. *Modern Plastics Encyclopedia*, Vol. 56, no. 10A, McGraw-Hill, 1978.
2. P.M. Hall, "Forces, Moments, and Displacements During Thermal Chamber Cycling of Leadless Ceramic Chip Carriers Soldered to Printed Boards," *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, Vol. CHMT-7, no. 4, December 1984.

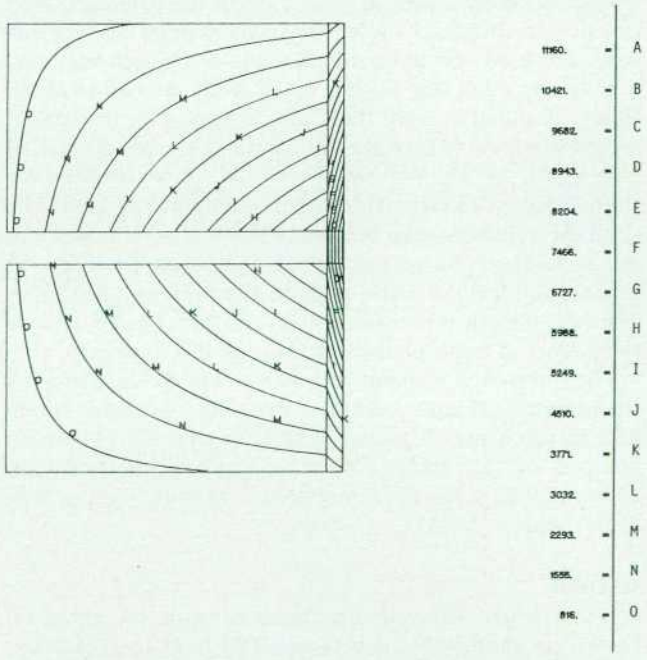


Fig. 8. *Von Mises stress distribution at point B of the panel.*

Element A

λ	$\sigma_{x\ psi}$	$\sigma_{y\ psi}$	τ_{psi}
0.10	−89.7	18830	283
0.15	−19.3	12590	188
0.20	7.46	9470	126
0.25	18	7592	86

Element B

λ	σ_x	σ_y	τ
0.10	369	18600	−283
0.15	95.4	12330	−188
0.20	−42.9	9213	−126
0.25	−119	7346	−86

Element C

λ	σ_x	σ_y	τ
0.10	1090	2213	283
0.15	792	1817	188
0.20	647	1503	126
0.25	559	1255	86

Compressive strength[1] = 35-80 kpsi
Tensile strength[1] = 35-85 kpsi
Shear strength[1] = 14-25 kpsi

Fig. 9. *Maximum stresses at elements A, B, and C of the panel (see Fig. 4).*

Deflections, Forces, and Moments of a Printed Circuit Board

Closed-Form Solutions

Fig. 1a shows an 18 × 12 × 0.062-in printed circuit panel with a Young's modulus (E) equal to 1,600,000 psi and a Poisson's ratio (ν) equal to 0.28. This panel is subjected to a uniform load, w = 0.013 psi, and is simply supported along x = 0 and x = a and is free to move along y = ±b/2. It can be shown that the closed-form solutions for the deflections (d), the bending moments (M_x, M_y), the twisting moment (M_{xy}), and the transverse shears (Q_x, Q_y) are given as follows.

$$d = \frac{wa^4}{D} \sum_{m=1,3,5\ldots}^{\infty} \left[\frac{4}{\pi^5 m^5} + A_m \cosh \frac{m\pi y}{a} + B_m \frac{m\pi y}{a} \sinh \frac{m\pi y}{a} \right] \sin \frac{m\pi x}{a}$$

$$M_x = -w(\pi a)^2 (d_{xx} + \nu d_{yy})$$

$$M_y = -w(\pi a)^2 (d_{yy} + \nu d_{xx})$$

$$M_{xy} = -w(\pi a)^2 (1-\nu) d_{xy}$$

$$Q_x = -w\pi^3 a (d_{xxx} + d_{yyx})$$

$$Q_y = -w\pi^3 a (d_{xxy} + d_{yyy})$$

where

$$\alpha_m = \frac{m\pi b}{2a}$$

$$A_m = \frac{4}{m^5\pi^5} \frac{\nu(1+\nu)\sinh\alpha_m - \nu(1-\nu)\alpha_m \cosh\alpha_m}{\Delta}$$

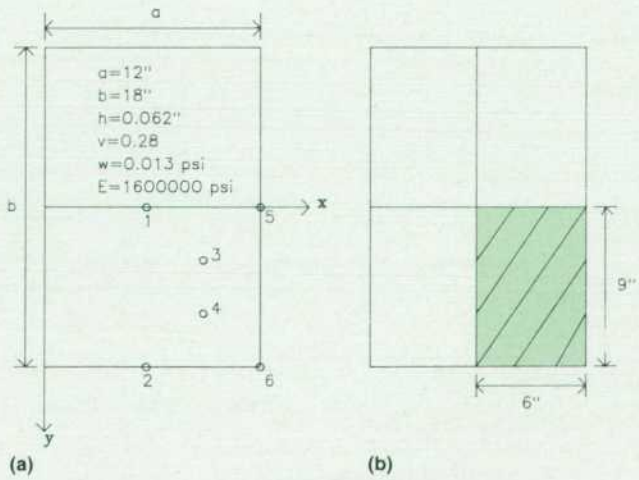


Fig. 1 *(a) Printed circuit panel. (b) Shaded area shows portion subjected to finite element analysis.*

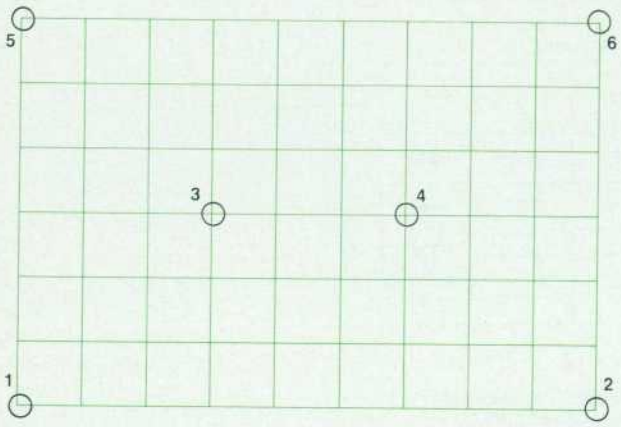


Fig. 2. *Finite element model for a quarter of the printed circuit panel.*

$$B_m = \frac{4}{m^5\pi^5} \quad \frac{\nu(1-\nu)\sinh\alpha_m}{\Delta}$$

$$\Delta = (3+\nu)(1-\nu)\sinh\alpha_m\cosh\alpha_m - (1-\nu)^2\alpha_m$$

$$D = \frac{Eh^3}{12(1-\nu^2)}$$

$$d_{xx} = -\sum_{m=1,3,5,\ldots}^{\infty} m^2\left[\frac{4}{m^5\pi^5} + A_m\cosh\frac{m\pi y}{a} + B_m\frac{m\pi y}{a}\sinh\frac{m\pi y}{a}\right]\sin\frac{m\pi x}{a}$$

$$d_{yy} = \sum_{m=1,3,5,\ldots}^{\infty} m^2\left[A_m\cosh\frac{m\pi y}{a} + m^2B_m\left(2\cosh\frac{m\pi y}{a} + \frac{m\pi y}{a}\sinh\frac{m\pi y}{a}\right)\right]\sin\frac{m\pi x}{a}$$

$$d_{xy} = \sum_{m=1,3,5,\ldots}^{\infty} m^2\left[A_m\sinh\frac{m\pi y}{a} + B_m\left(\sinh\frac{m\pi y}{a} + \frac{m\pi y}{a}\cosh\frac{m\pi y}{a}\right)\right]\cos\frac{m\pi x}{a}$$

$$d_{xxx} = -\sum m^3\left[\frac{4}{m^5\pi^5} + A_m\cosh\frac{m\pi y}{a} + B_m\frac{m\pi y}{a}\sinh\frac{m\pi y}{a}\right]\cos\frac{m\pi x}{a}$$

Locations	Coordinates (x,y)	Deflections (d) (in)	
		Closed-Form Solution	Finite Element Solution
1	(6,0)	0.10076	0.10080
2	(6,9)	0.11679	0.11690
3	(9,3)	0.07236	0.07237
4	(9,6)	0.07496	0.07498
5	(12,0)	0	0
6	(12,9)	0	0

Fig. 3. *Deflections of the panel at various locations (see Fig. 2).*

$$d_{yyy} = \sum m^3\left[A_m\sinh\frac{m\pi y}{a} + m^2B_m\left(3\sinh\frac{m\pi y}{a} + \frac{m\pi y}{a}\cosh\frac{m\pi y}{a}\right)\right]\sin\frac{m\pi x}{a}$$

$$d_{xxy} = -\sum m^3\left[A_m\sinh\frac{m\pi y}{a} + B_m\left(\sinh\frac{m\pi y}{a} + \frac{m\pi y}{a}\cosh\frac{m\pi y}{a}\right)\right]\sin\frac{m\pi x}{a}$$

$$d_{yyx} = \sum m^3\left[A_m\cosh\frac{m\pi y}{a} + m^2B_m\left(2\cosh\frac{m\pi y}{a} + \frac{m\pi y}{a}\sinh\frac{m\pi y}{a}\right)\right]\cos\frac{m\pi x}{a}$$

The numerical results of the deflections at locations 1 through 6 of the panel (Fig. 2) are shown in Fig. 3. The bending and twisting moments and transverse shear forces at location 1 are given in Fig. 4. These results were obtained by summing the above series up to m = 15.

Finite Element Solutions

Because of double symmetries (see Fig. 1b), only a quarter of the panel (shaded area in Fig. 1b) is modeled, as shown in Fig. 2. It consists of 54 doubly curved shell elements. Each element has 8 nodal points. Each nodal point has six degrees of freedom. The displacement boundary conditions are exactly the same as those shown in Fig. 3 on page 30. Finite element results for the deflections, bending and twisting moments, and transverse shear forces are summarized in Figs. 3 and 4. It can be seen that they are in excellent agreement with the closed-form solutions.

Location	Bending Moments				Twisting Moment		Transverse Shear Forces			
	M_x (in-lb/in)		M_y (in-lb/in)		M_{xy} (in-lb/in)		Q_x (lb/in)		Q_y (lb/in)	
	CF	FE	CF	FE	CF	FE	CF	FE	CF	FE
1	0.05932	0.05926	0.22974	0.23000	0.00001	0.00004	0.00011	0.00031	0.00254	0.00246

CF = Closed-Form Solution
FE = Finite Element Solution

Fig. 4. *Bending moments, twisting moment, and shear forces at location 1.*

Reliability Theory Applied to Software Testing

The execution-time theory of software reliability is extended to the software testing process by introduction of an accelerating factor. It is shown that the accelerating factor can be determined from repair data and used to make prerelease estimates of software reliability for similar products.

by H. Dean Drake and Duane E. Wolting

THE APPLICATION OF RELIABILITY theory to software is in its early stages. The marketplace, however, demands that the reliability of software in new products be improved and that this key quality attribute somehow be quantified. An observation by three British mathematicians in 1952 underscores early recognition of the problem, but provides little insight as to its definition or how to measure software reliability:

Those who regularly code for fast electronic computers will have learned from bitter experience that a large fraction of the time spent in preparing calculations for the machine is taken up in removing the blunders that have been made in drawing up the programme.[1]

Today's reliability engineer certainly has better methods and models to help understand software reliability than in 1952, yet a practical way to conceive of software reliability still eludes the industry. Even simple discussion of software reliability gives rise to terms such as "system crash" and "bug" that are foreign-sounding to those who routinely apply reliability theory to hardware.

While software reliability does differ from hardware in some aspects, it can be conceived of and applied using a model based on conventional reliability theory. Measurement of software reliability in terms of failure rate (spanning both prerelease and postrelease periods), treatment of software testing as a form of accelerated life cycle testing, and use of a contemporary reliability model for software all form a basis for reliability analysis of software.

Theoretical Model

Conventional reliability theory[2] often describes hardware reliability in terms of an instantaneous failure rate such as shown in Fig. 1. Based on this model, reliability analysis proceeds by assuming a distributional form for the time to failure, obtaining observations on this random variable, and then using the data to arrive at a statistical estimate of the failure rate.

The software characteristic, also shown in Fig. 1, is derived from a theoretical model proposed by Musa,[3,4] which can be classed as a reliability growth model. The model reflects a basic intuitive supposition: software generally improves with age. As a program matures, defects are discovered, the number of potential failures is reduced when the defects are fixed, and the rate at which new failures occur is also reduced.

Musa calls his model the execution-time theory of software reliability because the failure rate curve shown for software in Fig. 1 is actually a function of execution time instead of calendar time. Execution time is the time spent executing the program being analyzed. It excludes idle time, which a model using calendar time would include. For hardware, it is self-evident not to count the time the unit is not operational when taking data. The actual time a program spends executing, on the other hand, is often obscured by various factors such as time allocated to an application in a multitasking system environment.

The execution-time model predicts that the instantaneous failure rate for software can be described by (see Appendix):

$$h(x) = h_o e^{-(h_o/N_o)x} \qquad (1)$$

where h_o and N_o are fixed parameters that are characteristic of a given program and x is execution time.

If the assumptions are made that time spent explicitly testing a software product is synonymous with execution time and the testing environment is more stressful than the normal user environment, then the model can be extended to the software testing process. Consideration of test time and execution time as being equivalent is straight-

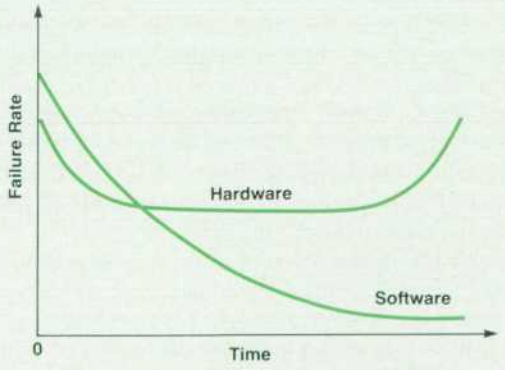


Fig. 1. *Characteristic failure rate curves.*

forward. Time spent testing an interactive application or compiling test modules for a new language is obviously time spent executing the software. Software doesn't wear out as hardware systems characteristically do; however, it does have an aging mechanism. As a given software product is exercised through use, more and more untried, discrete input cases are executed from the total possible input space of the product and previously undiscovered defects are found when particular cases result in failures. Since software testers are trying to cause failures, they create an environment for the software in which input cases occur with greater frequency than they would with normal use and therefore failures occur at a higher rate.

This higher failure rate in testing is accounted for by modifying Musa's basic model to include an accelerating factor C such that the failure rate in testing is:

$$h_a(x) = Ch(x) = Ch_o e^{-(h_o/N_o)x} \quad (2)$$

where $h_a(x)$ is the software failure rate in testing with respect to test time (or execution time), and C is a constant greater than or equal to one. Introduction of the constant C implies that the ratio of the failure rate h(x) to the accelerated failure rate in testing $h_a(x)$ is constant for all values of x.

Application of the Software Reliability Model

The process used to test terminal firmware at the HP Roseville Terminals Division has been designed with the software reliability model in mind. Testing is performed by teams of engineers responsible for various features of the firmware. They test on a part-time basis (as opposed to one or two engineers testing features sequentially on a full-time basis). This ensures that test cases are applied concurrently across all of the firmware's functional areas and helps to keep the occurrence of a failure independent from other failures and test coverage congruent with use coverage (see assumption 3 in Appendix). Time spent testing is logged to provide failure data as a function of test time (execution time) instead of calendar time.

Test data is then compiled in the context of the software reliability model. When defects are found, a report is generated by the testing engineer. These reports are counted each test day and divided by the day's test hours to determine a test day failure rate. A failure rate graph is then constructed by plotting successive average failure rates against cumulative test time.

Such a plot for the HP 2393A and HP 2397A Terminals is shown in Fig. 2 (both terminals contain the same firmware). Also shown is a curve that is a fitted regression model of the form Ae^{Bx}. By fitting the exponential curve to the data, it is assumed that the underlying trend is the general form of the decaying exponential predicted by the software reliability model in equation 2 without actually knowing the parameters C, h_o, or N_o.

Although the model shows failure rate as a continuous function in Fig. 1, it predicts actual rates to be constant between revisions (see Appendix). Given this attribute, the ending failure rate of the terminal's firmware at release is the value of the regression function at the end of testing, labeled $x_{Release}$ in Fig. 2. This is determined by evaluating the regression function at the final test time value of 647 test hours. This terminal's initial reliability is therefore a regression function of the form Ae^{Bx}, where the parameter estimates from the test data and regression analysis[5] are A = 83.9 and B = −0.00162. Thus $h_a(x) = 83.9e^{-0.00162x}$, and the accelerated failure rate at $x_{Release}$ is $h_a(647) = 29.4\%$ failures/test hour, that is, 29.4% of the terminals would be expected to fail per test hour if the test were continued.

This is the expected failure rate for the firmware's initial release, and until the firmware is revised, it should remain constant. This failure rate must, however, be converted to an equivalent rate in a user environment. It must also be expressed as a function of calendar time to reflect the customer's perspective. For terminals, the relationship between execution time and calendar time is assumed to be linear because their firmware is exercised whenever customers use their terminals. Experience has shown that customers operate HP terminals an average of 2000 hours out of 8760 per year, so the calendar-time equivalent failure rate becomes:

$$h(t) = (1/4.38)h(x) \quad (3)$$

where t is calendar time. From equations 1, 2, and 3, the calendar-time failure rate in terms of accelerated execution-time failure rate is

$$h(t) = (1/4.38)(1/C)h_a(x)$$

or on an annual basis,

$$h(t) = (1/4.38)(1/C)(8760)h_a(x) = 2000(1/C)h_a(x) \quad (4)$$

Using the released failure rate for this terminal, the calendar-time failure rate for that version of the firmware then becomes:

$$h(t) = 2000(1/C)h_a(647) = (1/C)58800 \text{ \% failures/year} \quad (5)$$

Computing Accelerating Factor

At this point a prerelease estimate of the reliability of this firmware could be made if the accelerating factor C were known. Since h(t) is expected to be constant for the duration of the initial revision, a measured failure rate for that period of time can be equated to equation 5 to solve for the accelerating factor.

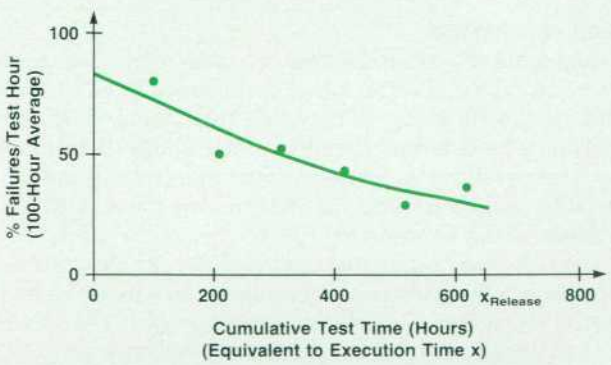


Fig. 2. *HP 2393A and HP 2397A prerelease failure rate.*

But how can such a failure rate be determined for firmware? There is no warranty to cause failure data to be gathered, nor is the industry in the habit of taking data of this nature. An answer lies in the fact that firmware (or software) can only be repaired by revising the customer's product. Those repairs can be tabulated in much the same fashion as hardware failure rate data is tabulated from warranty repair data. By assuming that a unit's firmware is repaired when its firmware is revised, and further, that a failure occurs on the same date as the repair, a failure rate can be computed based on unit revisions.

The initial revision of the HP 2393A/7A firmware was produced in all units from September 1985 to June 1986. If the sum of the individual unit revisions is divided by the sum of the monthly shipments over that period and multiplied times 100, the half-year average failure rate is found to be 0.615%. It is the half-year rate because only repairs to a unit that occur in the first six months from its date of shipment are reported and can be tabulated. If this rate of failure for a six-month window from shipment date is assumed to be constant, then the annualized failure rate is twice the six-month rate. Therefore, the overall average annualized failure rate for the version of this firmware that resulted from the testing process for the HP 2393A/7A is 1.23% failures/year.

Rearranging equation 5, substituting the average annualized failure rate, and solving for C yields:

$$C_1 = 58800/h(t) = 58800/1.23 = 47805. \quad (6)$$

C_1 is the accelerating factor for the HP 2393A/7A testing process.

A second terminal, the HP 2394A, was subsequently developed and its firmware tested similarly to the HP 2393A/7A's. The test-time failure rate data for this terminal is shown in Fig. 3. Following the same steps as above, its final failure rate from testing is determined by evaluating an exponential function fitted to the data at the final cumulative test time. The calendar-time equivalent failure rate from equations 4 and 5 is:

$$h(t) = 2000(1/C_2)h_a(384) = 67200/C_2 \text{ \% failures/year} \quad (7)$$

As before, this is the failure rate for this terminal's initial firmware release given some accelerating factor resulting from the testing process.

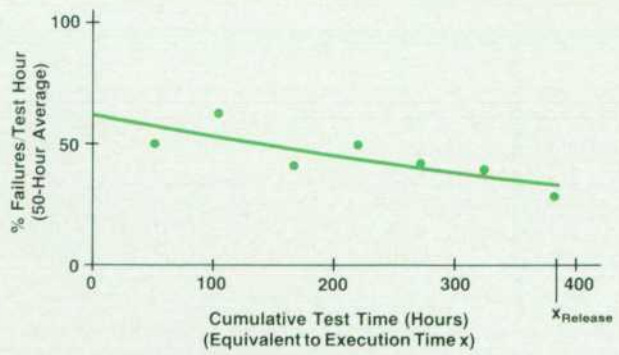


Fig. 3. *HP 2394A prerelease failure rate.*

Since this terminal has a predecessor for which an estimate of the accelerating effect of the testing process is available, a prerelease reliability estimate can be made. Substitution of C_1 for C_2 in equation 7 gives a prerelease estimate of the HP 2394A's reliability to be 1.41% failures per year, once released and in use by customers. Of course, this assumes that the accelerating effect of the second testing process is the same as that of the first, but it does provide an estimate of the reliability of the firmware before release and a potential measure to help decide when software or firmware is ready to release. As other projects are measured in this fashion, a history of computed accelerating factors will be built up, and a statistically more precise accelerating factor can be derived to estimate prerelease reliability.

The HP 2394A's initial revision of firmware was produced in all units from June 1985 to January 1986. Using the same technique as above for determination of a postrelease failure rate based on number of firmware revisions, this terminal's overall average failure rate was 1.58% for the first revision of the firmware. Rearranging equation 7 and solving for the accelerating factor gives $C_2 = 45532$. The failure rate after release compares favorably with that predicted before release, and the closeness of the accelerating factors (47805 to 45532) indicates that the testing process was fairly consistent in both cases.

Implications for the Testing Process

Several issues have arisen out of the application of the execution-time model to software. Use of an accelerating factor to describe the effect the testing process has on software reliability also raises questions about the variability of the testing process and the reliability improvement that results from it. Test coverage in the testing process is a critical aspect. How good a fit is required to conclude the model's applicability and what to do if the data from testing doesn't fit the model are other issues that merit discussion.

Examination of the testing process reveals a system that does more than verify reliable operation of the software. In a broader context, the failure data can be thought of as a measure of the performance of the reliability improvement process. Using principles of statistical process control, this data provides insight into ways to improve the process. A control chart facilitates the interpretation of such data. It also provides statistical evidence of improvement when reliability growth occurs. Fig. 4 is an example of a "u" control chart constructed from data for the HP 2394A. The chart is based on the Poisson probability distribution.[6] It is constructed from n successive paired observations on the number of failures detected and the corresponding number of test hours required. For each of the paired observations, the sample failure rate, u(i), is computed:

$$u(i) = w(i)/t(i) \quad (8)$$

where w(i) = number of errors detected or failures (detection of an error for software also causes a failure) in the ith sample and t(i) = number of test hours required to detect the w(i) errors. The central line for the chart, $\bar{u}$, is simply the weighted average sample failure rate:

$$\bar{u} = \frac{\sum_{i=1}^{n} w(i)}{\sum_{i=1}^{n} t(i)} \quad (9)$$

The control limits depend on the number of test hours recorded for each observation. The upper control limit, UCL(i), for the ith observation is:

$$UCL(i) = \bar{u} + 3\sqrt{\bar{u}/t(i)} \quad (10)$$

The chart is constructed by plotting u(i), $\bar{u}$, and UCL(i) versus cumulative test time. For practical purposes in this instance, the lower control limit is zero.

In this example, the process is "out of control" at x = 53 and at x = 124 hours. Logbook entries indicate that these times correspond to the addition of new features to the base software. This was confirmed as the assignable cause of the variations leading to the failure rate spikes. This observation highlighted segments of code that were unusually prone to error and suggested where engineering effort should be focused. The process also shows signs of statistically significant change after 272 hours. Beyond this point, 9 of 13 consecutive points lie below the centerline, a signal that the underlying failure rate has decreased. This provides additional statistical evidence that the reliability of the software has grown or been improved as a result of the testing process.

Coverage in testing must meet a basic assumption of the model (see assumption 3 in Appendix) to produce unbiased failure rate estimates. If coverage is not representative of the use space of customers, failure rates from testing will not correlate with actual use of a product. To ensure that estimates of the failure rate are unbiased, it is necessary to cover all of the features in the testing process. It is especially important that no untested features are released to production. This minimizes the risk of significantly overestimating the reliability perceived by the customer.

The fitted curves shown in Figs. 2 and 3 were obtained using the method of least-squares applied to log-transformed failure rate data. Statistical tests indicated that the respective models are reasonably good fits to the observed trends. In both cases the strength of the exponential trend was significant compared with the residual variation or "noise." Further, the fitted curves accounted for most of the observed variability in failure rate. Other sources of systematic variability were not apparent.

What if the failure rate data doesn't fit the decaying exponential form very well? If the data is not described by some form of monotonically decreasing function, it may signal the presence of a reliability problem. With hardware, early wearout mechanisms detected in life testing are likely to cause the product to be sent back for major redesign. The same applies to the development of software. If the reliability is not growing (i.e., the failure rate is increasing), the design may have a fundamental problem that will require redesign. If, on the other hand, the failure rate is decreasing but not exponentially, a reliability growth model that incorporates another functional form can be used.

Conclusion

Software reliability can be quantified through the application of a reliability growth model to test data. Simple structuring of the testing process provides the data necessary and the use of an empirically derived accelerating factor offers a way to make prerelease estimates of a software product's reliability. Combined with correlation of postrelease failure data over a history of products, these estimates can be made with increasing statistical precision. These techniques can be implemented easily without major modification of existing processes and can be of immediate

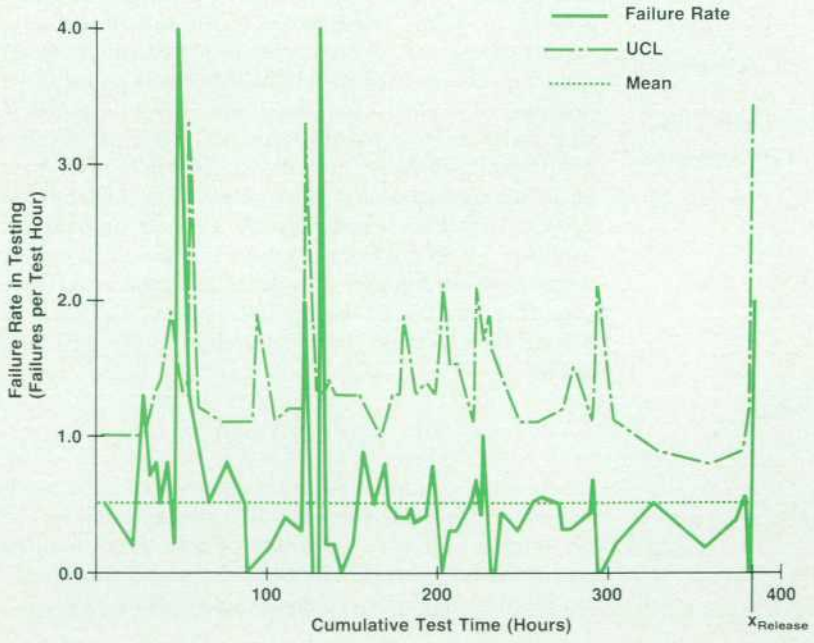


Fig. 4. *HP 2394A testing process control chart.*

use to help understand and improve this important quality attribute.

References

1. R.A. Brooker, S. Gill and D.J. Wheeler, "The Adventures of a Blunder," *Mathematical Tables and Other Aids to Computation*, Vol. 6, no. 38, 1952, pp. 112-113.
2. J. Freund and I. Miller, *Probability and Statistics for Engineers*, Prentice-Hall, 1985.
3. J. Musa, "A Theory of Software Reliability and Its Application," *IEEE Transactions on Software Engineering*, Vol. SE-1, no. 3, September 1975.
4. J. Musa, "Validity of Execution-Time Theory of Software Reliability," *IEEE Transactions on Reliability*, Vol. R-28, no. 3, August 1979.
5. *Statistics Library for the 200 Series Computers*, product number 98820A, Hewlett-Packard Company, 1982.
6. E.L. Grant and R.S. Leavenworth, *Statistical Quality Control*, Fifth Edition, McGraw-Hill, 1980.
7. R.E. Barlow and F. Proschan, *Statistical Theory of Reliability and Life Testing Probability Models*, Holt, Reinhart and Winston, 1975.

Appendix
Derivation of the Software Reliability Model

The approach taken in this paper uses a statistical model to characterize the reliability of software. The model is founded in theory that is routinely applied to diverse problems in the life sciences, economics, and hardware engineering. Although this theory has been used extensively in the past, its application to software testing is a comparatively recent development.

Derivation of the model begins with a probabilistic definition of reliability.[7] Here, T is a continuous random variable that denotes the elapsed time (or more generally the stress) to failure for a system under consideration. Let f(t) be the probability density of T and let F(t) denote the associated cumulative distribution function. The reliability of the system at time $T = t$ is simply the probability that the elapsed time to failure exceeds t. This variation of this probability with time is described by the reliability function, R(t), which is the complement of the cumulative distribution function:

$$R(t) = 1 - F(t) \quad \text{(I)}$$

The reliability function defines the unconditional probability that the time to failure exceeds t. Often, however, it is necessary to know the probability that the system will fail at any particular moment, given that it has not failed up to that time. The conditional probability of failure in a finite interval $(t, t+\Delta t)$ given that failure has not occurred up to time t, is:

$$\Pr[t<T<(t+\Delta t)|T>t] = [F(t+\Delta t) - F(t)]/R(t) \quad \text{(II)}$$

The instantaneous failure rate at time t, h(t), is obtained by first dividing this conditional probability by Δt, then taking the limit as Δt approaches zero. This reduces to:

$$h(t) = f(t)/R(t) \quad \text{(III)}$$

The function h(t) is also known as the hazard rate. It can be shown that the reliability function of the system can be obtained directly from the hazard rate alone. In general:

$$R(t) = \exp\left[-\int_{-\infty}^{t} h(u)du\right]$$

where u is a variable of integration.

The hazard rate is a particularly useful concept for two reasons: 1) It is a readily interpretable quantity that describes the tendency to fail as a function of elapsed (failure-free) time, and 2) all salient functions of T can be derived from it, including the probability density and reliability functions. In this context, knowledge of the hazard rate is sufficient to describe completely the reliability of the system under consideration. It also plays a central role in the development of the software reliability model.

Development of the Theoretical Model

Musa[3,4] showed that these concepts can be used to describe the reliability of software. In his approach, the stress variable is the actual CPU time used in execution. The model is therefore known as the execution-time model, of which the present model is a modified version. The key assumptions are:

1. Failures of the software system are caused by defects in the system. Any such defects are independent of each other and are distributed at any time with a constant average occurrence rate (per instruction) throughout the code between failures or repairs.
2. Types of instructions are well-mixed. Execution time between failures is large compared to average execution time per instruction.
3. Test cases are applied across the breadth of the product's features. Testing represents the environment in which the software will be used.
4. The set of inputs for each run of the software, whether during test or during customer use, is selected randomly.
5. All failures that occur are detected.
6. The defect causing each failure is removed without introducing new defects. The defect is removed before execution resumes or its rediscovery is not counted.

Under the assumptions, the instantaneous failure rate is piecewise constant, and is proportional to the residual number of errors in the system:

$$h(x) = k(N_o - n) \quad \text{(V)}$$

where N_o is the number of inherent errors in the program before testing begins, n is the number of errors corrected by (cumulative) execution time x, and k is a proportionality constant.

The instantaneous rate at which errors are being removed is identically equal to the instantaneous failure rate:

$$dn/dx = h(x) \quad \text{(VI)}$$

Equations V and VI, when combined, lead to a differential equation in n and x. Its solution is:

$$n(x) = N_o\left[1 - e^{-kx}\right] \quad \text{(VII)}$$

Using equation V, the instantaneous failure rate as a function of time becomes:

$$h(x) = h_o e^{-\frac{h_o x}{N_o}} \quad \text{(VIII)}$$

where $h_o = kN_o$ is the failure rate at the onset of testing. Equation VIII provides the theoretical basis for fitting an exponential curve to the empirical failure rate data and is an example of a reliability growth model, the decreasing failure rate representing "growth" of the software's reliability.

Reader Forum

The HP Journal encourages technical discussion of the topics presented in recent articles and will publish letters expected to be of interest to our readers.
Letters must be brief and are subject to editing.
Letters should be addressed to:

Editor, Hewlett-Packard Journal, 3200 Hillview Avenue, Palo Alto, CA 94304, U.S.A.

Editor:

I was fascinated by the description of HP Precision Architecture in the August 1986 issue of the *HP Journal* ("Hewlett-Packard Precision Architecture: The Processor," p. 4). While the architecture is interesting, there is no publicly available data that proves that RISCs are economically beneficial, in the sense that they really reduce software costs in the design, coding and maintenance phases. I am researching this issue in a graduate course on software engineering economics.

HP has a unique position as a leader of RISC technology, but at the same time needs to solve the problem of compatibility. I assume that HP has assessed this issue and has data to shed light on it. I would be very grateful if you could provide me with such data.

While the purpose of this research is purely academic, please be advised that most graduate students in this course, myself included, are also employed by engineering companies.

Yoav Talgam
University of Texas at Austin

You ask a question of great importance to all users of computers: What impact does RISC-style processor architecture have upon the cost of software?

The characteristics of processor architecture that most affect software costs are those that are directly reflected to higher-level language programmers. For example, if a small address space "shows through" the compilers to the users of a language, requiring them to separate their code and data into unnatural fragments, then there is clearly an incremental cost in design, coding, and maintenance of such software. On the other hand, if the compilers "mask" such characteristics from programmers, then no additional software costs are incurred (though run-time performance may suffer). The HP Precision Architecture imposes fewer addressing constraints than most. Because at least 48-bit virtual addresses are supported, this is not a cause for worry.

A simpler instruction set is not, in itself, a burden, because the compilers and optimizer take care of the mapping from higher-level languages to the machine. If a programmer requests an integer division from Fortran, for example, the compilers arrange for the division to be done by the most expeditious means—with or without a DIVIDE *instruction.*

To first order, Precision Architecture machines are neutral with respect to software costs. This is so because we successfully achieved our objective of efficiently supporting all important languages and their data types. As a result, there is no increase in the complexity or volume of user software, and hence in user software costs. (You might wonder about the cost of our compilers, because they are more complex than nonoptimizing compilers, but this is compensated by simpler hardware and the absence of microcode.)

The second-order effects of reduced-complexity machines are positive. Improving the cost/performance of computer systems permits a greater emphasis on software productivity, which is frequently compromised by the need to conserve precious processor cycles. For example, the HP 9000 Model 840 Computer has an integrated symbolic debugging capability for C, Pascal, and Fortran 77. This represents a use of the greater addressability and speed of the machine to deliver improved software productivity. The ALLBASE relational/network data base system is another example of using additional raw power to provide greater application ease and flexibility.

Any task involving the coordinated storage, retrieval, and management of information can be improved by the appropriate application of computer power, and the production of reliable, effective software is no exception. You should expect to see a new generation of computer-aided software engineering tools emerge, aided by the availability of powerful, low-cost programming workstations. The cost/performance advantage of HP Precision Architecture machines will play an important role in bringing about such integrated software development environments.

Michael J. Mahon
Manager
Computer Language Laboratory

HEWLETT-PACKARD JOURNAL

April 1987 Volume 38 • Number 4

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

HEWLETT PACKARD

5953-8559