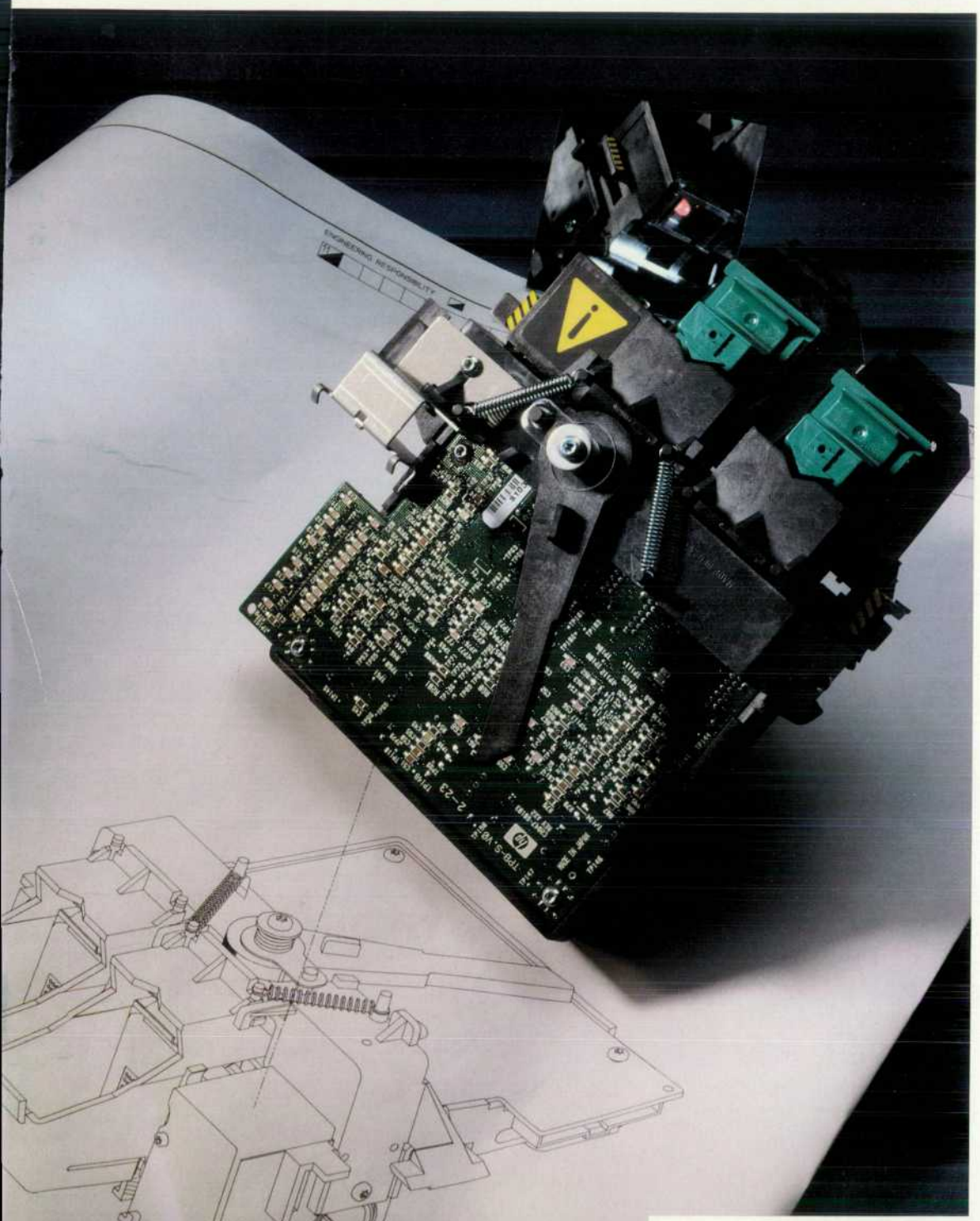
HEWLETT-PACKARD
JOURNAL

December 1992



HEWLETT-PACKARD
JOURNAL

December 1992 Volume 43 • Number 6

Articles

Departments

The Hewlett-Packard Journal is published bimonthly by the Hewlett-Packard Company to recognize technical contributions made by Hewlett-Packard (HP) personnel. While the information found in this publication is believed to be accurate, the Hewlett-Packard Company disclaims all warranties of merchantability and fitness for a particular purpose and all obligations and liabilities for damages, including but not limited to indirect, special, or consequential damages, attorney's and expert's fees, and court costs, arising out of or in connection with this publication.

Subscriptions: The Hewlett-Packard Journal is distributed free of charge to HP research, design and manufacturing engineering personnel, as well as to qualified non-HP individuals, libraries, and educational institutions. Please address subscription or change of address requests on printed letterhead (or include a business card) to the HP headquarters office in your country or to the HP address on the back cover. When submitting a change of address, please include your zip or postal code and a copy of your old label. Free subscriptions may not be available in all countries.

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

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

In this Issue

Drafting plotter technology occupies a good many of our pages this month, with two design groups reporting on their new large-format drafting plotters—the HP DesignJet and the HP DraftMaster Plus.

HP's thermal inkjet technology makes the DesignJet plotter as low-cost as a pen plotter and as fast as an electrostatic plotter. The pens, or print cartridges, are HP DeskJet printer cartridges, proved reliable by many years of DeskJet experience. The plotter accepts either vector commands or raster commands, uses regular drafting plotter media, and automatically cuts and stacks roll media. The article on page 6 introduces the DesignJet and discusses the issues of print quality, media, media handling, and the user interface. The electronic and firmware design, including built-in vector-to-raster conversion and three custom integrated circuit chips, is the subject of the article on page 16. Because it wouldn't have been able to plot fast enough with one pen, the DesignJet uses two, and how it keeps the two pens properly aligned is explained in the article on page 24. The unusual chassis is a rugged, precise unit made up of low-cost nonprecision parts held rigidly in place by a cleverly designed structure (page 28). In developing the mechanical architecture, the designers found that extra time invested in communication before beginning prototype design paid off in a shorter overall project and lower-than-expected costs, as they explain on page 32.

The DraftMaster Plus drafting plotter is an enhanced version of the DraftMaster pen plotter. Improved plotting reliability, improved media handling, and an improved user interface all contribute to increased user productivity by reducing the need for operator intervention during plotting. The most common customer complaints about pen plotters concern reliability—pens running out of ink, drying out, or clogging. The SurePlot plotting system includes improved pens, extra pens, and a sensor system that detects defective lines and automatically replaces faulty pens so that 998 out of 1000 drawings meet user expectations (page 35). Media handling is improved by roll feed and a new media cutter and tray (page 42). The enhanced user interface offers a redesigned front panel and simplified selection of pens, settings, and drawing quality (page 49).

Hewlett-Packard's reduced instruction set computer architecture, called PA-RISC, provides for multiprocessor implementations in which several processors share the work. Adding a processor board to a system turns out to be a very cost-effective way of improving the performance of a computer system, particularly for online transaction processing (OLTP). The operating system schedules the processors and keeps them from interfering with each other. The first multiprocessor implementation of the PA-RISC architecture supports up to four processors and is available in versions that run either the HP-UX* operating system or the MPE/iX operating system. The article on page 56 tells how the HP-UX kernel was modified to support multiprocessor operation and shows how much the performance is improved by adding processors. A feature of multiprocessor HP-UX is that, unlike many multiprocessor operating systems, it can run on a uniprocessor system with no performance penalty.

State-of-the-art very large-scale integrated circuits like microprocessors can contain millions of transistors and have hundreds of inputs and outputs. For various technical and logistical reasons, such chips are rarely mounted directly on printed circuit boards, but instead are supplied in packages, which are soldered onto the boards. While some packages provide high-performance electrical and thermal environments and some are easy to assemble and rework, the ideal package would meet all of these requirements. A packaging technology that comes closer to the ideal than any of its predecessors is described in the article on page 62. Based on an existing technology called tape automated bonding, or TAB, it maintains TAB's advantages of good performance and lower cost, but doesn't require TAB's expensive facilities and is easy to demount and rework. Called demountable TAB, or DTAB, it was developed by HP's Integrated Circuits Business Division.

As explained in our August 1992 issue, the HP 9000 Series 700 computer workstations have a built-in, or core, input/output system that provides a variety of industry-standard I/O ports. Because many customers will need higher performance or want to expand their systems, the Series 700 workstations also offer an I/O expansion bus, which accepts plug-in cards that provide additional industry-standard I/O ports. The EISA bus (EISA stands for Extended Industry Standard Architecture) was selected for the Series 700 expansion bus because it is a high-performance bus that is expected to meet the needs of HP customers and because it meets HP's goal of converging to an industry-standard I/O bus for all new workstations. The article on page 78 describes the adapter that provides the EISA bus to the outside world, acting as a bridge to the internal Series 700 system bus. Four types of plug-in EISA I/O cards are now available for the Series 700: an IEEE 802.3 (Ethernet) LAN card, an IEEE 488 (HP-IB) card, a SCSI (Small Computer Systems Interface) card, and a programmable serial interface card. Discussed in the article on page 83, the cards use EISA bus master and DMA (direct memory access) methods to take advantage of the EISA burst-cycle protocol for high-speed data transfer. The software for the EISA SCSI interface card is explained in the article on page 97, which also talks about recent extensions to the SCSI standard.

The article on page 109 is from HP's Worldwide Support Systems (WSS) organization, where they develop mission-critical systems for HP's worldwide customer support business. Seeing the emergence of client/server solutions based on open systems concepts, WSS began to migrate their applications to this new paradigm while it was still in its infancy and standards were still being defined. They developed a technical architecture that provides a framework for designing modern, integrated client/server applications. The article explains the architecture and relates WSS's experiences in migrating two existing applications.

December is our annual index issue. The 1992 index starts on page 120.

R.P. Dolan
Editor

Cover

The pen carriage of the HP DesignJet large-format thermal inkjet drafting plotter is shown with a DesignJet plot. The two print cartridges (HP DeskJet printer type) can be seen, as can the lever or adjustable cam that is part of the mechanism for aligning the pens in the media direction (scan direction alignment is done by adjusting pen-firing timing). The red light-emitting diode (in the mirror) is part of the adjustment system—it illuminates a test line that's used to measure and correct pen misalignment. (The mirror isn't part of the pen carriage. We just put it there to show the LED. We also simulated the LED light for this photo.)

What's Ahead

The February issue will be entirely a lightwave technology issue, featuring articles on the design of the following fiber optic test instruments:

- HP 8504A precision reflectometer
- HP 8146A optical time-domain reflectometer
- HP 83440 Series lightwave detectors
- HP 8167A and 8168A tunable light sources
- HP 81534A return loss module for the HP 8153A lightwave multimeter.

Leading off the issue will be a paper summarizing the contributions of HP Laboratories' photonics research program to HP's lightwave product line.

A Large-Format Thermal Inkjet Drafting Plotter

The HP DesignJet drafting plotter combines the low cost of pen plotters with the speed of electrostatic plotters. Throughput is almost independent of drawing complexity. The plotter uses the same roll and sheet media as pen plotters, and in roll mode, automatically cuts and stacks plots for unattended operation.

by Robert A. Boeller, Samuel A. Stodder, John F. Meyer, and Victor T. Escobedo

The major contribution of HP's first large-format drafting plotter, introduced in 1981, was high performance at a much lower price than had previously been available. Through the application of HP's proprietary inkjet technology, the new HP DesignJet large-format drafting plotter offers customers the same kind of contribution in performance at low cost while providing greater reliability of the ink delivery system and greater user friendliness.

The DesignJet plotter (Fig. 1) can plot a complex D-size drawing in three minutes on commonly available media. It is quiet and can produce several hundred plots without a pen change. It accepts single sheets or, for unattended plotting, a roll of media. When a roll is used, a built-in cutter separates the plots into sheets, which fall into a media tray below the plotter. The plotter accepts vectors (HP-GL/2 instructions) or raster data (HP Raster Transfer Language). Resolution is 300 dpi.*

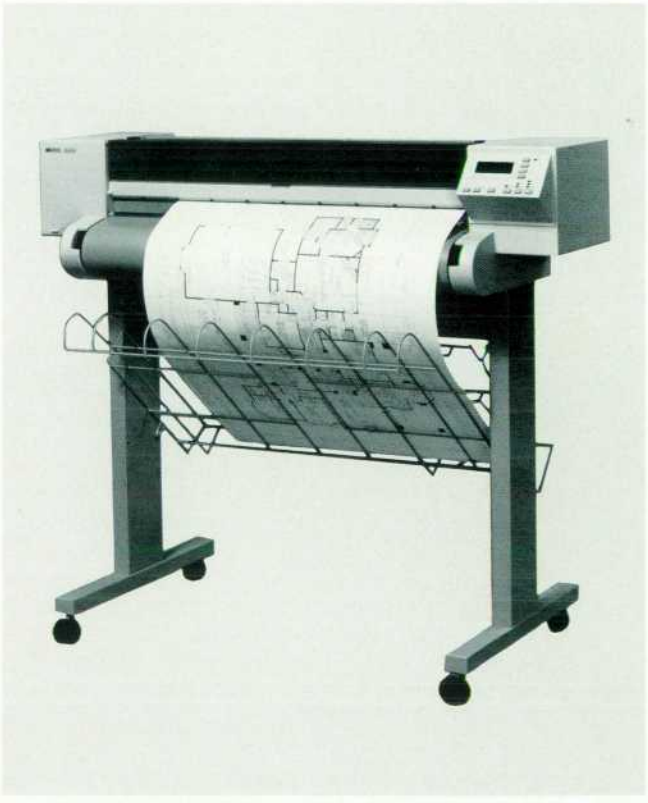


Fig. 1. The HP DesignJet plotter uses thermal inkjet technology to produce large-format plots on commonly available drafting media. A roll-feed mode allows unattended plotting with automatic cutting and stacking of completed plots. Resolution is 300 dots per inch and plot time for a D-size plot is about three minutes.

Low-cost plotting has been dominated by the traditional pen plotter. Pen plotter performance is usually characterized in terms of acceleration and velocity. The greater the complexity of the image, the longer it takes to plot. An architectural plot on E-size paper might be in the range of 100,000 to 1,000,000 vectors. A typical throughput profile for a 4g, 25-ips plotter might look as shown in Fig. 2.

In contrast, the DesignJet plotter's performance does not change much with drawing complexity. Throughput is more a function of the page size and the vector transmission and conversion time. The DesignJet plotter's built-in Intel 80960 processor and electronic architecture make vector transmission and conversion time very small. A typical throughput profile for the DesignJet plotter is shown in Fig. 3.

HP inkjet technology also provides customers with an improvement in writing system reliability. The inkjet pens each hold about 38 cc of ink, and no priming is needed when starting up. The DesignJet plotter uses two pens, which will last for about 200 reasonably complex E-size plots. The plotter can be left unused for many weeks, and the writing system will always work without intervention when it is needed.

* A 600-dpi addressable resolution version known as the DesignJet 600 is now in production.

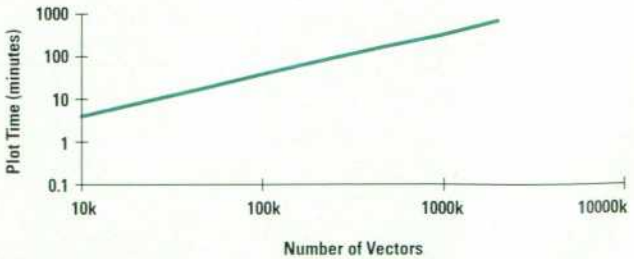


Fig. 2. Pen plotter plot time as a function of plot complexity.

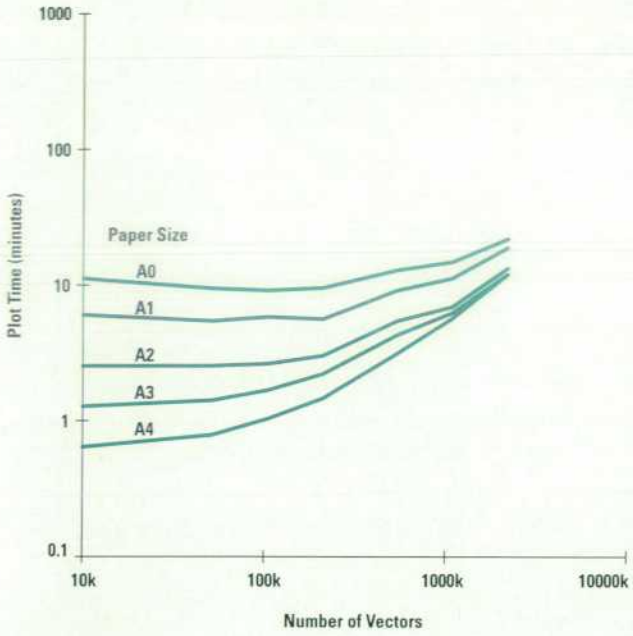


Fig. 3. HP DesignJet plotter plot time as a function of plot complexity and size.

The DesignJet plotter represents a major contribution towards making a large plotter more friendly. Sheets and rolls load easily from the front of the machine and are ejected toward the front, making plot retrieval easy. The built-in cutter works well on all media types, including Mylar, which is usually a challenge because of its silica content. A passive media stacking system stacks up to 20 plots.

The DesignJet plotter is one of the first HP products to incorporate a modular I/O slot that will allow many HP peripherals to use the same I/O cards. At the product's introduction, there were MIO (modular I/O) cards available that allow the DesignJet plotter to connect to some networks and to HP-IB (IEEE 488, IEC 625) systems. The MIO strategy promises to provide our customers with a much broader range of connectivity solutions as new cards are developed.

Design Challenges

We knew that print quality is our customers' most important need. But larger-format plotters require very tight tolerances over the large distances required to span an E-size sheet of paper. Unfortunately, the laws of physics dictate that large beams lose their stiffness rapidly as they get longer. Therefore, we had to engineer the product right from the beginning to be stiff enough and accurate enough to hold the required tolerances.

To increase the plotting throughput, the DesignJet plotter incorporates two pens instead of just one. Two pens can print twice as fast as one, but must be aligned accurately, so that the customer does not see any banding or gaps at the junction between pens. The same throughput requirement put demands on the vector-to-raster converter that transforms the HP-GL/2 input into the raster data that the pen needs.

We needed to make a major improvement in the ease of using and loading the machine. Large-format plots are difficult to handle, and our goal was to try to make the plotter as easy to use as an HP DeskJet printer. We also needed a way to stack plots as they came out of the machine.

We needed to make the machine reliable so that it could be used unattended. This required careful design of the media path and the cutter system so that paper would never jam or tear and the machine would be guaranteed to work all the time without user intervention.

We needed to be very cost conscious, and this required care in meeting design-for-assemblability and design-for-manufacturability goals. It also created a need to integrate more of the electronics.

We had a tremendous number of functions to implement in the firmware. Some of this functionality provides compatibility with other HP plotters, such as front-panel control of line types and widths. Other functionality provides new opportunities for using the plotter, such as our RTL (Raster Transfer Language), which allows vectors and raster data to be mixed on the same plot, and our MIO support, which allows the plotter to be used on a network directly, without connecting through a server.

Major Features

The DesignJet pen carriage holds two print cartridges, along with a mechanism that allows one of the cartridges to move slightly relative to the other so that optimal print quality can be achieved. The carriage also holds optical sensors for automating this adjustment and for automatically detecting the paper edges.

The Y-axis drive servo system, which moves the carriage back and forth on precision rails, includes a linear encoder for maximum accuracy. The same Y drive also programmatically engages a separate cutter carriage that cuts roll-feed media. The media cutter consists of a carriage-mounted rotary blade that is spring-loaded against a fixed linear blade.*

A pen service station caps the inkjet cartridges to prevent the ink from drying out when the plotter is not in use. The service station includes a wiper to help maintain a clean nozzle surface, which helps print quality. Also included is an ink-drop detector, which is used to determine if the cartridge is firing properly.

The X-axis drive servo system moves the media. The media drive system can handle a wide range of media widths up to 36 inches, and is also capable of driving a wide range of media types. A precisely dimensioned rubber roller provides a friction drive.

The roll-feed assembly accepts rolls of media up to 50 meters in length. A sheet stacking system collects the plots as they are cut off the roll.

System Block Diagram

The DesignJet plotter block diagram, Fig. 4, shows the relationships of all of the major functional areas of the plotter.

* The cutter design is similar, but not identical, to the media cutter of the HP DraftMaster Plus plotter (see article, page 42).

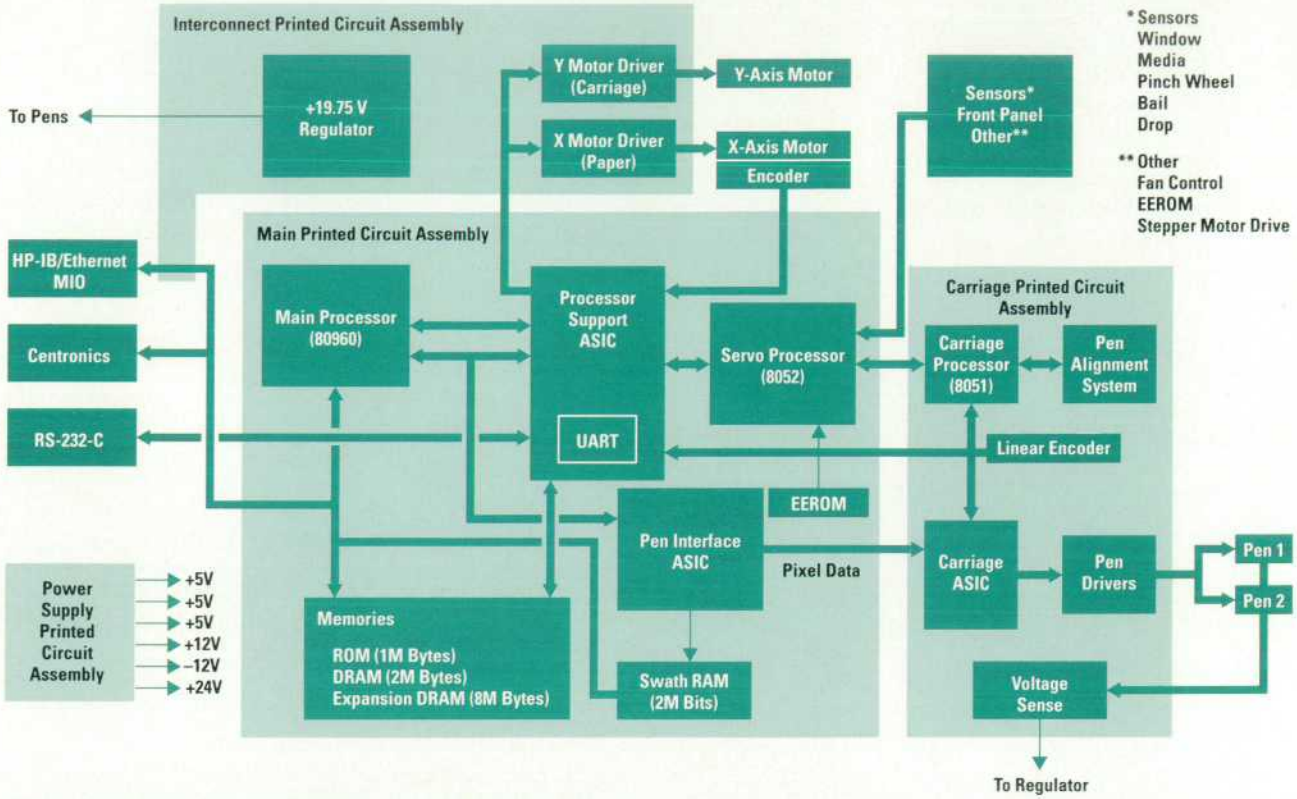


Fig. 4. Simplified system block diagram of the HP DesignJet plotter.

The input/output ports are treated as memory addresses. The RS-232 universal asynchronous receiver/transmitter (UART) is part of the processor support ASIC (application-specific integrated circuit). There are three ASICs in the DesignJet plotter.

1M bytes of system read-only memory (ROM) is used to store system firmware. 2M bytes of dynamic random-access memory (DRAM) is standard. Expansion slots allow expansion of the DRAM to 10M bytes. 2M bits of RAM is used to store swath information. An electrically erasable read-only memory (EEROM) IC is used as nonvolatile memory to store variables that must be retained when power is off.

DRAM addressing is controlled by the processor support ASIC. The main processor, an Intel 80960, communicates with the servo processor (an 8052) through the processor support ASIC, since the two processors run at different clock frequencies.

The pen interface ASIC removes swath pixel data from the swath RAM and transfers the data to the carriage ASIC, which is located on the pen carriage printed circuit assembly, through the trailing cable. The carriage ASIC creates the proper signals for the pen drivers.

The processor support ASIC receives encoder feedback data from the X-axis and Y-axis motors and outputs the data to the servo processor for calculation of the necessary pulse width modulation (PWM) signals to drive the motors. The processor support ASIC outputs the PWM signals to the motor drivers, which are located on the interconnect printed circuit assembly.

Front-panel and all sensor input (except the line sensor) goes to the servo processor, which also controls the fan, the stepper motor that moves the pen nozzle wiper blade, and the EEROM. A voltage regulator on the interconnect printed circuit assembly controls the pen drive voltage.

The pen carriage line sensor output goes to an analog-to-digital converter (ADC), which outputs the converted signal to the carriage processor (an 8051).

DesignJet Print Quality

Good print quality is of prime importance for any printer or plotter. An understanding of the customers' print quality needs and how to meet them was critical to the success of the DesignJet plotter.

During the investigation phase of the DesignJet project both internal and external surveys were conducted to determine how well a large-format, monochrome, thermal inkjet plotter would be perceived by the target market. Pen plotter, electrostatic, and inkjet plots were shown to customers. Customer comments on print quality were recorded and categorized. Next, specific print quality attributes were identified and special surveys were conducted to quantify what specifications had to be achieved to meet the customers' needs. These special surveys included media and print mode testing, banding, vertical line straightness, and cockle testing.

Media and Print Mode Testing

The target market is primarily composed of pen plotter users who need higher throughput. These users prefer to use the media types that are commonly available today for pen plotters. We found that most commonly available plotter bond papers exhibited excellent print quality when printing with one pass of the inkjet pen over the paper. However, when printing area fills on some vellums and translucent media, two passes of 50% ink density per pass were required

before customers would consider the samples to be final-quality plots (see Fig. 5). The two-pass print mode worked quite well to improve the uniformity of area fills. Commonly available polyester film media did not work well even with multiple passes. Therefore, a special film that is more receptive to the ink was developed to work with a two-pass print mode.

Banding and Line Straightness

To determine what were acceptable limits for banding, a survey was designed to test user sensitivity to this print quality attribute. Plots were generated on a highly accurate drum printer. Adjacent swaths were printed with known displacements from the ideal. This resulted in a range of plots, each with a different level of discrete swath boundary misalignment (bands). When adjacent swaths are placed too far apart a light band or gap is visible and when adjacent swaths are placed too close together a dark band or overlap appears. Users were asked to rank the plots for acceptability (see Fig. 6). By evaluating the survey data we were able to determine how accurate the specifications for the printing mechanism would have to be in the paper axis direction.

Line straightness was tested in much the same way. Plots were printed with discrete offsets at swath boundaries in the direction of swath scan. By evaluating the survey results, we were able to determine how accurate the specifications for the printing mechanism would have to be in the pen scan direction.

Cockle Testing

Cockle is the wrinkled appearance of paper after having being soaked in water and then left to dry. The high water content of the HP DeskJet pen ink results in significant cockle when applied in large area fills. Analysis of customer plots showed that most users do not plot with high print densities. However, solid area fills are used for small logos and thick lines.

To better understand user sensitivity to cockle, a test was designed to determine how dense a plot could be and still have acceptable cockle. Plots were printed with a pattern of evenly spaced square area fills. Area size and spacing were varied from plot to plot. Users were then asked to rank the plots for cockle acceptability. The survey results showed us that for area fill sizes and spacing that would be typical for our target market, there would be no problem with cockle.

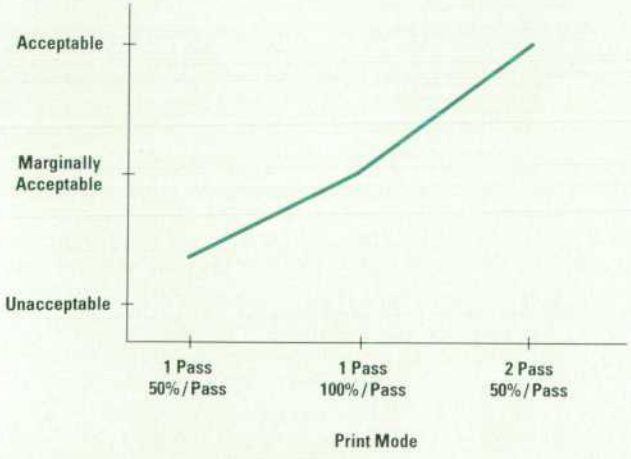


Fig. 5. Plot acceptability as a function of print mode for large-format inkjet plots (vellum and translucent media).

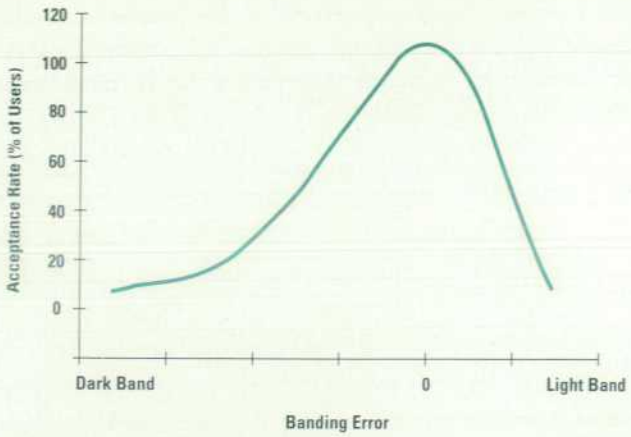


Fig. 6. Plot acceptability as a function of banding error.

Design Issues and Technical Risks

After the primary print quality specifications were established based on user needs, the design effort commenced to find ways to achieve these goals at the lowest possible cost. Key elements of the design were closely analyzed and risks were balanced. Worst-case analysis was done early to prove that various concepts could meet the required error budgets. The areas in which it was most difficult to meet the required specifications were banding and line straightness.

Banding error is the sum of errors caused by paper advance accuracy, pen-to-pen alignment in the paper axis, and failed or weak pen nozzles. The banding tolerance specification is taken up mostly by pen-to-pen alignment and paper advance errors.

On the DesignJet plotter, two DeskJet printer pens are used essentially as one larger pen to meet throughput goals. Worst-case analysis of the pen carriage tolerances and DeskJet pen tolerances showed that this concept would result in banding because of pen misalignment in the paper axis, and that the banding would be much greater than the customer would tolerate. An automatic pen alignment system was designed to minimize the amount of error budget consumed by pen alignment.

The allowable error remaining for the paper advance system resulted in a considerable design challenge. Several alternative designs were analyzed using Monte Carlo simulation methods. The winning concept uses an indexing worm drive gear transmission scheme. This approach virtually eliminates cyclical errors caused by the motor, encoder, and worm pinion. To reduce the errors caused by variation of the effective diameter of the drive roller, a calibration is performed on the production line using the DesignJet's own internal reference.

Achieving the line straightness goal also proved to be a difficult task. The DesignJet plotter not only requires two DeskJet pens to act as one larger pen but also needs to print bi-directionally to achieve the throughput goals. Much attention had to be given to the alignment of the pens in the scan

direction to achieve the vertical line straightness goal. A scheme was implemented by which pen alignment error in the scan direction and bidirectional error (carriage deadband) are measured with an optical sensor. These errors are then corrected by varying the timing of ink-drop firing.

The difficulty arose when we tried to align pens that exhibit excessive spray along with the main ink drop. The effects of spray on alignment get worse as the pen-to-paper distance increases, but moving the pens close to the paper risked the possibility of a pen drag because of cockle in a high-ink-density plot. The problem was minimized by doing a careful worst-case analysis of the pen-to-paper distance and moving the pen as close to the media as possible. Cockle amplitude was measured as a function of time for paper and translucent bond to determine how close the pen could be to the paper without any possibility of touching (see Fig. 7).

As mentioned above, cockle dictates how close the pen can be to the paper, which ultimately affects line quality. To minimize cockle, the paper path is designed to constrain the paper physically as much as possible. The paper is wrapped around a drive roller and is tensioned by an overdrive roller. In this way, paper cockle in the printing area is forced to a high-spatial-frequency, low-amplitude state, allowing the pen to be located as close as possible to the paper.

Production Control of Print Quality

Meeting print quality goals is dependent not only on careful design but also on careful production control to ensure that no product is shipped that does not meet all print quality specifications. On the DesignJet production line, several tests measure the performance of each machine for each print quality specification (see Fig. 8).

The banding specification is checked by four tests. First, cyclical errors in the paper drive are automatically measured and then analyzed with an FFT (fast Fourier transform) algorithm. From this data, a faulty part in the drive train can be identified and replaced. Second, accurate paper movement is ensured by an absolute accuracy calibration test. Next, weak or faulty pen firing is checked by visual examination of a test plot. Lastly, print cartridge alignment in the paper direction is measured with the aid of a video

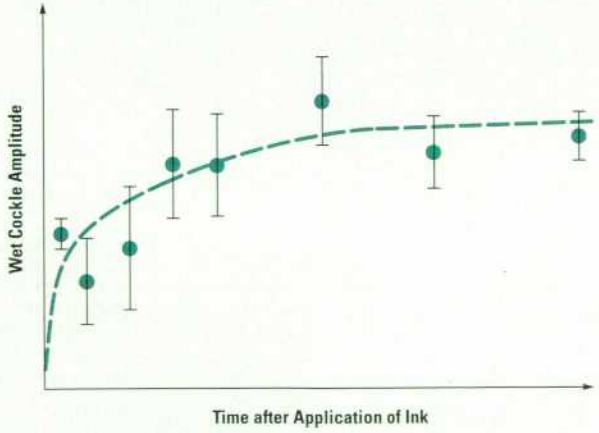


Fig. 7. Measurements of wet cockle (paper wrinkling after being wet and dried) as a function of time.

Fig. 8. Assembly line print quality attribute control for the HP DesignJet plotter.

microscope measurement system. Vertical line straightness is also ensured by measuring a print sample with the video microscope system.

Proper pen height above the paper is checked by two separate tests. First, slider rod straightness is measured as part of the chassis assembly procedure. Second, pen-to-paper distance is measured directly for each machine by installing a height gauge in the pen stall and measuring the distance to the platen at three different locations.

The final check for print quality is done by human inspection of a demonstration plot at the last station on the assembly line.

DesignJet Media

The media set plays an important part in the customer acceptability of a plotter. It was a requirement that the Hewlett-Packard media recommended for the new DesignJet thermal inkjet plotter provide consistent high quality, equal to or better than other brands on the market. The project goal was to use the current Hewlett-Packard pen plotter media set for the DesignJet plotter, the only exception being the drafting film. This goal presented a challenge in two areas: opaque bond paper and vellum.

Opaque Bond Paper

The then-current Hewlett-Packard bond was being changed concurrently with the development of the DesignJet plotter because of Hewlett-Packard environmental concerns and media archivability issues with the existing acid-process bond paper. A project had been initiated to develop a new alkaline plotter bond paper that would be better for the environment, provide improved archivability, and have superior print quality. Adding the requirement to support a thermal inkjet plotter increased the complexity and risk of the project and necessitated meeting the more aggressive plotter development schedule.

A major impetus for this alkaline paper project was the environmental issue. Alkaline paper processes use chemicals that are less harmful to the environment. Paper companies were changing their "plain papers" to alkaline partially in response to this concern. Plotter papers were not being considered for this change by the majority of plotter paper manufacturers, but we felt that Hewlett-Packard should be among the leaders in this effort.

Pen plotter and thermal inkjet inks have water as a major component. In typical pen plotter inks, water makes up slightly more than half of the ink. Other cosolvents are present depending on the type of pen and the manufacturer. In thermal inkjet ink, water makes up 90% or more of the ink. The cosolvent, while a much smaller percentage of the total ink mixture, is much more aggressive. The challenge in developing an alkaline paper that is compatible with both pen plotter ink and thermal inkjet ink involves the basic components of the paper sheet.

The common plotter papers were and still are typically acid-process and were developed to meet the requirements of pen plotter inks. This means a moderately high level of internal sizing compared to plain papers such as office paper or copy papers. Surface control of the image components of the ink is important to maintain fine line quality. Pen plotters emulate manual drafting methods, using pen strokes to create the lines and structures. Therefore, controlled removal of the solvents (water and others) from the surface of the sheet is not as critical as it is for the DesignJet thermal inkjet plotter.

Alkaline papers use the same paper fibers as acid paper and the process is very similar, using the same paper machine equipment. Alkaline papers differ from acid papers in two major areas: sizing and fillers. Sizing imparts water resistance along with other properties to the paper either through internal sizing or surface sizing. Fillers are used to fill spaces between the paper fibers, to improve printability, and to replace the more expensive paper fibers. The filler of choice for alkaline papers is calcium carbonate, and for acid papers the filler typically used is clay.

Extensive testing of alkaline sizing agents indicated that their water resistivity is equal to or better than that of the acid sizing agents. Therefore, sizing did not appear to be the critical component. The next component of paper to evaluate was the calcium carbonate. Sample papers with varying amounts of calcium carbonate were evaluated with both pen plotter inks and thermal inkjet inks. The level of calcium carbonate in the paper sheet was inversely proportional to the print quality level of both pen plotter and thermal inkjet plotter output. Controlling the calcium carbonate level improved print quality but the improvement was not sufficient for Hewlett-Packard products, so sizing was revisited.

The last component considered was the surface sizing. Our paper manufacturer had been working concurrently on paper components to improve thermal inkjet performance on alkaline papers. The particular surface sizing component developed for other products was tested on our candidate paper, and resulted in improved control of the thermal inkjet ink on the surface of the paper. This provided the needed improvement in print quality to make the paper acceptable.

The final effort in developing the paper was to characterize the current media, understanding the requirements of pen plotters and thermal inkjet plotters, and then recreate the physical characteristics of the acid paper in the alkaline paper. Rigorous testing of the then-current acid paper was performed to develop the model to present to the paper manufacturer. Testing included all image and handling characteristics of the paper sheet. These included both media handling characteristics such as thickness, stiffness, surface smoothness, surface friction, tear strength, tensile strength, and moisture and image characteristics such as opacity, brightness, paper color, and porosity.

The model was presented to the paper manufacturer along with a plotter test bed. As a result of the early investigation and the investment in developing a detailed model of the sheet, the new alkaline opaque bond met all the print quality and schedule requirements for both pen plotters and the new DesignJet plotter. Print quality for the DesignJet was optimized and critical performance parameters such as media handling, color response, and sheet feeding for pen plotters were maintained or improved.

Vellum

The vellum presented a different challenge. Development-phase testing of the DesignJet plotter with the vellum resulted in marginal performance because of the characteristics of the new improved thermal inkjet ink. Thus a new vellum was necessary to meet the print quality goals. The DesignJet schedule required a successful vellum within six months.

(continued on page 13)

DesignJet Plotter User Interface Design: Learning the Hard Way about Human Interaction

How many times have you picked up a product and found it easy to use? If a product is easy to use, it probably was no small task to make it that way. With some products, even simple ones, I can be frustrated because I can't make it do what I expect it to do. Either I don't know how or it doesn't have the capability. The manual may be within reach but I have no interest in consulting it. With any product, users hope to combine their own intuition with external clues to determine the machine's capabilities and correct operation. What a confidence builder it is when a person can walk up to a new machine and operate it correctly!

We engineers, with our generally analytical minds, think that the simple solution is to publish a manual with step-by-step instructions. Although users expect such a manual, at least 50% will attempt operation without even opening it. Apparently users do not find it pleasant to build their mental model of a machine's operation by using only diagrams and text from written instructions.

Although the cause is noble, achieving intuitive operation is not so easy. A designer, familiar with each intricacy of the mechanism, will express an opinion and come up with the initial user interface design. Is this going to work for all users? I think not, but it is a place to start.

Our brain is parallel processing input from five senses and filtering it through past experiences. No amount of analytical thinking in a serial fashion can possibly predict the human response. The designer's experience with the development precludes any useful help in the user interface area. Perhaps, we might think, an experienced person could help us determine how people will react. Mistake! No one person can determine the best user interface that will appeal to the most people. This is a very difficult concept. When your manager tries out your design and has a certain difficulty, that can seem like the highest-priority problem, but it may just be a corner case. User testing is the key.

The DesignJet plotter started life as a gleam in the eyes of an architectural team. Prototypes were built, and along with proving the functionality came early user testing of some of the concepts the designers were concerned about. For example, to load roll media, the user needed to preselect roll format on the front panel, place the roll correctly on the spindle (it can go four ways), insert the media into the load slot, and lift the pinch roller release lever to align. Was the roll-to-spindle orientation intuitive? Would people lift the lever when instructed by the front panel?

The designer's opinion was that we needed to redesign for automatic lever lift; we should not be asking people to raise the lever. Mistake! No one person, especially the designer, can determine user reaction. A week's worth of investigation resulted in an unacceptable impact to cost and schedule and so the idea was reluctantly shelved. Later user testing showed that people had no problem or frustration when the display read "Lift lever." They simply lifted the only lever on the machine.

The same initial user testing showed that all users intuitively oriented the roll on the spindle backwards. This sent us into another redesign, this time for two months, resulting in a prototype that allowed the roll to be installed this way. Mistake! Users look for clues and parallel process all information. This new prototype displayed a new set of problems of both function and ease of use. Going back to the original design, a more subtle change was made. A graphic label installed under the roll cover was the clue that users were looking for. It was only in the absence of any other information that they showed the opposite preference for roll orientation.

With the initial concerns addressed by user testing, we thought we could continue with hard tooling. Mistake! Not all user interface problem areas can be predicted. The entire system must be tried by typical users not familiar with the product.

Testing after hard tooling revealed that people will not select sheet or roll mode before trying to load the media. Initially a button on the front panel toggled two light-emitting diodes that displayed the type of media the DesignJet plotter expected. From the previous problem, we had already learned that the DesignJet did not need to be redesigned; we simply had to provide more clues. Now, after the load is initiated, the plotter pauses and asks the user, by means of the front-panel display, to select which format of media has been loaded. This tested well and has the added benefit of providing the user with time to stop and decide (with hands off) if the plotter has a good, even grip on the media before continuing. Unfortunately in this case, the hard tooling needed to be modified as an unexpected expense.

At the same time, it was found that first-time users almost always created a paper jam while trying to load. This was, of course, unacceptable. Observation of user tests showed the various techniques people were using to load. The DesignJet plotter grabs the media after a set amount of time when the media passes a sensor in the insertion slot. People did not know that they needed to wait for this time and then let go of the media so that it could move into the mechanism. We needed to provide some clues. An audible click, which was part of the original concept, was not enough. Almost by accident, it was discovered that if we moved the media quickly at the start, users would instinctively let go.

Another problem was that some people were using a line on the side of the machine to adjust the media squareness. People complained that there needed to be a guide on the side to align the media. It wasn't until we thought about the comment that we realized this was how they expected to keep the media square. They were frustrated because the DesignJet would reject that load as misaligned. It was not possible to use the marks on the side to adjust the squareness. Our other plotters use a side surface to reference the media and these people were accustomed to that type of system. The correct DesignJet method is to push the media against the pinch rollers which are, unfortunately, out of sight. The lines on the side were only an approximate left/right location reference. Within 1/4 inch of the side line would have been fine. Our solution was to reduce the length of the line on the right so it didn't look like something to align to and to add a label in the area that explains in six different languages to "Push media against rear stops." A better solution might have been to make the pinch rollers visible to the user.

The position selected for the label is not very eye-catching but has the benefit of not cluttering the appearance of the machine. While looking at this solution, the general comment was, "No one will ever see it." Mistake! Don't ask people to predict how others will perceive. User testing of first-time users showed that they were unsure of how to load and were actively searching for clues. Almost all noticed and comprehended the label. A bail lift timing change bought us some extra design margin which, combined with the improved user interface, gave us a vastly improved and acceptable design.

The design might be improved further now that we have gained a clearer understanding of DesignJet user perceptions. But to give an accurate model of user perception, user testing requires a complete product, and a complete product is not receptive to many changes. This creates a minor dilemma neatly expressed as: "In every product's development, there comes a time when you must shoot the engineer and go into production." In my case the wound was not fatal and I'm recovering nicely.

P. Jeffrey Wield
Design Engineer
San Diego Technical Graphics Division

The manufacturing of a translucent 100% cotton vellum results in internal and surface characteristics different from those of bond paper. The vellum has an added resin component that fills in the voids between the fibers to impart the translucency to the sheet. This results in a sheet that has essentially no porosity compared with the relatively porous plotter bond. This further reduces the ability of the ink solvents to migrate into the sheet. In addition, some transparentizers, as the resins or oils are called, are not compatible with the thermal inkjet inks. The project required understanding the thermal inkjet ink chemistry interactions with various transparentizers and selecting the best transparentizer for the best print quality on pen and thermal inkjet plotters.

As with the bond paper, extensive testing of the existing vellum was undertaken to determine the desired physical characteristics. The data was used to develop a model for the vellum similar to that for the bond paper. The vellum manufacturer used this information to develop a vellum that not only provides excellent print quality on the thermal inkjet plotter but also has improved characteristics on pen plotters.

These products, along with the remainder of the media set, were exhaustively tested under a variety of environmental conditions on the DesignJet plotter to ensure Hewlett-Packard quality for our customers.

DesignJet Media Bin

The Designjet plotter provides "unattended plotting." This involves handling multiple output sheets with no user intervention. The user should be able to pick up a finished plot without the added inconvenience of unrolling and hand-cutting a plot, or picking it up from the floor. We needed to go beyond the usual catch tray, which could wrinkle or damage the plot. The DesignJet plotter provides an automatic one-axis cutter that will trim the plot to the right length after plotting from a roll. The trimmed sheets are then handled by an automatic sheet stacking system that won't damage the plots. This function is provided at a low cost increment.

System Requirements

To define the media stacker requirements, we conducted user surveys, surveys of the media distributors, and focus group studies. From the collected data, we created a typical user model and determined the following requirements for the media stacker:

- It should work with popular media from most manufacturers.
- It should work with media sizes from C size or equivalent up to E size or equivalent.
- It should work with all media types, including vellum, chart paper, translucent, and polyester film.
- It should work under a wide range of environmental conditions.
- It should not exceed the footprint of the machine because of space constraints in the office environment.
- It should have low incremental cost.
- It should stack up to 20 sheets without user intervention.

Because it would have been impractical to test all of the possible media permutations, we defined a representative media sample from the most popular manufacturers to be evaluated and tested.

Design Considerations

Cost, time, and available resources pointed toward a passive system. This implied that system performance would be heavily influenced by the behavior and properties of the media.

We identified the critical properties that would affect a passive system and then proceeded to characterize these properties at extremes of temperature and humidity because the physical properties of most media types change drastically under different environmental conditions. We also looked at the inherent properties of roll media, since rolls are the media format used during unattended plotting (cut-sheet mode requires the user to load and remove plots individually, unlike roll mode, in which plots are automatically cut and stacked).

Media Properties

Curl-Set. Curl-set is curvature induced in the media because of internal stresses as it is deformed to wrap around the core of the roll. This induced curvature is more pronounced closer to the core because it is proportional to the radius of the core and the tension used to wind the media around the core. We characterized curl-set as a function of sheet position within the roll, and we also measured curl-set relaxation as a function of time. Curl-set is affected by temperature and humidity, so we tested at extreme environmental conditions (see Fig. 9). A section of roll media with curl-set so pronounced that it tends to roll on itself once the sheet is cut is considered not stackable. We found that the stackable portion of the media roll varies considerable from roll to roll, and from media type to media type. It appears that many media vendors do not have good control over roll-winding tension, which affects curl-set directly, so different rolls vary from 100% to 60% in the percentage of their length that is stackable. The amount of curl-set relaxation over time is not enough to make a difference in media performance.

Friction. We measured the friction coefficient of the representative samples to try to determine the optimal slide angle for this type of stacking system. Fig. 10 shows curves of tray depth as a function of slide angle for 44-inch sheets. The optimum angle is where the depth is a minimum. This angle varies with the coefficient of friction.

Ink Dry Time. The ink dry time is the time required for the ink to dry once it is applied to the media. This is an important parameter because we needed to be sure that the ink will be dry by the time it touches the ramp of the media stacker or other sheets, so that the plot won't smear.

Stiffness. The rigidity of the sheet as it is feed from the plotter to the stacker affects the geometry required to guide the media properly. Moisture content affects stiffness to a considerable extent. Some media samples could collapse before being stacked because of reduced stiffness.

Geometry and Operation

To keep the media stacker bin within the limits of the plotter footprint, we devised a ramp that inverts the sheet direction under the machine.

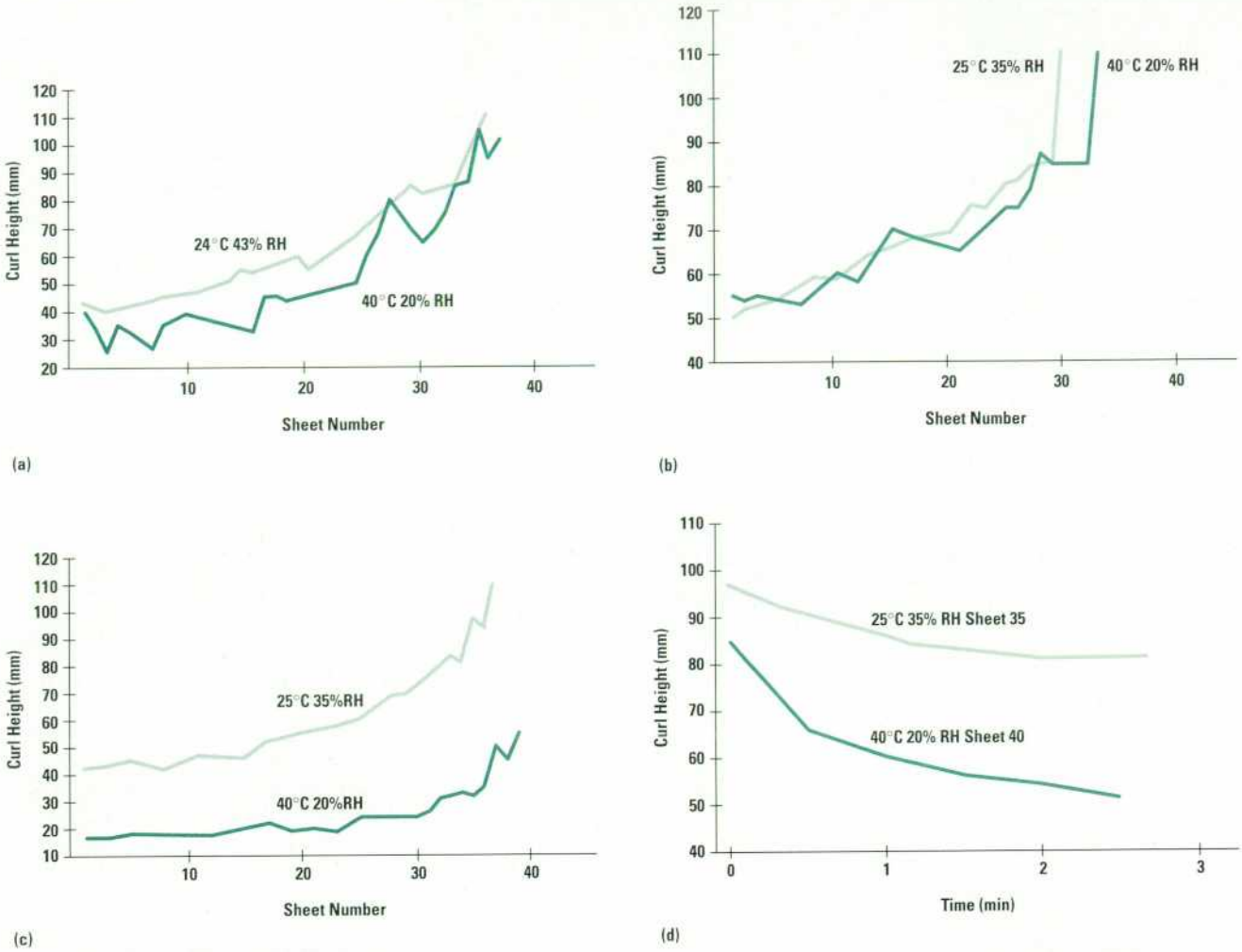


Fig. 9. Curl-set and curl-set relaxation measurements for various media types. Curl height as a function of sheet number for (a) HP Paper (b) JRG vellum V-20 (c) Clearprint 1000H vellum. (44-inch sheets. Sheet 1 is on the outside of the roll.) (d) Curl height as a function of time for Clearprint 1000M vellum.

The principle of the system is to guide the sheet using a ramp at a relatively steep angle into a stop. The sheet is then allowed to form a loop outside the ramp as it is fed, so that when the sheet is cut, the weight and position of the loop induce it to fold outward, so that half of the sheet ends up inside the bin and the other half hangs on the outside. Subsequent sheets follow the same path and are stacked on top of the previous sheets (see Fig. 11).

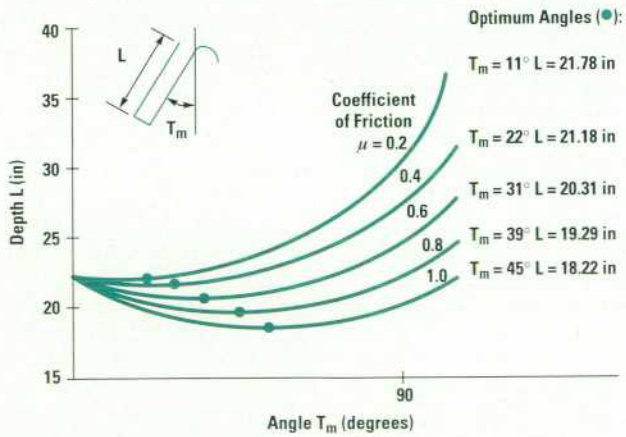


Fig. 10. Measurements of depth versus slide angle for the DesignJet media stacker.

To handle curled media from a roll, we devised a feature close to the bottom of the bin to serve as a stop for the curling media. If the media is considerably curled, the leading end of the sheet will tend to curl on itself. When the sheet is cut, it will collapse into a roll. The stop feature stops the leading edge before it curls on itself (see Fig. 12).

The parameters that have a first-order effect on stacking are bin depth, the position of the bin relative to the output gap, and inclination angle of the ramp. These parameters can be optimized for a particular media type and length of sheet. Unfortunately, there are large variations in media properties and dimensions. We compromised the geometry of the system so that it would function with the most popular combinations of media type and sheet length, and we provide a three-level depth adjustment to handle different sheet lengths.

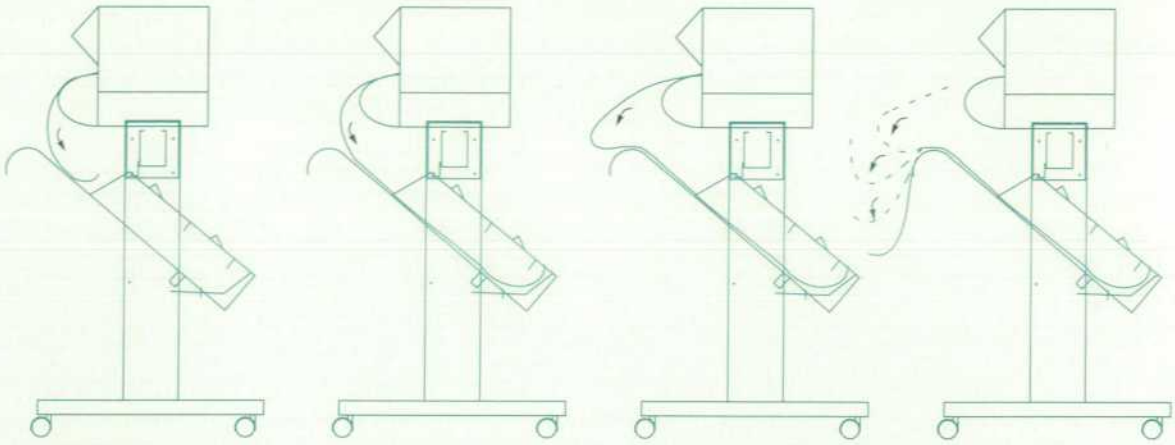


Fig. 11. Operation of the DesignJet media stacker.

Acknowledgments

The DesignJet development team was highly dedicated, professional, and motivated. The team members worked together well and respected the values that each area of expertise brought to the program. In addition to the people mentioned in other articles in this issue, the authors would like to thank the following for their notable contributions beyond the call of duty: Alecia Jay and Nancy Helff for the great manuals and user need studies, Dan Goese, Andy Tallion, and Steve Flack for the great marketing support they gave the program, Sharon Jensen, Joe Milkovits, Dan Caputo, Russ Bergen, Rob McCline, Rick Hermes, Katherin Ault, Scott England, Carey Enslow, Nancy Huelsmann, Adrian Huges, Nils Madden, Vinh Nguyen, Richard Szapacs, and Warren Werkmeiser for the excellent job they did in working with our suppliers and making sure that quality and delivery were priorities, Rod Degesero for the product packaging, which has proven itself capable of standing up to the worst transportation abuse, Irene Caravantes, Mike Duffy, Sandra Boldt, Vern Hudson, Cameron Light, Donna Ogilvie, Virginia Pollack, Jusitine Prehatney, Carey Ramos, Bob Stuart, and Christine St. Ives for the work they did in creating market visibility and market support for the product, Peter Morris, Bruce Mueller, and Allison Rap for getting our consumables in place, Rick Brown and Andy Tricario for the production support they gave during the design phases, and Stephen Glass, Tracie Middleton, Scott Bonnet, Tom Barker, and Scott Roleson for the excellent role they played in making sure the product met worldwide agency approvals and in ensuring that the product meets customer expectations. Glenn Gaarder, Don Hiler, Lynn Palmer, Jeff Stong, Darren Wilcox, and David Walker are additional mechanical engineers who contributed to the program greatly.

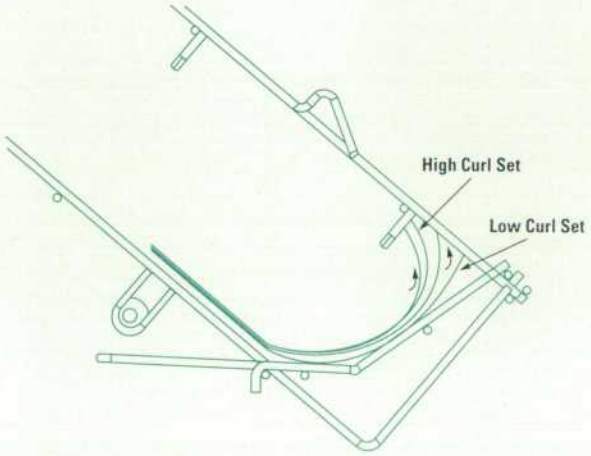


Fig. 12. Curl stop functioning.

Electronic and Firmware Design of the HP DesignJet Drafting Plotter

High-performance vector-to-raster conversion and print engine control are provided by a RISC processor, two single-chip processors, and three custom integrated circuits. Development of the electronics and firmware made extensive use of emulation and simulation.

by Alfred Holt Mebane IV, James R. Schmedake, Iue-Shuenn Chen, and Anne P. Kadonaga

The HP DesignJet raster inkjet plotter project required contributions in the design of vector-to-raster conversion and print engine electronics. The project was constrained by cost and schedule, but the performance of the vector-to-raster converter and the inkjet print engine was considered of prime importance in meeting our user's needs. Our traditional approach to plotter electronics, while meeting cost and schedule goals, would have fallen short of the required performance goals. Existing electrostatic vector-to-raster converters, while meeting performance goals, were too expensive for our market. The approach we took was a fresh look at the requirements of both the vector-to-raster converter and the print engine.

Vector-to-Raster Converter

Two major tasks are performed by the vector-to-raster converter. The first is HP-GL/2 language parsing. The second is conversion of the parsed objects into the dot patterns printed on the page. In the DesignJet plotter, these two tasks are serial and do not occur simultaneously.

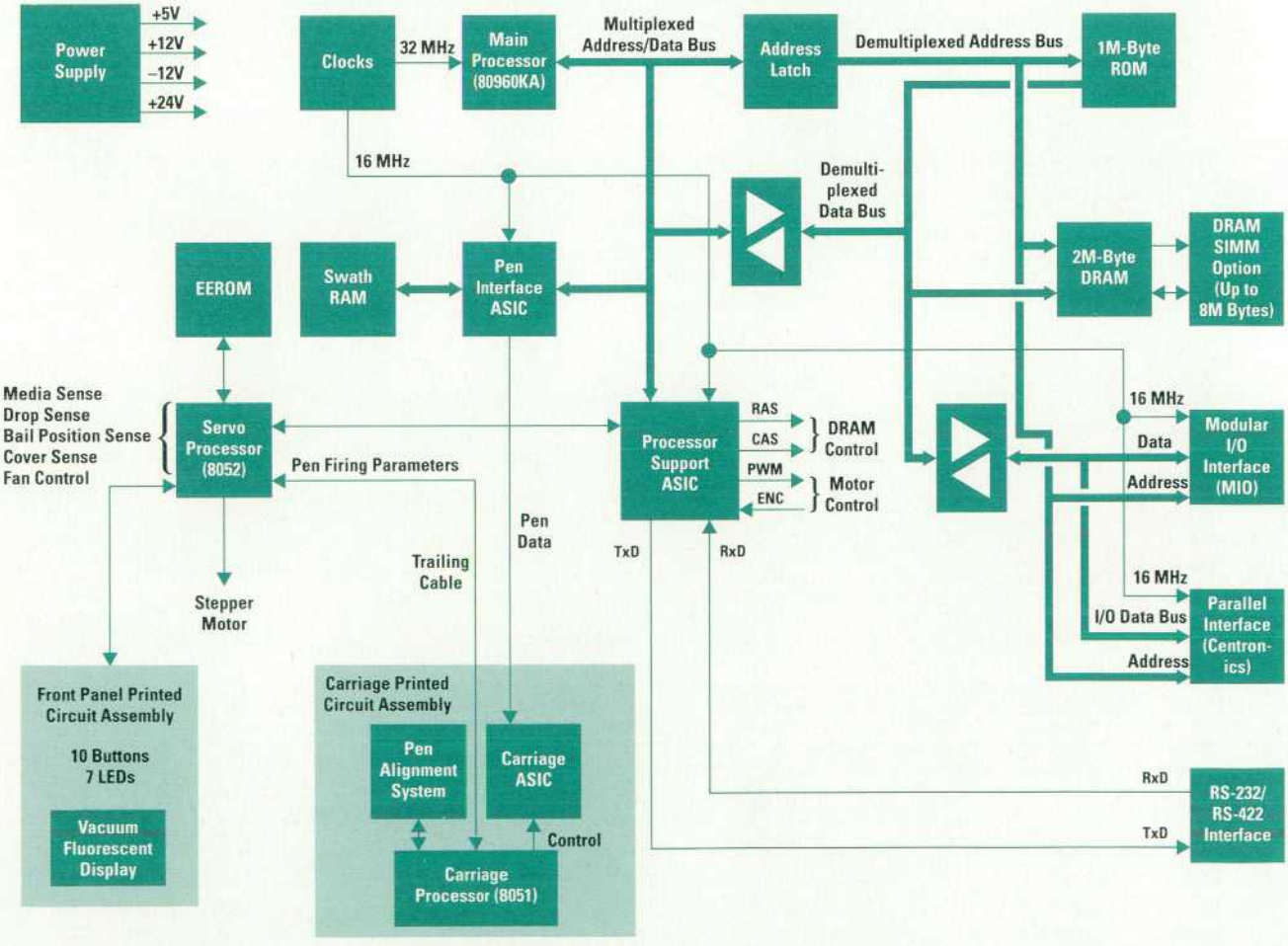


Fig. 1. Electronic block diagram of the HP DesignJet large-format inkjet drafting plotter.

An early decision was made to include the vector-to-raster converter in the base machine electronics even though several low-cost competitive raster products omit this feature. The design team felt that a vector-to-raster converter could be integrated together with the print engine control logic at a reasonably low incremental cost. The major constraint placed upon this implementation was that it be of high enough performance to outpace the print engine in all but the most complex plots. It was acceptable to slow the print engine during very complex plots, but only for the affected carriage scans.

Initial investigations centered on the use of specialized graphics processors. These processors were more than adequate for the rasterization of lines, but were far too slow when parsing HP-GL/2. Next, an approach was investigated in which a general-purpose processor performed the parsing and print engine control while a graphics processor did only the raster conversion. The fundamental weakness of this approach is the cost of using two processors even though only one is required at any point in time.

The final design is based on a single, high-performance, RISC processor for both print engine control and the two vector-to-raster converter functions. The selection criteria were cost, a performance benchmark based on an existing HP-GL/2 parser, and an engineering estimation of potential raster conversion performance. The Intel 80960KA was selected from the available choices, which included both RISC and CISC microprocessors. Although they were not part of the original criteria, two other features of the 80960KA made it attractive in the DesignJet application. The first of these is the multiplexed address and data bus, which reduced the pin count on the application-specific integrated circuits (ASICs) that were being designed for the product (see "DesignJet ASIC Development" on page 18). The second is the existence of the 80960KB processor, which includes hardware floating-point capability. If at any time during the project the need for faster floating-point performance had arisen, the -KA version could have been replaced by the -KB version without any circuit board changes.

The remainder of the processor portion of the electronics consists of 2M bytes of DRAM, 1M byte of ROM, I/O, and DRAM memory expansion. A processor support ASIC in this section provides a DRAM controller, wait state timing for both RAM and ROM, interrupt control, and 80960 reset synchronization. This ASIC is described later in this article.

Fig. 1 is the electronic block diagram of the HP DesignJet plotter.

Input/Output

The DesignJet plotter was originally defined as a plotter solution for pen plotter users who require greater throughput on monochrome plots. This definition suggested incorporation of the three built-in I/O ports—RS-232-C, HP-IB (IEEE 488, IEC 625), and printer parallel—that are included in existing HP plotters. The DesignJet R&D team decided to pursue a more flexible approach. A strategy that had been adopted for the LaserJet IIISi printer was investigated in which there is no built-in I/O but instead a standardized, modular I/O slot (MIO) capable of accepting many different types of I/O options. After reviews with manufacturing and service representatives it was decided to keep the RS-232 and parallel interfaces built-in to allow the DesignJet plotter to leverage existing test and repair systems. The built-in HP-IB port has been replaced with an MIO slot to allow the DesignJet plotter to use some of the I/O options developed for the LaserJet IIISi. At introduction both the HP-IB and Novell Ethernet were available options.

Memory Expansion

One of the more difficult specifications to set for the DesignJet plotter was DRAM memory size. In a pen plotter, each vector is strobed out as it is received, so plotting can begin as soon as the first vector is parsed. The DesignJet plotter must parse and store all the incoming vectors before making the first carriage sweep. It is an inconvenient feature of vector languages that the last vector received may cross the portion of the page covered by the first carriage scan. This leads to a limitation on plot complexity based on the memory size available to store parsed vectors. The trade-off is the cost of memory versus the complexity of the possible plotted images. Hewlett-Packard electrostatic plotters solve this storage problem by means of magnetic disk storage. The drawbacks of a DesignJet implementation of a similar solution were cost and the difficulty of providing a mechanical mounting to allow a disk drive to survive the shipping and operating environments expected for the DesignJet plotter. The chosen solution is industry-standard, 72-pin single inline memory modules (SIMMs). This memory is added in the form of 1M-byte or 4M-byte SIMMs, which are available from HP for a variety of products in the personal computer and peripheral areas. Two sockets are available under a panel in the rear of the plotter. Addition of two 4M-byte SIMMs gives a user an additional 8M bytes of vector storage for complex plots. The use of SIMMs for memory expansion adds very little cost for users with needs fitting into the standard 2M-byte RAM and provides a relatively low-cost upgrade path for users requiring more.

Print Engine Control

Print engine control is a much less demanding task than vector-to-raster conversion. Print engine control includes two-axis servo motor control, front-panel control, keyboard scanning, front-panel display update, optical sensor scanning, and thermal inkjet pen service station control.

The servo motor control functions of position decoding and pulse width modulation of the motor voltage were added to the processor support ASIC. The other functions required mostly pins with very little logic, so alternatives were investigated that could implement them more efficiently. The result of this investigation is an approach that surprised most of the design team. A single-chip processor, the Intel 8052, was able to perform these functions with a lower production cost than an ASIC and for a fraction of the development cost. It also performs all of the real-time servo control, offloading this from the 80960KA.

The 8052 is designed into the architecture as a slave processor to the 80960KA. A bidirectional command and mailbox port is implemented in the processor support ASIC to allow processor-to-processor communication. Through this port the 8052 is able to return data to the 80960KA in response to commands or to generate one of several interrupts. Among these interrupts is the operating system time slice interrupt.

A vast portion of print engine control is the electronics required to support the thermal inkjet pens. The most difficult function performed in this area is the mapping from the image generated in memory by the vector-to-raster converter to the series of timing pulses sent to fire the pens. This task is made difficult by the fact that the DesignJet plotter uses two 50-nozzle pens that are not accurately aligned mechanically with each other. The mapping and alignment compensation are performed by two ASICs, one located on the main board and a companion part located on the circuit board that travels on the pen carriage. The two ASICs are connected by a serial link that runs through the trailing cable. The pen interface ASIC on the main board is initialized with the measured distances between the two pens and is able to select from image memory all the dot positions that are covered by pen nozzles at a given carriage position. As the carriage scans across the page, this ASIC sends groups of 100 bits up the serial link to the carriage ASIC at 1/300-inch intervals. The carriage ASIC buffers the 100 bits and creates the timing patterns used as inputs to the drivers that generate the firing waveforms for the thermal inkjet pens.

Pen Calibration

Offsets between the two pens are measured by another set of electronics located on the carriage board (see article, page 24). DesignJet pens are aligned by drawing a series of patterns on a page and then using an optical sensor to measure the relationship of patterns drawn with one pen to the patterns drawn by the other. These offsets are written to the thermal inkjet support ASICs as described in the paragraph above. The electronic components of the optical system are controlled by an 8051 microprocessor located on the carriage board. The decision to use the 8051 on the carriage is based on the same criteria used to select the 8052 for the main board—the 8051 can perform the sensor control functions at a lower part cost and a much lower development cost than a special-purpose ASIC. An added feature gained by using the 8051 is that it provides the serial communication path to the 8052 on the main board. This communication path is used both for the optical system and for sending pen firing constants to the ASIC on the carriage.

DesignJet ASIC Development

Often the custom IC design is in the critical path of electronic systems development. This was the case for the HP DesignJet plotter project. Three ASICs were needed to provide the necessary functionality and performance in the DesignJet plotter at a low cost.

The main challenges were clear right from the beginning. The team had to deliver three working ASICs on schedule. A turnaround in any of the ASICs would have meant a serious schedule slip, since the fully functional ASICs were needed to start much of the system-level testing. The team also had to provide the functionality of these ASICs before the first silicon became available to enable parallel development of the printer mechanisms and firmware. These needs had to be met with a limited number of engineers and a given budget to avoid adversely affecting other projects.

Processor Support ASIC

The processor support ASIC interfaces to both the main processor (80960KA) and the servo processor (8052) and performs various assist functions for each processor. The block diagram of the IC is shown in Fig. 2. The main processor side of the processor support ASIC consists of the 80960KA interface, the DRAM controller, the serial I/O interface, and the fire pulse controller. On the servo processor side, the IC contains the 8052 interface and the motion controller. The processor support ASIC also provides two modes of communication between the processors: a polled, bidirectional mailbox and an interrupt-driven, unidirectional block transfer buffer.

The 80960KA interface assists the processor during burst-mode ROM and DRAM accesses and handles the queueing and prioritizing of the incoming interrupts. The DRAM controller supports up to 20M bytes of memory of different sizes

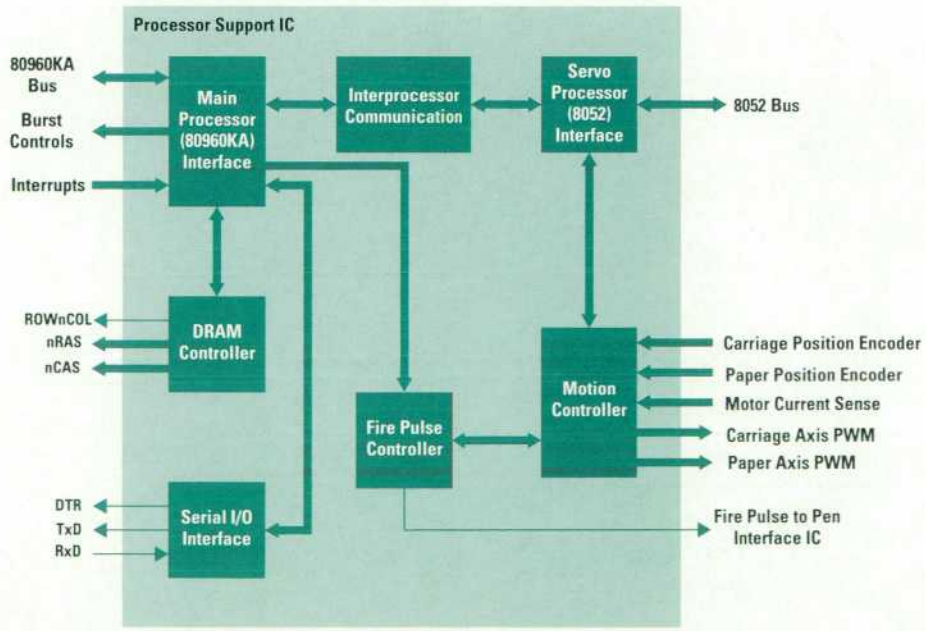


Fig. 2. Processor support ASIC block diagram.

and access times. The serial I/O interface consists of a baud rate generator and a UART (universal asynchronous receiver/transmitter). The fire pulse controller generates a synchronizing pulse for each column of pen data. The output is extrapolated from the carriage position encoder counts at one, two, or four times the frequency.

The motion controller decodes the position information for the carriage and paper axes. It also generates the PWM (pulse width modulation) signals for the dc motor drivers as needed for servo control of the carriage and paper axes. A watchdog timer monitors the servo loop and disables the PWM outputs in the event of a servo processor malfunction. A status register is also provided to log various motion control error conditions.

Pen Interface ASIC

A shuffler stage is needed to map the row-oriented image data into the column-oriented DeskJet pen nozzle data. In a departure from previous shuffler designs, the pen interface ASIC uses a dedicated external memory array to store its own copy of the image data and a programmable internal sequencer array to hold the shuffle pattern. The shuffling is done by copying an entire swath of image data from the system memory into its local memory and fetching the pixel data for each pen fire sequence according to the preloaded shuffle pattern. This approach offloads these tasks from the main processor bus and allows greater flexibility for supporting different pen nozzle configurations.

The pen interface ASIC contains three bus interfaces: the main processor (80960KA) bus, the swath memory interface, and the serial link to the carriage ASIC. The block diagram of the pen interface ASIC is shown in Fig. 3. In the copy mode, the data and address path from the main processor interface to the swath memory is enabled. The pixel counter counts the number of pixels to be printed, and its output is used to calculate the plot density and ink use. In the shuffle mode, the path from the swath memory to the serial interface is enabled. The serial data transmitter assembles the pixel data fetched from the swath memory for each fire sequence into a serial bit stream and sends it to the carriage ASIC. The pen interface ASIC and the carriage ASIC contain identical pixel checksum counters to check the integrity of the serial transmission.

The pixel address generator is the heart of the pen interface ASIC. It consists of an SRAM array for the programmable sequencer, a logical column counter, and an adder. The sequencer is preloaded with the pixel address offsets for each pen nozzle of a fire sequence. The offsets contain the column adjustment delays for pen alignment correction (see article, page 24). A mask pattern is tagged onto each entry to aid different print modes. The column counter is either incremented or decremented depending on the print direction, and its content is added to the pixel address offset from the sequencer to generate the physical DRAM address of each pixel's data.

Carriage ASIC

The carriage ASIC resides on the printed circuit board that is mounted on the moving pen carriage assembly. Its primary function is to generate the data and address signals for the pen driver ICs. The relative timing and the pulse widths of these signals are carefully controlled to adjust the pen-to-pen offsets, the bidirectional offsets, the pen-to-paper-axis deviation errors, and the pen turn-on energy variations.

The carriage ASIC contains three bus interfaces: the carriage processor (8052) bus, the serial link from the pen interface ASIC, and the pen driver IC interface. The block diagram of the IC is shown in Fig. 4. The processor bus is used to access the timing control registers. The serial data from the pen interface ASIC is shifted into a serial-to-parallel converter. A checksum counter monitors the serial input data to verify the integrity of the serial link. A parallel pipeline register follows the shift register to provide the double column buffering. The buffer outputs are divided into four delayed pipeline registers, each of which is delayed by the value in its corresponding delay time register. Each of the four data

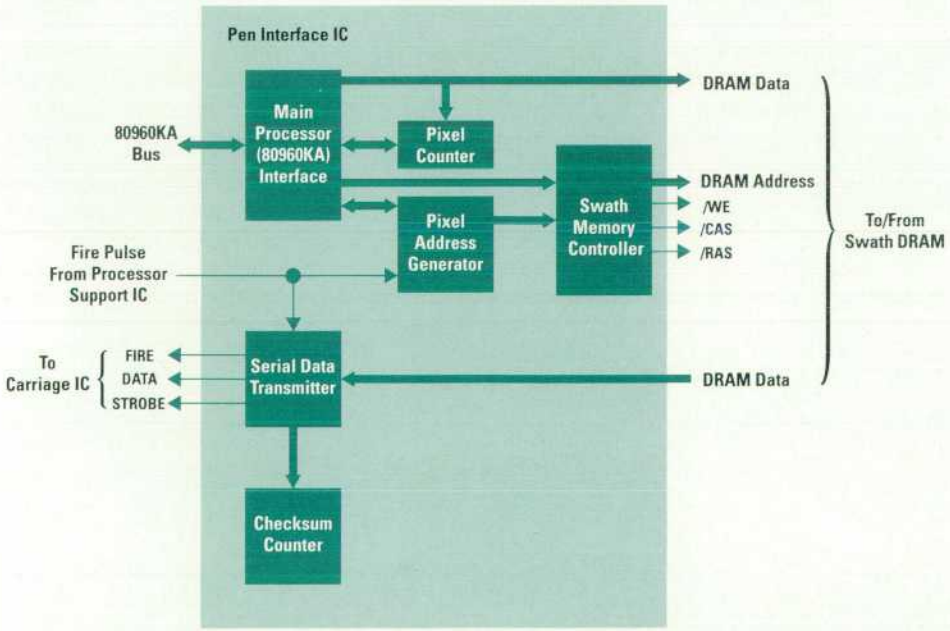


Fig. 3. Pen interface ASIC block diagram.

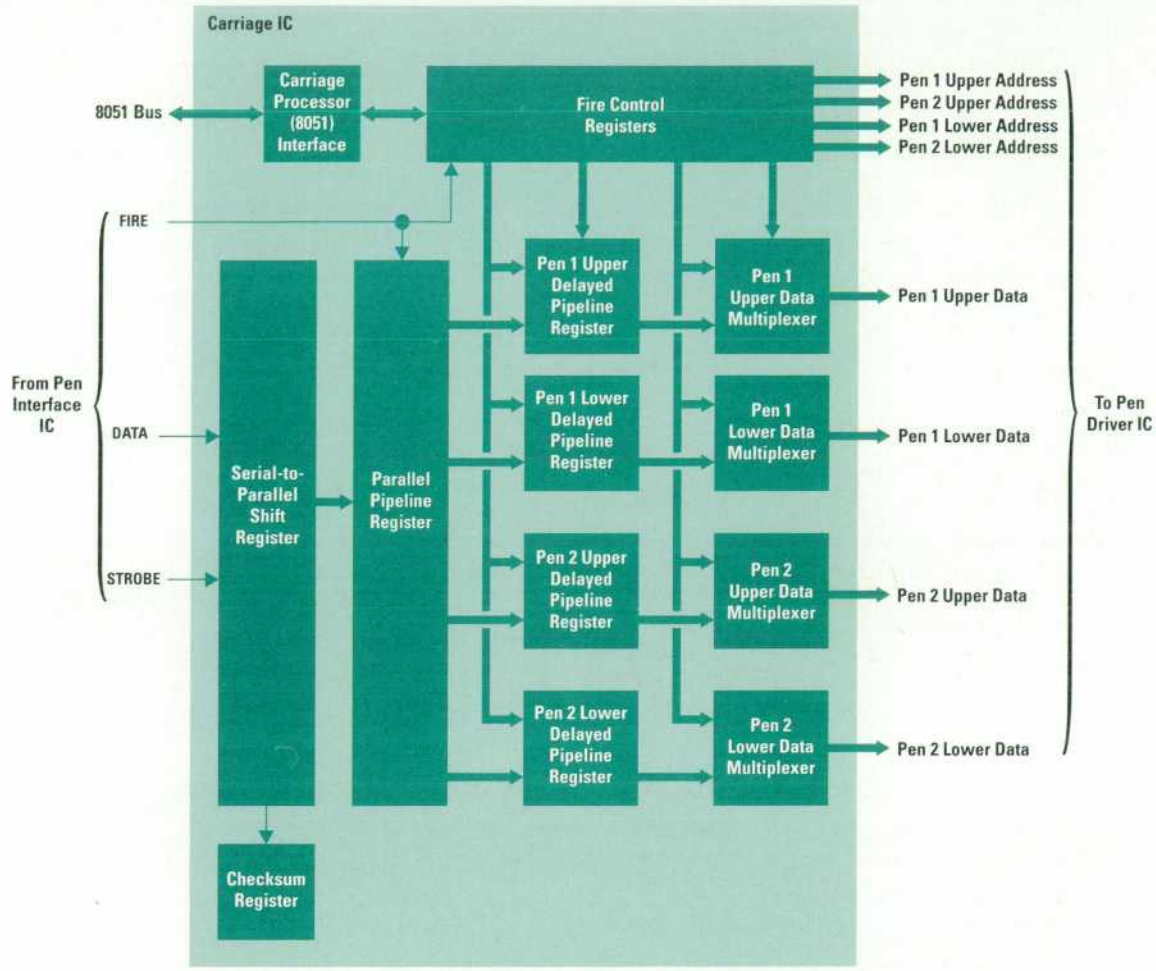


Fig. 4. Carriage ASIC block diagram.

multiplexers selects the data to be driven for one half of each pen.

Design Approach and Tools

Behavioral Simulation. The shuffler algorithm implemented in the pen interface ASIC and the carriage ASIC is a completely new design for any HP product. The shuffle path from the image data in the system memory to the pen outputs includes many hardware blocks and spans several different bus interfaces. The need to verify the correctness of the shuffler algorithm at a high level before any hardware design was apparent. A behavioral model of the algorithm was written in the C programming language. A graphics driver was added for both the input image data and the pen outputs. A complete graphical animation showed both the image data and the printed data as the pen swept across the screen. Any error in the shuffler algorithm was observable right on the display screen. This high-level verification approach turned out to be very valuable. The shuffler algorithm was completely debugged during the simulation phase before any hardware was designed.

Emulator Strategy. It was decided to build emulators for the three ASICs despite the substantial amount of additional resources required. As mentioned earlier, the functionalities of the ICs were needed before first silicon to enable parallel development of the printer mechanisms and the firmware. The emulators were able to meet this need. If there had been a major bug in the first silicon, the emulators could have continued to provide the necessary hardware platform at least for firmware development if not for further mechanical integration and system testing. Another important consideration was that in the absence of a system-level simulator, the emulators combined with the rest of the electronics supplemented the chip-level simulation in the overall functional verification effort. The emulators also proved very useful in isolating and diagnosing anomalies encountered while integrating the first silicon into the system.

The emulator for each IC presented a unique set of design requirements. The major problem for the processor support ASIC was the schedule because of its late start and complexity. To keep pace with the other two ASICs, much of the emulator and IC design was done in parallel. The emulators for both the processor support ASIC and the pen interface ASIC required fast, hence low-density, PAL (programmable array logic) parts to meet the 16-MHz clock frequency requirement. The processor support ASIC emulator consisted of fast PALs, standard logic parts, and a UART. The pen interface ASIC emulator used fast PALs for most of the logic and a high-speed SRAM for the pixel address sequencer. The swath DRAM parts, which reside outside of the IC, were also included in the emulator to avoid electrical problems related to board interconnects. The requirements of the carriage ASIC emulator were a large number of registers and a small board area. The slower clock frequency of 12 MHz

allowed the use of high-density field-programmable gate arrays, which offered a large number of flip-flops per part.

Vendor Selection. After a preliminary screening of many ASIC suppliers, several vendors of both standard cell and gate array parts were closely evaluated. In addition to nonrecurring engineering charges, cost per part, and prototype lead time, also considered were the availability and cost of the hardware and software toolset, the quality of the technical support during development, and expected responsiveness after release to production. We decided that all three ASICs would use the same vendor and toolset for both technical and logistical reasons. We chose the CMOS34 standard-cell technology of the HP Circuit Technology Group, which offered the LogicArchitect ASIC development toolset, some of which is described below in more detail. The vendor also delivered the UART megacell for the processor support ASIC and the SRAM array for the pen interface ASIC.

Logic Synthesis. A logic synthesis tool was used extensively for transforming the control logic into the standard cells. The automation of this laborious task not only saved much time during the initial mapping of the logic into a combination of gates but also reduced the number of iterations through the compose-verify-correct loop. The synthesis tool was also used to generate the custom registers, decoders, and multiplexers. This approach was preferred over the use of TTL macro cells* because it made it possible to implement only the necessary functions using minimum-area cells. TTL macro cells were used for such blocks as counters, where they could be modified to provide only the necessary functions with the minimum-area cells.

The synthesis tool also offered area-versus-delay optimization capability, but this was of limited use for most blocks because the CMOS34 process is fast relative to the clock frequencies of 16 MHz and 12 MHz. In many cases, setting the area constraint at minimum sufficed. The exceptions were in control blocks where both the rising and falling edges of the clock were used, effectively doubling the frequency, and some critical timing path blocks. For these blocks, the delay constraint was set as needed with some margin. The worst path delay estimates included in the report files were useful in quickly verifying the timing margins at the block level.

Timing Analysis. Unlike a functional simulator, which requires a set of test vectors to exercise the circuit to verify the delay timing, a static timing analyzer calculates the delays through the specified paths based on the circuit structure alone. The static timing analyzer of the LogicArchitect toolset complemented the functional simulator by rapidly searching through the circuit delay paths before any test vectors were written. It was used to identify any unexpected critical timing paths and to verify the margins in the known critical timing paths. This tool was especially useful in determining the margins in input setup times and output delay times with respect to the clock edges.

Functional Simulation. As expected, the functional simulation took a major chunk of the total IC design time. A test vector generation interface in the LogicArchitect toolset saved a great deal of time by allowing the test vectors to be assembled from the subroutine functions and macros. Distributing the simulation jobs among various workstations also helped. Although the blocks were simulated at all levels in a bottoms-up order, the emphasis was different at each level. For example, once a comparator was verified at a low level with a semiexhaustive set of input combinations, it was not subjected to another input combinations test at a higher level. Instead, more time was spent exercising its interaction with the control logic. At the top level, special care was taken to set the correct input and output timing values and to choose the right types of load and delay models. A full set of top-level test vectors was repeated with the capacitance values extracted from the layout, and a special check of paths with potential skews was done.

Design for Testability. Three types of test support circuitry are designed into the DesignJet ASICs: scan paths for the automatic generation of the stuck-at fault test vectors, a boundary scan path in the carriage ASIC, and pad tristate control for the board testers. The HP CMOS34 standard cell methodology offers automatic test generation capability for stuck-at faults. The estimated hardware overhead for providing the necessary scan paths was about 10%. The designers of all three ASICs decided to support automatic test generation to achieve the highest possible fault coverage with the minimum of test vector generation effort. The scan paths were created by replacing each flip-flop with a scannable type and providing scan controls. Special controls were designed for the flip-flops clocked on the opposite edge of the clock, for bidirectional buses, and for asynchronous signals. A boundary scan path is implemented in the carriage ASIC to drive the output pads directly, bypassing the complicated internal configuration. The tristate control is provided for all pad drivers on each ASIC to allow the board tester to drive the IC pins directly.

ASIC Results

The DesignJet ASIC development team delivered the three fully functional ASICs on schedule. All three ASICs were released to manufacturing without any design changes. The processor support ASIC, packaged in an 84-pin PLCC (plastic leadless chip carrier), has the equivalent of 11,000 gates and a die area of 4.96 by 5.90 mm. (One equivalent gate represents four transistors needed to implement a two-input NAND gate.) The pen interface ASIC, also packaged in an 84-pin PLCC, came in at 4,000 gates and a die area of 4.87 by 5.56 mm, including the SRAM array. The carriage ASIC, packaged in a 68-pin PLCC, has a final gate count of 10,000 and a die area of 4.92 by 5.24 mm.

Each ASIC was successfully integrated into the system within a day of the arrival of the first silicon. This was possible partly because the system was already debugged using the ASIC emulators. With further code development later, a few corner-case problems that required minor firmware workarounds were uncovered. No other functional or electrical problems were found. This proved that our design approaches and the toolset were basically sound, but needed improvement in testing the corner cases. Even with the IC emulators, the corner-case testing was difficult because the code development continued well past the IC tape release. A collaborative effort by the ASIC and firmware

* Most ASIC vendors offer macro cells for widely used off-the-shelf logic parts such as TTL parts. A macro cell is a collection of library standard cells and interconnections that performs the same function as the corresponding off-the-shelf logic part.

designers involving their respective toolsets would help develop a better functional verification strategy. The design for testability allowed quick destaffing of the ASIC design team because no additional fault coverage test vectors were needed after the tape release.

DesignJet Firmware Development

The use of the Intel 80960KA processor in the DesignJet plotter created several development challenges for the firmware design team. First, hardware to exercise the code would not be available for several months after coding had begun. Second, the 80960 was a new processor to our division, so we needed to start almost from scratch on our development system. Third, we had extremely limited personnel resources and time to perform the necessary tasks. In the past, a main source of schedule investment was in the integration of independently developed code sections and the correction of timing problems between these independent sections of code. In addition, we generally spent much time tuning (if not totally redesigning) the user interface of the plotter once the code was integrated into the target. For the DesignJet project, we needed to minimize this largely nonproductive use of time. The solution was to build a development environment that was largely independent of the hardware, to leverage code and algorithm designs from outside groups, and to debug our code at the source level on both the target and host (simulator) systems. To ease integration and help with timing-related bugs, we chose to use an internally developed full-featured operating system.

Development Environment

The DesignJet development environment was different from any environment our division had used in the past. On past projects, almost all firmware engineers used a target-based emulator for all development work. Hardware was generally reused from past projects to run the emulators in an environment that closely matched the new system. Much of our past work had been done in assembly language. Therefore, the use of emulators was really a method of "source-level" debugging. On the DesignJet plotter project, our lack of hardware limited the use of emulators. In addition, there were no emulators available that interfaced with our HP 9000 Series 300 workstations, which were used for compiling and linking. These restrictions forced us to reconsider our past development methods. We decided that our code should run, as much as possible, on both our target system and our workstations. Since our coding was to be almost 100% in the C language, we maintained object modules compiled both for the target and workstation, or host, systems. Only where there were differences between the two systems was there any additional code to support the host system.

The main difference between the host and target systems was in the I/O and print engine subsystems. The I/O subsystem was easily modified to accommodate a host-based development system. The main input path was set up such that when running on the host, input was done from the host file system rather than a Centronics or RS-232 port. None of the code outside of one function "knew" where input was originating. The print engine that ran on the target was almost identical to the print engine on the host. The main difference was where to send data to print. On the target, hardware was the destination of completed bands, or swaths, of print engine data. On the host, a simulator was used. We obtained an X Window-based simulator that we were able to modify for our use on the DesignJet plotter. The simulator was passed an array of bitmap swaths, and when so instructed, opened an X Window on the workstation to display the data. The simulator had the ability to highlight individual swaths, scale the bitmap, and zoom in on specified areas of the bitmap. This allowed inspection of plotter output at the individual dot level, something that would have been next to impossible to do accurately on the target system. The simulator allowed all development of nontarget-specific code to be done on the host, especially debugging using `cdb` and `xdb`.

Large-format plotters require much interaction with the user, particularly in the areas of front-panel control and media loading. The front panel was developed with the aid of a front-panel simulator that ran on a PC. This important area of the user interface was then designed independently of the hardware and firmware of the plotter. The front-panel simulator was designed such that nonfirmware personnel, such as marketing and user interface experts, could design and fine-tune their own front panel. This freed a firmware engineer to work on other tasks.

The media-loading interaction was also subject to a great deal of modification. The basic loading algorithms were developed by mechanical engineers on simple breadboard mechanics. These generic motor controllers allow nonfirmware engineers to control motors precisely with simple HP-GL-like instructions from any computer. Typically, a mechanical engineer wrote BASIC programs on an HP 9000 Series 200 computer to develop media-loading sequences and algorithms. The generic motor controllers have general-purpose inputs and outputs, thus allowing the development engineers access to sensors and actuators as desired. Offloading these types of development activities allowed the firmware engineers to focus on software engineering problems. When algorithms were basically complete, the mechanical engineers documented their work graphically, allowing the firmware engineer to translate the algorithm into the DesignJet framework.

Code Reuse

A major portion of any printer or plotter is the language subsystem, in our case HP-GL/2. On the DesignJet plotter, we clearly did not have the resources to reinvent this wheel. We were able to use a language subsystem developed at our division that had been previously used on several products, including the LaserJet III printer. This proved to be extremely beneficial in several respects. Certainly, the time spent integrating this code was much less than would have been necessary to rewrite it. The most beneficial aspect, however, was that far less time was needed in testing because the code had already been thoroughly tested and debugged in previous products. This gave us the high-quality language subsystem that we wanted with a minimum of investment in time from our product team. A second area that was heavily leveraged was the vector-to-raster converter. The basic high-performance design that had been used in other raster products from HP was leveraged for the DesignJet plotter. While the original vector-to-raster converter was written in assembly language, we chose to rewrite the code in C. The previous products had used a different processor,

so rewriting the code was necessary anyway. Using C, reuse in future products would be far easier no matter what processor might be chosen.

Debugger

The lack of emulators on the target system required the use of a new tool for our division: the retargetable remote debugger. We found that the GNU debugger, `gdb960`, worked very well on our system. `gdb960` was developed by both The Free Software Foundation and Intel for use on the 80960. This source-level debugger uses a simple monitor on the target system while the main debugger runs on the host. Communication for debugging is via RS-232. We modified the monitor code so that we could download executable modules to the target system via our built-in Centronics interface. The monitor program, `NINDY`, was supplied by Intel and modified by us for our hardware. The modifications were minimal and were only in those areas of the code that dealt with serial I/O and downloading. The download capability was modified because the default method, RS-232, was too slow. With Centronics downloading, we were able to cut download times from about 10 minutes to about 10 seconds.

Using `gdb960`, we were able to resolve problems that appeared only on the target very quickly. The debugger functions much the same as the debugger `cdb` of the HP-UX* operating system. With our remote-reset and login capability, we were able to debug target-executing code from home on those nights when long hours were called for. The debugger is completely source-level; that is, breakpoints are set on C source lines, not particular addresses. Structures and arrays print as such; there is no need to examine memory addresses and reconstruct the data types of interest. If a bug manifested itself as a processor fault, the debugger showed us the entire stack frame and allowed us to move about within stack frames. This enabled us to find the real source of a problem, which was often several stack frames up from the fault itself. In short, debugging on the target was generally no more trouble than debugging on the host.

Operating System

At the outset of our design cycle, we determined that use of a formal operating system would be beneficial in two ways. First, it would provide a stable interface for interprocess communication. This would ease the process of integrating code that had been developed independently by several firmware engineers. Second, since the operating system we used was a preemptive operating system, timing problems typical of custom operating systems would be greatly reduced. Rather than invest in methods and hardware to debug complex timing problems, we decided to invest in an operating system that would prevent these problems. While sounding overly simplistic, this logic turned out to be quite accurate; we had very few timing-related problems. In addition, when hardware became available, code integration went very smoothly. There were very few problems with interface specifications changing because the operating system defined the interface.

The operating system was developed independently using a PC-based development board supplied by Intel. When hardware became available, the operating system was fully debugged and ready to install. The operating system is written in assembly language, translated from code that runs on the HP 9000 Series 300 workstations. This was another major subsystem for which we leveraged the design. The benefit of the operating system running on both the host and the target is obvious; the underlying code has no knowledge of its operating environment.

Firmware Summary

The development process of the DesignJet firmware was unlike any other project at the San Diego Division. Code was developed in a largely hardware-independent fashion. We strove to eliminate system timing problems through the use of a formal operating system. This had the additional benefit of defining a strict, stable interface between processes. The user interfaces of the plotter were developed by experts separate from the firmware design team. Debugging was accomplished using `cdb` on the host and `gdb960` on the target. These debuggers provided source-level interfaces that greatly enhanced the engineers' productivity. The language subsystem consisted of a reusable code base, and the vector-to-raster converter subsystem we produced will be reused in future products. We believe that our development methodology was a key factor in meeting our project's goals.

Conclusion

The DesignJet plotter electronics provide a high-performance raster plotting system at a very aggressive price. The distribution and selection of ASICs and processors enabled us to design a robust, flexible, and cost-effective system. This design not only meets current customer needs, but contains the features necessary for leverage into future raster products.

Acknowledgments

We would like to acknowledge contributions by other members of the electronics and firmware teams: Curt Behrend, Craig Bosworth, Jack Cassidy, Keith Cobbs, George Corrigan, David Ellement, Diane Fisher, Milt Fisher, Dan Johnson, Tom Halpenny, Bob Haselby, Janet Mebane, Kent Takasuki, Teri Tracey and Irene Williams. In particular, we would like to thank the DesignJet firmware and language reuse group for being open to new development methodologies. We would also like to acknowledge our electronics project manager, Larry Hennessee, for providing the creative and productive environment, and our firmware manager, Jennie Hollis, for allowing us the freedom the explore new ideas. Our thanks also go to our ASIC partners at the HP Circuit Technology Group: Kwok Cheung, Bert Frescura, and others.

Pen Alignment in a Two-Pen, Large-Format, Inkjet Drafting Plotter

Misalignments are found by using a quad photodiode sensor to measure test patterns printed on the media. Scan-direction errors are corrected by timing adjustments. Media-direction errors are corrected algorithmically and mechanically.

by Robert D. Haselby

Print quality limitations of thermal inkjet printers arise from several factors. These may include writing system resolution, pen-to-pen alignment in multiple-cartridge printers, and manufacturing tolerances in pen and carriage mounting systems.

Also contributing to print quality problems are variations in inkjet drop velocity. The uncertainty of drop velocity makes it difficult to correct for print quality errors when printing in a bidirectional mode with either single-cartridge or multiple-cartridge printers. Since the drop must travel a finite distance before striking the media, the lateral placement of the resulting drop on the media depends upon carriage speed, nozzle height, and drop velocity. These factors are repeatable enough so that unidirectional printing results in acceptable print quality with single-pen printers. Bidirectional printing of text is acceptable if the row of text is printed completely in one scan. This is because there is separation between rows of text and misalignments are not noticeable as long as each row of text is printed in a single swath direction.

For throughput considerations, the HP DesignJet plotter was designed with two HP DeskJet printer multiple-nozzle pens and requires bidirectional printing of fully populated graphic data. Mechanical tolerance studies quickly revealed the necessity of an alignment scheme for the two DesignJet pens. Alignments in both the swath scan (carriage motion) direction and the media advance direction are required.

We determined that one-time factory alignment would not be sufficient, since we have no control over pen mechanical tolerances and the tolerances just in installing the pen in the carriage exceed our print quality goal. We considered having the customer adjust the pen alignment manually, but the time and customer interaction required and the subjectivity of a manual alignment procedure made this alternative unacceptable. It seemed that only an automated system of alignment initiated automatically whenever a pen is changed would be acceptable to the average customer.

Automatic Alignment System

The automatic alignment system for the HP DesignJet plotter is based on the following principal steps:

- Printing a test pattern on the media that is sensitive to some error mechanism
- Measuring the test pattern with a carriage-mounted sensor
- Applying a correction calculated from the measured error.

The correction is applied by changes in timing for scan-direction vertical errors or by physically moving one cartridge relative to the other to correct errors in horizontal swath banding.

An optical sensor with high accuracy is required for this technique. Printing errors are noticeable if they are larger than a visible threshold, which was determined by visual print quality studies. Automatic correction of these errors requires a measurement system that is capable of measuring less than about one third of the visual threshold.

A design goal was to make the alignment scheme as inexpensive as possible. This ruled out a high-accuracy, high-resolution scan encoder scheme with a single detector such as a bar-code sensor. Instead, the accuracy and resolution had to be in the sensor. This allows the distance in the scan direction to be measured between two lines, which are printed in a test pattern during normal operation. The method requires that the carriage be positioned so that the test pattern is within the maximum sensing range of the sensor. Thus the maximum range must be larger than the repeatability of the carriage scan-axis positioning system.

The test pattern for the scan-axis calibration is a line drawn with both print cartridges in both scan directions. Ideally this forms a continuous vertical line with no discontinuities visible. The test pattern vertical line is measured relative to the sensor without moving the carriage. This is done by moving the media under the sensor while the carriage remains stationary. Each area of interest on the test pattern vertical

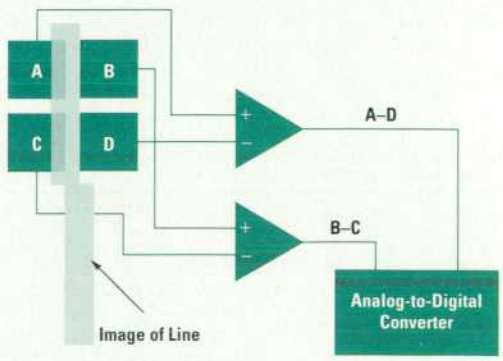


Fig. 1. Quad photodiode sensor circuit diagram.

line is measured by moving the media until the area of interest is under the sensor. Several sensing schemes were considered before settling on this solution to the alignment problem.

Quad Sensor

A quad photodiode array is used as the sensor with illumination from a diffuse light-emitting diode. A simple two-lens optical system images the test pattern onto the quad photodiode. Fig. 1 shows how the quad photodiode is used to measure the position of lines imaged upon it.

For detecting line displacements in a single direction, a dual photodetector would be sufficient. However, we were interested in both horizontal and vertical sensing. A quad photodiode allows the orientation to be controlled electronically. Fig. 1 shows two channels of analog signals that are converted to digital information in an analog-to-digital converter (ADC). Each channel is generated by taking the difference of two diagonally opposed elements of the quad photodiode. By arithmetic addition and subtraction of these two signals, (A–D) and (B–C), the sensor can be used as either a vertical or a horizontal differential sensor:

$$(A–D) – (B–C) = (A+C) – (B+D)$$

$$(A–D) + (B–C) = (A+B) – (C+D).$$

The first equation above shows that subtracting the two converted analog signals gives a differential detector that is essentially like a dual photodiode with one element being segment A and segment C and the other element being segment B and segment D. This is a dual differential detector that is sensitive to the horizontal position of the images of a line in the vertical direction as shown in Fig. 1. If the whole quad is illuminated uniformly except for the image of the line, Fig. 2 shows the output after subtraction that results from horizontally scanning a vertical line image through the quad photodiode array. A similar curve would result from scanning a horizontal line image through the quad sensor vertically while looking at the sum of the ADC channel outputs (second equation above).

Error Correction in the Scan Direction

The vertical sensor center region is calibrated each time the alignment process starts. The short-term stability of the sensor is sufficient to allow calibrating it only once for the entire alignment process. Calibration is automatically accomplished

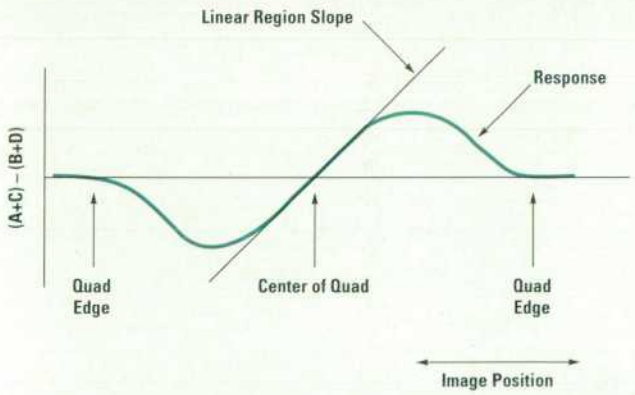


Fig. 2. Response of the difference between the two quad sensor outputs to the position of a vertical line.

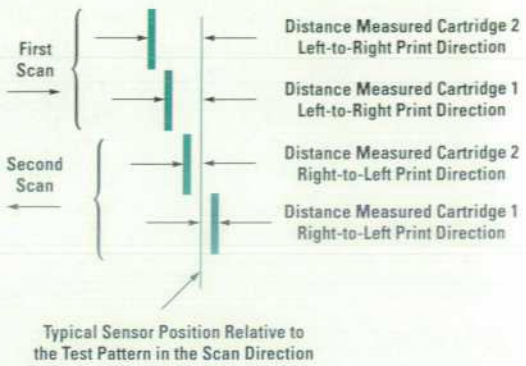


Fig. 3. Measurements for error correction in the scan direction.

by first printing a vertical line with one inkjet pen in a unidirectional print mode. Each swath prints a vertical line approximately 0.015 inch wide and steps over one dot column for each swath printed. This is repeated 26 times to generate the sensor calibration pattern. The media is then rewound so that the carriage-mounted sensor is over the nominal center of the slightly diagonal vertical line. The line is scanned across the sensor in the horizontal direction by advancing the media one swath for each column position and recording the resultant signals from the sensor. If these results are plotted against the swath advance a curve like that shown in Fig. 2 will result.

This data is placed in an array and processed to find the linear region and the slope of the linear region in counts per resolution element. A resolution element is defined as the horizontal distance between adjacent dots; for 300-dpi resolution, a resolution element is 1/300 inch. The linear center region is found by crosscorrelating the data array with a template that looks like the center portion of the ideal sensor curve. Once the peak of the crosscorrelation function is found, the center point and several points on either side of the center point are fit using linear regression to find the slope of the sensor curve. This slope is then used to measure the difference of all other line segments used in the horizontal calibration of the carriage alignment. Limitations on this process are mainly because of differences in line width and contrast ratio, which are caused by drop volume differences between print cartridges and different directions of printing.

Fig. 3 depicts a vertical test line drawn with two pens. A line is first printed in a left-to-right scan direction. This is followed by a swath advance, and then another vertical line is printed in the right-to-left scan direction. Fig. 3 shows the resulting error that would be measured on each segment of the "vertical" line. These measured errors are used to calculate the drop firing delays for each inkjet pen for each print direction so that the resulting test line will be continuous, with no discontinuities between either print cartridges or swaths. The method of calculating these delays attempts to drive each line segment to some average position, which is determined by averaging the individual errors. The correction delay (or advance) for each pen is calculated by taking the difference between the measurement for that pen and the average. This is the basic scheme for all corrections in the swath direction.

The actual process is somewhat more complex. Each inkjet pen is divided into two primitives: an upper primitive (nozzles 26-50) and a lower primitive (nozzles 1-25). Rotational errors can be corrected by dividing each cartridge into two primitives and measuring and calculating delays for each. Thus the actual system uses eight line segments for final correction instead of the four shown in Fig. 3. To minimize the effects of measurement noise, several lines are measured and the results averaged. Also, the sensor is affected by nonuniformity of the media. To correct for this, a background measurement is taken and saved before printing the test pattern. This background data is then subtracted from the differences measured for the test pattern.

Error Correction in the Media Direction

Pen-to-pen vertical separation errors are corrected algorithmically and mechanically, since in this direction, the cartridges are related mechanically and not by timing. By using 48 of the 50 nozzles on each pen and adjusting which 48 to use on each cartridge, corrections can be made to the nearest half nozzle. However, this alone is not accurate enough to meet the print quality specification. Therefore, a second, mechanical means of vertical pen alignment is necessary.

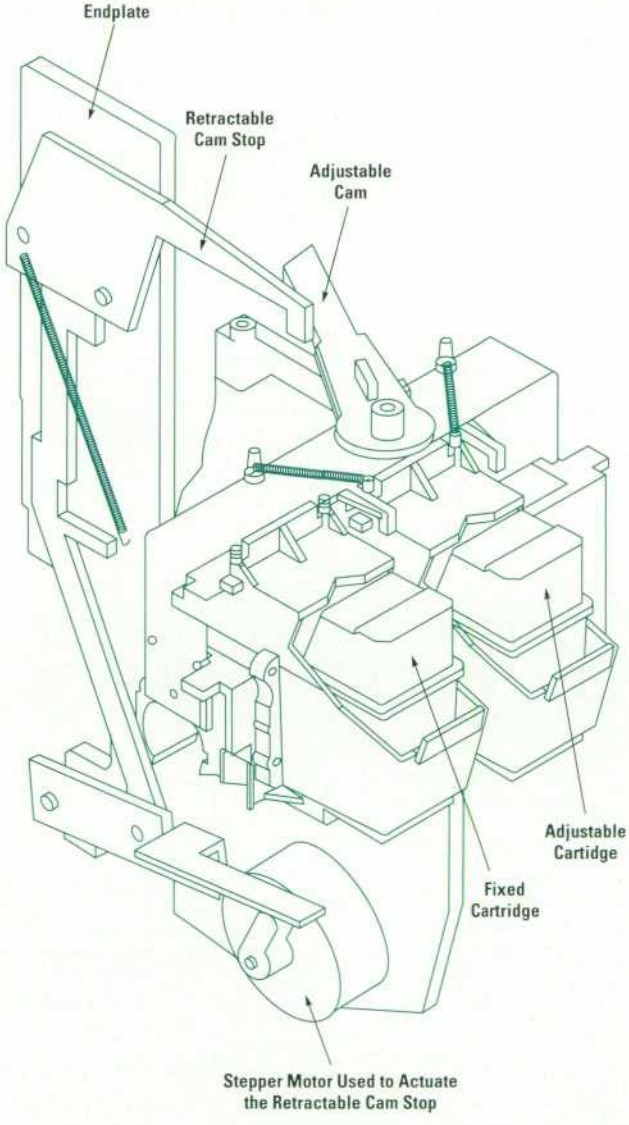


Fig. 4. Print cartridge alignment mechanism for error correction in the media direction.

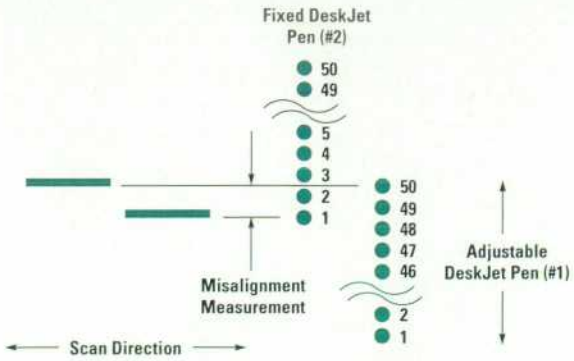


Fig. 5. Measurement for error correction in the media direction.

The need for automatic mechanical adjustment of the relative overlap distance between the two DeskJet pens on the DesignJet carriage was evident early in the project. Ideally, it was desired that the mechanism not require additional expensive components for actuation. These requirements are met by a carriage with a movable cam that can be actuated with the motion of the carriage against a retractable cam stop (Fig. 4). The cam stop is actuated by means of a linkage from the stepper motor that controls the pen nozzle wiper blade. The carriage cam is designed so that about 5 cm of carriage motion is required to adjust the pen-to-pen spacing by about three nozzles. This motion, along with the algorithmic selection of appropriate nozzles on each pen, allows the DesignJet plotter to correct for up to ±4 nozzles of manufacturing tolerance.

The resolution of the media drive axis of the DesignJet print mechanism is nominally 12,500 counts per inch. The gearing is set up so that, nominally, the worm gear rotates two revolutions for each swath of media advance. This significantly reduces cyclical gear errors, since the gear errors have a periodicity that corresponds to one rotation of the worm gear. The high resolution and high repeatability of the media axis controller allow an alternative error measurement scheme to be applied to the media axis.

To measure errors in the media axis direction, two horizontal, short, nonoverlapping lines are drawn on the media, as shown in Fig. 5. The fixed inkjet pen (#2 in Fig. 5) draws a horizontal line with nozzle 1. The adjustable inkjet pen (#1 in Fig. 5) draws a horizontal line with nozzle 50. These lines are long enough so that scan-direction positioning of the carriage is not critical. This pattern is repeated several times with a one-nozzle advance between each swath. This gives a wider line and a larger signal when these lines are later scanned. The media is then rewound and the carriage positioned so that the sensor is scanned through the line drawn by nozzle 1 of cartridge #2. The rewind distance is large enough so that an array of values can be stored that contains the entire sensor response, which is like that shown in Fig. 2. The resultant data array is then processed by correlation and linear regression to solve for the center of the sensor. The media advance distance used for the data scan is one nozzle pitch (0.00333 inch). This allows the units of the independent variable in the linear regression to be nozzles. The slope of the best-fit linear line through the center of the

sensor response is used to solve for the nozzle and fraction of a nozzle where the linear line crosses the mean of the data array. This process is repeated for the line from nozzle 50 of inkjet pen #1. The difference between the measurements of nozzle 1 of cartridge #2 and nozzle 50 of cartridge #1 is used to calculate how much to adjust the pen-to-pen spacing on the carriage. The adjustment value is used in an algorithm that decides which 48 nozzles to use on each inkjet pen. The adjustment remainder is used to calculate the final position of the adjustable carriage cam. After this calculation, the carriage and the retractable cam stop are used to position the cam to correct the remaining error.

Acknowledgments

I would like to thank Mike Nguyen, the mechanical designer of the carriage, for implementing the mechanics of the optical design, and the rest of the firmware team for answering the many questions that arose while I came up to speed in their domain. Diane Fisher implemented the many test pattern generators that were required in the early stages of development. Curt Behrend and Keith Cobb were valuable sounding boards for ideas and help when I couldn't see the forest for the trees. Keith also performed valuable algorithm analysis and redirection late in the project. Teri Tracey handled the sensor electronic implementation on the carriage printed circuit assembly. Al Hockley and the HP Optoelectronics Division did major work in developing a diffuse bright LED. I would also like to thank the whole management structure for allowing me to convert to a computer scientist for a year and a half and for keeping the faith when told that alignment would be the last thing to work since it had to be invented after all other systems were more or less in place.

DesignJet Plotter Chassis Design: A Concurrent Engineering Challenge

Instead of the expensive prestraightened slider rods used in previous designs to form the guideway for the pen carriage, the DesignJet chassis uses rods that are straightened during assembly and held in place by a low-cost rigid structure. The chassis components, assembly process, and assembly tooling had to be developed concurrently.

by Timothy A. Longust

Like an automobile, the HP DesignJet large-format drafting plotter has a chassis, which serves as the foundation to which all of the other parts are attached. The DesignJet chassis also provides a precision guideway for the pen carriage to move back and forth over the media.

The DesignJet chassis is rugged and made of low-cost components, yet it is very precise. The concept of integrating ruggedness and precision is an approach that departs from previous chassis solutions. Instead of purchasing precision components and assembling them in a relatively random manner, this approach uses nonprecision components and assembles them in a precise, systematic manner.

The DesignJet chassis concept became a reality as a result of a significant engineering effort that involved concurrent iterations of component design, assembly tool design, and assembly process development.

System Requirements

The design of the chassis was largely driven by the requirement for good print quality. The two HP DeskJet pens that the DesignJet plotter uses require a very consistent pen height above the media to achieve good print quality. The guideway for the pen carriage needs to be straight or true across a 36-inch span. It is expected to remain true even when subjected to external vibrations. It is not allowed to oscillate, thereby disturbing the pen heights.

The chassis also needs to be very rigid since it is the backbone of the entire product, and it needs to be rugged to

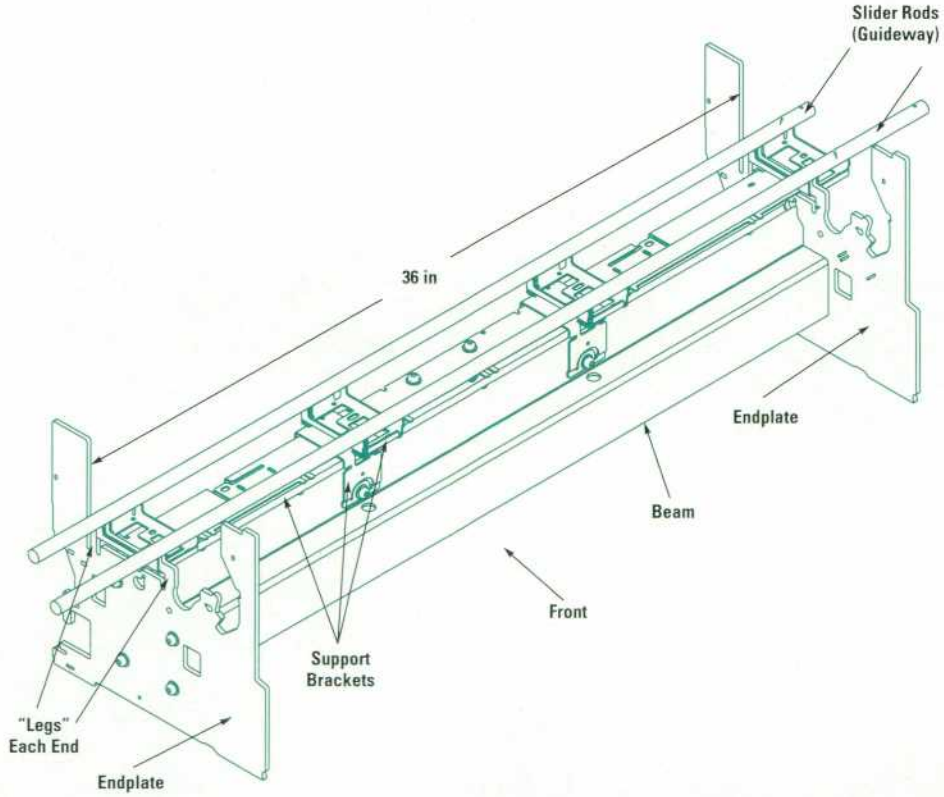


Fig. 1. Chassis assembly of the HP DesignJet large-format drafting plotter.

withstand shock loads and to be immune to vibration and thermal cycling. These characteristics are required to be sustained throughout the life of the product and to be achieved at minimum cost.

Design Concept

Fig. 1 shows the final DesignJet chassis configuration. It uses two slider rods to form the precision guideway for the pen carriage. The hefty beam and two endplates form a rigid structure to serve as the backbone for the product. Several support brackets are used to connect the rods to the structure. Each of these components is assembled using screws.

Previous chassis concepts use expensive prestraightened slider rods for the guideway and other expensive structural components to mount them. The DesignJet plotter chassis concept uses nonprestraightened slider rods for the guideway and a low-cost structure that simply holds the slider rods straight.

In the DesignJet plotter the straight guideway is achieved by deforming the long rods into a straight position, building and connecting a rigid structure onto the rods, and then releasing the rods. Although the rods want to spring back to their original free-state condition of being bowed and/or twisted, they are held tightly in place by the rigid structure. Rods that are bowed as much as 0.020 inch initially can be held straight to within 0.006 inch once assembled.

This concept represents a significant improvement in performance and cost over previous chassis concepts. It achieves twice the guideway straightness of any previous solution. It also represents a significant cost savings in the manufacturing of the components.

The challenge of the DesignJet plotter chassis concept was to derive a system of components that could securely hold the slider rods at the lowest possible cost. This system has to withstand the mechanical loads caused by the strain in the slider rods as well as the loads from shock. The system of support brackets and the rigid structure serve this purpose. However, assembly of these components is not straightforward.

Early in the development it was discovered that there are considerable interactions between the components as they are screwed together. The interactions depend on how the individual components are attached to each other and the order in which they are assembled. In several cases individual components act together as groups to behave differently than predicted.

Because of these complex interactions of the components within the chassis, the assembly process and assembly tooling became as critically important as the component designs. All three areas required simultaneous development. This concurrent engineering effort is illustrated in Fig. 2.

Component Design

The mounting of the slider rods was carefully studied. Having intermediate supports allowed the diameter of the rods to be significantly reduced. Without the two center supports

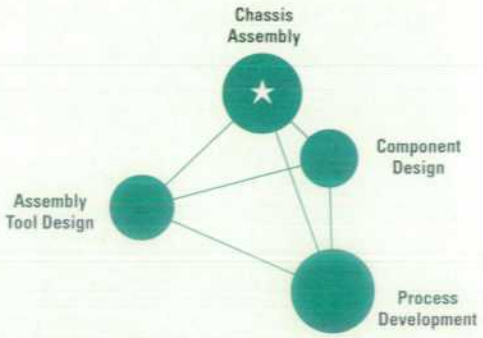


Fig. 2. The DesignJet chassis development process required concurrent component design, process development, and assembly tooling design.

each rod would have had to be approximately four inches in solid diameter to meet the straightness and vibration requirements. Of course, some equivalent tubular cross section of the rods could have been used, but not without considerable cost. Intermediate supports seemed to be a good trade-off to minimize slider rod material and weight and add considerable rigidity to the rods, making the entire chassis assembly less susceptible to external vibrations.

The design of the support brackets was rigorously analyzed to find the lowest-cost method of rigidly holding the slider rods in all directions. It was believed that by taking advantage of geometry, sheet metal could be used for all of the support brackets. Normally, sheet metal is considered very weak for structural applications because it is so thin. However, all of the sheet-metal brackets in the DesignJet chassis are designed to take mechanical advantage of the slenderness ratios of sectional geometries, that is, they are rigid in some directions while being flexible in others. Some of the brackets are made more rigid by forming flanges to stiffen sections. Some brackets are made more flexible by adding relatively large open areas to allow the bracket to twist or flatten as it is assembled. All of the support brackets ultimately work together as a system to provide the overall cost-effective solution for supporting the rods.

A summary of the individual components is shown in Fig. 3. This chart shows each component, the primary function it serves, the manufacturing process used to make it, and the advantages of the selected manufacturing process. A concerted effort was made to match the design requirements to the manufacturing process to provide a cost-efficient solution for each component.

For performance reasons the slider rods are made of plated steel. For cost reasons, the beam is made of aluminum. Since these components are made of dissimilar materials and both span the 36 inches between the endplates, there is a thermal expansion issue that needed to be resolved. In the final solution, "legs" were added within the endplates beneath the slider rod mounts. These legs flex elastically under the thermal loads that develop after a temperature change, without affecting the overall strength of the chassis. This thermal expansion design feature had negligible effect on the cost of the endplates.

Chassis Component	Primary Function	Manufacturing Process	Primary Process Advantage
Beam	Rigidity	Extrusion	Extremely Rigid Cross Sections Feasible
	Mounting Support Brackets	Machining	Flexible Mounting Configurations
Endplates	Mounting Slider Rods	Fineblanking	Accurate Features
	Mounting Nonchassis Components		Infinite Mounting Configurations Feasible
Support Brackets	Mounting Slider Rods	Sheet Metal	Flexible Mounting Configurations

Fig. 3. DesignJet plotter chassis components, functions, and manufacturing processes.

Process Development

The initial goal of process development was to develop a basic understanding of how to achieve the performance goals. Later, full characterization of the chassis assembly would be done under dynamic and environmental conditions.

Development of an assembly process began after a prototype assembly tool was created. Each component was studied individually to determine whether it performed in the desired manner. Interfaces between components were monitored to determine whether adequate friction developed to prevent components from slipping relative to each other. However, continued testing showed that the way in which the components are fixtured during assembly affects the resultant friction between interfaces after assembly. Issues like this, along with complex component geometries, made continued theoretical modeling difficult. Empirical testing was required.

Empirical testing included many tests using actual component materials and geometries. After careful studies, considerable insight was gained to discover which components and interfaces were most significant to overall performance. Particular groups of these components were tested separately from the assembly. Forces and deflections (stiffnesses) were measured to optimize each group's performance. After achieving a fundamental understanding of the chassis assembly in a static situation, process development progressed to dynamic conditions.

In shock, components within the rigid structure slipped relative to each other. Fastener tightening torques were increased to gain more clamp load at many interfaces. However, in some applications, the maximum recommended loads for fasteners were found to be insufficient. Instead of increasing the size of the fasteners, it was found to be more cost-effective to pretap some of the holes instead of allowing them to be self-tapped by the fasteners. By reducing the drive load required to seat the fasteners, more clamp load became available to prevent relative component slippage.

To prevent component slippage further, it was believed that additional fasteners between the major components of the chassis assembly would be better. However, after testing, this idea proved to weaken the shock resistance of the assembly. The additional fasteners reduced the flexibility within the chassis assembly and thereby transmitted shock loads to more sensitive areas. The final solution uses a minimum number of fasteners to optimize flexibility and strength.

Assembly Tooling Design

Specialized assembly tooling is required to bend and hold the rods in a nearly perfectly straight position and allow the rigid structure to be built around them. This tooling is required to hold each of the components in its proper position and allow access for fasteners and driving tools.

The assembly tooling uses a programmable sequencer that precisely controls the order in which components are loaded, the order in which the fasteners are tightened, and the tightening torques that the fasteners receive. Deviations in the assembly sequence are minimized.

Other important design parameters for the assembly tooling include operator friendliness, ergonomic considerations, and assembly time.

Since each chassis is assembled using the specialized assembly tooling, each assembly is considered factory calibrated. This implies that the chassis assembly is not serviceable by service personnel or the customer. Tamper-proof recesses in the heads of all machine screws are used to serve as a reminder that these screws should never be loosened or removed outside the factory. This was a carefully monitored aspect of the design.

The Concurrent Challenge

With component design, process development, and assembly tooling design occurring simultaneously, the entire development effort was very challenging. Interrelationships between two areas frequently caused ripple effects in the third.

Early testing of the assembly process showed major problems. For example, accessibility of the fasteners and their driving tools into the necessary locations of the assembly forced significant design changes in the components and assembly tool.

Providing locating features and clamping areas for repeatable component loading into the assembly tool also added complexity to both the components and the assembly tool. These features were iterated several times.

Conversely, limitations of assembly tool and operator ergonomics forced many modifications to the assembly process.

Later testing demonstrated that small changes in the assembly order or the clamping order greatly affected the finished chassis assembly's overall performance. Assembly order was iterated many times to optimize performance.

To cope with the volatile atmosphere of three moving targets coupled together, several techniques were used to make this development effort a success.

The first technique was to adopt a philosophy of maximizing design flexibility from the very beginning of the development. This philosophy translated into an attempt to limit each component to serving a single function. It was believed that single-purpose functionality would make design iterations somewhat easier to predict than they would be if the functions of components were allowed to be consolidated. Overall, this philosophy proved extremely valuable.

DesignJet Plotter End Covers Produced by Coinjection

Industrial design objectives combined with physical constraints made it a difficult task to manufacture the end covers for the DesignJet plotter. The deep-cored parts with no-draft outside surfaces made conventional plastic injection molding impractical. The entire weight of the DesignJet plotter is applied to the end covers when a customer lifts the instrument. The structural integrity required to make this possible, along with the need for low cost, negated the use of painted metal parts. These design requirements indicated that the coinjection process would best meet our needs.

The coinjection process can be described as a sandwich of conventional cosmetic thermoplastic skin with a structural foamed plastic core. For the DesignJet end covers, a polycarbonate/ABS blended material is used for the skin. This same material with a foaming agent added is used for the core. Both materials are injected sequentially on a two-barrel injection press through a proprietary nozzle. A controlled volume of skin material is injected followed by a controlled volume of foamed core material. As the skin is propelled by the core material, it adheres to the cool mold surfaces, allowing the core material to occupy and flow through the middle. This requires a minimum wall thickness of 6 mm.

Normally, a wall thickness of this size in a conventional plastic injected part would result in cosmetically unacceptable sink marks on the surface of the part. In the coinjection process, the foam additive in the core material expands in the mold. This action provides internal pressure to pack the thermoplastic skin material against the mold surfaces and prevent sink marks. Gate location on the part is also important to minimize blush and flow marks. Mold flow analysis techniques were used to determine gate locations and flow patterns for proper mold design.

The coinjection process provides a cosmetically acceptable part with no secondary painting required. The sandwich structure is rigid and impact-resistant. All secondary processes associated with conventional injection molding, such as ultrasonic insertion of hardware, are applicable to these coinjected parts. Although more plastic material is used than in conventional injection molding, the cost savings over painted metal parts is substantial.

In addition to these advantages, coinjection has afforded the DesignJet plotter the opportunity to use recycled resin in the structural foam core.

Steven R. Card
Mechanical Engineer
George F. Nasworthy, Jr.
Development Engineer
San Diego Technical Graphics Division

The second technique was to involve experts early in the development. Experts in manufacturing helped select the best processes for the components. Expert tool designers were involved very early and agreed to the goals and expectations of the assembly tooling. Expert production operators were involved early and built many assemblies themselves to communicate their sensitivities to the assembly process and tool design groups. Having experts readily available facilitated effective decision making.

The third technique was to avoid making several changes at once. Investigations moved slowly and methodically from one area to another, resolving one issue at a time. The experts were kept informed and agreed to the trade-offs, thereby preventing past issues from arising again. By recording dependencies of component attributes and behaviors as they were discovered, the process development and assembly tool design were eased. Careful documentation was important. Many controlled experiments allowed continual improvements in rod straightness and shock robustness. This effort continued until a minimum level of performance was achieved. It is probable that additional efforts could achieve still higher performance.

Acknowledgments

Dave Petersen was instrumental in conceptualizing the initial design. He was extremely valuable as a team player and effectively added theoretical understanding and mechanical experience. He is listed as a coinventor in the U.S. patent for the DesignJet chassis. Norm Ashley and Dennis Culver helped guide the initial concept away from past problems and towards a better alternative. Joe Milkovits provided manufacturing process knowledge and component design support. Steve Card and John Furman provided years of tool design experience. Andy Tricarico provided early production engineering input to the assembly tool design.

DesignJet Plotter Mechanical Architecture Development Process

By investing several months in designer communication before beginning detailed prototype design, an architecture was developed that was subsequently never changed, allowing the project to reach manufacturing release a month early. Costs for most subsystems were lower than expected.

by David M. Petersen and Chuong Ta

The mechanical design team for the HP DesignJet large-format drafting plotter used a mechanical architecture development process that proved to be very effective. The process emphasized designer communication early in the product development cycle and resulted in an architecture that was never changed after it was initially defined. The development teams for follow-on products based on the DesignJet architecture have found the configuration to be highly leveragable.

The mechanical architecture development is the first and last chance to make major design approach decisions about product geometry without causing project schedule delay. In the beginning of a project, very little effort has been invested in the detailed design, so it is easy to accommodate the design needs of other subsystems. However, thoroughly anticipating these design needs at this time is not easy. The DesignJet team used an architecture development process designed to make the needs of each subsystem visible so that they could be accommodated by other subsystems and competing needs could be balanced to the benefit of the overall product. The process emphasizes communication of design requirements and facilitates early feedback about design concepts.

History

Historically, mechanical architecture development for one of HP's large-format plotters was done by only one or two people. It began by assuming a design for servomechanisms to provide pen and paper movement. Ideas were collected about manufacturing processes that could be used to create a structural chassis to support the servo mechanics.

Taking one low-cost pen plotter as an example, friction rollers and a belt-driven carriage were chosen for the paper and pen servo mechanics. An aluminum extrusion was selected as the primary structural element for the chassis, and injection-molded side panels, on which the servo mechanics were mounted, were bolted to each end of the extrusion. Once the basic architecture had been established, the other members of the mechanical team designed their subsystems to mount on this chassis. Some of these subsystems included printed circuit boards, cosmetic enclosures, vacuum fans for paper flatness, and ground rods for pen-axis guideways.

The penalties of this approach to architectural development became apparent when trying to integrate the subsystems. The connection features for each subsystem were often not anticipated when the chassis structure was defined. This typically resulted in the addition of awkward, bulky, and costly parts, which were used only to mount other subsystems. Additional constraints were encountered as independent subsystems competed to be located near accurate chassis surfaces. Electrical ground paths were not integrated and unexpected structural resonances occurred. The injection-molded side panels of the low-cost pen plotter, which were initially conceived of as flat plates, grew into two of the largest and most complex plastic parts that our division has ever built. These caused several months of product development schedule delay.

The low-cost pen plotter design was a new platform for large-format plotters and was subsequently used as the basis for two other plotters. Although these products successfully achieved the established design goals, the mechanical architecture approach negatively affected the development schedule time and the production cost of all three products.

The primary factor that causes this type of approach is the urgency felt by the product development team to produce the first integrated hardware. Pressure to meet schedules can be extreme and the first integrated prototype is the key checkpoint showing that the product development is progressing. Teamwork is another significant factor. Lack of cooperation or working synergy between the design engineers and between the engineers and engineering management can

Fig. 1. Phases of the DesignJet plotter mechanical architecture development process.

encourage compartmentalization of designs and responsibilities. The result can be the assignment of the entire architecture job to one or two individuals and a nonoptimum mechanical product architecture.

Mechanical Design Team Selection

The DesignJet project began with a strong belief that teamwork can have a profound affect on development time, so a conscious effort was put into selecting design team members who would work synergistically with one another. Designers were chosen whose technical skills and working strengths complemented each other to ensure a solid technical team background. Personality compatibility was addressed by soliciting the inputs of existing team members about subsequent candidates for the team. The selection was limited to engineers who were not entrenched in other projects. A specific team personality was not the goal of careful team selection, but rather it was aimed at creating a team that was balanced in technical skills and compatible in interpersonal relations.

Architecture Development Process

The DesignJet mechanical architecture development occurred during the investigation phase that preceded the detailed design of the first prototype. Fig. 1 shows four distinct phases within this investigation period:

- The prearchitecture phase is used to identify and resolve issues needed before the architecture process can begin.
- The architecture phase is the core process in which the integration of subsystem geometry occurs.
- The postarchitecture phase allows time for resolving difficult integration issues.
- The architecture convergence phase brings together the final vision of the integrated mechanical designs before the detailed hardware design of the first prototype begins.

Prearchitecture Phase. Four mechanical engineers staffed the prearchitecture effort, which lasted about three months for the DesignJet plotter. The goal of this phase is to complete enough of the product definition and resolve the critical risk issues so that the architecture development will be realistic. The DesignJet product definition, including a use model, was 90% complete at this time. The fundamental feasibility issues were already understood or investigated during this phase. These included plot throughput, pen type and number, paper cockle, and ink-to-media compatibility.

The most significant issue studied was print quality. An extensive tolerance analysis of print quality errors was performed. A model of the plotter mechanics and electronics as they would affect print quality had to be assumed. Error budgets were assigned to each of the subsystems. Print quality acceptance surveys using graphic plots generated on HP DeskJet printers were performed to define acceptable error limits. The tolerance analysis was frequently updated during the investigation phase as the architectural concept matured. The error budgets became critical design goals for each subsystem.

Several design solutions were defined that became the starting points for the architecture geometry. An accurate paper drive system using a 2.5-inch-diameter elastomer-covered tube and pinch wheels to move the paper was developed and tested. Tests on this main drive roller were performed with paper to characterize cockle (wet paper bubbling) and paper drive accuracy. The ability of vendor processes to meet the tightly toleranced runout specifications was studied. A pen alignment scheme to measure and adjust the location of the two pens was developed. This required inventing a system to measure the ink location on the paper and an adjustment system to reposition the pens relative to each other.

At the end of the prearchitecture phase a complete list was generated of mechanical subsystems and subsystem design objectives. Among these subsystems were the pen carriage, pen carriage guideway, automatic media cutter, product stand, media stacking bin, product cosmetic enclosure, and pen service station. Also included was electrical hardware such as printed circuit boards, cooling fans, power switches, interconnect cables, various sensors, and the user interface front panel—these were the required subsystems identified by the electrical engineering team. The design objectives were related to both product performance and project development goals. Some of the product performance goals for the structural chassis included stiffness requirements to resist functional vibration and shock, an ability to tolerate temperature changes, and a pen carriage guideway straightness requirement that was part of the error budget from the print quality tolerance analysis. Some of the project development goals were concerned with product cost, design time, and design risk or the potential for unforeseen problems in a design choice.

Architecture Phase. The goal of the DesignJet mechanical architecture development was to choose each subsystem design approach to best meet the needs of the overall product and not just the independent needs of each subsystem. To achieve this goal, the DesignJet mechanical architecture team met in an isolated room for three weeks to conceive the product architecture. The assembled engineers included four mechanical designers, two manufacturing support experts, and one mechanical lead to act as facilitator. We stayed at the company site to have daily access to information and engineering tools, but felt that isolation was necessary to speed and focus the architecture phase.

Prepared with the design goals for each subsystem, we picked the paper drive system as the first to develop. Architecturally, it is the largest and most central part of the print mechanism. It was also one of the high-risk designs that had already been prototyped in the prearchitecture phase to determine its performance requirements. We began by presenting the design goals for the paper drive. Then various possible geometries were suggested, resulting in half a dozen alternatives. The merits and issues of each were discussed by all the architects present. The engineers who knew the most about the paper drive defended the design that they favored most and gradually the process converged on several choices with differing advantages and issues. During the discussions, all architects gained a clear knowledge of the design needs of the paper drive. They each offered their own insights about the options being discussed, along with concerns on how a geometry might affect another design. These insights were absorbed by the one architect who would later be owner of this design.

Next, the structural chassis was the topic of a similar process. It generated a greater variety of geometries than the paper drive had. For this large, tightly toleranced, and

potentially costly subsystem, several afternoons were spent gathering cost-versus-tolerance data about different manufacturing processes. CAD sketches of the different options were created to allow a common conceptualization within the group. Finally, several options were voted as superior.

The pen-axis drive system was next, followed by the pen service station, pen alignment, electronics enclosure, stand, and on through the subsystems until all had gone through the process, resulting in one or occasionally two suggestions for configurations. As we progressed through the subsystems the architecture team's understanding of the importance of interrelated design goals increased. Each architect began to own the design goals for all the subsystems. When sound ideas were presented they were reinforced by the team. If a marginal design idea was being pushed too hard, the others on the team would identify the concerns in enough detail to allow the advocates to rebut them or at least to realize the risks. As a result, each design included in the architecture process was subjected to significant scrutiny by the entire team and the design choices made reflected their combined knowledge and insight. In most cases, the final selection of the design approach was not democratic, but was left to the architect who would be responsible for doing the design. It is essential that the design owner fully support and believe that the design approach selection is appropriate within the context of the subsystem and overall product goals.

Occasionally during the three-week architecture phase it was necessary to stop the meetings and perform tests or gather data. Some design ideas were well-liked, but contained unknowns that were too risky to delay investigating.

For the architecture phase, each subsystem was given to one or two architects who were responsible for keeping track of and sketching the various concepts. The responsible engineer either owned the design later or transferred it to a designer who joined the team late in the investigation phase.

By the end of the architecture phase, all subsystems had been conceptually integrated into the plotter design. Each was understood in enough detail that 90% of the parts and processes were known. Simple two-dimensional assembly layouts were created. The process and the resulting architecture were presented to the other involved departments for review. Inputs and concerns were received from service engineering, customer assurance, marketing, industrial design, other support areas, and lab management (who were quite relieved to see some design output).

Postarchitecture Phase. The issues and parallel paths identified in the architecture phase were the focus of the postarchitecture phase. 2.5 months was spent investigating detailed feasibility. Testbeds were designed and testing was performed for user media loading, media cutter design, the structural chassis, media stacking, and electronics cooling. These tests allowed the final selection of design approaches to be made. Towards the end of this period the design team more than doubled in size as design ownership was distributed.

Architecture Convergence Phase. Two final weeks were spent in the isolated room to bring together the decisions on alternate design paths and complete the final vision of the mechanical architecture. This time all the designers including the original architects were present. Engineers from the electrical engineering team whose inputs had been previously solicited were included in these meetings. The plotter subsystems were all reviewed. A detailed plotter layout reflecting inputs from each designer was available. Minor issues were resolved and the detailed design of the first integrated prototype was ready to begin.

Results

The DesignJet mechanical architecture development process proved to be very successful. This process reduced the total product development cycle time, although primarily after the first prototype design, build, and test cycle was completed. The overall product program beat the original schedule to manufacturing release by one month. We never changed the architecture although we tried unsuccessfully to change some areas after the first prototype was built. In one instance, it was decided that the media loading design could be made easier to use. Some time was spent studying possible architecture changes to improve this area, but the conclusion was that the original choice was still the best for the overall system.

Communication between the designers was excellent all through the development indicating an understanding of the needs of adjacent subsystems. Arbitration of conflicting design objectives by engineering management rarely was needed. The team members' responsiveness to requests for help from other designers on difficult design areas was impressive.

For most of the designs, costs were lower than expected. The opposite usually occurs as architectural mistakes are encountered that require costly redesign. We met the zero-week availability goal which allows customers to purchase the plotter off-the-shelf on the introduction date. During the production ramp-up period, the production line was operational 68% of the time, an improvement of 100% above the previous divisional best. Six months after the start of production, the production support engineering staff had been reduced to half the typical number.

Our belief is that the effectiveness of this architecture process comes from two communication mechanisms. The common understanding of the product requirements each architect gains by participating in the development discussions of each subsystem gives that person the data needed to balance the design goals between subsystems to the benefit of the whole product. Also, exposing each subsystem to design input and critique by all the architects and others before the design begins prepares the designer with many insights into the design while there is still time to react to them.

Acknowledgments

We would like to make visible that the DeskJet mechanical architecture development process was allowed to develop because of the patience and trust of Bob Boeller, whose belief in management by objectives gave us the freedom to try something new. Significant contributions to this process were made by Gerold Firl, Dan Kline, Michael Nguyen, and Sam Stodder. Early manufacturing participation by Tim Longust and Tom McCue helped provide excellent realism and accuracy in the selection of subsystems. We wish to thank the many other designers and support people who made valuable refinements to the DesignJet architecture.

Improved Drawing Reliability for Drafting Plotters

The SurePlot drawing system, a feature of the HP DraftMaster Plus drafting plotter, significantly enhances drawing reliability and unattended plotting ability. The system is based on a noncontact color optical line sensor that verifies the writing of the pens.

by Robert W. Beauchamp, Josep Giralt Adroher, Joan Uroz, and Isidre Rosello

The lines drawn on an HP DraftMaster plotter, HP's high-end large-format pen plotter, compare favorably in precision and smoothness with those drawn by a human. The graphical output appears to be essentially perfect to the unaided eye.[1] The maximum specified pen speed of 110 cm/s and the maximum acceleration of 55.5 m/s^2 have been unsurpassed for over a decade. Although raster printing devices are faster, pen plotters are less costly and are able to satisfy the need for high line quality and multicolor large-format drawings.

To a large extent the value of a drafting plotter is a function of how quickly it can produce drawings of adequate quality, thereby maximizing the daily productivity of the user. Therefore, we saw an opportunity to provide a superior value in pen plotting by reducing the time invested by the user in operating the HP DraftMaster plotter.

HP DraftMaster Plus Plotter

The HP DraftMaster Plus plotter incorporates three types of improvements that enhance the user's productivity:

- Improved reliability, high enough for unattended operation
- Improved media handling
- A friendlier user interface.

These improvements have been made without increasing the cost of the product.

The most common customer complaints about pen plotters have always concerned reliability, mainly pens running out of ink, drying out, or clogging. The SurePlot drawing system is designed to provide a major improvement in the reliability of the drawing system, so that the plotter consistently generates plots without defects, and without requiring constant or frequent monitoring of the pens by the user.

Media handling is improved by roll feed, an automatic media cutter, and a new media tray. Users can retrieve their drawings at their convenience without individually loading every page before plotting or unwinding a takeup spool and cutting plots manually afterwards.

The enhanced user interface offers a redesigned front panel and simplified selection of pens, settings, and drawing quality. Immediate action keys and direct menu access keys give fast access to the functions most commonly used. The information appears in a larger, more easily readable vacuum fluorescent display.

This article focuses on the design of the SurePlot drawing system. The media handling system and the user interface are discussed in the articles on pages 42 and 49, respectively.

Common Pen Problems

Liquid ink pens based on capillary-action ink feed are usually used for final-quality plots and give excellent line quality. However, they require careful handling by the user. Liquid ink pens dry out quickly if they are not capped properly when not in use. Also, commonly used fine-width pens, say 0.25 mm, can experience insufficient ink flow after plotting a certain distance. This is because the relatively high velocity between the pen and the page abrades the paper fibers, which obstruct the end of the pen capillary, ultimately resulting in the clogging of the pen so that it quits writing. This failure mechanism depends on multiple parameters: the thickness of the pen, the type of media used, and the plotting speed and force. Another common occurrence is for one of the pens in the multistall carousel to run out of ink, becoming the cause of an unsatisfactory drawing.

When a pen fails in any of these ways, lines that should have been drawn are missing, and the entire drawing has to be plotted again with a considerable waste of time. The alternative is for the user to monitor the correct writing of the pens, resulting in a significant investment of time.

SurePlot Drawing System

The SurePlot drawing system includes SurePlot pens, a noncontact color optical line sensor, a pen distance monitor, and extra pens. SurePlot drafting pens have ceramic tips for clog-free plotting and regulators to make them leak-free. The optical line sensor allows the plotter to detect the most common pen failures: ink out, dry out, and clogging. The pen distance monitor detects when a pen is in danger of running out of ink on a drawing, based on the lower bound for the life of the pen. Failed pens and pens nearing the end of their lives are automatically replaced.

The system verifies the writing of the pen on the page by periodically sensing lines just placed on the page to verify that the print contrast is adequate for the given pen. The system previously learns the print contrast for a good pen of each type, thickness, and color. If the print contrast is unsatisfactory, the defective pen is replaced and the plot is either

restarted from the beginning or retraced from the last good verification, or the plotter halts to allow the user to select the appropriate corrective action.

An improvement of 40 times in the number of drawings not meeting user requirements has been measured for plotters using the SurePlot drawing system over plotters without it. At the 90% confidence level, 998 out of 1000 drawings meet user needs.

Optical Color Line Sensor

The design objectives for the line detection sensor for the HP 759X Series DraftMaster Plus plotters were:

- Low cost. The design should not cost more than the existing digitizer.
- Small size. The line sensor package should fit into a space measuring 1 by 1 by 0.5 inch.
- Low weight. The moving part of the line sensor must weigh less than 4 grams.
- Depth of focus. The line sensor must work over a 3.0-mm range.
- Pen colors. The line sensor must detect black, blue, red, green, violet, aqua, and brown. We wanted yellow and orange but were not willing to pay the added cost to include them.
- Media. The sensor must work with chart paper, vellum, polyester, translucent media, and tracing paper.
- Check with pen. The sensor must focus past the end of the pen tip but not be so low as to touch the platen when the paper feeler is touching the bottom of the pen groove. This objective means that the pen does not have to be stowed while sensing lines.
- Lighting. The plotter must work under normal office lighting conditions (lights off/lights on) or in ambient light up to 1000 lux.

Average User Plot

The performance of a hard-copy device almost always depends on the graphic information being sent to it. This means that an exhaustive performance test of a pen plotter requires that a complete test be run for every drawing in a set of drawings representing all targeted applications. This amount of test effort seemed excessive for testing the SurePlot drawing system for the HP DraftMaster Plus plotters, since we were interested in an average performance measure that would allow us to do summary comparisons with other products.

For this reason, we developed the average user plot, or AUP. This is a single, standard plot that synthesizes the graphic contents of a set of drawings representing the targeted applications at the time it was created (1990). The AUP drastically reduces the test effort and allows easy comparisons between products.

To create an AUP for pen plotters, we chose a number of user drawings representative of state-of-the art drawing complexity and applications. We then extracted the graphic information contained in each of the sample plots. This was done in two ways: by vectorial composition and by external shapes composition. To determine vectorial composition, drawing files were analyzed with a special parser that produces histograms of vector lengths, length plotted, number of pen up/down cycles, and other parameters. To determine external shapes composition, we measured the actual percentages of the total plotting length of each drawing that represented curves, lines, characters, filled areas, and so on. After extracting the graphic information in these two ways, we averaged all the information to obtain the composition of the AUP. We then created a unique drawing that reasonably matches this composition and meets our specific needs.

The cost objective and space limitations proved formidable. No sensors existed that met these aggressive requirements. This made it necessary to develop a custom sensor. The resulting sensor meets all of the design objectives and saves tooling costs by using off-the-shelf parts. Assembly costs were kept low by considering the part assembly from the

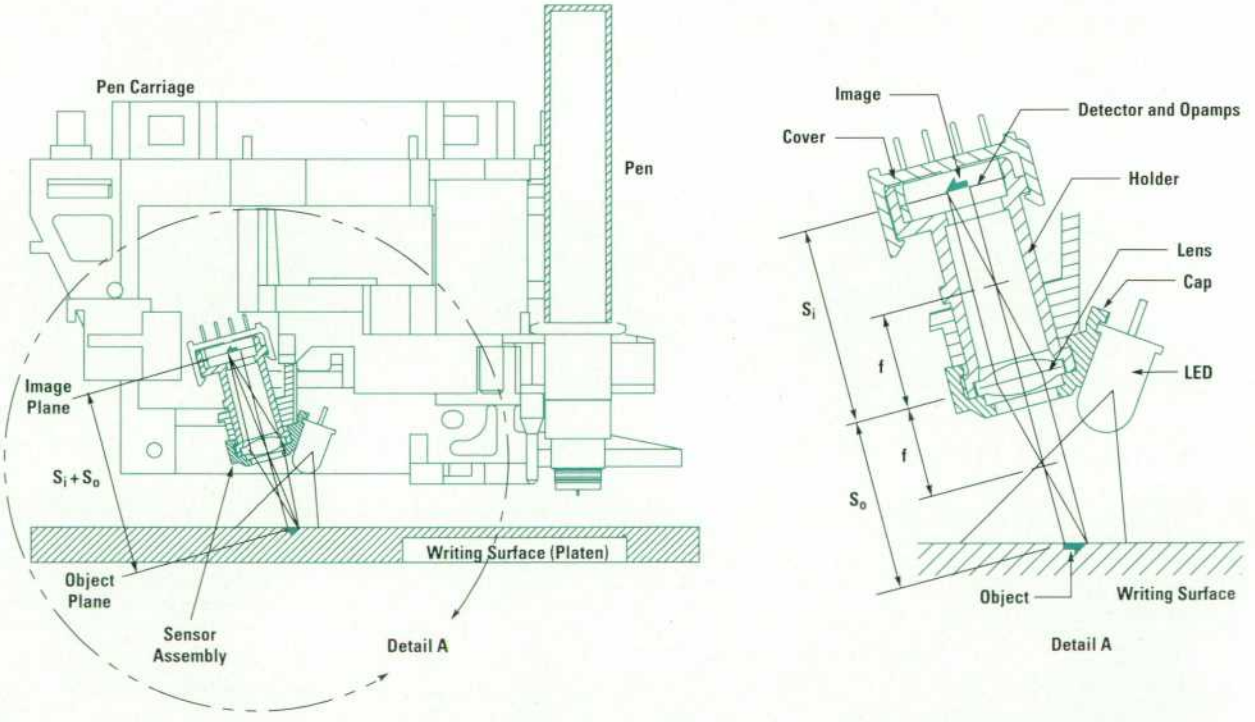


Fig. 1. Noncontact optical line sensor mounted in the HP DraftMaster Plus plotter.

Acceptable Quality Level Index

Acceptance sampling is the set of techniques employed to estimate the quality level of a given set of products by measuring the quality of a limited number of sample products.

Since these techniques are widely applied to incoming inspection of parts to be used in production, most of the literature specifically refers to incoming inspection topics.[1,2,3] Nevertheless, the basic underlying concept is applicable to any product. In our case, we wanted to describe the overall achievable quality of the output obtained during the life of a pen plotter through knowledge of the measured quality of a sample of drawings obtained during a limited time.

Sampling plans can be divided into attribute plans and variable plans. In attribute plans, a sample is taken from the lot and each unit is classified as good or bad. In variable plans, a measurement of a specified quality characteristic is made on each unit.

Since our quality descriptor for every drawing tested was whether it was acceptable, and because we couldn't make any assumption about the underlying distribution of the data, we used the U.S. military standard MIL-STD-105D procedures for sampling by attributes.[2]

The quality index in MIL-STD-105D is the acceptable quality level, or AQL, which is defined as "a nominal value expressed in terms of percent defective or defects per hundred units, whichever is applicable, specified for a given group of defects of a product."

References

1. J.M. Juran, *Quality Control Handbook*, McGraw-Hill Book Company, Inc., 1962.
2. *Sampling Procedures and Tables for Inspection by Attributes*, MIL-STD-105D, U.S. Government Printing Office, 1963.
3. I.W. Burr, *Engineering Statistics and Quality Control*, McGraw-Hill Book Company, Inc., 1953.

start and talking to the vendors at each step of the design process.

Mathematical Model

A simple mathematical model was used in the development of the line sensor. Other scanning sensors focus the emitter light and the detector at the same spot and therefore are very sensitive to changes in height. The basic configuration chosen for the HP DraftMaster Plus line detection system achieves a large depth of field by having the emitter illuminate a wide area and focusing only the detector at a point. This design removes the sensitivity to changes in height.

Fig. 1 shows the sensor design. The LED emits light with an intensity of I_v. The intensity incident upon the media, I_m, is directly proportional to I_v and inversely proportional to the square of the distance between the LED and the media. The light I_m is reflected by the media or ink into the optics, where the lens refocuses the light from the media (object plane) to the detector (image plane). A simple relationship relates the image distance S_i, the object distance S_o, and the focal length f:

$$1/S_i + 1/S_o = 1/f.$$

The depth of focus for a single lens with a focal length f is a maximum at $S_i = S_o = 2f$. This was chosen as the operating point for the HP DraftMaster Plus design.

The variable used to determine whether or not a line is present is the print contrast ratio, or PCR. The PCR is the ratio of the drop in light intensity at the detector resulting from light absorption by the plotted line to the intensity of the general white level of light reflected by the media. Using the PCR to determine the presence of a line removes any dependence on the light intensity.

Ideally, the spot size sensed by the sensor is an infinitesimal point. In reality, the detector needs some area to collect light. The lens also adds to the spot size because it does not perfectly focus the line onto the detector as a result of the aberrations of an inexpensive spherical lens. The spot grows even more when the lens is out of focus because of errors in positioning the object plane.

The voltage output of the detector can be modeled as the convolution of functions representing the line, W, the lens, L, and the detector, d (see Fig. 2). From a system perspective, the input W is a step function of width X_W (the line width). The system is modeled as the convolution of L and d, or $L \star d$, and the output V is the voltage output of the sensor.

The PCR can be calculated using the above model:

$$PCR = (V_W - V_{min})/V_W$$

$$V_W = I_m R_m$$

$$V_{min} = I_m R_i \phi(W) + I_m R_m (1 - \phi(W))$$

$$\phi(W) = \int_{-0.5X_W}^{0.5X_W} (L \star d)dX$$

$$PCR = \phi(W)(R_m - R_i)/R_m,$$

where R_m and R_i are the reflectivities of the media and ink, respectively.

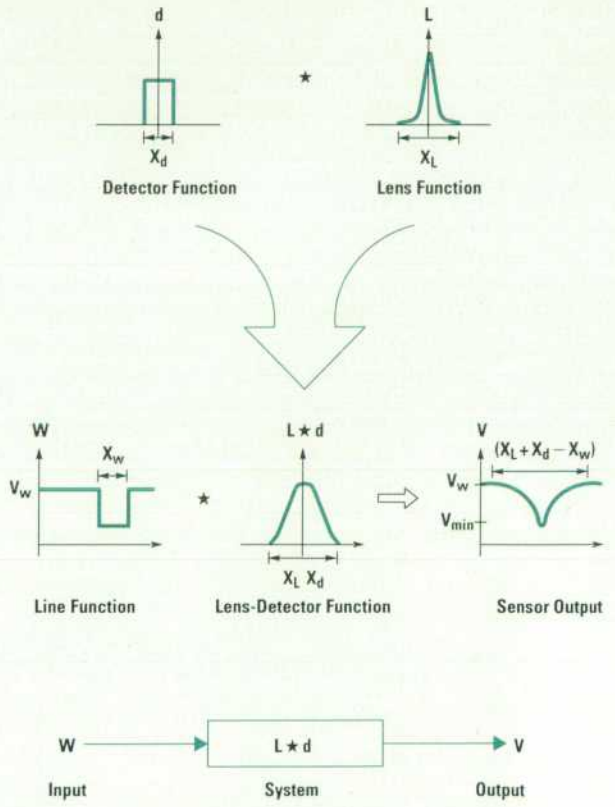


Fig. 2. Mathematical model for the optical line sensor.

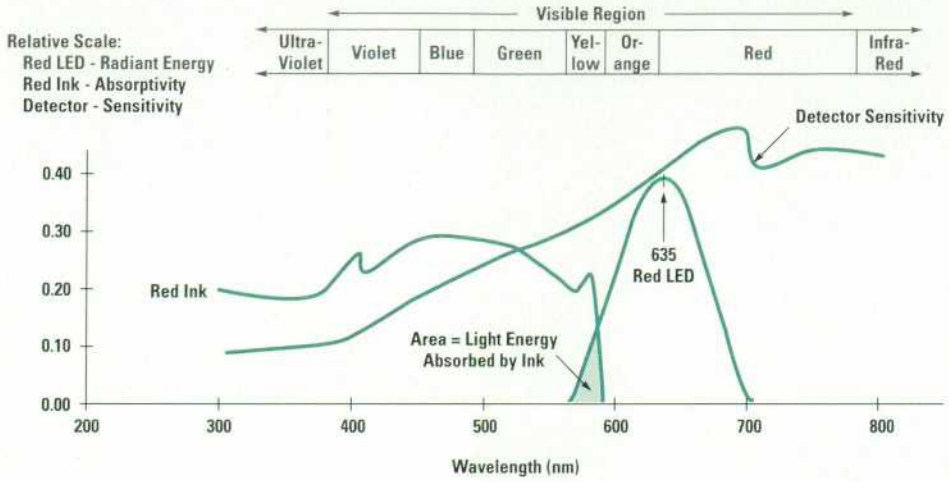


Fig. 3. Red LED and red ink absorbance spectra and detector sensitivity for the noncontact optical line sensor. The shaded area is a measure of the light energy absorbed by the ink.

This analysis demonstrates in equation form how R_i, R_m, and X_W influence the PCR. For example, the absorptivity of the ink is not so important as the difference between R_m and R_i. This implies that some consideration must be given to the fact that media with different reflectivities will produce different PCRs for the same ink. For this reason a white strip lies directly beneath the sensor to boost the PCR for transparent media with very low reflectivities.

As long as the sensor has not been saturated (too much light), the PCR calculation is independent of intensity I_m. This independence was verified by measuring V_W and V_{min} with the room lights on and off.

Light Source Selection for Color Detection

An important consideration, omitted in the above analysis, is the spectral characteristics of the LED, ink, media, and detector. It was tacitly assumed that the reflectivity of the ink, R_i, is the average reflectivity of the ink weighted over the spectral sensitivities of the LED and the detector:

$$R_i = \int_0^\infty R_i(\lambda) I_v(\lambda) I_d(\lambda) d\lambda.$$

Therefore, the PCR can be maximized by minimizing R_i. Because inks are normally characterized by their aborptivity rather than their reflectivity, minimizing R_i implies that the absorptivity A_i needs to be maximized. Thus, a light source should be selected that emits light at wavelengths where the absorptivity of the ink is a maximum. Halogen light sources are very good candidates because of their broad emission spectra but are not used in this application because they get hot and require much more current than LEDs. Blue LEDs are good candidates to detect yellow lines but they are costly, emit very low-intensity light, and would have problems detecting blue lines. Red LEDs have the highest-intensity output but they have a known problem sensing red ink. HP's high-intensity green LEDs were chosen for their low cost, low current, and emission spectrum centered in the visible spectrum. Green LEDs are also capable of sensing green lines because it is difficult to design an ink that absorbs blue light and red light but reflects green light. Fig. 3 shows the spectrum of a red LED compared to the absorption spectrum of red ink. Fig. 4 shows a similar comparison for a green LED/green ink combination. The shaded area represents the value of the above integral and is much larger for the green LED/green ink combination.

Figs. 5 and 6 show the results of sensing different colors of lines with red and green LEDs, respectively. Note the low signal produced by a red LED over a red line, whereas the green LED gives a good signal over a green line.

Detector

The detector has two distinguishing characteristics not found in most other detectors. One is its small size (0.25 mm

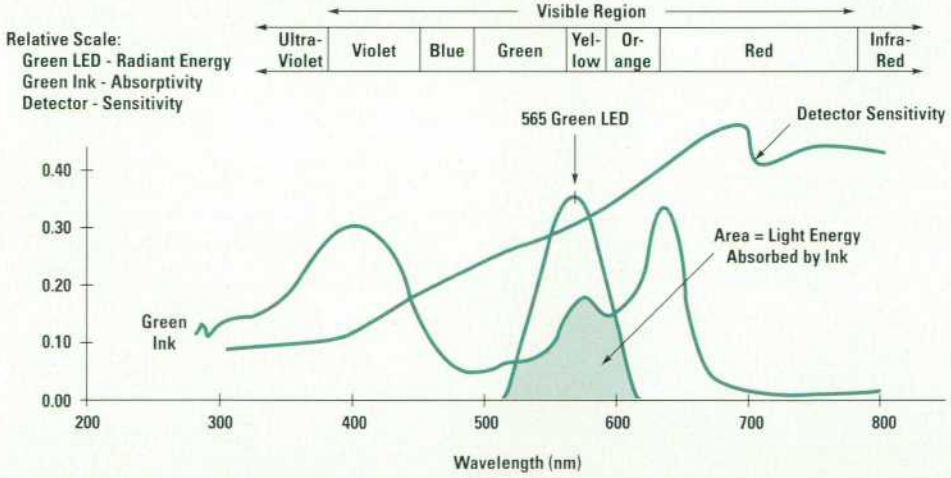


Fig. 4. Green LED and green ink absorbance spectra and detector sensitivity for the noncontact optical line sensor. The shaded area is a measure of the light energy absorbed by the ink.

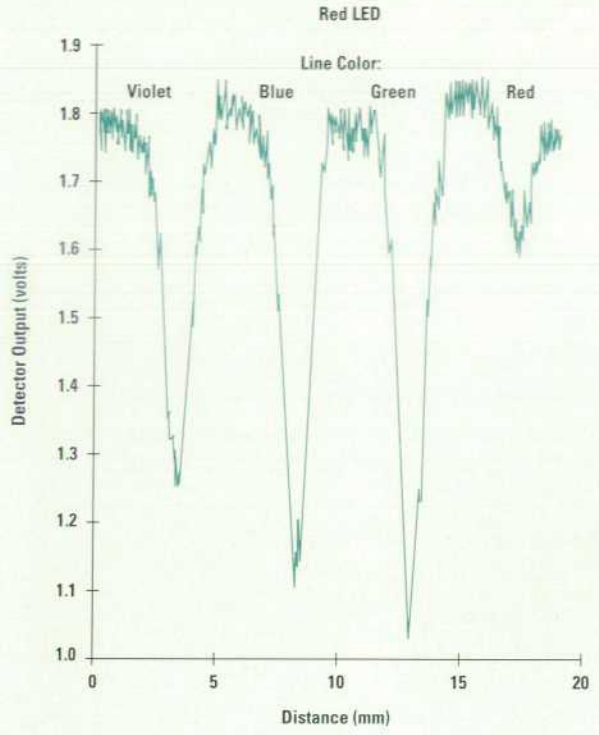


Fig. 5. Detector output for a red LED and violet, blue, green, and red lines.

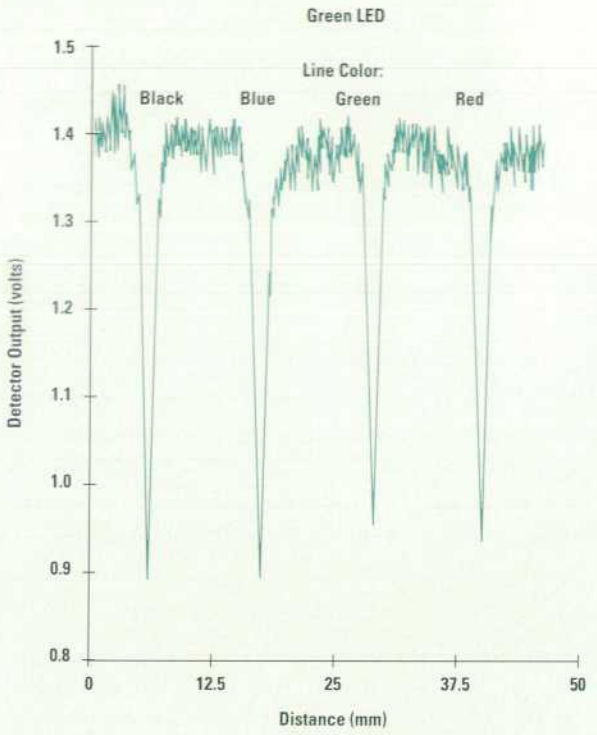


Fig. 6. Detector output for a green LED and black, blue, green, and red lines.

by 0.25 mm), and the other is a two-stage amplifier integrated on the same IC. The small detector size helps keep the spot size small, which is helpful for detecting small lines, while the two-stage amplifier boosts the signal before EMI can obscure it. This is important because the detector signal is sent across a 2-meter unshielded flex cable to the processor printed circuit board. Fig. 7 shows the detector circuit.

Sensor Assembly

Fig. 8 shows the line detection sensor assembly process. All parts are either snapped or press-fit together, thus eliminating messy, unreliable gluing processes and reducing the need for expensive assembly tooling. Excluding the wave solder pass (which is done in batch mode), it takes approximately 20 seconds to assemble the six parts that make up the sensor.

Testing the SurePlot System

Since the degree of attention needed to obtain usable drawings and the quality of the output are highly dependent on the operating conditions, we spent some time before beginning the tests of the SurePlot drawing system attempting to understand or model all of the possible real-life modes of operation and applications that users would be likely to apply.

We split the problem into three aspects: drawing contents, modes of operation, and required output quality. To keep the volume of tests required from becoming unreasonable, we synthesized in a single drawing the average graphic contents of a set of customer-representative drawings. We called the resultant drawing the average user plot, or AUP, (see page 36). Our knowledge of pen plotter customers, accumulated through continuing focus group sessions, customer visits, experience, and other means, allowed us to assume reasonable values for the components of the operating modes: roll feed versus cut sheet, different pen/media combinations, time between plots, workload, applications, and so on.

Finally, we set up the criteria to be able to evaluate the quality of the output obtained and recognize a failure situation. In some scenarios users are expecting high-quality output, so only high-quality plots were counted as acceptable. In other situations, draft-quality plots meet customer needs and are usable, so in these cases draft-quality plots as well as high-quality plots were counted as acceptable. In any

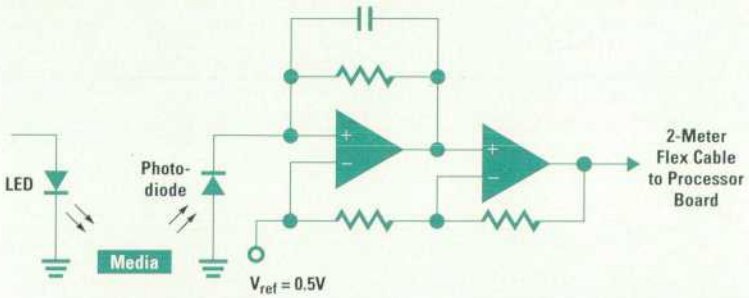


Fig. 7. Electrical circuit of the line detection sensor. The photodiode and the two-stage amplifier are integrated on the same IC.

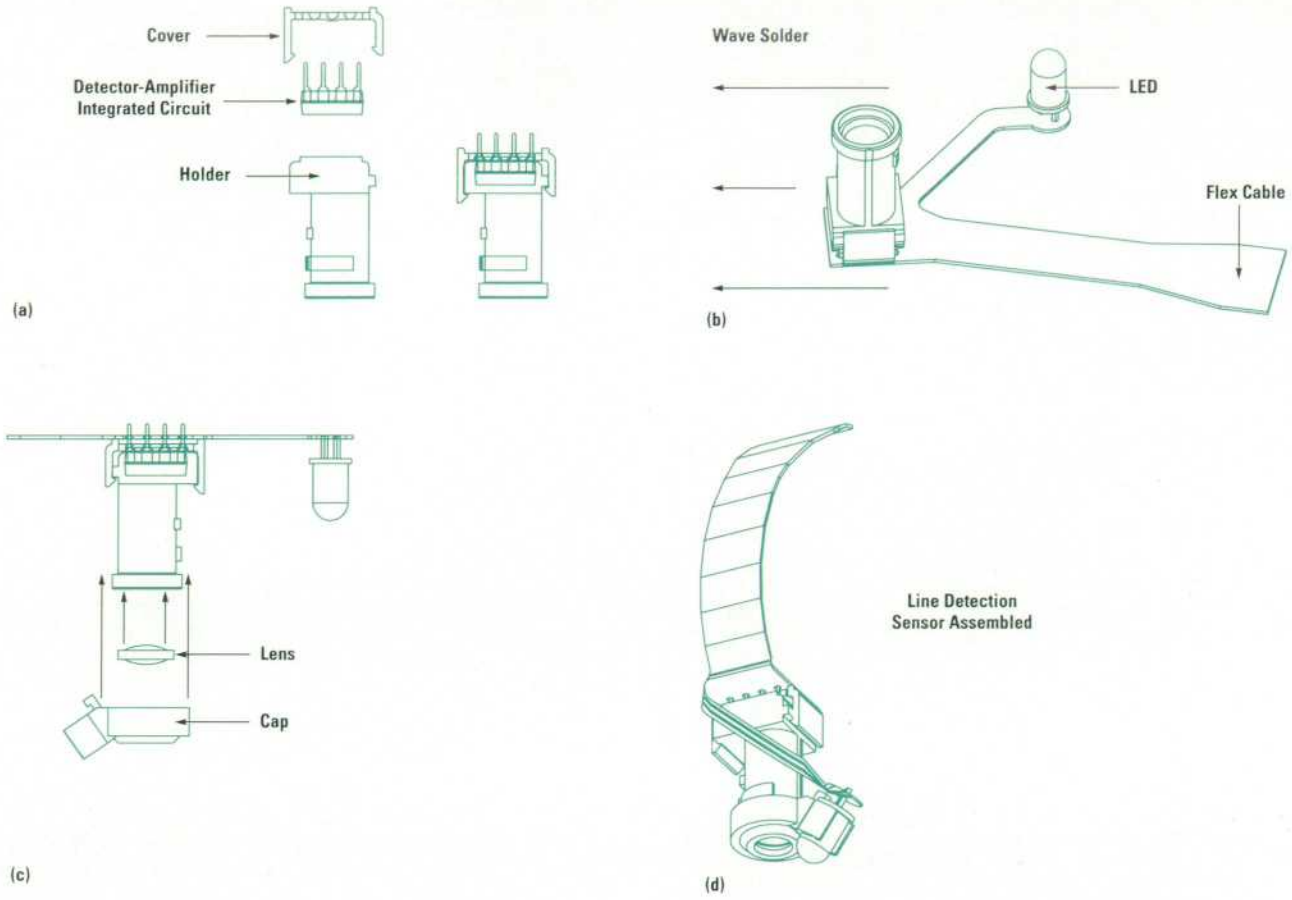


Fig. 8. Line detection sensor assembly process. (a) The detector package is captured between the cover and the holder, which snap-fit together. (b) The LED and the detector IC are wave soldered to the flex cable. (c) The lens is captured between the cap and holder, which press-fit together. (d) The LED is snapped into the cap.

case, a drawing with a missing vector was considered a failure unless it was recovered (only the HP DraftMaster Plus plotter was able to do this).

Test Description

To measure whether and by how much the HP DraftMaster Plus plotter is able to produce more usable plots with less user attention than another pen plotter, we had both an HP DraftMaster RX plotter and an HP DraftMaster RX Plus plotter plotting the AUP eight hours a day during two weeks of accelerated testing in all targeted modes of operation. The tests consumed ten rolls of media and 20 SurePlot and fibertip pens.

The fairly stable and known behavior of the SurePlot pens allowed us to accelerate the testing. These pens were used in a separate plotter until most of their ink capacity had been spent, and were put into the HP DraftMaster Plus plotter near the end of their life. These pens produced all of the HP DraftMaster Plus failure situations recorded. This technique permitted us to extrapolate the results of the 270 drawings plotted to be equivalent to 1500 plots, which represent 150 days of work on the basis of 10 plots per day.

During the test, we collected data on the number and duration of the user interventions necessary to keep the plotters working properly in the selected operating mode, and examined every drawing produced by both machines to classify them as final plots, draft plots, or plots with missing vectors.

Results

Figs. 9 and 10 show results of these tests. The qualitative conclusion is that the SurePlot drawing system allows the HP DraftMaster Plus plotter to deliver a higher percentage of valid plots than its predecessor while requiring only half as much of the operator's time.

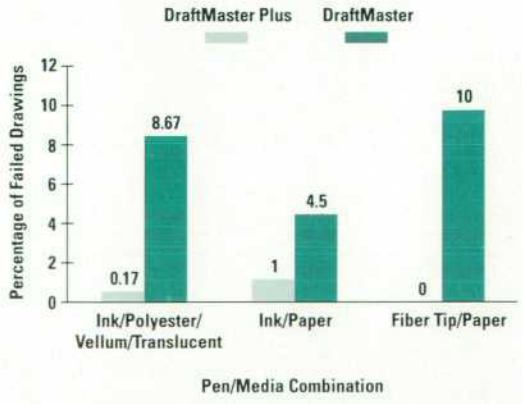


Fig. 9. Drawing reliability test results for the HP DraftMaster plotter and the HP DraftMaster Plus plotter with the SurePlot drawing system.

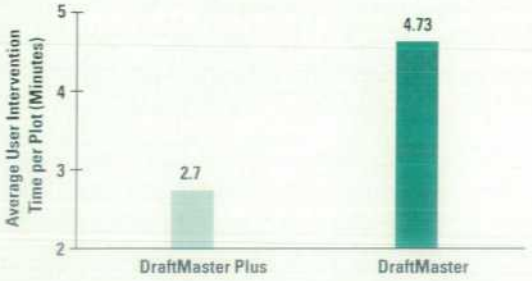


Fig. 10. Average user intervention times measured in the drawing reliability tests.

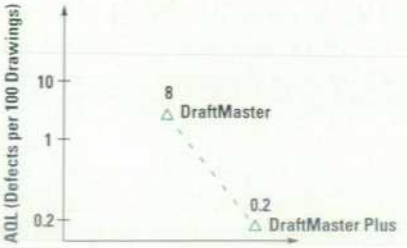


Fig. 11. In the drawing reliability tests, the SurePlot drawing system lowered the AQL (acceptable quality level) by a factor of 40.

Quantitative metrics are highly dependent on the operating mode. From the user's point of view, the question to answer is "How many good drawings can I obtain from a given batch?" To communicate that, we looked for a statistically representative quality index, a single figure that would represent the level of output quality that a customer would experience through the operating life of the plotter. We concluded that the most appropriate index for extrapolating our test data to the life of the plotter was the AQL, or acceptable quality level, defined in a U.S. military standard as "a nominal value expressed in terms of per cent defective specified for a given group of defects of a product" (see page 37).

The AQL results are shown in Fig. 11. The HP DraftMaster Plus plotter had an AQL of 0.2 defects per 100 drawings, compared to an AQL of 8 for the DraftMaster. These results had a confidence level of 90%. This can be interpreted to mean that 90% of HP DraftMaster Plus plotter users, using recommended pens and media, will find that 998 out of 1000 drawings will meet their needs, compared to 92 out of 100 drawings for the DraftMaster. This represents an improvement of 40 times in the percentage of unacceptable drawings over the original DraftMaster plotter.

Summary

The development of a noncontact color optical sensor that verifies the writing of the pens on the actual drawing greatly improves the drawing reliability of a pen plotter. The system operates on an extremely wide range of media, pens, and graphic pattern contents based on its ability to learn the right print contrast. This added functionality does not add cost to the product, since it replaces an existing accuracy calibration sensor. The system can detect errors on the actual traces and recover automatically. On the basis of the AUP and the test results, 90% of HP DraftMaster Plus users will find that 998 out of 1000 drawings will meet their needs. This represents an improvement of 40 times in the percentage of unacceptable drawings over plotters without the new sensor.

Acknowledgments

The authors are grateful to Luis Fernandez, Steve Vanvoorhis, and the entire project team.

Reference

1. M.L. Patterson and G.W. Lynch, "Development of a Large Drafting Plotter," *Hewlett-Packard Journal*, Vol. 32, no. 11, November 1981, pp. 3-7.

An Automatic Media Cutter for a Drafting Plotter

This simple, reliable, low-cost cutter is a classical rotating and linear blade design. It requires no separate drive motors and does not interfere with normal plotting performance. To quantify its performance, cut quality parameters and measurement methods were defined.

by Ventura Caamaño Agrafojo, David Perez, and Josep Abella

One of the main user needs driving the HP DraftMaster Plus plotter development was to reduce the requirement for operator attention to the plotter without increasing the product price. It was very clear that an automatic media cutting device that allowed the user to obtain drawings already cut to the desired size was an essential part of a set of features designed to satisfy this user need. With an automatic cutter, the cost of the product could be substantially lowered for some customers since the existence of a cutting device could eliminate the need to buy a takeup spool.

Determining Customer Needs

To help define the objectives for the cutter development, the technique called quality function deployment (QFD) was used to analyze customer needs and wants. QFD helped to identify the feature set the automatic cutter should have to satisfy customer expectations.

The first step was to collect customer needs by means of focus groups. After analyzing this data, customer needs were classified into groups, each of which represented an overall customer concern such as reliability or performance. The next step was to define the design characteristics that would address the customer needs. To translate cut quality into quantifiable terms we defined a set of measurable parameters. These parameters were used not only to set cut quality goals but also to make comparisons with competing products. The definitions of these cut quality parameters and the procedures for measuring them are presented on page 46. Design characteristics were analyzed based on customer responses and experiments to determine their relationships to the customer needs. On the basis of this analysis, we built the relationship matrix shown in Fig. 1, which indicates how much each design characteristic affects each customer need. Looking at the relationship matrix we could see the features that would meet the customer needs.

One of the most important benefits from using QFD was that it provided the means for concurrent development across all functions—marketing, R&D, quality, and manufacturing. As result of this coordination, design changes were minimal, which helped us to meet our schedule.

Mechanical System Design

Having defined the user needs, the cut quality parameters, the basic design goals, the present state of technology, and the constraints imposed by the product itself, the next step was to determine the cutter design configuration that would be best able to satisfy the various requirements. From an analysis of different possible cutting systems in the light of these requirements, it was apparent that the best solution was a cutter based on the classical rotating and linear blade approach. Instead of being driven by dedicated motors, such a system takes advantage of the plotter's existing drawing driving system.

A breadboard prototype with design parameter adjustment capabilities was built to determine appropriate values of the following design parameters:

- Rotating blade diameter (see Fig. 2).
- Inner rotating blade cone angle (see Fig. 2).
- Cutting speed.
- Rotating blade sharpness.
- Side force. This is the contact force between the rotating blade and the linear blade, which is produced by the compression spring (see Fig. 3).
- Depth of penetration. This is the amount of overlap between the rotating blade and the linear blade (see Fig. 2).
- Shear angle. This is the angle between the rotating blade perimeter and the vertical faces of the linear blade (see Fig. 2).[1]

Using experimental design techniques,[2] we were able with very few iterations to determine that all of these parameters were important, but only three of them were critical, since cut quality was very sensitive to them. These three parameters are depth of penetration, shear angle, and rotating blade inner cone angle. Their acceptable values were measured to be within the following ranges:

- Depth of penetration: >0.1 mm to 0.5 mm
- Shear angle: >0 to 15 minutes
- Rotating blade inner cone angle: >0 to 15 minutes.

With regard to sharpness, the rotating blade edge can be made blunt as long as the contact surfaces between this blade and the linear blade are two sharp points. This is also good for safety and economy.

Although media type is a very important quality parameter, the goal was that all the other parameters should be adjusted to make cut quality as independent of media type as possible. For this reason, no plotter media type ranges were established.

Customer Requirements	Design Characteristics 1	2.1	2.2	2.3	2.4	2.5	2.6	2.7	3	4	5	6	7
Reliability													
○ Reliability	S												
Performance													
• Cut Quality													
○ No Media Tear			S	S									
○ Clean Cut			S	S				S					
○ Square Edges		W	W	W	S	S		M					
○ Straight Cut		S	M	M									
• Throughput													
○ Quick Cut									S				
• Media Handling													
○ Accurate Size					W	W	S						
○ No Media Waste							W						
• Media Type													
○ Cut All Media Types												S	
Usability													
• Maintenance													
○ Replacement Frequency											S		
○ Easy to Replace												S	
• Human Factors													
○ Quiet													S
○ Safety													S

Design Characteristics:

1. Mean Number of Cuts Before Failure (MCBF)
2. Cut Quality Parameters:
   - 2.1. Straightness
   - 2.2. Waviness
   - 2.3. Perpendicular Deformation
   - 2.4. Parallelism
   - 2.5. Perpendicularity
   - 2.6. Accuracy
   - 2.7. Edge Effect
3. Cut Time
4. Mean Number of Cuts to Replacement (MCTR)
5. Replacement Time
6. Media Types
7. HP Class B2 Environmental Requirements

Relationship:

S: Strong

M: Medium

W: Weak

Fig. 1. Relationship matrix for the DraftMaster Plus plotter automatic media cutting system.

Cutter Operation

The cutter system consists of two basic subsystem devices: the slitting device and the engagement and driving device (see Fig. 3).

The cutter carriage stays in a rest or parking position on the left end of the pen carriage guide arm while the pen carriage is being operated during plotting (see Fig. 4). When the plot has been completed, the paper is moved to the position where the cut is desired. The engagement and driving device, a swiveling lever with a hook on the end, is activated by means of a voice coil, causing the pen carriage to grab the cutter carriage from its parking position and drive it along the pen carriage guide arm. The paper is cut by the slitting mechanism consisting of the rotating blade and the linear blade.

The rotating blade contains a press-fitted O-ring that is held against the media by the leaf springs. Friction between the O-ring and the media causes the rotating blade to rotate while contacting the linear blade at two points (the leading point is the cutting point).

After the cutting operation, the paper is moved out of the cutter path to avoid contact with the rotating blade. The pen carriage moves back to the left end of the guide arm and the cutter carriage is disengaged, leaving it in the parking position. The pen carriage can then be used for a new plot.

The voice coil that activates the engagement and driving device is the same voice coil that moves the drafting pens up and down.

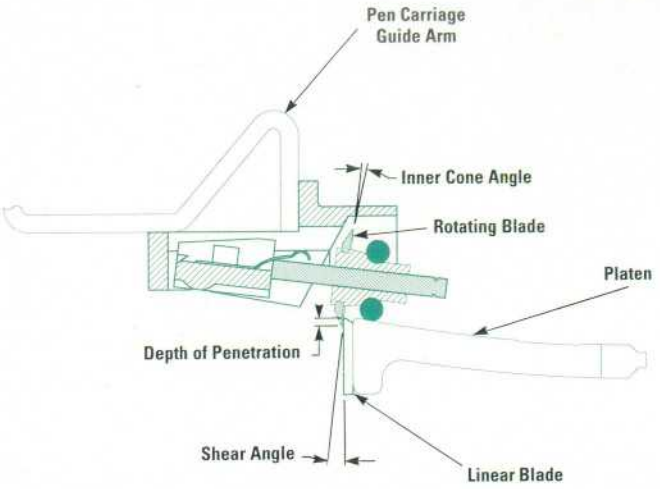


Fig. 2. Design parameter definitions for the cutter, which is a classical rotating and linear blade design.

Slitting Device

As shown in Fig. 3, the slitting device consists of the cutter carriage, the rotating blade assembly, and the linear blade. It uses the same guide arm as the pen carriage, taking advantage of an existing cavity underneath the arm. The cutting speed is fast enough to complete the cut before the cut sheet of media begins its falling motion.

The life of the slitting mechanism depends mainly on the rotating blade because the parts of this blade are subject to the most wear. The blade hub is made of acetal because of its low friction and high wear resistance. Its wear resistance allows it to withstand the high reaction forces against the shaft that are induced by the compression spring. Low friction reduces tangential forces on the media, which could produce buckling and nonstraight cuts.

The geometry of the rotating blade is designed to ensure high cut quality. The side that faces the linear blade is conical so that the contact between the two blades is only at two points. This ensures near-perfect shearing of the media rather than tearing.

For durability, the rotating blade is made of AISI 410 steel tempered to a hardness of 63 Rockwell C. This material also offers high corrosion resistance and reasonable cost. The blade diameter is as large as possible to minimize the number of turns per cut, and the compression spring force is minimized to reduce wear. Since precise sharpening of the blade is not required, no grinding is required in its fabrication, which lowers its cost.

The O-ring is important for obtaining high cut quality because the blade overlap or depth of penetration depends on the diameter of its torus section. It needs to adhere to the surfaces on which it rolls, while avoiding tangential forces on the media. It is made of a rubber that meets these objectives and is resistant to wear caused by the media and media dust.

The O-ring is pressed against the media by the spring-loaded shaft assembly. The friction resulting from this force makes the rotating blade rotate while the cutter translates.

Engagement and Driving Mechanism

Since the DraftMaster plotters can draw along virtually the entire path of the pen carriage, it was necessary to develop an engagement mechanism that would leave the way clear for the plotter to draw. The existing voice-coil mechanism

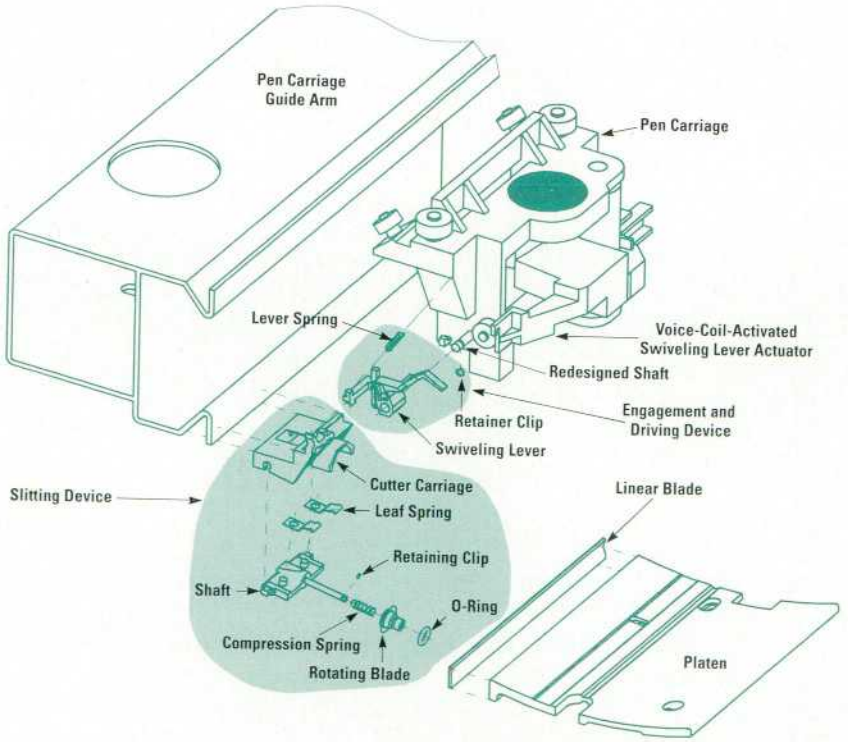


Fig. 3. Exploded view of the automatic cutter, showing the two main parts: the slitting device and the engagement and driving device.

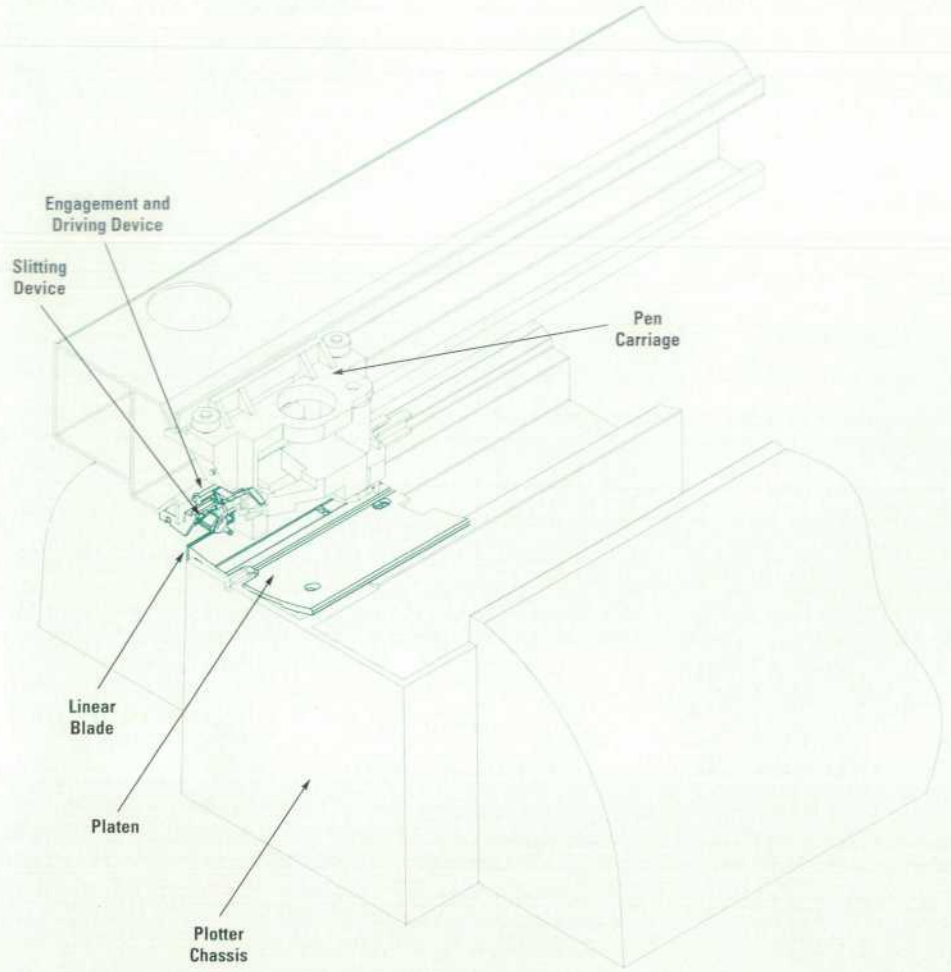


Fig. 4. The parking position of the cutter is at the left end of the pen carriage guide arm.

that moves the pens up and down was easily controllable and turned out to be the best candidate for driving the engagement mechanism. This avoids the need for additional solenoids or motors.

The upper 3 mm of voice-coil travel was not used for any drawing purposes. The HP ME30 three-dimensional mechanical design system was used to find the maximum possible angle of rotation for the rotating engagement lever given just 3 mm of vertical drive motion. Changes in the firmware were made to ensure that the pen carriage does not invade the upper 3 mm of vertical travel during normal plotting.

The pivot axis of the rotating engagement lever is parallel to the horizontal direction of motion of the pen carriage. Thus the inertia forces caused by acceleration of the pen carriage are orthogonal to the lever motion and do not induce any swiveling motion that could disturb the pen carriage and influence drawing quality.

Reliability Testing

The cutter carriage assembly is designed as a consumable part. It has to be replaced when the cut quality becomes unacceptable because of cutter degradation. At the beginning of the project a 5000-cut life was set as a reliability goal for the cutter carriage assembly. This was specified as an MCTR (mean cuts to replacement) greater than 5000 cuts. The MCTR specifies the mean number of cuts a cutter carriage assembly is able to perform before it has to be replaced because of an uncorrectable failure.

According to our user model, a DraftMaster Plus plotter will perform about 67,000 cuts during its 10-year life when used in an environment with a high and continuous workload. Consequently, the reliability goal for the cutter system was set at an MCBF (mean cuts before failure) greater than 67,000 cuts. The MCBF specifies the mean number of cuts the cutter system is able to make before its first failure. The MCBF does not include failures that can be corrected by replacing the cutter carriage assembly.

A nonaccelerated life test was developed to verify the reliability of the cutter system over its lifetime. This test was performed on three pilot-run units with stable parts, avoiding the usual difficulties of prototype testing. The units under test cut HP media continuously at ambient conditions for two months at a rate of 1590 cuts per day per unit. To simulate customer use, three media types were used in the test in the same proportion as an average user: approximately 50% paper, 30% polyester, and 20% vellum. To shorten the test time, strips of media were cut as narrow as possible. Such narrow strips could not fall off on their own, so a set of pressurized air nozzles was installed to blow the strips out of the cutter path.

Definitions and Measurement Procedures for Cut Quality Parameters

Because of the nonexistence of any standard method for measuring the quality of a paper cut, we had to define cut quality parameters, especially those related to the edge finish of the cut, and develop measurement procedures for these parameters before we could determine what the cutter design objectives should be.

According to the literature[1] and our experience, cut quality is usually judged in a rather qualitative fashion, by visually inspecting the fibers that project from the cut paper edge. The less apparent these fibers are, the better the cut quality.

Since the length and density of these fibers are very difficult to quantify, it was necessary to search for other parameters that could represent cut quality as the user sees it and that could also serve our purposes.

It was found that there was a very high correlation between visual cut quality, average edge fiber length, fiber density, and cut waviness. Because cut waviness is reasonably easy to measure, it was selected as one of the main parameters.

Hardcopy media are paper, vellum, and polyester. The cut quality parameters selected define both the quality of the cut edge by itself and the quality of the cut edge integrated in the media. The parameters are:

- Cut edge finish: straightness, cut waviness, perpendicular waviness
- Media geometry: parallelism, perpendicularity, accuracy, edge effect.

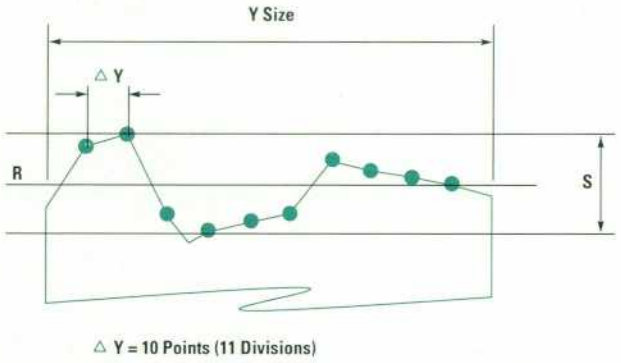


Fig. 1. Definition of straightness S.

Straightness. Straightness (S) is a parameter that measures how straight the edge of the cut is. It is measured as shown in Fig. 1. The location of the edge is measured at ten locations across the cut using a coordinate measuring machine with an optical probe. The regression line R is computed and the maximum positive and negative deviations from R are added to give the straightness S.

Cut Waviness. Waviness (W) is the parameter that measures the undulation of the cut edge along the Y axis. It is measured with a profile projector at 50× magnification in five zones per cut (zone width = 8 mm) as shown in Fig. 2. In each zone, the distance from the deepest valley to the mean crest (where light starts being visible through the fibers) is measured by visual estimation (Fig. 3). Waviness W is the maximum of the five measurements.

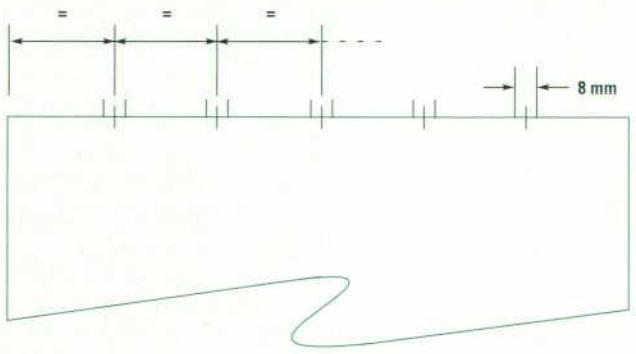


Fig. 2. Measurement points for waviness W and perpendicular waviness Z.

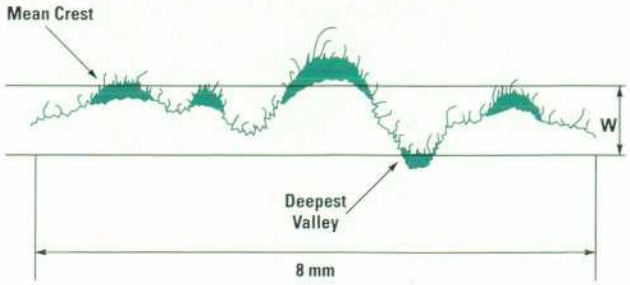


Fig. 3. Definition of waviness W.

Perpendicular Waviness. Perpendicular waviness (Z) is the parameter that measures the undulation of the cut edge along the Z axis (perpendicular to the media plane). It is measured visually using a manual coordinate measuring machine with a suspended knife-edge and a magnifying glass (Fig. 4). Five zones per cut are measured as shown in Fig. 2. For vellum, the characteristic perpendicular waviness is undulatory (Fig. 5); in each zone, the distance between a minimum and an adjacent maximum is measured. For paper, the characteristic perpendicular waviness is not undulatory (Fig. 6) and the total deformation is measured in each zone. No perpendicular waviness is observed for polyester. The perpendicular waviness Z is the maximum of the five measurements.

Parallelism. Parallelism (L) is the parameter that measures how nearly parallel are two consecutive cuts an arbitrary distance apart. It is measured as shown in Fig. 7 using a coordinate measuring machine with an optical probe. First, the regression lines R1 and R2 for the two edges are computed. The parallelism L is the angle between R1 and R2.

Perpendicularity. Perpendicularity (T) is the parameter that measures how perpendicular two consecutive cuts are to a reference line drawn on the paper. A line parallel to the Y axis is used as a reference because of its stability (it depends only on the straightness of the guide arm, while a line at 90 degrees depends on the media tracking, humidity, and other factors). Perpendicularity is measured using a coordinate measuring machine with an optical probe. The regression lines

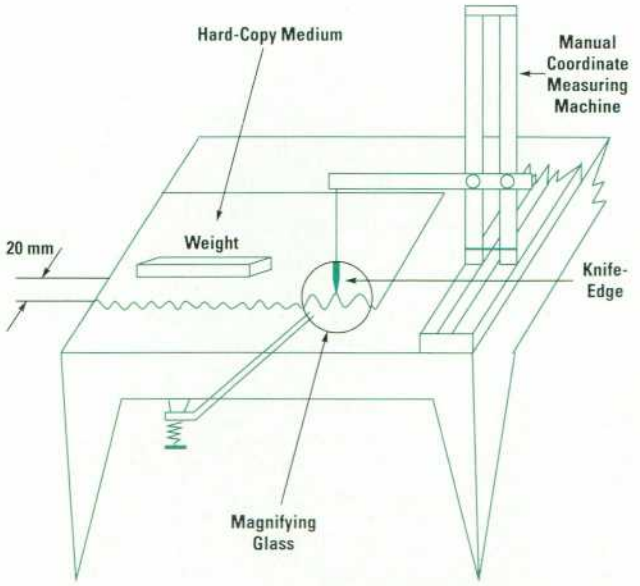


Fig. 4. Measurement setup for perpendicular waviness Z.

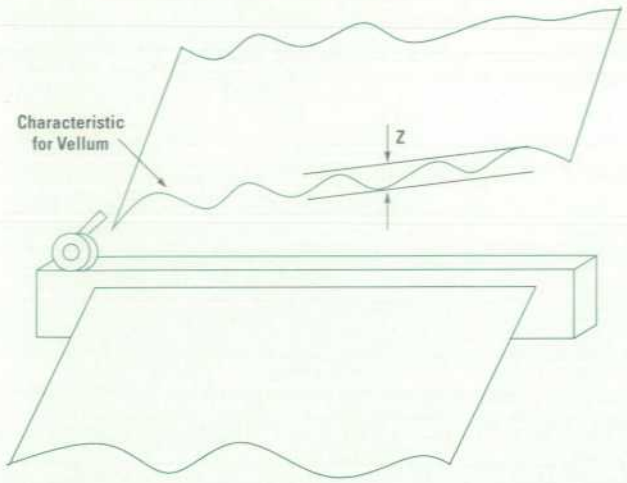


Fig. 5. Definition of perpendicular waviness Z for vellum.

R1 and R2 are obtained as in the parallelism measurement (see Fig. 7). The perpendicularity is the two angles between R1 and the reference line and between R2 and the reference line. Since the reference line is parallel to the Y axis, the ideal perpendicularity is zero degrees.

Accuracy. Accuracy (A) is the parameter that measures the difference between the required and actual sizes of the cut hard-copy media. It is measured using a coordinate measuring machine with an optical probe. One point is defined at each cut edge a distance of 30 mm from the uncut edge of the media at which the cut begins (the offset makes the accuracy measurement independent of edge effect). These points are P1 and P2 in Fig. 7. The accuracy is the difference between the theoretical size and the projected distance on the X axis from P1 to P2.

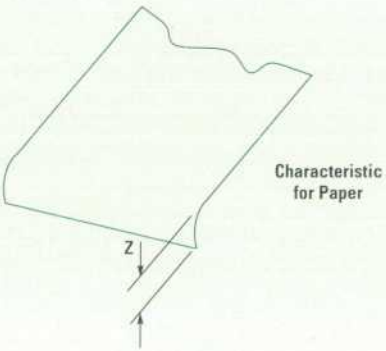


Fig. 6. Definition of perpendicular waviness Z for paper.

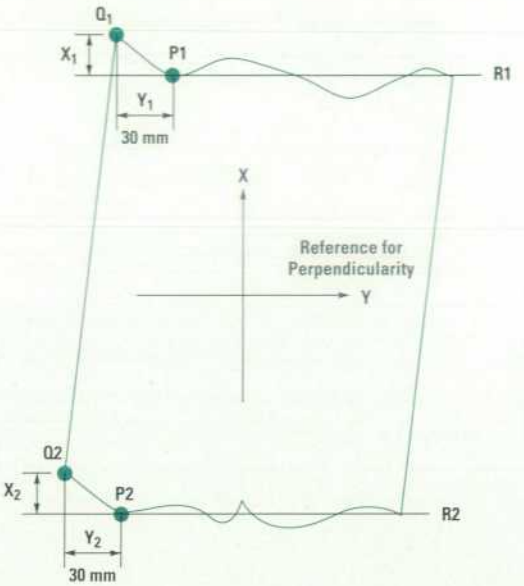


Fig. 7. Definitions for measurements of parallelism L, perpendicularity T, accuracy A, and edge effect Q.

Edge Effect. Edge effect (Q) is the parameter that measures the curve produced at the beginning of the cut. It is measured using a coordinate measuring machine with an optical probe. The starting point of each cut edge is measured. These points are Q1 and Q2 in Fig. 7. The edge effect Q is the maximum of the two projected distances on the X axis between Q1 and P1 and between Q2 and P2.

All measurements are recorded as functions of temperature and relative humidity.

Reference

1. *Wear of Paper Slitting Blades*, Tribology International, December 1980.

Periodically, the cutter carriage assembly, linear blade, pen carriage guide arm, and engagement and driving mechanism were visually inspected for wear. Cut quality degradation was controlled by measuring cut waviness. Cutter carriage assemblies were replaced with new ones when their performance became unacceptable.

At the end of the test each of the three units had accumulated 67,000 cuts. Fourteen kilometers of media had been cut into 200,000 strips. The test did not identify any major problem. Only two risk areas were identified and corrective actions were taken to address them.

Cutter carriage assemblies replaced after 15,000 cuts showed flattened O-rings and some wear on the plastic frame. Cut quality began to degrade when a cutter carriage assembly had performed 10,000 cuts.

On the basis of these results it was concluded that the cutter system meets the MCTR and MCBF specifications.

One of the risk areas we were concerned about was the possible degradation of the coating on the pen carriage guide arm caused by the friction of the cutter carriage. This degradation could result in poor cosmetics and even corrosion under unfavorable environmental conditions.

An accelerated test simulating very intensive use was developed to understand this failure mode. This test consisted of moving the cutter carriage assembly and pen carriage back and forth at high speed along a small span of the pen carriage guide arm without cutting any media. The coating flaked off only in the areas where the cutter carriage assembly and the pen carriage bearing wheels had rolled. No coating degradation appeared in the areas where just one of them had rolled. This test proved that some deterioration of the pen carriage guide arm cosmetics might show up at the half-life of the product when used very intensively. This led to a change of cutter carriage material, which completely eliminated this problem in subsequent tests.

Conclusions

Concurrent development and testing of this system in the development phases, treating it as if it were in production, was key to obtaining a product whose characteristics and performance can satisfy user needs.

This experience taught us that development tools such as QFD are very helpful if used with discretion. It is important to assess the limits within which these tools can be of great value to the definition of the product and beyond which their contribution may be diminished or even become a drag on the development efforts.

The result of this project is a cutting system with the following advantages:

- Independent motors and driving means for the cutter are not required, making it simpler, more reliable, and more economical.
- The cutting mechanism of rotating and linear blades offers simplicity, reliability, and low cost and provides high-quality cuts on different types of media with no need of any mechanism to hold the media. It also provides high durability because it is self-sharpening.
- The cutting system adds little weight to the pen carriage during its normal operation as a drafting device, allowing it to accelerate and stop rapidly and have all of the inherent advantages of a low-inertia design. Thus the same pen carriage can perform both drafting and paper cutting functions while maintaining the original plotter performance.
- The same driving means is used for engaging and disengaging the cutter carriage and for raising and lowering the pens without hampering drawing performance or restricting the paper sizes that can be used.
- The cutting device can be retrofit to plotters manufactured without cutting devices.
- The only maintenance needed consists in replacing the entire slitting device with a new one. This is done when the user observes a degradation in performance or when the plotter warns that it should be replaced (the plotter counts the number of cuts performed). The elastic support system, which keeps the slitting device in equilibrium between the compression and leaf springs, makes it easy (by means of a supplied insertion tool or even directly with the fingers) to separate and raise the rotating blade away from the linear blade to take the cutter out and install a new one.

Acknowledgments

The authors would like to thank Josep Maria Pujol for his contributions to the definition and measurement of the cut quality parameters, Diego Torres, Agusti Comadran, and Joan Uroz for their QFD work, and Felix Ruiz, Joaquim Brugue, Carles Viñas, and Robert Beauchamp for their contributions to the cutter design. Enric Guasch was our consultant on experimental design techniques. Jordi Balderas managed the quality assurance program for the project. Steve Vanvoorhis was project manager.

References

1. *Wear of Paper Slitting Blades*, Tribology International, December 1980.

2. G.E.P. Box, W.G. Hunter, and J.S. Hunter, *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*, John Wiley and Sons, 1978.

Reengineering of a User Interface for a Drafting Plotter

An existing user interface has been successfully reengineered and plotter usability enhanced by selecting, combining, and adapting software prototype techniques and standard software development methodologies.

by Jordi Gonzalez, Jaume Ayats Ardite, and Carles Castellsague Pique

HP's large-format pen plotter family has been evolving for the last ten years, mainly by introducing new models that enhanced the functionality and performance of previous products. While adding more features and providing better performance, new models have usually required more complex and less intuitive user interaction. To facilitate the use of the new functionality of the HP DraftMaster Plus plotter, it was decided to redesign the plotter's user interface.

The reengineering of the user interface was successfully accomplished by applying and adapting a combination of software development methodologies and prototype techniques. The key steps in the evaluation, design, and development of the new user interface were:

- Development of a user interface software prototype for early evaluation and usability enhancement

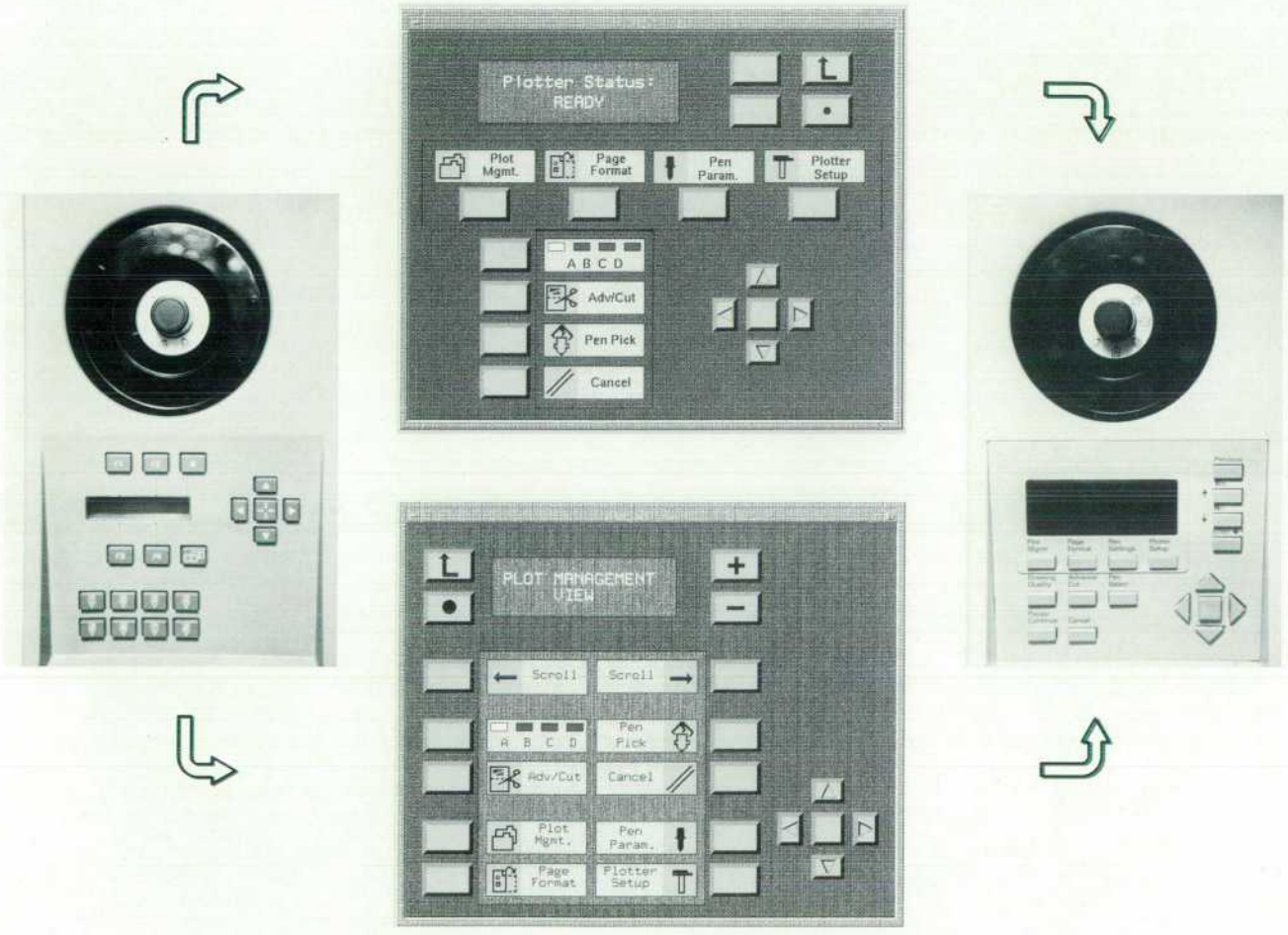


Fig. 1. (left) The old HP DraftMaster plotter front panel. (center) Two of the software prototype options tested. (right) The final HP DraftMaster Plus plotter front panel. In the final design, four keys are reserved for menu navigation and option selection. The most frequently used options are assigned special keys.

- Detailed specification, both graphical and textual, of the user interface menu options, dialogues, and messages
- Design of the user interface for easy localization (translation) of all displayed text
- Use of best practices for design and development: structured analysis, structured design, system testing, design walkthroughs, and code inspections.

User Requirements

Improvement of the HP DraftMaster user interface was identified as a priority as a result of focus groups and customer surveys. The major customer complaint was the readability of the LCD display, especially from different angles. The readability was worst under light conditions that created reflections and shadows. Another area for improvement was the basic user interaction required for menu navigation, function selection, and option setup. Another major limitation was the number of characters allocated to describe each menu option, especially for some languages, such as German and Spanish.

The readability problem was easily addressed by replacing the LCD display with a light-emitting display technology to provide good readability even in poor light conditions. Among the several different display technologies considered, the best combination of cost and features was offered by a vacuum fluorescent display (VFD). To eliminate reflections, the display is covered by dark plastic. The display window has been enlarged to facilitate readability from different angles and greater distances.

To enhance the usability of the front panel, a broader understanding of how users interact with a peripheral was required. Aspects of front-panel design that had to be considered included the layout, type of keys, number of keys, labeling of keys, localization, and aesthetics. Usability aspects included menu selection, option setup, operation feedback, and others.

Our approach to investigating and defining the most appropriate user interface was to use a software prototype tool to build and test different types of front panels.

Rapid Prototyping

The study began with the definition of numerous and highly diverse types of user interfaces. Key people from various disciplines were involved in the definition process: R&D, marketing, quality, product support, and industrial design.

Software prototypes of the different front-panel layouts were generated easily and rapidly using an HP software tool called LogicArchitect. The prototypes allowed people to try different options in a very efficient manner. Very soon the range of options under consideration was narrowed to a small number.

The use of software front-panel prototypes proved to be a very useful and powerful tool with multiple advantages. First, it was useful as a communication tool to describe a particular front-panel alternative. Software prototypes were easily and rapidly sent back and forth between the U.S.A. and Spain through a computer link. Second, as a testing tool, software prototyping allowed the quality department to evaluate concepts early in the investigation phase. This made it possible to design the user interface test suite sooner. The prototype also served as a reference for checking the correctness of the user interface implementation. Third, consensus about which user interface to choose was achieved much sooner with software prototyping, since the advantages and disadvantages of each option were easy to demonstrate. Fourth, the prototype allowed the manual writer to start writing much earlier and helped make the manual more accurate because the writer was able to interact with the prototype. Fifth, the prototype speeded development because the software engineers were able to reuse some of the data structures and texts from the software prototype. Finally, the main advantage was that it was easy to demonstrate the usability of the chosen front-panel option before development.

Fig. 1 shows the old HP DraftMaster front panel, two of the software prototype options tested, and the final front panel.

Risk Assessment

The initial plan was to develop the new user interface in the following pen plotter project. The early consensus on the type of user interface to develop encouraged us to consider advancing its development.

Before committing to the development of the new user interface an estimation of the risk of adding new functionality late in the project was required. This risk assessment was based on estimates of the precision of the current project schedule and the defect removal effort for the project. The conclusion was that the project could be done on schedule, but there would be little margin for error. Therefore, the user interface development had to be done in one cycle with very little time for rework.

The methods used to evaluate the precision of the schedule and the defect removal effort were the keys to a good risk assessment and are explained in more detail in the following sections.

Precision of the Project Schedule. The precision of the project schedule is a measure that can be derived from the project delay over time: the greater the project delay the less accurate the project schedule.

Planned project progress can be measured as the number of planned test cases. A test case is a set of tests done to validate a certain unit of functionality. Actual project progress can be defined as the number of test cases completed up to a given time. The assumption is that a test case is performed as soon as the associated functionality is available for testing. By observing the planned test cases and the actually completed test cases over time one can quantify the project delay and the precision of the project schedule. If a test case finds the functionality under test to be invalid, that test case is considered open until the defects are fixed. Open test cases are not counted as completed. This is because extra time is required to fix the defects, resulting in an extra delay in the project schedule.

The number of planned test cases, the number of test cases actually performed, and the number of completed test cases (actual minus open) can be plotted as functions of time as shown in Fig. 2. Project delay can then be estimated as the difference in time between the date a certain percentage of the test cases are actually completed and the planned completion date. The projection of this difference over time

Fig. 2. Plotting the planned test cases (Plan), those actually executed (Actual), and those for which any defects found have been fixed (Actual minus Open) makes it possible to estimate the precision of the project schedule. The estimated delay for this project was six weeks.

for the DraftMaster Plus project resulted in an estimated delay of 6 weeks.

Rework Effort Estimation. The defect removal effort at the point when we were about to undertake the development of the new user interface was estimated as:

Rework effort = (Number of defects) × (Average time to fix a defect).

The average time to fix a defect varies from person to person. It is also related to the laboratory's software development environment. An estimate can be based on the laboratory's defect history by computing the average time it has taken to fix a defect in the past. However, the defect tracking records in the short history of our lab were not sufficient to give an accurate average, so we took the approach of doing our best team estimation. This resulted in an estimate of 2.5 hours per defect, not including test time.

Several statistical techniques are available for forecasting the number of defects. The one that worked best for this project is based on the defect density per test case in each of the system regression tests. The same metric has been used as a project progress measure and as a software stabilization measure. The key benefit of this technique is that it gives a good estimate of the number of defects early in the project.

At the time we assessed the number of defects there were parts of the code that were undergoing the third regression, while others were still in the first. We found that the number of defects per test case and per regression was very close to a straight line, except for the first regression, where the straight-line pattern appeared after 20% of the test cases were executed (Fig. 3). Therefore, we established three conditions for estimating the number of defects:

- At least 20% of the first regression is executed.
- At least 10% of the second regression is executed.
- At least 5 regressions are estimated to be required, based on past project experience.

To evaluate the total number of defects we extrapolated the defects to be found in the first regression at 100% completion, and the same for the second. The the ratio of the numbers of defects in the first and second regressions was calculated. Values for the third, fourth, and fifth regressions could then

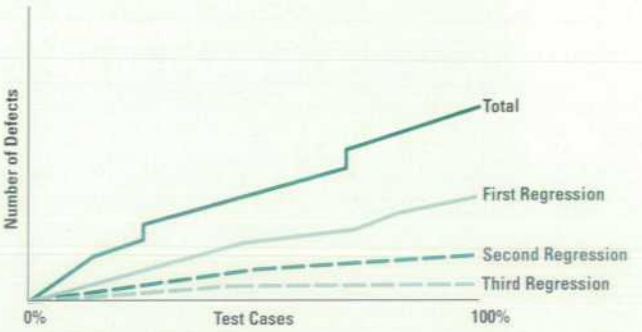


Fig. 3. Number of defects found as a function of test cases executed and the number of regressions. The top curve shows the total defects found for the project.

be determined. The result was that at 20% execution of the first regression, we estimated 96 defects. The actual value at the end of the project was 110 defects for the entire project.

On the basis of the estimated 96 defects and 2.5 hours to fix each defect, the required rework effort was estimated to be 1.5 engineer-months.

Using these techniques we were able also to forecast the number of defects that would be found in the next month. To do this we used our regression test plan and the straight-line relationship between test cases, regressions, and the number of defects. The results are shown in Fig. 4.

Graphical and Textual Specification

As a software good practice, we decided to do as much formal specification as possible of all new software functionality before the design and code development phases. The challenge was to describe the general operation of the user interface formally. To achieve this objective a special graphical syntax, combined with text, was designed as an easy and intuitive way to describe a generic menu-driven user interface. The main objective was to have a working document, the *DraftMaster Plus User Interface Internal Reference Specification* (IRS), which would be easy to review and update for all the different functional areas involved in the user interface development: marketing, quality, R&D, product support, industrial design, and manual writing. The user interface IRS document had to describe the static aspects of the user interface (menu hierarchy, list of options, buttons, etc.) as well as the dynamic and interactive aspects (dialogues, event sequences, option setups).

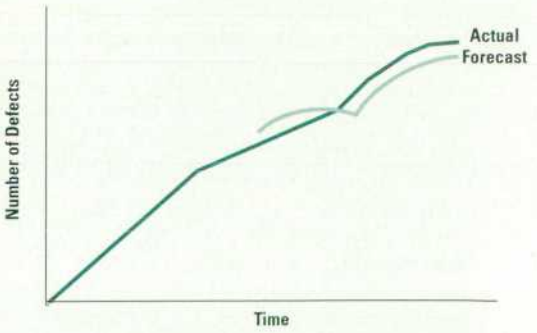


Fig. 4. Number of defects found versus time.

The user interface physical display is one of the basic objects used to describe the user interface. It is represented graphically as a rectangle with shadowed edges. A front-panel button is represented by drawing an outline of its real shape with an icon on top of it for identification. Another element used in the user interface description is the screen, defined as the text being displayed at a particular instant in the user interface display. A screen is described graphically by a rectangular display symbol with the particular text in it. Particular menu options, messages, and option setup screens are graphically represented as screen elements.

The description of a series of user interactions is captured as a dialogue. A dialogue is a sequence of events mainly driven by the user—for example, the setup of an option such as baud rate or the number of copies. To set up a particular menu option, the user goes through a sequence of screens, making selections, setting values, and pressing buttons. Dialogues are described graphically as screens connected by arrows that define a time sequence. In a dialogue, the transition from one screen to another can be triggered by the user's pressing a particular key. This is captured graphically as two screens connected by arrows, with the button graph in between. An example of such a description from the user interface IRS is shown in Fig. 5.

There is a limit on the amount of detail this graphic representation can describe effectively. Details and complementary information are better described as text.

This notation allowed an excellent review of the user interface requirements before the design phase, and facilitated a broad consensus among the different functional areas. The detailed specifications in the user interface IRS also proved to be very useful for system test development. All of the menus and options were grouped into 25 equivalence classes according to the number and kind of buttons to be pressed to reach them, and one menu or option of each class was chosen and fully tested. Status and error messages were tested in the same way, grouping them in classes according to priority, and testing all possible combinations of messages belonging to different classes.

Misunderstanding or incompleteness of an IRS results in software defects. As a result, code and tests have to be reworked. In this project, the detailed specifications minimized this effect and saved a significant amount of time.

Analysis and Design Methodology

Working on a tight schedule, there is little time to redesign previous designs. Things have to be done right the first time to minimize the time to market. With this in mind, we decided to use structured analysis and design practices. The structured analysis, based on data flow diagrams,[1] allowed a team of software engineers to work efficiently. The goal was to minimize the interaction among the engineers while limiting the the analysis to a reasonable level of detail.

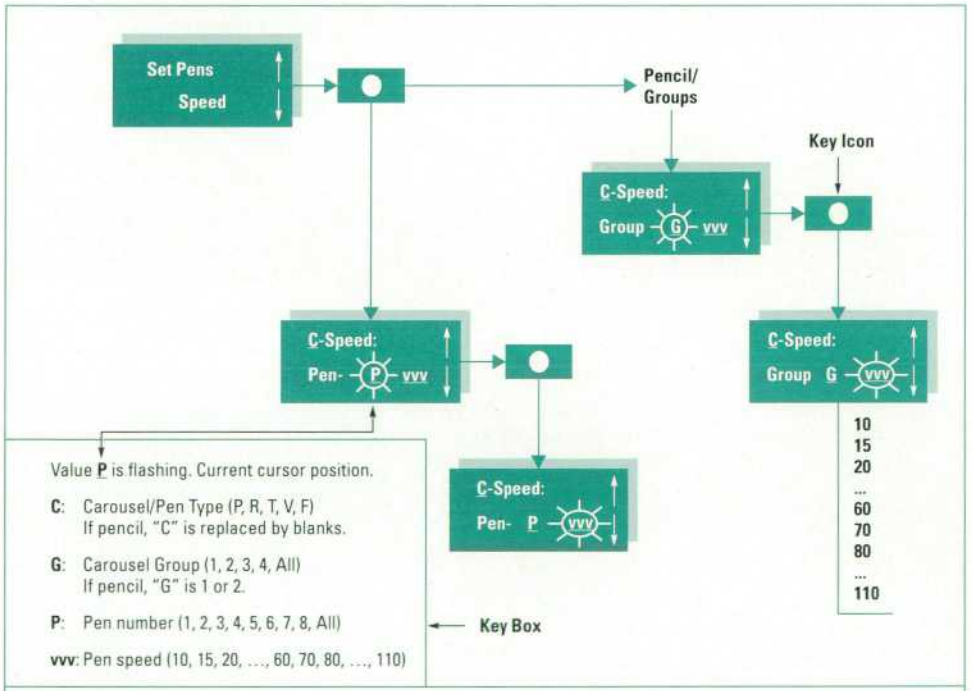


Fig 5. Specification example, with the full graphical and textual description, of the pen speed menu and related dialogues. The combination of graphics, text, and tables proved to be a simple and effective way to describe the dynamic behavior of the user interface menus and dialogues. The display, with the current text message, is represented as a rectangle with shadowed edges. A user event, such as the pressing of a key, is represented by the key icon. Arrows connect the screen sequence to the user action that triggered the event. Special display effects, such as flashing characters or cursor position, are also graphically represented. Lists of values for specific options are represented by variable names with possible values fully detailed in a key box.

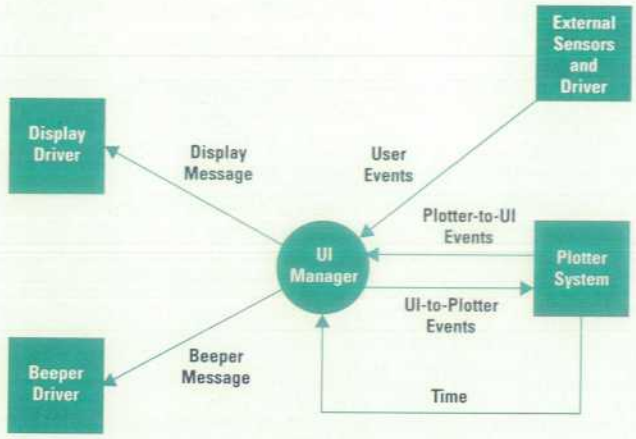


Fig. 6. Context diagram of the user interface manager. It communicates with the plotter system through events. Events are either messages to display, commands to execute, requests for response data, or notifications of plotter state changes. The user interface manager interacts with the outside world through buttons and sensors, a display for menus and messages, and an alarm beeper.

To get a high-level picture of the user interface we drew a context diagram and a data flow diagram. The context diagram (Fig. 6) shows the interaction of the user interface with the plotter system through events and with the outside world through sensors and buttons, the display, and an alarm beeper.

The plotter system is the plotter itself and includes many submodules that can interact with the user interface: I/O, plot management, graphics engine, media handler, vector manager, pen handler, and so on. These modules interact with the user interface manager to notify or warn the user through messages or alarms, show the status of the plotter, show current menu values, set new parameters, and execute user interface commands.

The data flow diagram of the user interface manager (Fig. 7) shows the main procedures in this module: user interface event manager, message handler, menu handler, and display manager. The user interface event manager takes care of all kinds of events, including plotter system, keyboard, and timing.

The menu handler navigates along the menu tree and executes the menus at the tree leaves. It receives the menu events, including keyboard events (a button pressed by the user), data supplied from the plotter, timer events (like an inactivity timeout), and special menus triggered by the plotter from the graphics engine. The menu tree is stored in the menus data structure. The root is the status menu, which shows plotter status, such as ready, busy, paused, paper out, and so on. From the status menu, six main menus are available, four of them directly available from buttons. These have several submenus. At the tree leaves are the menu dialogues—customized menus that can show, toggle, and set parameters or simply trigger with or without confirmation of user interface commands. The user interface allows the nesting over the menu tree of special menus triggered from the graphics engine, such as the digitize menu, and several direct menus for easy access to the cancel, select pen, pause, and other often-used menus. Fig. 1 shows the user interface key layout.

The message handler displays and removes messages. The messages come from the user interface event manager (from the plotter) or are internally generated by the menu handler (to notify or warn the user). The messages data structure holds all the messages, classified by class and priority. Messages of different classes and priorities can be nested. Each class has an associated menu behavior. Informative repetitive messages are displayed until any key is pressed, and are redisplayed after an inactivity timeout. Error messages from the graphics, I/O, and other submodules are displayed until any key is pressed. Informative timeout messages are shown for a few seconds to inform or warn the user. User action request messages require the user to press the **Enter** button to confirm. Progress status messages

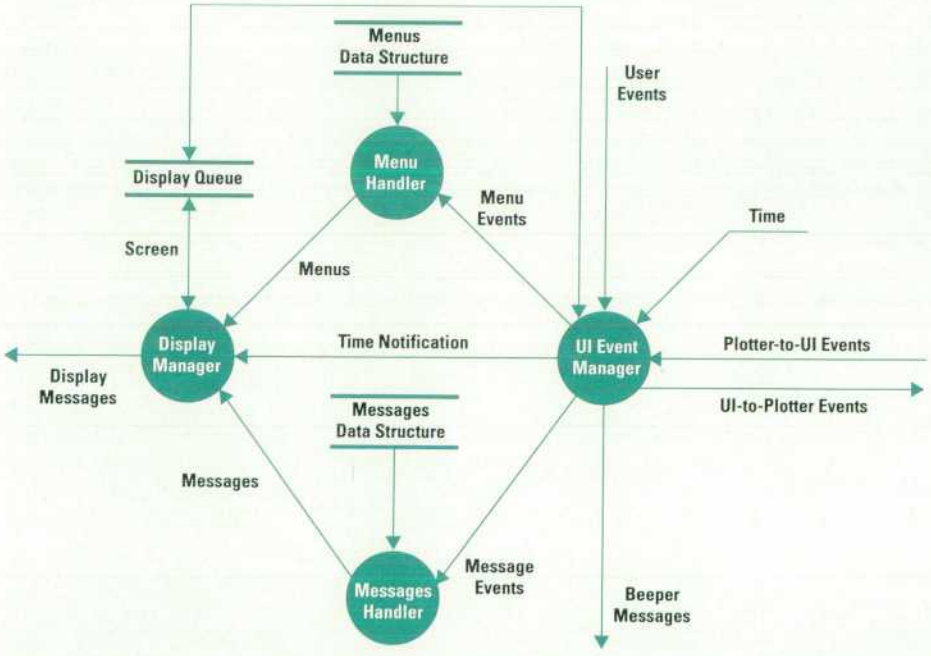


Fig. 7. Data flow diagram of the user interface manager. There are four procedures. The user interface event manager looks for events from the plotter system, the keyboard, or the sensors. It also watches for timing events. The message handler manages messages coming from the plotter or from the user interface on a priority basis. The menu handler navigate along the menu tree and executes the menu dialogues at the menu leaves. The display manager manages the display queue and displays the menus and messages on a priority basis.

are shown while a critical action is being performed. Critical error messages indicate failures.

The display manager controls the vacuum fluorescent display. It manages the display queue and displays the menu choices. It also sets some of the display options, such as flashing, character set, and brightness.

Data Structures

The next step in the user interface design was the design of the data structures, which are mainly composed of menus, messages, and words. Static data structures were specially designed for easy localization to support the six targeted languages: English, French, German, Spanish, Italian, and Japanese. From experience, we clearly understood the need for an automated procedure to allow further refinement and correction of all the texts and their translations.

Fig. 8 shows the building process for the menus and messages data structures. Fig. 9 shows the text data structures.

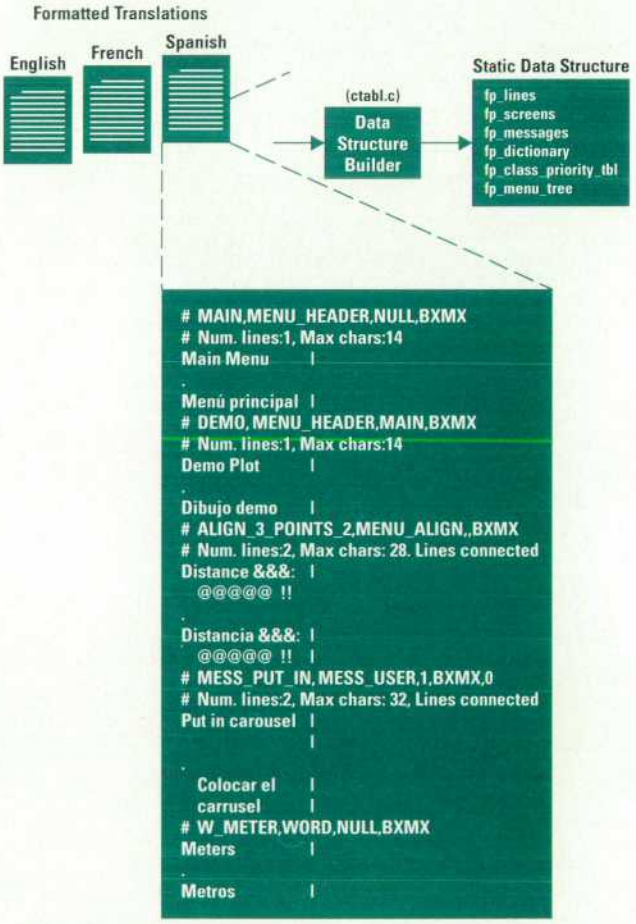


Fig. 8. The building process for the menus and messages data structures. The input files are six files with the six localized menu screens, messages, and words. The program ctab filters and merges these files. The output is a C include file with the needed data structures. In the input files, there are menu screens that hold the menu displays used in menu tree navigation, formatted menu screens for the menu dialogues, message screens specifying message classes and priorities, and localized words for the dictionary to be formatted in some menu screens.

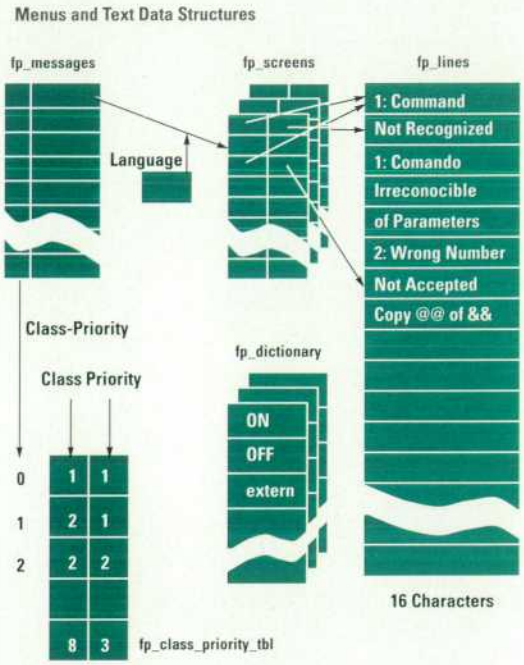


Fig. 9. Text data structures. Messages are indexed by their identifiers. Each entry contains an index to a class priority table to select the appropriate behavior. Another index points to the screen table. For the selected language, there are two indexes to the first and second lines of the message forming a screen. A screen can have parameter fields, specified by the symbols @, &, !, $. The size of a parameter field is specified by repeating the same symbol. When the message is formatted, the parameter values or localized words from the dictionary are substituted for the parameter fields. A similar process is applied to the menus, which are stored in the fp_menu_tree data structure, which has indexes to the fp_screens structure.

In each case, the English version was built first. Some parameters were specified, such as name, type (menu, message, word), menu linkage (for menus), and message class and priority (for messages). There was a field for the English text and another for adding the translation to a single language. This file was sent to the five translators, who had to fill in the translations field.

We created a tool called the data structure builder for automatically building all the data structures from the six files. The tool produces a C include file with all the menu and message structures. When changes had to be made, we just edited the six source files and reran the data structure builder again.

The last step in the design phase was a walkthrough to detect design defects. This walkthrough proved to be very useful. It showed some inconsistencies, but primarily it highlighted a major implementation issue: how to implement the menu dialogues. This issue is covered in the next section.

Implementation

When it came to implementation, there were some inherited constraints that made the reengineering of the user interface more difficult. Most of the code had been written more than ten years earlier when low-level languages were more the rule than the exception. The software system architecture

was interrupt-driven and had no operating system for task scheduling and dispatching. The new user interface was constrained to exist in this environment. The plotter system runs as a background process and the user interface is triggered at an interrupt level by a periodic interrupt event. The user interface manager and the plotter system communicate through queues as shown in Fig. 10.

The plotter system was too hard to model, so we decided instead to have a clear, well-specified interface to the user interface queues and then surgically remove the old user interface references and add the new ones.

The user interface manager is scheduled through a periodic interrupt. It is basically a state machine with states, input events, and outputs. When activated, it checks for an event (button press, timing, or plotter) and handles the menus or messages depending on its state and the event.

Another limitation, because of the lack of an operating system, is that the dialogues (menu leaves) were designed as independent tasks. They are called upon entering a dialogue menu and they end when the user operation is completed. They are implemented in the user interface manager at the interrupt level. Therefore, we had to design a small operating system subset just to put the menu dialogues to sleep when waiting for events and wake them up at the next interrupt. We supplied a local stack so the user interface manager would be better isolated from the rest of the system and could more freely build its own display screens.

The last important issue was how to support the design team of three engineers so that they could work efficiently on the same set of modules and within the schedule constraints. We succeeded thanks to a well-established software management control system for supporting parallel development. This system is based on RCS and includes some scripts to better automate the generation of code releases. It also supported our intensive use of code merges. The data structure builder already mentioned made it easy to update the menus and messages.

Results

The HP DraftMaster Plus user interface reengineering project met its planned introduction date and its main project objectives. The number of software defects was very low compared to previous projects in this lab.

Fig. 10. Interaction between the plotter system and the user interface manager. The user interface can request the plotter system to schedule the execution of a particular function or procedure. The user interface manager can also send a request for data maintained by the plotter system. The plotter system interacts with the user interface manager by means of messages and update events.

The project took 6.5 months. One month was spent in selection of the best proposal. Three months were invested in the definition of the IRS and the system tests—a measure of the amount of specification effort. The final 2.5 months were spent on coding and testing.

The project complexity, measured by the amount of new C code written, was 15 KNCSS (thousands of noncomment source statements).

The number of defects related to the front panel found before introduction was 37. To determine the quality of the specification and coding activity, the defects were classified as specification defects (13%), coding and design defects (84%), or hardware defects (3%). The specification category includes such defects as misunderstandings between team members, side effects of specifications, incomplete specifications, and wrong specifications. The low percentage of these defects is a clear improvement over previous projects, indicating that the specifications were clear enough that every team member was able to understand the expected product behavior.

Defects found in new and old modified code were:

- New code: 38%
- Old code: 62%.

The quantity of code written in the old assembly language was much smaller than the quantity of new code, proving again that the modification of old patched code is much harder than writing brand new code. The prerelease defect density in the new C code was 2.5 defects/KNCSS. This is a significant improvement over previous experience in our lab. At introduction, there were no open defects in the user interface.

Acknowledgments

We would like to recognize the contributions of Juan Jose Gimenez in implementing and supporting rapid software prototypes of the different user interface concepts, Ken Larsen who contributed multiple comments and ideas on the design specification documentation, Herb Sarnoff and Josep Giralt for their significant role in turning the user interface project into a successful reality, and Carles Muntada for his contribution on the user interface project and the help he provided on the revision of this article. In addition, we appreciate the efforts of the engineers at the HP San Diego Technical Graphics Division who helped in the project. Special thanks to the whole project team that made the DraftMaster Plus a real product in the market.

References

1. T. DeMarco, *Structured Analysis and System Specification*, Yourdon Press, 1978.

Bibliography

1. D. Hatley and I. Pirbhai, *Strategies for Real-Time System Specification*, Dorset House Publishing, 1987.
2. M. Page-Jones, *The Practical Guide to Structured Systems Design*, Yourdon Press, 1980.
3. T. DeMarco, *Controlling Software Projects*, Yourdon Press, 1982.
4. M. Ould, *Strategies for Software Engineering: The Management of Risk and Quality*, John Wiley & Sons (UK), 1990.
5. D. Youll, *Making Software Development Visible*, John Wiley Series, 1990.

A Multiprocessor HP-UX Operating System for HP 9000 Computers

The system supports up to four processors in the HP 9000 Model 870 computer, significantly increasing online transaction processing (OLTP) performance without degrading uniprocessor performance.

by Douglas V. Larson and Kyle A. Polychronis

The kernel of the HP-UX* operating system has been modified to support PA-RISC multiple-processor systems in a symmetrical manner (PA-RISC is Hewlett-Packard's reduced instruction set computer architecture). In a symmetrical multiprocessor system, any processor can run any task—user or kernel—on the system. The first release of this product, HP-UX 8.06, supports up to four processors with the HP 9000 Model 870 hardware.† Although the current system is implemented for up to four processors, there is no fundamental design limitation on the number of processors that can be supported.

The Model 870 multiprocessor HP-UX systems represent a significant technical milestone for HP for several reasons. First, the hardware, which is used by both the HP 9000 Model 870 and the HP 3000 Series 980 computer systems, is the first multiple-processor implementation of HP's PA-RISC architecture. Multiprocessor systems are an important technical direction for the industry because they offer the most cost-effective means of improving the performance of a given platform. Adding a processor board to a system is an extremely effective way of adding power.

Second, the Model 870 multiprocessor system raises HP-UX online transaction processing (OLTP) performance to a new level. In fact, at the time of the Model 870's introduction, it had the best performance in the industry for UNIX*-system computers running the TPC-A benchmark (Transaction Processing Performance Council Benchmark A). We had to solve a myriad of difficult technical problems at both the operating system level and the database manager level to achieve this performance.

Finally, we advanced the state of the art of multiprocessor UNIX-system computers by effectively harnessing the power of multiple high-performance RISC processors. Previous UNIX-system multiprocessor machines have featured relatively low-powered microprocessors. Using a processor with the capacity of the Model 870 with its fast, large caches is a fundamentally harder multiprocessor problem; adding a single Model 870 processor is equivalent to adding half a dozen Intel 80386s, for example.

PA-RISC Multiprocessor Hardware

The PA-RISC architecture includes specification for multiprocessor systems with a tightly-coupled shared main memory model. Each processor has its own cache (see Fig. 1). Perhaps the most important characteristic of the PA-RISC multiprocessor design for software is the automatic maintenance of consistency in the system caches. Although cache coherency is transparent to software, the costs of cache coherency may be significant. Another PA-RISC multiprocessor concept is that of a *monarch* processor. At power-up, the processors arbitrate to determine a monarch, which subsequently performs the boot process.

In a Model 870 multiprocessor system, up to four Model 870 CPU boards can be inserted into the chassis (see Fig. 2), where they interface to the system memory bus (SMB). From the SMB, a bus converter connects to the MidBus and the HP CIO channel adapters for I/O cards (see Fig. 1). All processor clocks and the SMB clock run asynchronously with respect to each other. This necessitates software resynchronization to provide a consistent system-wide clock to the software.

As mentioned above, cache consistency is maintained automatically by hardware in the PA-RISC multiprocessor architecture. However, in the Model 870 multiprocessor system the overhead to maintain cache consistency is high. For example, a cache miss on one processor when the same cache line is dirty in another processor's cache results in a worst-case (typically 170-cycle) miss penalty, which is far greater than the typically 70-cycle penalty for a uniprocessor cache

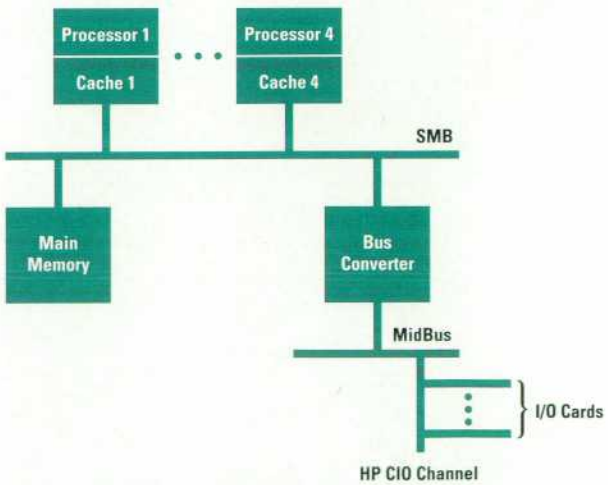


Fig. 1. HP 9000 Model 870 multiprocessor hardware.

† After this article was written, a new multiprocessor system was released. See page 58 for more about HP-UX 9.0 on the HP 9000 Model 890 computer.

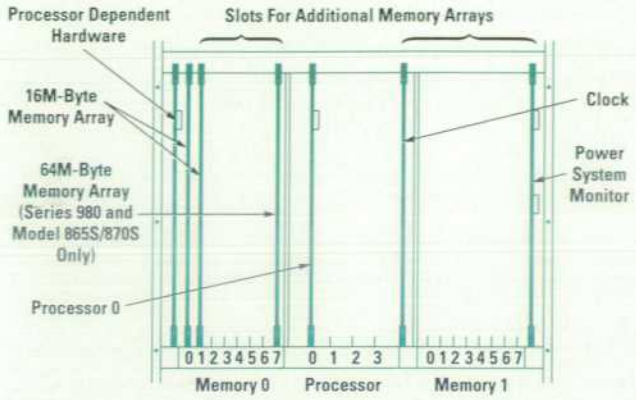


Fig. 2. Up to four processor cards can be installed in the HP 9000 Model 870 computer.

miss. This condition is exacerbated when data is shared between processors on a Model 870 multiprocessor system. The cache miss rate can go up dramatically, resulting in a dramatic decrease in the system's throughput. Working to minimize cache effects has dominated the Model 870 multiprocessor tuning effort.

Another challenging situation was the speed of the processor relative to the SMB memory bandwidth available in the Model 870 chassis. With the Model 870, we placed an extremely powerful 1990-vintage processor into a chassis that was designed in 1983.

Implementation Strategy

Our multiprocessor implementation of HP-UX 8.06 needed to meet the following requirements:

- Support symmetrical multiprocessor execution for up to four processors.
- Be transparent to user-level code (source and object).
- Provide industry-leading performance running the industry-standard TPC-A, TPC-B, and multiprocessor SPECmark benchmarks.
- Support all features of HP-UX 8.0 (except C2 security and diskless clusters, which would have added significantly to the schedule).
- Run the same kernel on uniprocessor and multiprocessor systems, but have negligible overhead on uniprocessors.

The HP-UX operating system is multitasking and supports running numerous processes at any one time. In a uniprocessor system, only one process is executing at any given moment, and the consistency of HP-UX kernel global data structures is ensured because processes running in the kernel are allowed either to run to completion or to run until they voluntarily give up the processor. In either case, the global data structures are consistent at the time another process is scheduled, so there is no need to use semaphores or other concurrency control mechanisms that are seen on other multitasking operating systems. (The only explicit data structure protection is between the kernel and interrupt service routines.)

In a multiprocessor system, however, this uniprocessor assumption is shattered. For adequate multiprocessor system performance, it is essential that multiple processes on different processors execute simultaneously in the kernel. In this multiprocessor scenario, kernel data structures are being updated simultaneously with unpredictable interleaving of updates, which inevitably leads to kernel data structure corruption.

The single greatest technical problem in adapting HP-UX for multiprocessor operation was adding the kernel data structure protection needed to allow simultaneous execution of processors in the kernel. This was achieved by adding semaphore and spinlock primitives to HP-UX, and using these primitives to synchronize accesses to global data structures. Every module in HP-UX was affected by these changes.

Another area of the kernel requiring significant modifications was the scheduler. Instead of just scheduling for a single processor, the scheduler must now manage the computing resources of multiple processors. Also, low-level operating system code such as TLB miss handlers and power-fail recovery procedures required extensive modification for multiprocessor operation.

Once the multiprocessor system was operating, a tremendous effort was required to tune the system to meet our performance objectives. In our view, the key challenge of multiprocessor system performance is tuning the system for the characteristics of the target hardware. Because of the numerous combinations of processor speeds, bus speeds, and cache configurations that are possible in multiprocessor systems, it is a virtually impossible task to develop a multiprocessor kernel that will perform well for everybody without a significant amount of tailoring.

Prototype Refinement

Our development strategy was to start with a coarsely semaphored prototype multiprocessor HP-UX system and successively refine it to address observed performance bottlenecks. For example, concurrency could be added where needed to relieve spinlock and semaphore contention.

Our starting point was a single-semaphore system, which essentially emulates the uniprocessor situation by only allowing one processor in the kernel at a time. This was useful for bringing up early multiprocessor prototype hardware, but has obvious performance limitations.

Our first concurrent multiprocessor kernel involved only four "empire" semaphores (file system, virtual memory, process management, and I/O), together with a handful of spinlocks. While the semaphoring here was very coarse, it provided the basis for measurement, analysis, and further rounds of tuning.

Our tuning efforts since the initial four-empire system have led to a significant division of the virtual memory empire, with a lesser division of the file system empire, to add more concurrency to the system. However, we found that the bulk of our tuning cycles were devoted to reducing overhead resulting from cache effects on the Model 870 PA-RISC multiprocessor hardware.

The conventional wisdom for multiprocessor systems has been to add as much concurrency as possible up front (fine-grained semaphoring) and use this as the basis for tuning. This approach would have been disastrous on the Model 870 multiprocessor hardware because of the high cost of locking resulting from cache effects. This would have led to a high level of rework to address the real performance problems.

Next-Generation Multiprocessor HP-UX

The accompanying article describes the first-generation HP-UX multiprocessor system, which consists of the HP-UX 8.06 operating system running on the HP 9000 Model 870 computer hardware. A new multiprocessor system, consisting of the HP-UX 9.0 operating system on the HP 9000 Model 890 computer hardware, was released after this article was written. Hardware and software advances contribute to a substantially higher level of online transaction processing (OLTP) performance for this new system.

The Model 890 hardware incorporates new chassis and bus designs that address the limitations mentioned in the accompanying article. The main processor bus is dramatically faster and the cache coherency circuitry has been improved. The Model 890 also uses a higher clock rate.

The operating system software has been improved through increased parallelism in the I/O subsystem and additional OLTP performance tuning. In TPC-A benchmarking experiments, a more efficient client-server configuration is being used instead of the monolithic configuration used for the Model 870 measurements.

The result of these improvements is a TPC-A rating of 578 transactions per second for a four-processor HP 9000 Model 890 computer system in a client/server configuration, compared to 173 transactions per second for the four-processor Model 870 in a monolithic configuration.

Successive refinement of prototypes including adding concurrency where needed is the most pragmatic approach to retrofitting an originally uniprocessor system such as HP-UX for multiprocessor operation. Given the complicated interactions involved with HP-UX running on a multiprocessor system (cache effects, semaphore contention, etc.), it was impossible to predict precisely what modifications were needed for best system performance; experimentation with the multiprocessor technology was essential.

Concurrency Control Primitives

The heart of concurrency control within our multiprocessor system is the spinlock. HP-UX kernel primitives have been added to obtain and release spinlocks. If a processor attempts to obtain a spinlock that is held by another processor, it will busy-wait until the lock becomes available.†

Spinlocks are used to control access to data structures that will be held for a relatively short period of time. In such cases, a blocking semaphore doesn't make sense because the overhead of a context switch is likely to be longer than the time a processor will need to busy-wait for the lock. Spinlocks must be used during interrupt service routines, when context switching is impossible. Spinlocks are also at the heart of our blocking semaphore calls to control access to the semaphore data structures.

The conventional method for supporting spinlocks is to take advantage of a test-and-set instruction in the hardware. In PA-RISC, this is the load and clear word instruction (ldcw).††

† Spinlocks are at the core of HP-UX concurrency control. Procedure calls to spinlock routines specify the lock to be acquired as an argument. If the lock is free, the spinlock routine acquires the lock and marks it as busy. If the lock is already busy (that is, another process or processor holds the lock), the routine will busy-wait until the lock becomes available. The busy-wait is a program loop in which the lock is repeatedly checked until it becomes free. An important attribute of a spinlock routine is that the action of checking a lock and marking it busy is atomic—no other process or processor can acquire a lock after its availability has been checked and before it is marked busy.

†† In this paper, ldcw is used as an abbreviation to refer to the ldcws and ldcwx PA-RISC instructions.

However, because of cache effects associated with ldcw on the Model 870, the cost of an ldcw instruction is extremely high, and we realized a significant multiprocessor performance improvement (approximately 20%) by using an algorithm relying only on loads and stores for mutual exclusion.

Blocking semaphores are used to control access to regions of code that are associated with a set of data structures. With a blocking semaphore, a processor attempting to acquire a semaphore already held by another processor will put its current process to sleep and switch to another task. The assumption is that the expected time to busy-wait for the lock will be much greater than the overhead of a process switch.

There are three types of blocking semaphores:

- Alpha semaphores. These semaphores are relinquished when a process goes to sleep, so the data structures protected must be consistent whenever sleep is called.
- Beta semaphores. These semaphores are retained while a process sleeps.
- Synchronization semaphores. These are used to signal events rather than protect data structures.

Fig. 3 shows the decision process for choosing the appropriate protection mechanism for kernel data structures.

Race Condition and Deadlock Avoidance

Converting a uniprocessor operating system to a multiprocessor operating system adds two new classes of software problems: interprocessor race conditions and interprocessor deadlock conditions. Also, existing intraprocessor race and deadlock conditions become much more likely to occur.

A race condition occurs whenever data that should be protected by a lock (a spinlock or semaphore) is accessed without the appropriate lock being held. The thing that makes race conditions hard to notice is that almost every time code with race conditions is executed, it works with no problem. However, occasionally it will fail, and it usually fails in a drastic and difficult-to-diagnose way (e.g., a system crash).

To attack the problem of race conditions we developed a tool that we called SDTA (semaphore data trap analysis). Fundamentally, we gave the tool (which was built into the

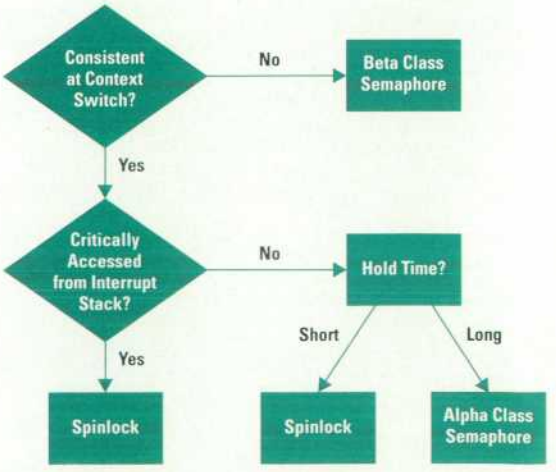


Fig. 3. Decision process for choosing protection mechanisms for kernel data structures.

kernel as a driver) a list of data structures and the corresponding locks protecting each of these structures. Then we used a feature of the PA-RISC architecture that allowed us to trap whenever one of these data structures was accessed. Upon trapping, SDTA tested to see if the appropriate lock was held, and if not, an error message was logged. In a continually changing development environment such as ours, with many people working on the code, this is a strong regression testing tool.

Deadlock conditions occur whenever it is possible for each of two processors to be waiting for the other processor to release a lock so that it can acquire it. In this state they will wait indefinitely for each other (see Fig. 4). More complicated cases with three or more processes are also possible. We chose to solve this problem by instituting a simple, well-known deadlock avoidance algorithm. This algorithm requires that all the locks in the kernel be taken in a particular order. We implemented this by giving each lock an *order* (an integer), and then enforcing the rule by testing within the semaphore code to ensure that whenever a semaphore was taken, no higher-order semaphore was already held. By enforcing the acquisition of locks in lock order, the cycles that lead to deadlock can never occur. This enforcement is done through a series of assertions within the semaphore code. Assertions in the code are statements of the form:

```
ASSERT(<condition that should be true>,"statement of problem");
```

The programmer decides what conditions must always be true at a particular point in the code and puts the assertion in. If the condition is violated then an error condition is noted. Assertions can be turned off at compile time, and because of their impact on performance we do not ship the product with the assertions turned on. Testing with the assertions turned on and compiled allows us to detect problems early. Because assertions affect timing and could potentially cause other problems, we also test with assertions turned off. In HP-UX 8.06 there are well over 1000 assertions, over 200 of which are multiprocessor-specific. Different sets of assertions can be turned on and off at will.

Processor Scheduling

In our early multiprocessor prototypes, a single run queue was maintained, and the highest-priority process was assigned to the first processor available. This is the spirit of a symmetrical multiprocessor implementation: any task can execute on any processor. However, we soon recognized that cache effects made the cost of migrating a process between processors significant. A process builds up a cache context on a processor, and if the process is migrated, that context needs to be rebuilt completely on another processor. The situation is worse if dirty cache lines are associated with the original processor when a process migrates because this causes the worst-case cache miss overhead when those lines are reaccessed by the new processor.

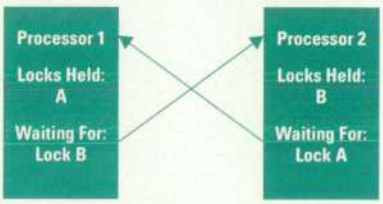


Fig. 4. Processor deadlock.

To address this situation, we implemented a new design with a run queue for each processor, and we added heuristics to control the migration of processes between processors to minimize cache effects from process migration. Controlled migration is necessary to allow load balancing, but processes should stay on the same processor whenever possible. This can be viewed as *heuristic processor affinity*.

For certain applications, our heuristics are inadequate for optimally assigning processes to processors. For these cases, we have implemented *explicit processor affinity* with an interface to allow applications to make process-to-processor assignments. In our performance tuning, we have been able to achieve a 10%-to-20% improvement in TPC-B performance by explicitly assigning processes from the application level through a proprietary processor affinity interface.

We are not widely promoting explicit processor affinity as an available feature of the product because our interface is nonstandard. We plan to move to a standard interface for processor affinity as soon as one emerges. Until then, our proprietary interface is being documented through the field organization for use in customer situations where explicit affinity is necessary for adequate performance. The heuristic processor affinity with automatic load balancing will perform well for most workloads.

Interrupt Handling

All interrupts are directed to the monarch processor. If the monarch processor holds the I/O semaphore, or if the I/O semaphore is free, then the monarch will acquire the I/O semaphore if necessary and handle the interrupt. If the I/O semaphore is held by another processor, then the monarch will forward the interrupt to the processor holding the I/O semaphore.

This implementation is somewhat asymmetrical, but we have shown through experimentation that this is superior in performance to the symmetrical implementation of broadcasting the interrupt to all processors and having the processors arbitrate to determine who handles it.

I/O Drivers

There are many I/O drivers in HP-UX, and significant modifications to all of them for multiprocessor operation would have greatly added to the expense and risk of HP-UX 8.06. Because of this, we provide *uniprocessor emulation* for drivers. This allows existing I/O drivers to be incorporated into the multiprocessor system with few if any changes.

Uniprocessor emulation is possible because a single semaphore is used for all I/O, and spl (the call used to raise the interrupt level) is supported in multiprocessor HP-UX. However, there are performance penalties for this, because all I/O is single-threaded and because spl calls in the multiprocessor system are expensive because of contention for the underlying spinlock. Contention from these sources has been dealt with in the tuning of HP-UX 8.06 for certain

commonly-used I/O drivers. For example, the HP-FL disk driver has been broken away from the I/O semaphore to allow it to run concurrently with other I/O, and we have reduced the spl calls where possible in other key drivers.

Multiprocessor Boot Process

At system power-on, the monarch processor is bootstrapped. IPL (initial program loader) will load HPUXBOOT on the monarch, and the monarch will then load the HP-UX kernel. When control is transferred to the kernel, the system is still using only the monarch processor. Kernel initialization is performed on the monarch, and the last step in kernel initialization is to activate the other processors on the system and put the system into multiprocessor operation.

Uniprocessor Overhead

A requirement for multiprocessor HP-UX was the need for an extremely low multiprocessor overhead on uniprocessor systems. This is because the majority of HP's computer sales are derived from uniprocessor systems. To achieve a negligible level of multiprocessor overhead on uniprocessor systems, we have added special-case code for uniprocessors in key places, and we "back-patch" most multiprocessor semaphore primitive calls at boot time to eliminate locking overhead on uniprocessor systems. In other words, the system automatically modifies itself at boot time to specialize itself for uniprocessor or multiprocessor operation. As a measure of our success, HP-UX 8.06 on a Model 870 uniprocessor system outperforms HP-UX 8.0 on the same machine because of negligible multiprocessor overhead on the uniprocessor coupled with performance improvements that improve both uniprocessor and multiprocessor operation.

One consequence of our success in reducing uniprocessor degradation by effectively creating a uniprocessor-only system at run time is that our multiprocessor improvement ratios over uniprocessors are reduced. The ratios would look much better if our multiprocessor kernel were run unmodified on a uniprocessor for the basis measurement, which is common industry practice.

Performance

As mentioned previously, our development approach was based on the successive refinement of a series of prototypes. This was especially necessary for Model 870 multiprocessor performance tuning because the complex interactions between many hardware and software components resulted in unpredictable performance results beyond the current set of modifications.

Our performance tuning was an iterative process as follows:

1. Measure and analyze the current system.
2. Propose a candidate set of modifications.
3. Perform a quick implementation of these modifications on an experimental system.
4. Evaluate the performance of the experimental system and choose the modifications to put into the production system.
5. Perform detailed design and review on the selected modifications and integrate them into the main body of multiprocessor kernel source code.
6. Go to step 1.

We relied primarily on kernel profiling to analyze system performance. The profiling was done while running a variety of industry-standard and proprietary benchmarks. Through the profiling, we saw where the system overhead was excessive.

A change that we made from previous HP-UX system profiling is what we call time-based profiling. In the past, HP-UX profiling was based on instruction counts, but this hides the high overhead of multiprocessor cache effects. For example, a load and clear word (ldcw) instruction is many times more costly than most other instructions, but still accounts for only one instruction in an instruction-count profiling system. With time-based profiling, the actual time spent in routines is measured, including time costs for cache effects.

The profiling output was organized according to procedure calls, and ordered according to the cumulative time spent in these routines. In this way, the costly routines were highlighted, and we investigated them further to analyze why they were consuming excessive processor bandwidth and what steps could be taken to optimize them.

Other tools that we used for evaluating performance included instrumentation that we added to measure semaphore and spinlock contention, and measurements by AWAX, a proprietary tool that supports accurate low-level measurements and logic analysis.

We iterated through the tuning cycle approximately six times during multiprocessor HP-UX development. Each cycle took approximately eight weeks, with the first four weeks devoted to analyzing system performance and quickly prototyping a set of candidate changes, and the next four weeks spent doing detailed design, performing code reviews, and running regression tests. Careful attention to quality while integrating performance modifications with the system allowed us to continue performance tuning well into the system test phase and practically up to release.

Benchmark Performance

Table I summarizes the performance of the HP 9000 Model 870 multiprocessor system with one to four processors for the TPC-A and SPEC Aggregate Throughput benchmarks. TPC-A is an industry-standard OLTP benchmark that rates systems in transactions per second, and SPEC Aggregate Throughput is an industry-standard benchmark for measuring system throughput under an artificial workload.

Table I
HP 9000 Model 870 Multiprocessor Performance

	Model 870/100	Model 870/200	Model 870/300	Model 870/400
TPC-A	74.5	111.2	not measured	173.2
SPEC Aggregate Throughput	35.4	67.3	95.3	127.7

The multipliers for multiprocessor systems over a uniprocessor base are very dependent on the system workload. For compute-bound processing, we observe better results. In fact, you can run four copies of a Spice simulation in parallel on a Model 870/400 in the same time it takes to run a single copy. The reason for this is that on compute-bound

jobs such as Spice, there is no data sharing between jobs and little time is spent in the kernel, so there is little multiprocessor contention for resources and no appreciable cache coherency thrashing.

For this reason, it is no surprise that the SPEC Aggregate Throughput benchmark, which is fairly computationally intensive, shows better multiprocessor multipliers than TPC-A, which is more I/O intensive and spends a lot more time in the kernel. For a Model 870/400, the performance multiplier with respect to a uniprocessor is 3.6 for the SPEC benchmark but only 2.3 for TPC-A.

Another factor tending to reduce the Model 870's multiprocessor multipliers is the self-modification of the HP-UX 8.06 kernel on uniprocessor systems, as described previously. The multiprocessor HP-UX uniprocessor mode transparently produces a kernel with virtually no multiprocessor overhead, which significantly reduces the denominator in the multiplier calculations. This is a departure from common practice, which is to compute multipliers with respect to unmodified multiprocessor code running on a uniprocessor.

Finally, performance effects caused by applications cannot be ignored. In our tuning for TPC-A, a considerable amount of effort went into working with the database management system developers to reduce the multiprocessor contention generated by that level of software. Database systems typically manage their own sets of locks, which can be a great source of contention on multiprocessor systems.

Application Portability

Multiprocessor HP-UX is transparent to applications in that the system call interface has not changed, but there are some pitfalls that application developers need to anticipate. In particular, applications involving a set of cooperating processes are vulnerable to problems in a multiprocessor environment that their developers should anticipate. It is dangerous to assume that such applications will work correctly on a multiprocessor system without regression testing. Although nothing needs to be modified (or even recompiled) to run on the multiprocessor Model 870, some profound problems may be experienced.

For an application consisting of multiple processes, the timing characteristics will be radically different on a multiprocessor machine. The progress of execution through individual processes will be greatly affected by the ability of processes to execute concurrently on different processors. The consequence of this is that timing problems may be exposed that were never seen on uniprocessors, although they were always possible. We have already seen this firsthand in moving complicated, multiple-process system tests over to multiprocessor systems. Careful regression testing on the multiprocessor system is essential to screen for timing hazards.

Applications making use of real-time priorities (rtprio) need to take special care in moving to the multiprocessor system. On a uniprocessor, it is fair to assume that nothing else will run while the highest-priority real-time process is running. However, on a multiprocessor system another process may be running in parallel with the real-time process.

If a set of cooperating processes share a resource, for example a segment of shared memory, then care must be taken to check whether competition for this resource negates the computing power available from the additional processors. On a uniprocessor, this is not a problem because only one process can be executing at any time. However, on a multiprocessor system, it is possible to have effectively only one processor because the others are waiting on the shared resource.

The unit of scheduling in our initial multiprocessor implementation is the process. Thus, there is no concurrency within a given program, and a single instance of a program will not run faster on the multiprocessor system. The benefit of multiprocessor operation lies in increasing system throughput. Although more total MIPS (millions of instructions per second) are available, those MIPS are not all available to a single program.

Summary

The HP 9000 Model 870 multiprocessor systems are a significant technical milestone for HP. These systems take HP's high-end HP-UX system performance to a new level, with industry-leading OLTP performance at the time of their release. These multiprocessor capabilities were added to the HP-UX system without penalizing its uniprocessor performance.

The multiprocessor capabilities are transparent to existing applications. However, the behavior of applications on a multiprocessor system may be different, and regression testing should be performed by customers moving their programs to the multiprocessor system.

Transforming uniprocessor HP-UX to a multiprocessor operating system was the largest set of modifications to the HP-UX system since its introduction. This project demanded new tools for analyzing systems and new designs to address the special characteristics of the Model 870 multiprocessor systems over more conventional multiprocessor systems. The development strategy of successive refinement of a sequence of prototypes allowed us to focus our efforts on the areas of greatest return. The availability of many intermediate systems during the course of development allowed greater in-house exposure to the system before release and resulted in a higher-quality system.

The HP-UX multiprocessor kernel for the 9000/870 is also a suitable base for future multiprocessor systems with more and higher-performance processors.

Acknowledgments

We wish to acknowledge the hard work of all the engineers who have been involved with multiprocessor HP-UX 8.06. We want to specifically recognize the work of Jon Bayh, Bill Groves, and Jonathan Ross. Their superb contributions were essential to the success of this project.

Advances in Integrated Circuit Packaging: Demountable TAB

State-of-the-art IC packaging, particularly with RISC architectures, demands performance at a high lead count. This paper presents some of the fundamental topics in IC packaging, formulates the principal criteria by which single-chip IC packages are judged, and evaluates existing industry-standard packages. A new packaging technology is described that addresses the unsatisfied packaging needs of modern digital systems.

by Farid Matta

A state-of-the-art integrated circuit can consist of millions of transistors in a thin small chip of semiconductor material. A prominent part of every chip is an array of terminal pads, which include the circuit's inputs and outputs and connections to the power supplies and the system's grounds. The number of terminal pads in an IC is often quite large, and as a rule, the more complex the chip, the more terminals it has. Some recent ICs have more than 500, and future chips are projected to contain up to a thousand terminal pads.

In part because of their large numbers, and in part for electrical considerations, a chip's terminal pads must be very small. Larger pads cause the chip to be bigger than it needs to be based on functionality, and bigger chip sizes mean higher cost. At the present time, terminal pads are typically designed as 75-to-125-micrometer squares on a 150-to-250-micrometer pitch, and the trend is toward smaller pads and narrower pitches.

IC Assembly

Complex as it may be, a single chip rarely makes a complete system. Normally, a number of integrated circuits (along with other components) must be assembled together to form a useful electronic apparatus. The most straightforward way to do this is to mount the ICs as made (i.e., in chip form) onto a substrate that provides connections to their terminal pads—a concept known as hybrid or multichip module technology. However "natural," this approach has so far been limited to a few special applications for various technical and logistical reasons.

Technical Issues. For the interconnection substrate to accommodate the direct mounting of the chips, it must have conductor features comparable in size to the chips' terminal pads. While generally available, such substrates are significantly more expensive than standard printed circuit boards. The approach also requires the systems' manufacturers to adopt precision chip-to-substrate bonding techniques vastly different from the traditional soldering processes used for the rest of the system, and to take responsibility for the mechanical and environmental protection of the chip during handling, assembly, and service. Understandably, few users are willing to make the investments and bear the additional cost except when there is a compelling need.

Logistical Issues. A major difficulty in using ICs in chip form stems from the practical impossibility of performing at-speed test and burn-in on bare chips. Without such testing, the system manufacturer runs the risk of including substandard or faulty chips in multichip assemblies. The inevitable result is to scrap whole assemblies (including many good chips) or to expend resources on lengthy diagnosis and rework. Another important logistical hurdle is the reluctance of IC suppliers to provide their products in chip form, since that may reveal their product yields and cost structures, and would complicate accountability for product quality. Finally, the approach blurs the distinction between component and equipment reliability that can be neatly drawn in the alternative packaged approach.

It is no accident, therefore, that chip-on-substrate assembly technologies have taken hold only in vertically integrated companies, where logistical barriers can be overcome more easily. Even then, most applications are mandated by performance or weight considerations that cannot be satisfied by the alternative approach of using packaged ICs.

This alternative approach to building electronic systems, which is much more prevalent, consists in attaching prepackaged components to printed circuit boards by means of

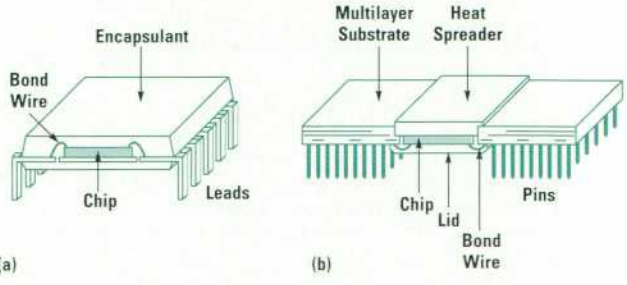


Fig. 1. Schematic representations of package categories. (a) Basic package. (b) High-performance package.

solder. For an IC to be usable in such a technology, the chip needs to be enclosed in a package, which is expected to:

- Serve as a space transformer between the IC terminal pads and the geometries characteristic of printed circuit board technology and solder-based attachment methods
- Provide mechanical and environmental protection to the semiconductor chip during handling, assembly, and service
- Allow the full testing and burn-in of the ICs by the chip vendor, thereby clearly fixing the responsibility and accountability for quality and failures.

The low cost and maturity of printed circuit boards and the safety and convenience of using packaged chips have resulted in the overwhelming dominance of this assembly approach (at least for now) over the direct chip-on-substrate assembly techniques.

Categories of IC Packages

Some IC packages may be better than others in one or more aspects, but most of them satisfactorily perform the three basic functions outlined above. There are, however, applications in which basic performance is not sufficient and the package must meet certain electrical and thermal conditions. Namely, in high-frequency (or high-speed) applications the package should present an adequate electrical environment, and when the chip dissipates a substantial amount of heat the package should provide a good conduit for that heat to a heat sink. We will call packages capable of satisfying these additional requirements "high-performance" packages, and the ones that cannot, "basic" packages. Fig. 1 illustrates the two categories of IC packages.

Basic Packages. A typical basic package is shown schematically in Fig. 1a. It consists of a metal lead frame patterned to match the chip's pads in the center and the printed circuit board's conductors on the outside. The chip's terminal pads are attached to the lead frame by fine wires, and the whole structure is enclosed in a molded plastic encapsulant. Examples of this arrangement are the dual inline package (DIP), the plastic quad flat pack (PQFP), and the thin slim-outline package (TSOP), among others. Basic packages also exist in which the plastic molding is replaced with ceramic casing to ensure hermeticity.

High-Performance Packages. A schematic representation of a typical high-performance package is shown in Fig. 1b. In this category of packages a better electrical environment is achieved through the use of a multilayer substrate containing power and ground planes as well as stripline and/or microstrip transmission lines for signal propagation. The chip is wire-bonded to the substrate and covered with a metal lid. The most prominent example of this approach is the pin-grid array (PGA).

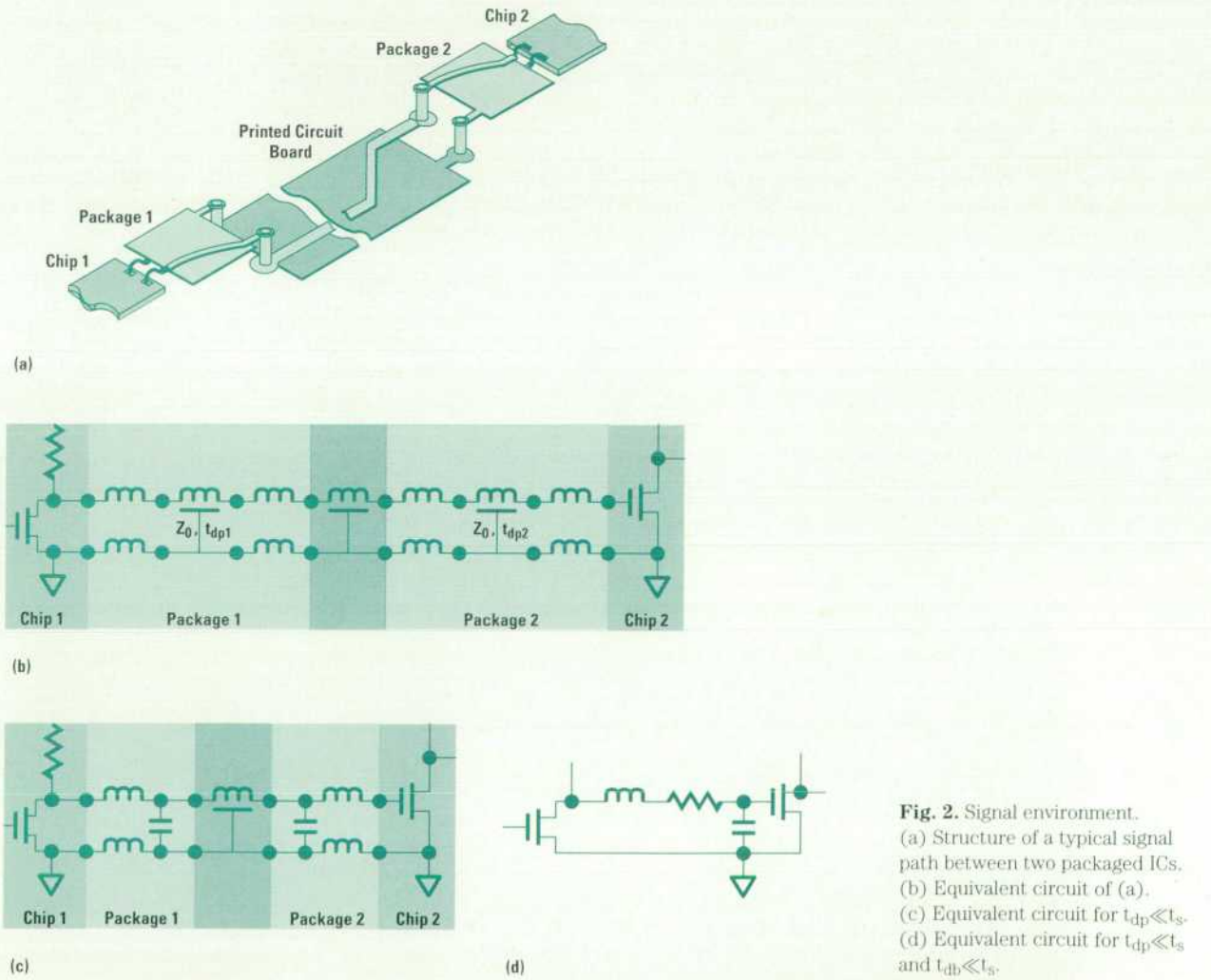


Fig. 2. Signal environment. (a) Structure of a typical signal path between two packaged ICs. (b) Equivalent circuit of (a). (c) Equivalent circuit for $t_{dp} \ll t_s$. (d) Equivalent circuit for $t_{dp} \ll t_s$ and $t_{db} \ll t_s$.

Electrical Performance

The choice of a package for an integrated circuit depends on the electrical and thermal conditions under which the chip is expected to operate. In other words, the package must satisfy a set of electrical and thermal requirements formulated for the application at hand.

The electrical operating conditions of an integrated circuit can be viewed as consisting of two distinct environments: one for signals and another for power. The requirements for these environments are substantially different.

Signal Environment

The signal's electrical environment is the arrangement of conductors and dielectric materials through which signals travel to and from the chip. Fig. 2a illustrates a typical signal path between two packaged ICs mounted on a common printed circuit board. Electrically, each segment of this path represents a transmission line with certain characteristic impedance Z_0 and time delay t_d as shown in the equivalent circuit of Fig. 2b. Also shown in Fig. 2b are parasitic parameters such as the inductances of the bond wires and package pins.

The transmission-line behavior of any section of this circuit manifests itself only when the time delay t_d in that section is comparable to the signal's switching time t_s. If the delay in a certain segment is significantly shorter than the signal's switching time, that segment behaves as a lumped-parameter element. For example, the equivalent circuit of Fig. 2c illustrates a situation in which the package's delay t_{dp} is much shorter than the switching time, while the board's delay t_{db} is not.

At appropriately low switching speeds all of the components of the signal path can be viewed as lumped reactances, resulting in the equivalent circuit of Fig. 2d. In this case the electrical process behaves like the simple charging of a capacitor; hence the commonly used terms "charging" and "discharging" of the signal line.

Ideal Package. For signal rise and fall times shorter than about 1 ns most commonly available packages must be treated using the transmission-line model. An "ideal" package for such conditions should conduct the signals between the chip and the interconnection substrate with minimal propagation delay, low attenuation, minimal reflections, little degradation of the waveform, and weak coupling with other signals. In terms of physical attributes, these requirements translate (respectively) to the following:

- Insulating materials with low dielectric constants
- Conductors with low sheet resistances
- Impedance control and matching with the printed circuit board's electrical environment
- No (or negligible) parasitic inductances or capacitances
- Low mutual inductance and coupling capacitance between lines.

The equivalent circuit of such an ideal package is shown in Fig. 3a. Understandably, this is a technical abstraction that does not exist in reality. It is shown here to illustrate how real packages deviate from it.

Basic Package. At the opposite end of the scale from the ideal is the basic package whose signal environment is represented by the schematic diagram of Fig. 3b. Here the leads are not of controlled impedance and each possesses substantial inductance and capacitance. The parasitic inductance includes the inductances of the leads and the bond wires, and the parasitic capacitance to ground is uncontrolled and depends on other structures. Relatively strong inductive and capacitive coupling (M and C_c respectively on the diagram) exist between the leads. Typical values of these parameters are shown in Table I for a 240-lead PQFP.

Table I
Typical Signal Environment in a 240-Lead PQFP

Parameter	Unit	Value
Lead self-inductance	nH	17.50
Mutual inductance (M)		
with nearest neighbor	nH	11.12
with third neighbor	nH	8.71
Coupling capacitance (C_c)		
with nearest neighbor	pF	0.96
with third neighbor	pF	0.16
Lead resistance	ohms	0.127

Waveform Degradation. To illustrate the significance of lead inductance for the signal environment, consider, for example, the condition shown in Fig. 4a, where a PQFP is connected to a matched 50-ohm transmission line. Ignoring (for simplicity) the capacitance between the leads, the circuit can be viewed as a low-pass filter with a time constant

$$\tau = L/R = (2 \times 17.5\ \text{nH})/50\ \text{ohms} = 0.7\ \text{ns}.$$

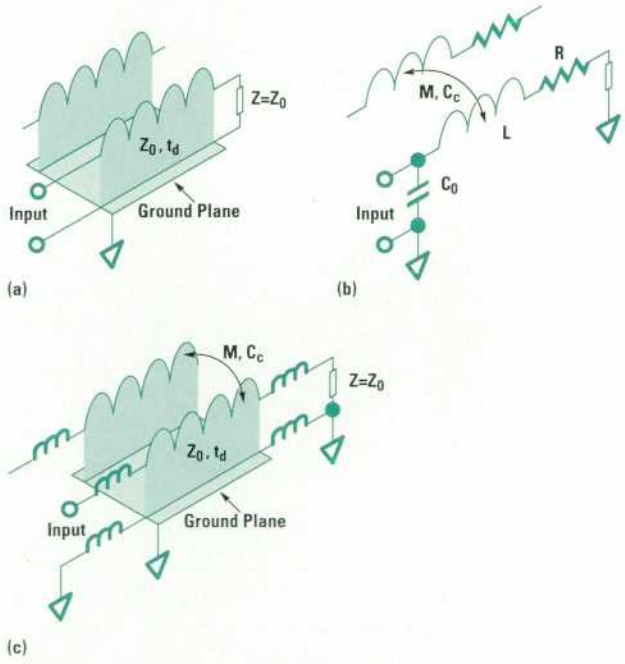


Fig. 3. Signal environments in various packages. (a) Ideal. (b) Basic. (c) High-performance.

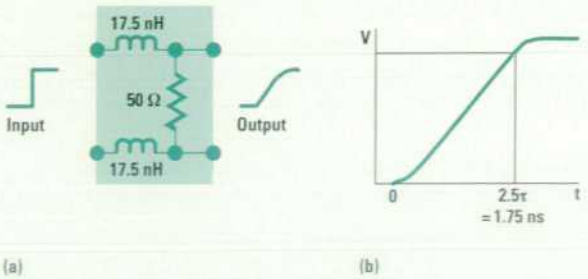


Fig. 4. Signal degradation in a basic package. (a) Equivalent circuit. (b) Effective delay.

In approximate terms, switching can be considered complete after about 2.5 time constants as shown in Fig. 4b. Therefore, it can be seen that the presence of parasitic inductances in the signal environment has resulted in an effective time delay of almost 1.75 ns.

High-Performance Package. Somewhere between the ideal and the basic lies the category of high-performance packages. The signal environment in these packages is schematically illustrated in Fig. 3c. Here, the lead does represent a controlled-impedance transmission line, but one that imposes a relatively long time delay and causes finite resistive losses. In addition, the line includes a parasitic uncompensated inductance representing the parts outside the multilayer substrate (in the case of the PGA, for example, that would be the wire bonds and the pins). The use of multiple conductor layers in high-performance packages can help reduce parasitic electrical coupling between signal lines compared to basic packages. However, such coupling is still fairly significant because the wires and pins are unshielded. Table II lists the electrical parameters of a 408-pin ceramic PGA as a representative example of high-performance packages.

Table II
Typical Signal Environment in a 408-Pin Ceramic PGA Package

Parameter	Unit	Value
Time delay in longest lead	ns	0.39
Uncompensated lead inductance	nH	2-7
Mutual inductance with nearest neighbor	nH	3.2-4.7
Coupling capacitance with nearest neighbor	pF	0.5-0.7
Longest-lead resistance	ohms	3.0

Power Environment

The package's power environment is the structure of conductors, insulators, and passive components involved in supplying the required voltages and currents to the chip within the package. An "ideal" package does not impede the delivery of electrical power from the power supply to the chip, and maintains constant potentials at the power supply terminals at all times regardless of the current being drawn by the chip. A schematic representation of this abstraction is shown in Fig. 5a.

In reality, power circuits look more like Fig. 5b. For a basic package, the inductance represents the power lead and the ground lead including any bond wires. High-performance packages, such as PGAs, have power and ground planes and their parasitic inductances can be significantly lower, consisting mainly of bond-wire and pin inductances. However, multilayer ceramic packages typically have thin conductors with relatively high sheet resistance. Table III lists some typical power-circuit parameters of basic and high-performance packages.

Table III
Typical Power-Circuit Parameters in Selected Packages

Parameter	Basic Package (PQFP)	High-Performance Package (PGA)
Power line inductance L_p	17.53 nH	2-7 nH
Ground inductance L_g	17.53 nH	2-7 nH
Power circuit resistance	0.25 ohm	0.2 ohm

Inductive Effects. Inductances in the power circuit cause instability of the potentials at the power and ground terminals of the chip. Consider, for example, the simplified case shown in Fig. 6a. When the circuit switches from logic low to logic high its output line charges through the package's power lead (with inductance L_p), and in the process draws a charging current that reaches its maximum value i_c in a time interval t_c. Similarly, when the device switches in the opposite direction the output line discharges through the ground lead (with inductance L_g), sinking into the ground a discharge current that reaches its maximum value i_d in a time interval t_d.

In both cases the parasitic inductances develop voltages that counteract the change in current. The magnitudes of these voltages are determined by the general relationship $V = L(di/dt)$ and their polarities are as shown on the schematic of Fig. 6a. As a result, during low-to-high switching, the potential of the chip's power terminal drops from its steady-state value V_s to a temporary low $V_c \cong V_s - L_p(i_c/t_c)$. Similarly, during high-to-low switching the potential of the chip's ground terminal rises to a temporary high $V_d \cong V_g + L_g(i_d/t_d)$. The first of the two phenomena is commonly known as power supply droop and the second as ground bounce. These potential swings couple directly into other lines connected to the same source.

Simultaneous Switching. In practical circuits more than one driver will be connected to common power and ground leads. Fig. 6b illustrates a typical case in which four drivers

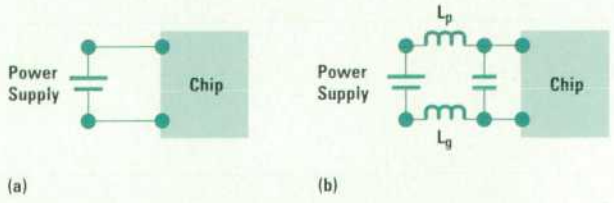
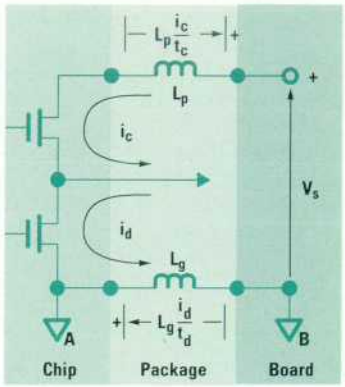
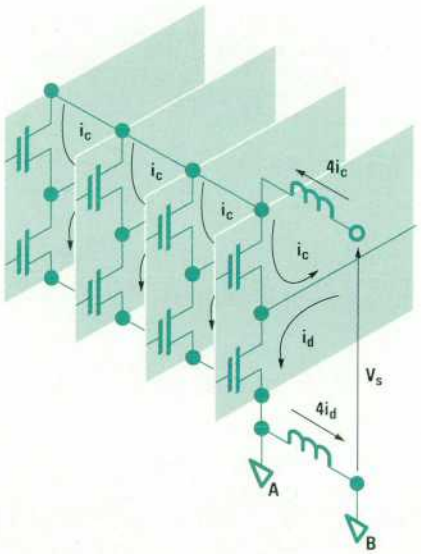


Fig. 5. Power environment. (a) Ideal. (b) Real.



(a)



(b)

Fig. 6. Power environment phenomena. (a) Inductive effects. (b) Simultaneous switching.

are served by one power connection and one ground connection. If all four drivers switch simultaneously, the currents involved and the resulting noise will be four times as great as in the case just discussed. To illustrate the magnitude of the issue consider a basic package with lead inductances of 15 nH each. If the four drivers switch simultaneously from low to high in 0.5 ns drawing 20 mA each, the potential of the chip's power terminal would drop by $(15\times10^{-9}) \times 4 \times (20\times10^{-3}) \times 1/(0.5\times10^{-9}) = 2.4$ volts. During high-to-low switching the chip's ground terminal potential will rise in a similar manner. Any other line connected to the same source or the same ground will temporarily experience a change of potential that may be sufficient to cause false switching.

High-performance packages feature lower power and ground inductances, thus expanding their range of switching speed, driver current, or number of I/O lines per power or ground terminal. Another advantage is the presence of significant capacitances between power and ground planes. Just as inductance impedes instant changes in current and develops an EMF to counteract them, capacitance impedes instant changes in voltage, and supplies the necessary current to prevent them. The phenomenon (or technique, if implemented by design) is known as capacitive bypassing or decoupling. However, bypass capacitances help neutralize the effects of upstream inductances only. Therefore, in a wire-bonded PGA the wire inductance is not remedied by the power-plane capacitance.

DC Voltage Drop. Newer chip technologies often use lower supply voltages and higher currents. Some state-of-the-art chips draw in excess of 20 amperes. For such chips, the resistance of the conductors in the power-supply circuit acquires particular importance since the voltage drop in the package can be large enough to interfere with the operation of the chip. Ceramic PGAs have thin conductors made of tungsten with fairly high sheet resistances.

Electrical Performance Specifications

Even the brief and simplified discussion presented here leads to the conclusion that packages cannot be easily specified for a certain operational parameter such as clock frequency. The ability of a package to operate at that frequency will depend on many factors, such as the number of I/O lines per power (and ground) lead, the driver current, the capacitance of the output lines, and other factors. It is quite possible, however, to point out the set of parameters that define a package electrically. Although these parameters cannot directly predict the package's performance in a given system, they can certainly be used to differentiate packages in terms of electrical performance. For the signal environment, these parameters are:

- Impedance control
- Minimal uncompensated inductances
- Short delay time, that is, low dielectric constants and/or short lines
- Low conductor resistance
- Low mutual inductances and coupling capacitances.

For the power environment, the relevant parameters are:

- Low power-circuit inductance
- Low power-circuit resistance
- High capacitance between power and ground connections.

Everything else being equal, the closer a package comes to meeting these criteria, the better its electrical performance will be in any system.

Thermal Performance

Every chip generates a certain amount of heat as part of its normal operation. Depending on the technology (MOS, bipolar, etc.), the chip's operating frequency, and a number of other factors, the thermal power produced can vary from a fraction of a watt to over 30 watts. Unless that heat is constantly removed, the chip's temperature can rise beyond the acceptable limit for proper operation and reliability.

For low-power ICs, the natural processes of conduction, convection, and radiation are often sufficient to dissipate the heat into the surrounding structures and environment. For chips that generate more than a few watts, special provisions must be made for the adequate removal of the heat produced. Specifically, a constantly replenishing coolant (such as forced air) is employed, and a heat sink is attached to the IC to facilitate effective heat transfer from the chip to the cooling medium.

An ideal package would be one that allows the unimpeded flow of heat from the chip to the environment or to the heat sink. In reality, packages always present a certain thermal resistance to the heat flow. Fig. 7a illustrates a typical IC cooling arrangement in which the heat produced by the chip is conducted through the package to the heat sink where it is removed by the flowing air. T_j, T_c, and T_a are the temperatures of the chip (junction), the package surface (case), and the air, respectively. A simplified schematic of the thermal path is shown in Fig. 7b, where Q denotes the thermal power and θ_p and θ_s are the thermal resistances of the package and the heat sink, respectively. The variables and parameters of the system are connected by the following relationships:

$$T_j - T_a = Q(\theta_p+\theta_s).$$

$$T_j - T_c = Q\theta_p.$$

$$T_c - T_a = Q\theta_s.$$

These relationships show that, given a certain power Q and air temperature T_a, the total thermal resistance $\theta_p+\theta_s$ defines the chip's temperature. Conversely, given a certain allowable chip temperature T_j and air temperature T_a, the thermal resistance determines how much power can be safely dissipated by the chip. From either perspective, the thermal resistance of the package is its principal thermal parameter, and the lower it is the better.

As an example, consider a 12-watt IC cooled by air that has been preheated (by other components) to $T_a = 50°C$. If the maximum allowable junction temperature T_j is 110°C, the total thermal resistance should not exceed 5°C/W (from the equation: $\theta_p+\theta_s = (T_j - T_a)/Q$). A practical heat sink may have a thermal resistance of about 3°C/W at reasonable air velocities, say 1 m/s. Accordingly, the package's thermal resistance must be below 2°C/W to satisfy the requirement. Table IV lists the thermal resistances of some typical packages.

The values in Table IV are the thermal resistances of the packages when attached to heat sinks (but not including the thermal resistances of the heat sinks themselves). If a package is operated without a heat sink, its thermal resistance will be significantly higher.

In most cases the thermal conditions are much more complex than the simplified picture outlined above. Secondary thermal paths exist through the leads, the housing, and other paths. This results in a multibranch equivalent circuit, but the general approach remains the same. Thermal resistances can also be strong functions of the air flow. However, in most packages the junction-to-case thermal resistance is fairly constant.

Table IV
Thermal Resistances of Selected Packages

Package Type	θ_p (°C/W)
Basic	
Dual inline	30
Plastic quad flat pack	28
High-performance	
Ceramic pin grid array	25
Ceramic PGA with thermal via holes	1.3
Ceramic PGA with metal heat spreader	0.45

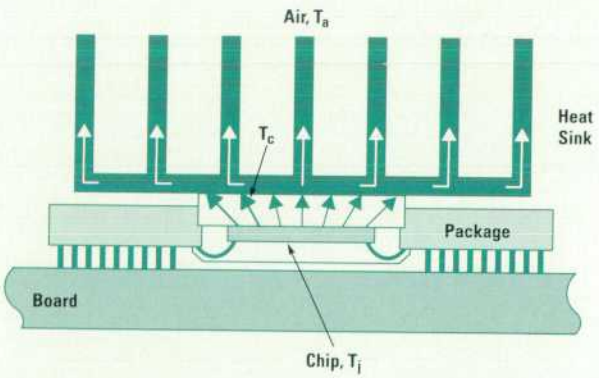
Thermal requirements for packages vary widely. Bipolar silicon and MOS GaAs devices dissipate much more heat than CMOS silicon chips. Heat generation also rises steadily with increasing clock frequency. Ensuring the proper thermal environment for the chip is crucial, since operating at higher temperatures will degrade the chip's electrical performance and will accelerate the failure mechanisms, which are normally exponential functions of temperature.

Manufacturability and Serviceability

Although the performance of a package is its principal attribute, other factors play important roles in the design and selection process. Prominent among those are cost, reliability, manufacturability, and serviceability. Cost and reliability are straightforward, easily quantifiable parameters. Manufacturability and serviceability require some elaboration, and towards that end a brief overview of printed circuit assembly methods may be helpful.

When the packaged ICs, along with other components such as connectors, passive parts, and so on, are mounted on a printed circuit board, the result is a printed circuit assembly, which is the main building block of most electronic equipment. The design and fabrication of printed circuit assemblies is carried out in one of three ways:

- Through-Hole Technology (THT). This traditional approach involves inserting the components' leads into holes in the printed circuit board before performing the soldering operation. The mounting holes, passing all the way through the board, waste much of the area available for interconnections in the board's inner layers, thus complicating its routability and limiting its density.



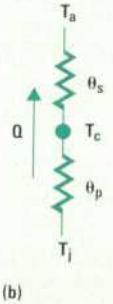


Fig. 7. IC cooling. (a) Typical arrangement. (b) Equivalent circuit.

- Surface Mount Technology (SMT). This more modern approach is based on soldering the components to pads on the surface of the printed circuit board, thus freeing more of the area of the inner layers for routing. It also allows double-sided assembly for even greater packing density. The current trend is toward a greater proliferation of SMT.
- Mixed Assembly. Most basic packages are available in SMT-compatible versions, but high-performance packages and some other components are still predominantly of the through-hole variety. As a result, some high-performance printed circuit assemblies, such as in computer applications, must use a mixed THT-SMT assembly process, which puts conflicting demands on the construction, metallurgy, and thickness of the printed circuit board and on the parameters of the soldering process.

Not all printed circuit assembly designs are equally suited for efficient, easy, and economical manufacturing. If a manufacturability index is formulated for printed circuit assemblies, it should include the following factors:

- Number of Processes. Lower score for products requiring more than one attachment method—for example, solder reflow and wave soldering.
- Process Window. Higher score for wider process window.
- Component Pretestability. Higher score for the ability to test all components fully before assembly.
- In-Process Testing. Higher score for the ability to test a complex assembly at intermediate points during the fabrication process.
- Ease of Rework. Higher score for easier diagnosis and rework, particularly of the more expensive components, and for multiple chances to rework.
- Product-Specific Fixturing. Lower score for the need to change production fixturing for different products.

When a hardware failure occurs in an installed system, it is typically diagnosed to the board level, and the culprit board is replaced and returned to the factory for diagnosis and repair. In much the same way as for manufacturability, a serviceability index can be developed to judge various packaging options. It should be based on the following factors:

- Component Replaceability. Higher score for easily removable parts. In addition to reducing the time and cost of physical repair, readily demountable parts facilitate diagnosis and access to other components.
- Component Testability after Removal. Higher score if the components remain intact and testable after removal. This eliminates unnecessary scrap, helps provide meaningful feedback to component suppliers, and helps build a database for effective troubleshooting.
- On-Site Repairability. Limiting on-site repair to the board level necessitates stocking a variety of expensive printed circuit assemblies in service centers around the world. The ability to diagnose and easily replace a bad component at the customer's site can produce sizable savings.

Compatibility of IC packages with advanced assembly methods (namely with SMT) and with manufacturability and serviceability criteria are becoming increasingly important. While basic packages generally meet the manufacturability requirements, and to a lesser degree the serviceability criteria, high-performance packages do not score high on either account. PGAs, for example, require mixed assembly, manual insertion, and application-specific fixturing. A printed circuit assembly that contains PGA packages cannot be reworked more than once or twice, and the rework process is tedious and messy and requires highly skilled technicians. Rework time estimates vary from 45 minutes to two hours per package, and the packages are ruined and untestable after removal. In addition, certain future reliability hazards are often inflicted on the rest of the assembly as a result of the rework and repair process.

Package Fundamentals Summary

Basic packages are simple and inexpensive. They meet most of the manufacturability and serviceability requirements, but electrically and thermally they are suitable only for nondemanding applications. Conventional high-performance packages, such as PGAs, provide significantly better electrical and thermal environments. However, they are fairly expensive and fail to meet most manufacturability and serviceability requirements. Their electrical performance is likely to become inadequate as signal rise and fall times drop below 1 ns.

Demountable TAB

Having defined the principal criteria used to evaluate single-chip IC packages for digital applications, and having reviewed the state of the art, we now proceed to describe a new technology that has been developed in response to the shortcomings of existing solutions. The technology is based on tape automated bonding (TAB).

Tape Automated Bonding (TAB)

TAB was originally conceived as a way to avoid wire bonding by attaching the lead frame directly to the chip. Since conventional, self-supporting lead frames could not be made with features fine enough to mate directly with the chip's pads, the method uses thinner lead frames laminated on a supporting dielectric film. The latter is typically fabricated in tape form with sprocket holes for automated manipulation; hence the term tape automated bonding.

Fig. 8a presents an example of a chip mounted on a TAB frame, used to replace the wire bonds in a standard package. The TAB frame's inner leads are bonded to the chip, while the outer leads are attached to the package's conventional lead frame. TAB can also be used for direct attachment to the printed circuit board as shown in Fig. 8b, in which case the method is known as TAB on board (TOB).

TAB Advantages

TAB (specifically TOB) offers the advantages of lower cost, lighter weight, and lower-profile assembly. Therefore, it is used widely in consumer products and low-cost technical applications. It has also turned out to be an attractive option for computer applications, for reasons of cost and performance.

The performance advantage consists in TOB's suitability for denser chip I/O geometries and finer printed circuit board features, and in its smaller footprint and lower electrical parasitics. When TAB frames are made with two conductor

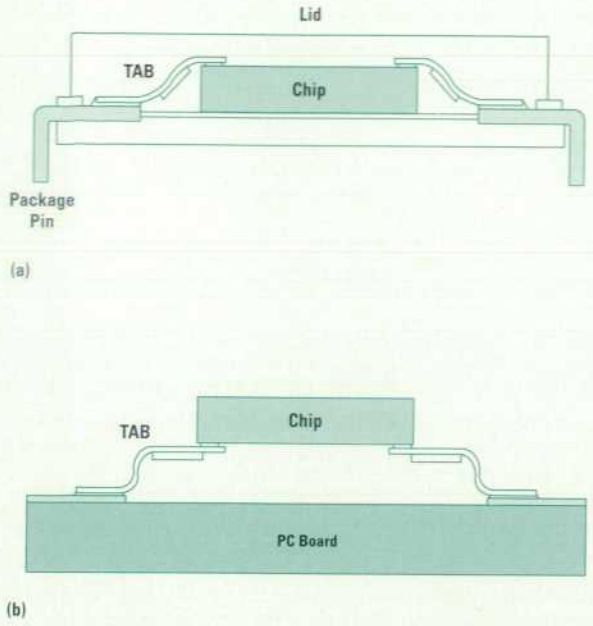


Fig. 8. Tape automated bonding (TAB) structures. (a) TAB in package. (b) TAB on board (TOB).

layers the signal leads can be designed as controlled-impedance microstrip lines, as will be shown later. The thick copper conductors (0.5 to 1 oz/ft^2) and the low dielectric constant of the polyimide insulator (3.4) account for low losses and short electrical length, respectively. Additional advantages are found in TAB's shorter design and fabrication cycles compared to other high-performance packages (e.g., PGAs), and in its preassembly testability, an important feature not found in other chip-on-board assembly approaches.

TAB Drawbacks

On the other hand, TAB has some shortcomings, chief among which are entry barriers, such as high startup cost, and the difficulty of rework and repair.

The startup cost barrier affects both the chip supplier and the equipment manufacturer. For the chip maker, getting into TAB entails substantial startup investments in specialized wafer-bumping and inner-lead bonding facilities. The industry-standard inner-lead bonding (ILB) process requires "bumping" the chip's I/O pads, that is, growing miniature gold bumps to which the TAB inner leads will be bonded. This is a fairly expensive process that requires special expertise and a large initial investment. Since bumping is done at the wafer level, the per-chip cost increases steeply at low wafer-sort yield (typical for advanced chips).

For the system manufacturer, the outer-lead bonding (OLB) processes also require expensive equipment with inherently application-specific tooling and relatively low throughput. For example, an outer-lead bonder with a throughput less than 10,000 TAB sites a month costs more than U.S.$600,000, requires a lead-form and excise module costing about U.S.$100,000, and requires a thermode costing over U.S.$10,000 for every TAB geometry in use. Most important, however, TAB represents a radical change in the board assembly culture, a change that requires significant physical, logistical, and behavioral modifications on the shop floor.

TOB packages are even more difficult to rework or repair than PGAs, especially at higher lead counts when the outer-lead pitch is very small. The task becomes even harder when the chip is attached to a heat spreader. Some laborious, barely acceptable techniques do exist, but they invariably entail the destruction of the chip and some damage to the board. Typically, such methods can be used only once on any specific TAB site. None of these methods was sufficiently practical to find acceptance in the industry. At the present time it is fair to say that TOB packages are basically not reworkable.

Therefore, despite TAB's advantages of higher electrical performance and lower cost compared with other packaging methods, its proliferation in the industry has been extremely slow and mostly limited to low-cost consumer products. Obviously, to potential users in the computer and instrument industries the issues of high startup cost and the difficulty of rework and repair have been sufficient to outweigh TAB's benefits.

Demountable TAB

Our solution to this problem is a version of TAB that, while maintaining the advantages of better performance and lower cost, does not require expensive facilities and is easy to demount for rework and repair. This solution was developed at HP's IC Business Division R&D center. It is called *demountable TAB*, or DTAB.

The demountable TAB idea is fairly simple. It is based on using a TAB structure that differs from the standard implementation in two ways. First, the expensive, bumped, inner-lead bonding operation is replaced with a simple bumpless process.[1,2] Second, the soldered outer-lead connections are replaced with separable pressure contacts.

Bumpless ILB lowers the TAB entry barrier for the chip maker by obviating the need for capital-intensive wafer-bumping facilities. Similarly, the solderless outer-lead connections allow the user to adopt TAB without the associated investments and profound changes in the assembly culture. Demountability also completely removes the other major drawback of conventional TAB, that is, the difficulty of repair and rework.

DTAB Structure

Fig. 9 illustrates the basic structure of the DTAB package. As received by the user, it consists of two parts: the package proper, and a spring/stiffener. The spring/stiffener is mounted on the opposite side of the printed circuit board to enhance its stiffness and to provide the spring action necessary for the pressure contact.

The package contains four protruding alignment bosses, and the printed circuit board has a set of matching holes. Mounting the package on the board consists merely of inserting the bosses into the holes and attaching the spring/stiffener on the backside of the board. The structure is self-aligned and one of the pin/hole pairs is offset so the package can be inserted only in the correct orientation.

The main element of the package is the TAB frame, which carries the IC. It is oriented so that its circuit side faces the printed circuit board while the back of the IC is attached to a heat spreader which doubles as a mechanical clamp. The

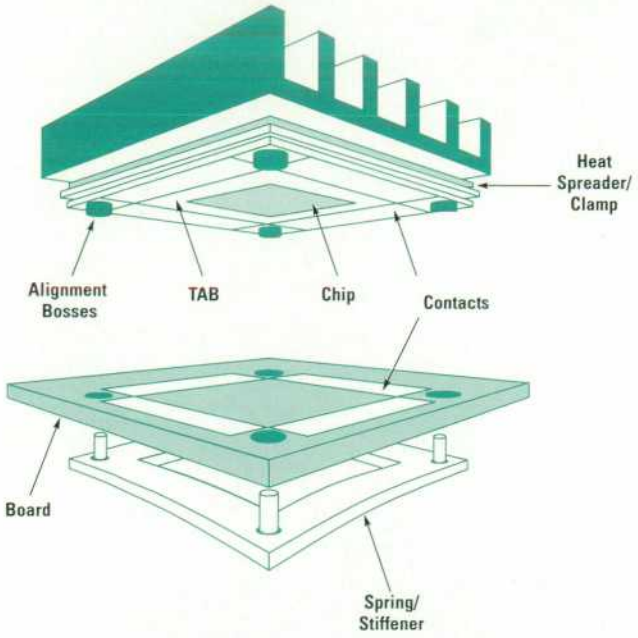


Fig. 9. Structure of the demountable TAB (DTAB) package.

TAB frame and the printed circuit board contain gold-plated pads which come into contact with each other when the package is mounted onto the printed circuit board. The spring/stiffener, together with an elastomeric planarizer backing the TAB frame, form a composite spring system designed to ensure that the outer-lead contacts are made and maintained under various operating conditions.

An example of a DTAB package with 432 leads is shown in Fig. 10. The outer-lead contacts on the TAB tape are arranged in an area array configuration, an arrangement that packs a large number of contacts into a relatively small area without requiring a fine-pitch printed circuit board technology or overly tight tolerances on the holes. For example, this 432-lead package has a footprint 5 cm (2 in) on a side, and the required printed circuit board can be made using a technology as coarse as 200-micrometer (0.008-inch) lines and spaces with standard hole tolerances. In this design, a misregistration of 250 micrometers (0.010 inch) can be tolerated without danger of losing continuity or shorting to neighboring contact pads.

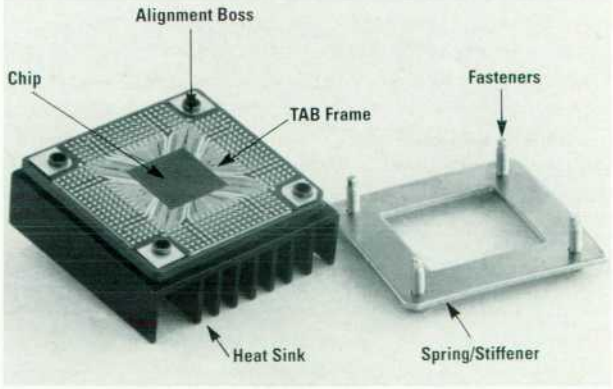


Fig. 10. 432-lead DTAB package.

Advantages of DTAB

DTAB not only avoids TAB's drawbacks while maintaining its advantages, but also adds a number of valuable features of its own. In product development, DTAB removes many of the constraints imposed on the printed circuit assembly design by mixed-assembly requirements. Easy demountability facilitates debugging in the breadboard phase. In manufacturing, it provides preassembly testability, unlimited reworkability, and ease of troubleshooting. DTAB does not upset established printed circuit assembly practices and fits easily into existing production lines. Another potential advantage is the possibility of having the printed circuit assembly without VLSI chips fabricated where manufacturing costs are lowest, then adding the DTAB-packaged VLSI chips just before shipment of the end product.

The importance of testability in manufacturing cannot be overstated and deserves more elaboration. Testability is one of the main advantages of TAB in general. However, to accommodate test pads for all the leads in the conventional technology, the TAB frame must be made much larger than it otherwise needs to be. An expensive fixture is used to access those pads, and the resulting electrical environment is so inferior to the actual system that the test coverage is often quite poor. It is also extremely difficult to cool a chip on TAB adequately during test and burn-in if that is required. In contrast, DTAB needs no test pads on the tape. The test fixture can be a simple DTAB site on a printed circuit board identical to the one on which the package will eventually be mounted. Cooling is handled in exactly the same way as in the assembled product.

In product support, the readily demountable DTAB package makes it easy to diagnose and repair faulty printed circuit assemblies. It makes troubleshooting and servicing the equipment possible at the chip level at the customer's site, which may eliminate the need to stock expensive subassemblies at many service centers around the world. Another major advantage is the possibility of field upgrades on the chip level.

The Cost of DTAB

DTAB's benefits do not come without cost. The concept imposes two special requirements on the printed circuit board: a noble finish on the traces, and four alignment holes on each site. The impact of these requirements varies depending on the specific application.

Trace Metallurgy. For the pressure contact to be reliable, the printed circuit board's copper traces must be coated with gold over a nickel barrier. The minimum gold thickness that has been characterized for reliable operation is 0.127 micrometers (5 microinches). The recommended thickness of the nickel barrier is 2.54 micrometers (100 microinches) or greater.

Alignment Holes. The alignment holes add to the cost of the printed circuit board in two ways. First, they use up some board area which might otherwise have been used for routing. This is a minor issue, and its effect on cost is negligible. Secondly, the required tolerances on hole size and placement are ±50 and ±125 micrometers (0.002 and 0.005 inch), respectively. While printed circuit board shops will normally provide a placement tolerance of about 100 micrometers,

meeting the size tolerance may require that the holes remain unplated. Therefore, they should either be drilled separately at the end (post-drilling) or "tented" to prevent side-wall plating. Either option implies some additional cost.

Mechanical Considerations

In general, three principal elements in the concept and the design of DTAB can be credited for its successful operation. These are a flexible contact element, a composite, controlled-relaxation spring system, and sealed contacts.

Flexible Contact Element. In theory, when two rigid bodies are brought into contact, only three points on each are actually touching.[3] However, when one of the two elements is flexible, as is the case in DTAB, the contact area is larger, the resistance is lower, and the electrical connection is more reliable.

Controlled-Relaxation Spring. Previous designs for pressure contact systems used an elastomeric spring to provide the contact force. The effects of time and temperature cause elastomeric materials to relax and lose up to 60% of the original stress, and accordingly, retain less than half of the initial force. They also have difficulty maintaining the contact pressure during thermal cycling, because of hysteresis. DTAB uses a composite spring system in which the loss of stress is minimal. The composite spring system is described in more detail later.

Contact Shielding. A major failure mode in pressure contacts is the corrosion of the contact surfaces in chemically aggressive environments. In DTAB, the contacts are virtually sealed and the circulation of corrosive agents around the contacts is all but eliminated.

Fig. 11 shows a cross section of the package. The heat spreader/clamp is made of a copper-molybdenum-copper laminate and the alignment frame is of molded plastic. The controlled-relaxation composite spring system consists of the elastomeric planarizer backing the TAB frame in series with the nonrelaxing metal spring/stiffener on the opposite side of the printed circuit board.

Fig. 12 shows the basic dimensions of the DTAB package. The first two members of the package family are DTAB-284 (replacing the 272-pin PGA), and DTAB-432 (replacing the 408-pin PGA). DTAB-284 and DTAB-432 are 3.75 cm (1.5 in) square and 5 cm (2 in) square, respectively. In comparison,

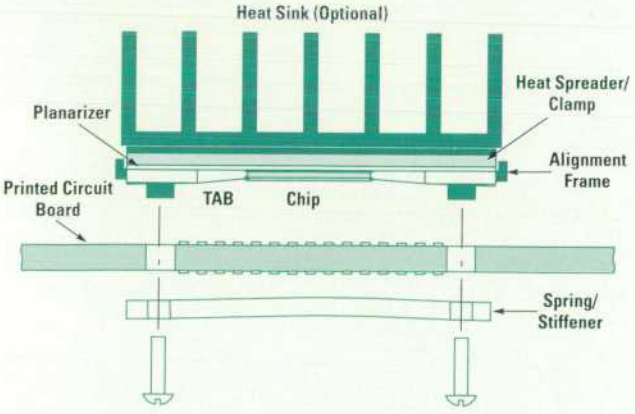
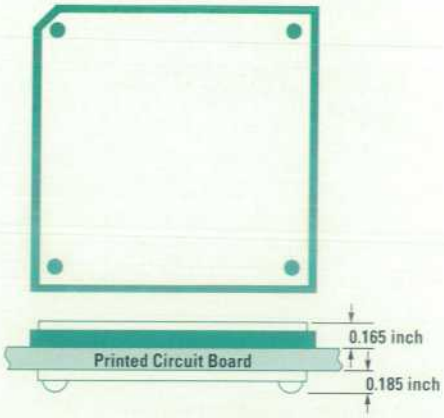


Fig. 11. Cross section of the DTAB package.



Lead Count	Package Size	Comparable PGA Size
284	1.5 Inches Square	2.1 Inches Square
432	2 Inches Square	2.3 Inches Square

Fig. 12. DTAB package dimensions.

the 272-pin PGA and the 408-pin PGA are 5.33 cm (2.1 in) square and 5.84 cm (2.3 in) square, respectively.

Composite Spring

Fig. 13 illustrates the effect of the composite spring. Fig. 13a shows qualitatively how an elastomeric spring relaxes (i.e., loses stress at a given strain) with time. Depending on the temperature, the relaxation typically tapers off at about 40% to 60% of the initial stress. In the composite spring, the metal spring compensates for the relaxation of the elastomer so

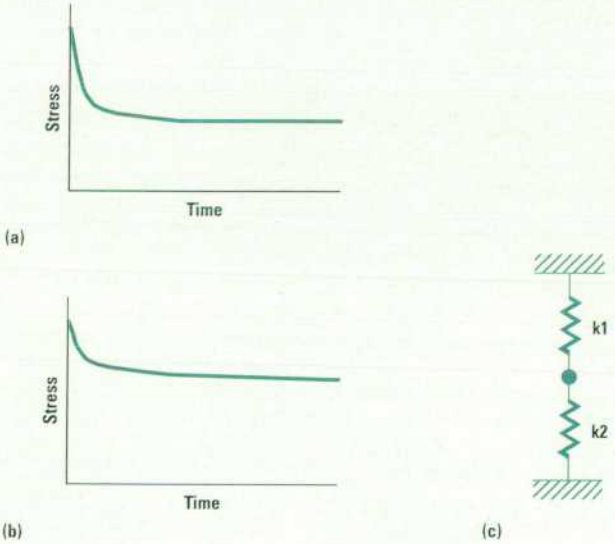


Fig. 13. Spring relaxation. (a) Elastomer only. (b) Composite spring (elastomer with metal spring). (c) Schematic representation of the composite spring.

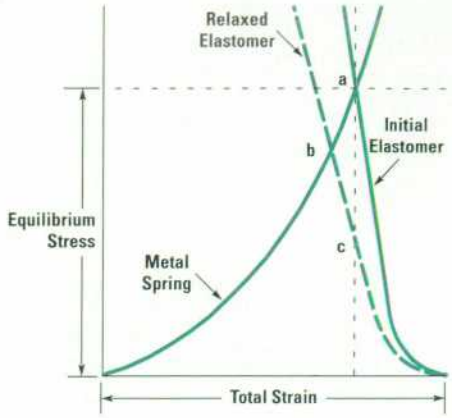


Fig. 14. Operation of a composite spring.

that the overall loss of force is considerably reduced, as shown in Fig. 13b.

A simplified mathematical description of the phenomenon can be obtained by analyzing the series connection of two linear springs with spring constants k_1 and k_2 (Fig. 13c). Assume that one of the springs, say k_1, is a relaxing type with force retention r_1, that is, r_1 is the fraction of the original force remaining in that spring after relaxation under constant strain. Assume also that the other spring is nonrelaxing with force retention $r_2 = 1$. It can be shown that the overall force retention R in the composite system is:

$$R = r_1 \frac{1 + \frac{k_2}{k_1}}{r_1 + \frac{k_2}{k_1}}.$$

Therefore, by adjusting the ratio of stiffnesses k_2/k_1, one can control the loss of stress in the composite system. Force retention R can be increased by reducing the ratio k_2/k_1.

The physics of the composite spring is illustrated by Fig. 14. The diagram represents a graphical method of finding equilibrium stresses and strains in the composite system. The stress/strain curves of the two springs are plotted with one of them mirrored around the stress axis. When the curves are spaced apart by an amount equal to the total system strain, the intersection point corresponds to the equilibrium stress. Conversely, if the stress is known and the curves are placed such that they intersect at that stress, the distance between them on the abscissa shows the total strain. The segments of the abscissa enclosed between the intersection point and the origins correspond to the component strains.

When the elastomeric spring relaxes, its stiffness is reduced, and its stress/strain curve shifts to the position represented by the dashed line in Fig. 14. The equilibrium stress drops from point (a) to point (b). If the system consisted of an elastomeric-only spring, the stress would have dropped to the lower point (c) instead. In addition to illustrating the physics of the composite spring, this graphical method can be used for the accurate analysis of real nonlinear systems. (Further discussion of the composite spring can be found in reference 4.)

Assembly of the DTAB package on the printed circuit board consists of torquing the the fasteners with a certain moment, which corresponds to a specific mechanical stress in the system.

Electrical Considerations

The main electrical structure in the DTAB package is the TAB frame. Depending on the required performance, a single-metal or a double-metal tape may be used. The single-metal version, whose cross section is shown in Fig. 15a, is considerably less expensive but offers limited high-frequency capabilities (similar to basic packages). In the double-metal structure (Fig. 15b) each lead, together with the common ground plane, forms a microstrip transmission line with a characteristic impedance of about 50 ohms. Such a structure represents a much better electrical environment for high-frequency signal propagation than the single-metal version.

DC Parameters

The main dc parameters of the DTAB package are the resistances and the current-carrying capabilities of the leads and the contacts.

Lead Resistance. The width of the signal lead (50 micrometers or 0.002 inch) is chosen to form a 50-ohm microstrip line with the ground plane and the polyimide insulator. The total lead resistance is proportional to the lead length, which is a function of the package size and of the position of the lead on the TAB frame. With the typical 1-oz/ft^2 copper on the signal side the lead resistance values are 0.06 to 0.15 ohm for DTAB-284 and 0.07 to 0.17 ohm for DTAB-432.

Contact Resistance. Difficult to calculate, the contact resistance has been empirically defined. Based on tens of thousands of measurements, the observed values are typically about 0.8 milliohm and do not exceed 2.0 milliohms.

Current-Carrying Capability. This parameter, although of great interest, is hard to specify, mainly because of the lack of definition of what it means for the case at hand. At the time of this writing, both the 0.002-inch lead and the individual contact point have been proven able to sustain indefinitely a dc current of at least 0.3A without change in resistance or appearance. This, however, is only a data point rather than a limit or a specification.

AC Parameters

As discussed earlier, for a package not to degrade the performance of the chip it houses, the package should provide a clean environment for signal propagation and unimpeded access by the chip to the power supply. A clean electrical environment means a transmission line of uniform known characteristic impedance, with minimal parasitics and losses, free from coupling with other signal lines. Unimpeded access to the power supply means low-impedance power leads with minimal series inductances to cause Ldi/dt potential bounce.

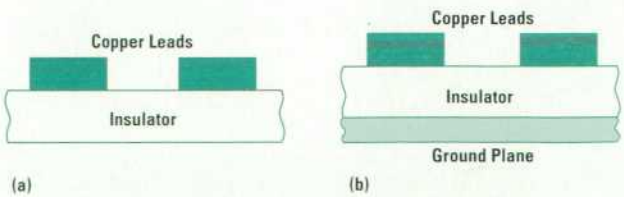


Fig. 15. TAB cross sections. (a) Single-metal. (b) Double-metal.

Signal Environment. The signal propagation environment in DTAB is shown in Fig. 16a. Two signal leads are drawn to illustrate coupling effects. A microstrip line is formed by each lead, the underlying ground plane, and the dielectric sandwiched between them. The equivalent circuit of the signal propagation environment is presented in Fig. 16b. The small parasitic inductance of 0.18 nH represents the uncompensated portion of the inner lead protruding beyond the edge of the ground plane.

The line delay and the mutual parasitics between lines are proportional to the lengths of the traces. Therefore, each of these parameters varies within a range for a given package. Table V lists the values obtained by validated modeling.

Table V
Signal Environment Parameters in DTAB

Parameter	DTAB-284	DTAB-432
Line delay, (ps)	50-60	80-100
Uncompensated inductance (nH)	0.18	0.18
Mutual inductance (nH)		
with 1st neighbor	0.61-0.88	1.1-1.6
with 3rd neighbor	0.13	0.14
Coupling capacitance		
with 1st neighbor	0.06-0.08	0.13-0.18
with 3rd neighbor	0.002	0.004

Power Environment. A power supply line on the tape should have the lowest possible impedance. Lower inductance will reduce Ldi/dt effects, and higher capacitance will help isolate the circuit from voltage fluctuations in the rest of the system. Therefore, a power supply line should be as wide as possible. Additional capacitive bypassing should be available as close to the device as possible.

Depending on the tape layout, a power supply line in a DTAB package can be as narrow as a signal line, or significantly wider, as shown in Fig. 17a. Available space and the criticality of the power line in question will dictate to the package designer the actual width. For typical situations in our experience, the power environment parameters are as shown in Table VI.

Table VI
Power Environment Parameters in DTAB

Parameter	DTAB-284	DTAB-432
Total line inductance (nH)	0.5	0.85
Total line capacitance (pF)	6	8
Sheet resistance (ohm/square)	0.5	0.5

Bypass capacitors may be placed in the locations shown in Fig. 18. Assuming a 0.070-inch-thick printed circuit board, the physical distances between the bypass capacitors and the chip pads are shown in Table VII. Comparing the physical distances separating the bypass capacitors and the chip pads in the DTAB packages with their counterparts in ceramic PGAs, it is clear that the differences are fairly insignificant. However, if a critical application does demand that bypass capacitors be closer to the chip, they can be mounted directly on the TAB frame. This approach has not been formally qualified, but its feasibility has been proven in the laboratory.

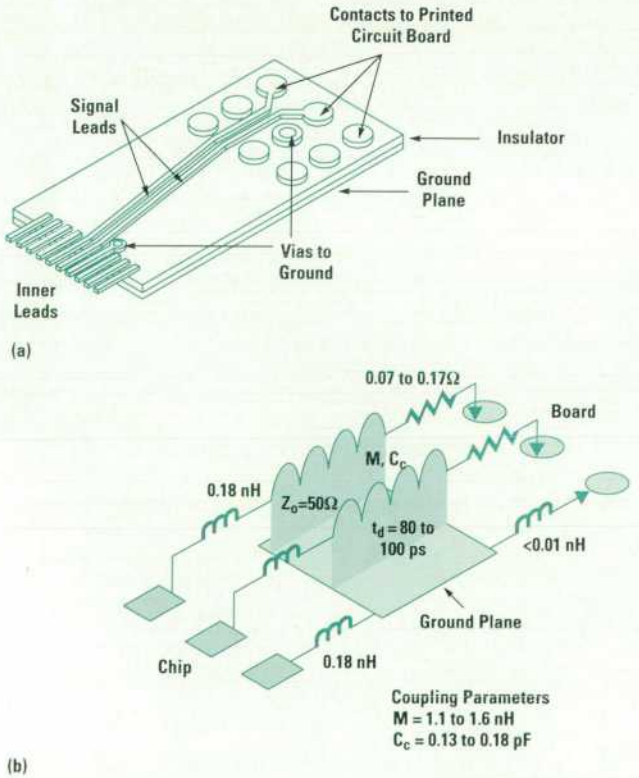


Fig. 16. Signal environment in the 432-lead DTAB package. (a) Structure. (b) Equivalent circuit.

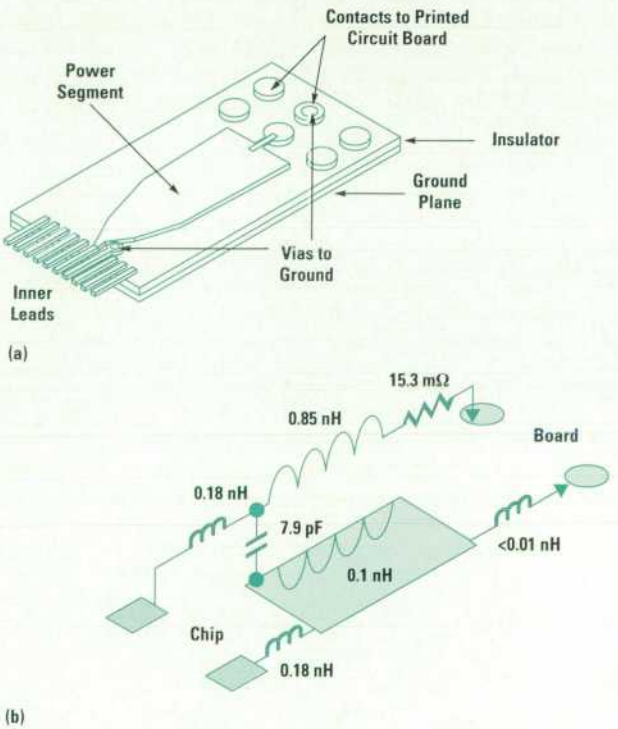


Fig. 17. Power environment in the 432-lead DTAB package. (a) Structure. (b) Equivalent circuit.

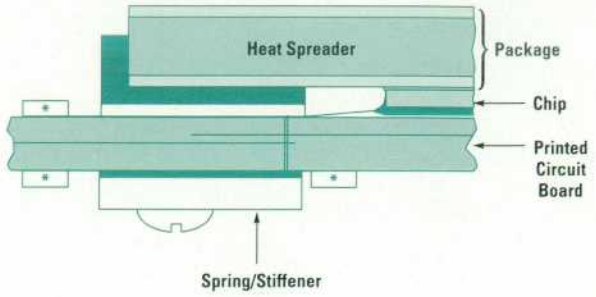


Fig. 18. Available locations (*) for bypass capacitors in DTAB packages.

Table VII
Distance to Bypass Capacitors

Parameter	DTAB-284	DTAB-432	408 PGA
Shortest trace (in)	0.454	0.499	0.435
Longest trace (in)	0.725	0.974	0.770

The DTAB concept offers additional options for enhancing the power environment—for example, segmenting the ground plane to reduce coupling between clean and dirty power sources and using a three-metal tape.

Table VIII lists some important electrical package parameters for DTAB-432 and its closest ceramic pin-grid array counterpart (408-pin CPGA).

Table VIII
Comparison between DTAB-432 and 408 PGA

Parameter	DTAB-432	408 PGA
Uncompensated line inductance (nH)	0.18	2-7
Line delay (ps)	80-100	300-400
Mutual inductance (nH)	1.1-1.6	0.5-0.7
Coupling capacitance (pF)	0.13-0.18	0.5
Total line inductance (nH)	2.50	4.6-6.0
Distance to bypass capacitor (in)	0.50-0.97	0.44-0.77
Sheet resistance (mohm/square)	0.5	12

System Performance

It was argued earlier that the parametric performance of a package is not a sufficient (although necessary) condition for adequate operation in a system, since the operation depends greatly on the engineering of the system. The system performance of DTAB has been evaluated in two challenging situations. The evaluations were conducted initially by simulation, and then the results were verified by direct measurements.

DTAB-432 in a 65-MHz System. A custom processor chip used in various HP computer products is normally housed in a 408-pin ceramic PGA. A simulation was conducted to predict its behavior in a DTAB-432 package at a clock rate of 65 MHz. The summary of the results is shown in Table IX, which compares the most pessimistic DTAB conditions to the most optimistic PGA operation. This simulated performance was subsequently validated by testing.

Table IX
DTAB-432 in a 65-MHz System

Parameter	Unit	408 PGA	DTAB-432
Cache address rise time	ns	4.2	3.9
Ground bounce energy	relative	1	0.47
Logic V_{dd} bounce	relative	1	1
Maximum crosstalk	relative	1	1.23

DTAB-432 in a 125-MHz System. A proprietary test chip was designed to operate at 125 MHz. A simulation was run to compare the operation of the chip housed in a DTAB-432 package and in a 408 PGA. Table X summarizes the results of those simulations. They indicate the overall superiority of the DTAB package in a system context. These results were also validated by measurements, as shown in the table. More details on this exercise can be found in reference 5.

Table X
DTAB-432 Performance at 125 MHz

Parameter	Unit	408 PGA	DTAB-432	
			Simulated	Measured
Reflected noise	mV	490	145	160
Crosstalk	mV	325	125	144
Dc IR drop	mV	102	23	
Signal delay time	ps	200	150	
Common-mode noise	mV	N/A	425	275
Differential-mode noise	mV	N/A	125	40

Thermal Considerations

The main thermal path in the DTAB package is through the heat spreader to the heat sink. A secondary thermal path exists through the TAB frame's leads and ground plane to the printed circuit board. The components of the main thermal path are the chip, the die-attach material, the heat spreader, and the heat-sink-attach material.

Although the material of the heat spreader (Cu-Mo-Cu) and its geometry were selected mainly to satisfy certain stringent mechanical requirements, the thermal environment of the package is at least equal to the best commonly used heat spreader materials. Table XI presents a comparative chart.

Table XI
Heat Spreader Materials

Material	Temperature Coefficient of Expansion ($\times 10^{-6}$/°C)	Thermal Conductivity (W/cm-°C)
Cu-Mo-Cu	5.07	2.08
Cu-Invar-Cu	5.3-6.7	1.88
Cu-Kovar-Cu	7.4-8.7	2.30
Cu-W	6.5	2.00
Note: Si	2.6	1.50

The die-attach material is an industry-standard silver-filled epoxy-based adhesive with a thermal conductivity of 0.02 W/cm-°C. The heat-sink-attach material is optional. In certain applications the heat sink is glued to the heat spreader with silver-filled epoxy; in others, demountability of the heat sink is required, and the epoxy is replaced with thermal grease.

The thermal performance of the DTAB package has been characterized at power dissipations up to 40 watts. Table XII lists the components of the thermal resistance of the DTAB package and their values measured at an air flow of 1.5 m/s.

Table XII
Thermal Resistances

Thermal Component	Thermal Resistance (°C/W)
Die attach epoxy	0.12
Heat spreader	0.08
Heat sink attach epoxy	0.08
Subtotal junction-to-case	0.33

Measurements conducted at various air flow rates are shown in Fig. 19 for a 432-lead DTAB package with a standard extruded aluminum heat sink. The change in the total thermal resistance is a result of the heat sink characteristics, while the thermal resistance of the package remains constant throughout the range of measurements. The DTAB package is compatible with virtually any heat sink or heat-sink-mounting configuration selected by the user.

Sometimes the power dissipation of the chip is sufficiently low that a heat sink is not required. In that case, the junction-to-air thermal resistance of the DTAB-432 package is 8.4 °C/W.

It is useful to compare these values with the best commercially available packages. For example, the expensive "slugged" 408-pin CPGA has a junction-to-case thermal resistance of 0.45 °C/W. The less-expensive slugless version has a junction-to-case thermal resistance of 1.3 °C/W when attached to a heat sink.

Reliability

To develop a reliable product in the shortest possible time, reliability testing has to start very early in the development phase and continue to be used as a development tool throughout the process. This is termed "developmental reliability testing" to distinguish it from the formal reliability qualification that takes place before the manufacturing release of a new technology.

For DTAB, possible failure modes were identified based on observations, experience, and theoretical reasoning. Certain stresses were defined to which the failure mechanisms were deemed particularly sensitive, and these stresses constituted the battery of developmental reliability tests. Table XIII presents a list of these tests and the conditions under which they were conducted.

Table XIII
DTAB Developmental Reliability Tests

Test	Code	Conditions
Temperature/humidity	85/85	85°C, 85% RH
Thermal aging	TA	105°C in air
Thermal cycling	TC	–55 to 85°C in air
Liquid thermal shock	LTS	–55 to 125°C
Corrosive atmosphere	CA	$H_2SO_4/SO_2/Cl_2/NO_2$ @25°C, 70% RH, 96 hr
Mechanical vibration	MV	Random, 15g, 0.06-inch amplitude
Mechanical shock	MS	2.5 ms, 6 axes

Table XIV shows a matrix of the potential failure mechanisms and the stresses designed to reveal them. The matrix demonstrates that all suspected failure modes are being addressed by at least one test, and, in some cases, by up to six tests. The test codes in Table XIV are defined in Table XIII.

When the development effort was deemed complete, a formal product qualification was performed according to the protocol summarized in Table XV.

In addition to these tests, a separate formal qualification of tape via integrity was conducted using specially designed coupons containing a total of 46,400 vias. The coupons were subjected to HAST (highly accelerated stress testing), LTS (liquid thermal stress), and high-temperature storage.

Alpha-Site Testing

After much of the development work was completed, two HP alpha sites were selected to demonstrate the operation of the package in real computer products. The products were selected to cover both through-hole and surface mount printed circuit assembly technologies, single and multiple VLSI assemblies, minicomputer and workstation products.

Bus Converter Board. This printed circuit assembly, used in a variety of computer products, is built on a 12-layer through-hole printed circuit board with nickel/gold surface finish. The bus converter chip is the only VLSI chip on the board. It is a 9-mm-by-9-mm NMOS device with 272 I/O pads, and is normally housed in a 272-pin PGA package. The bus converter chip operates at a 27.5-MHz clock frequency and an 8-MHz output rate, and dissipates 8 to 10 watts.

The strategy of the test was to package the bus converter chip in DTAB and redesign the bus converter board to accept the new package without disturbing the other components.

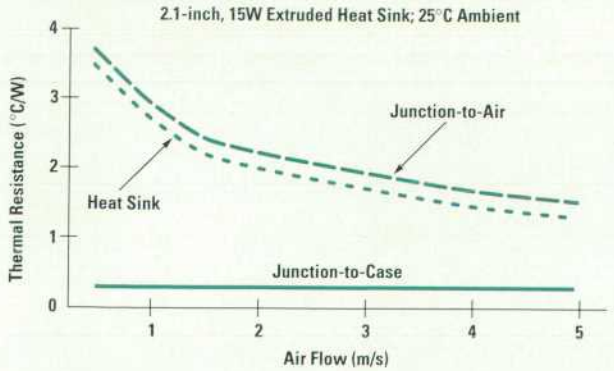


Fig. 19. Thermal resistances in the 432-lead DTAB package.

Table XIV
DTAB Stress/Failure Developmental Reliability Matrix

Failure	Stress						
	85/85	TA	TC	LTS	CA	MV	MS
Elastomer							
Relaxation		X					
Embrittlement		X			X		
Printed Circuit Board							
Flow		X					
Mechanical degradation			X	X		X	X
Contacts							
Diffusion	X	X					
Oxidation	X	X			X		
Corrosion					X		
Fretting	X		X	X		X	
Contamination	X				X		
Electrochemical phenomena	X						
TAB Tape							
Trace peeling	X				X		
Via failure	X						
Trace failure			X	X		X	
Chip							
ILB separation		X	X	X			
Encapsulation failure	X		X	X	X	X	X
Die attach separation	X		X		X	X	X

Five DTAB printed circuit assemblies were built and subjected to standard production testing and environmental qualification. The redesign of the board turned out to be a fairly simple task, and all tests were successfully completed.

Processor Board. This workstation processor board contains six VLSI chips, all of which are 9 mm by 9 mm and have 272 I/O pads. The chips are normally housed in 272-pin ceramic PGAs, and the board is all surface mount technology except for the PGAs. The system's clock rate is 30 MHz, and the maximum power dissipation per chip is 12 watts.

Ten DTAB printed circuit assemblies were built and exercised in operational workstations. They passed the full production test, and then were subjected to HP class C reliability testing, which they successfully passed. At the time of this writing, the assemblies have been functioning without failure in operational workstations for over two years.

Fig. 20 shows a photograph of the DTAB package used for both of the alpha-site exercises just described. As can be seen from the picture, the outer-lead contacts are arranged as a linear peripheral array rather than an area array. Peripheral DTAB is the original and more logical technical solution. Area DTAB was conceived as a response to customer concerns about the availability and cost of the fine-pitch printed circuit boards required for peripheral TAB. The package of Fig. 20 has 272 leads on a 400-micrometer (0.016-inch) outer-lead pitch. Peripheral DTAB designs with outer-lead contacts on 200-micrometer and 100-micrometer pitches were successfully used in other exercises.[6]

Table XV
DTAB Reliability Qualification Protocol

Test	Sample Size	Test Conditions
1. Package:		
HAST*	34	125°C, 85% RH, 14 psi, 5V bias, 96 hr
Thermal cycling	40	–55 to 150°C, 500 cycles
Liquid thermal shock	76	–55 to 150°C, 200 cycles
Thermal aging	129	110°C in air, 1500 hr
Mechanical vibration	32	Random, 15g, 0.06-inch amplitude
Mechanical shock	32	600g, 2.5 ms, 6 axes
Ozone exposure	30	
2. Package-to-Board Connections:		
Contact resistance#	6	
Mating/demating#	6	20 cycles
Liquid thermal shock#	6	–55 to 105°C, 25 cycles
Thermal aging#	6	110°C, 1 wk
Industrial environment#	6	Cl_2, SO_2, NO_2, H_2S, 1 wk
Moisture resistance	6	25-65°C, 80-98% RH, 2 wk
Insulation resistance	6	100Vdc, 1 GΩ
Dielectric withstand	6	100Vac, < 0.5 mA

* Highly accelerated stress testing

\# Tests performed sequentially on the same sample of 6

Extensibility of DTAB

The outer-lead pitch chosen for the area array designs (1.625 mm or 0.064 inch) was driven by the need to maintain compatibility with relatively coarse-pitch printed circuit boards. As printed circuit technology improves, the pitch can be easily shrunk to as little as 0.635 mm (0.025 inch), which will produce a 1000-lead package of roughly the same size as DTAB-432. In the peripheral version, prototypes have been successfully fabricated and tested with outer-lead pitches as low as 0.1 mm (0.004 inch). The resulting packages were only about 3.5 mm larger than the chip itself.

Two major hurdles have been hampering the successful development of a commercially viable, cost effective multichip module technology. One issue is the inability to test and burn in the component ICs before to assembling the multichip module. The second is the difficulty of replacing faulty components on a finished assembly. Without an acceptable solution to these two problems, it is difficult to envision a multichip module technology with a high enough yield to make it commercially competitive.

The prevailing methods of assembling ICs on multichip modules (wire bonding, TAB, and solder bumps) are at the core of these difficulties. Wire bonding and solder bumps require

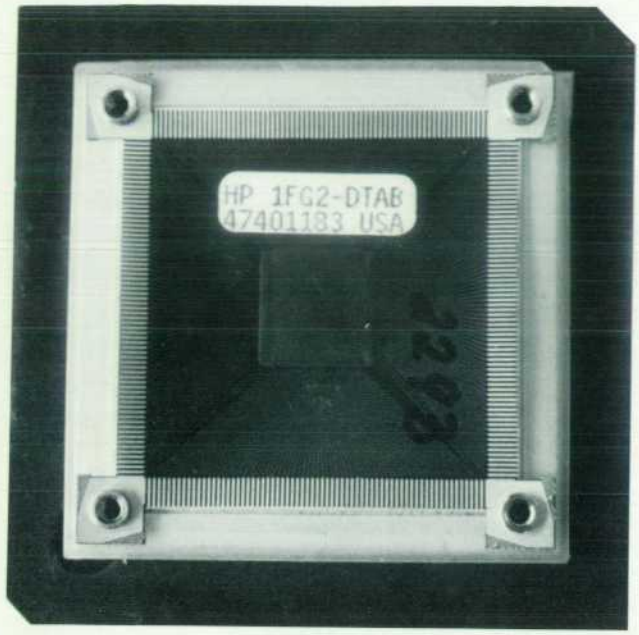


Fig. 20. 272-lead peripheral DTAB package.

obtaining bare chips from various IC manufacturers, but the techniques for testing and burning in bare chips are not yet available. TAB does provide a potential solution to the test and burn-in issue, but, like the other two methods, is not reworkable.

The DTAB concept presents a clear and easy solution for both test/burn-in and rework. Therefore, in addition to being a competitive single-chip packaging approach, it represents a very promising multichip module enabling technology and a migratory path from single-chip to multichip assembly.

Conclusions

Demountable TAB (DTAB) is a new packaging technology that capitalizes on TAB's advantages while avoiding its main drawbacks. The DTAB concept generates significant benefits in the areas of system design, manufacturing, and support. Electrically and thermally, the DTAB package compares very favorably with any existing high-performance package. Its performance has been proven at clock frequencies up to 125 MHz and power dissipations up to 40W.

DTAB packages were fabricated and tested with outer-lead area-array pitches of 1.6 mm and peripheral pitches of 0.4 mm, 0.2 mm, and 0.1 mm. The area-array version has successfully passed formal qualification.

The DTAB concept offers not only a competitive single-chip package, but also an enabling technology for multichip modules.

Acknowledgments

Raj Pendse, Bahram Afshari, and Kevin Douglas were the principal engineers involved in the development of the DTAB technology. Raj developed the composite spring system, and Bahram was responsible for the overall architecture and mechanical design. Bruce Heflinger, Vivek Rastogi, Sam Burriesci, Pauline Prather, Ron Lackey, Hank Schade, and Ken Scholz made important contributions. Bruce formulated the tape and board design rules, Vivek developed the area-array concept and was responsible for tape design, Hank engineered the test and burn-in hardware, and Ken designed the developmental reliability test plan. Ravi Kaw was the principal electrical consultant and Pete Dawson the thermal expert. Russ Parker, Jerry Gleason, Dave Sangster, and Dave Halbert participated in the program in different phases. Pedro Engel and Wayne Schar designed the qualification program. Rob McCullough, Nick Nichol, and Hilmar Spieth performed the alpha-site testing. David Yanaga designed and evaluated the 125-MHz system. Marcos Karnezos provided invaluable guidance and support for the program since its inception.

References

1. J.W. Jacobi, *Process for Bonding Integrated Circuit Components*, U.S. Patent 4 842 662, June 27, 1989.

2. J.L. Deeney, et al, "TAB as a High-Lead-Count PGA Replacement," *Proceedings of the 1990 International Electronic Packaging Conference*, Marlborough, Maryland, 1990, pp. 660-669.

3. R. Holm, *Electric Contacts: Theory and Applications*, Springer-Verlag, 1981.

4. R. Pendse, "Low-Relaxation Elastomeric Pressure Contact System for High-Density Interconnect," *Proceedings of the 1991 International Electronic Packaging Conference*, San Diego, California, 1991, pp. 750-758.

5. R. Kaw, D. Yanaga, and F. Matta, "Multimetal DTAB Package for a 125-MHz Computer," *Proceedings of the 1992 Electronic Components and Technology Conference*, San Diego, California, May 1992, pp. 450-457.

6. L. Hanlon, "A Readily Pretestable, Reworkable MCM," *Proceedings of the 1992 Electronic Components and Technology Conference*, San Diego, California, May 1992, pp. 321-325.

Bibliography

1. F. Matta and K. Douglas, *Demountable Tape Automated Bonding*, U.S. Patent 5 053 922, October 1, 1991.

2. F. Matta, B. Heflinger, R. Kaw, and V. Rastogi, "High-Performance Packaging at Competitive Cost Using Demountable TAB," *Proceedings of the 1991 International Symposium on Microelectronics*, Orlando, Florida, 1991, pp. 324-327.

3. R. Pendse, B. Afshari, B. Heflinger, F. Matta, and V. Rastogi, "DTAB: A New Path for TAB Technology," *Proceedings of the IV International TAB Conference*, San Jose, California, February 1992, pp. 9-25.

4. F. Matta, B. Afshari, K. Douglas, and R. Pendse, "Demountable TAB Aids Rework and Repair," *Connection Technology*, December 1991, pp 26-30.

5. F. Matta, B. Afshari, K. Douglas, and R. Pendse, "Demountable TAB: A High-Performance Low-Cost Package," *Proceedings of the 1991 International Electronic Packaging Conference*, San Diego, California, 1991, pp. 1036-1046.

6. B. Afshari, B. Heflinger, F. Matta, and R.D. Pendse, "Demountable TAB; Improving Manufacturability of TAB," *Proceedings of the 1991 International Electronic Manufacturing Technology Conference*, San Francisco, California, 1991, pp. 1-5.

7. B. Afshari, F. Matta, and L. Hanlon, *Shearing Stress Interconnection Apparatus and Method*, U.S. Patent 5,133,119, July 28, 1992.

8. M. Karnezos, R. Kaw, L. Hanlon, and F. Matta, *Flex Interconnect Module*, U.S. Patent 5,065,280, 1992.

9. F. Matta and K. Douglas, *Demountable Tape Automated Bonding System*, U.S. Patent 5,142,444, August 25, 1992.

10. F. Matta, *Via-Less Two-Metal Tape-Automated Bonding System*, U.S. Patent 5,142,351, August 25, 1992.

The EISA Standard for the HP 9000 Series 700 Workstations

The EISA interface on the HP 9000 Series 700 workstations provides a high-performance, expandable architecture that allows peripherals using different I/O standards to communicate with the system on the same I/O bus.

by Vicente V. Cavanna and Christopher S. Liu

The Extended Industry Standard Architecture (EISA) was selected for the HP 9000 Series 700 workstations to meet the I/O system performance and expandability needs of HP customers. The I/O system provides the communication link between the workstation and its external peripherals. Many of today's computer applications require a high-performance I/O link. EISA complements the performance of the Series 700 processors, and provides this high-performance path.

EISA is a descendant of the Industry Standard Architecture (ISA).[1] More address lines, data lines, and control signals were added for EISA to enhance performance. EISA provides separate 32-bit address and 32-bit data buses, supports multiple bus masters efficiently, provides for enhanced DMA functionality, and defines a synchronous data transfer protocol for burst cycles. An EISA configuration utility allows automatic configuration of add-on I/O cards installed in the computer. A technical summary of the major EISA features includes:

- Full 32-bit memory address bus
- Full 16-bit I/O address bus
- Multilevel, round-robin centralized arbitration
- 33-Mbyte/s burst rate (four-byte—one double-word—transfer per EISA clock cycle operating at 8.33 MHz)
- Efficient support of multiple, intelligent bus masters
- Enhanced DMA support for high-performance, inexpensive DMA devices
- 12 programmable edge-triggered or level-triggered interrupt lines
- DMA controller that has:
  - Four cycle types (ISA-compatible, Type-A, Type-B, and Type-C)
  - Three transfer modes (single, block, and demand)
  - Programmable transfer sizes (8, 16, and 32 bits)
  - Channel modes (autoinitializing, buffer-chaining, and ring-buffer)
- Programmable interval timers
- Vectored interrupt controller
- Automatic data bus translation that can handle:
  - Mismatched size (8, 16, and 32 bits)
  - Mismatched cycle type (system, ISA, EISA, burst, and nonburst)
- Arbitrary atomic transactions via LOCK mechanism
- Automatic card configuration
- Full compatibility with ISA cards.

Typically, there are many basic I/O capabilities that are built-in features of a computer. Many customers have unique requirements, so their needs may require additional I/O capabilities. The EISA I/O expansion bus gives customers the flexibility to add functionality to their computers.

Why HP Chose EISA

The primary I/O bus objective for all of the new HP workstations is to converge rapidly to a single industry-standard I/O expansion bus. HP selected EISA for the HP 9000 Series 400 and Series 700 workstation families because it is an industry-standard, high-performance bus that meets the needs of HP customers and the goal of a single bus for new workstation platforms.

EISA was built upon the ISA standard to provide compatibility with ISA cards. EISA is an electrical and mechanical superset of ISA which allows customers to tap into the vast array of ISA products. An industry-standard I/O bus like EISA allows HP customers to plug any investment of ISA or EISA cards into their EISA-based workstations, and be confident the cards will work given that the correct software drivers are available.

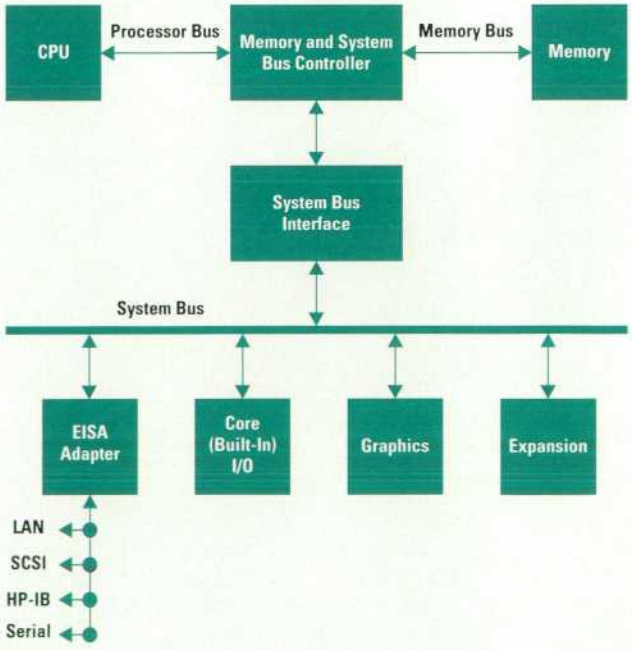


Fig. 1. System block diagram of the HP 9000 Series 700 workstation I/O subsystem.

HP cannot realistically design workstations to fill every market niche. Thus, some HP customers will take advantage of having EISA to customize their own workstation applications. In addition, business partners like system integrators, complementary-hardware vendors, and complementary-software vendors are willing to create solutions for HP customers. Typically, these solution creators are attracted to an open-bus architecture with a comprehensive set of hardware and software development tools. Many development tools are available through third-party ISA and EISA vendors. HP provides the EISA I/O services and routines to be used by all HP-UX* interface drivers written for ISA and EISA cards. An EISA open-systems toolkit is available from HP for developing drivers.

EISA Adapter

The EISA adapter resides, along with the built-in I/O devices and graphics, on a 100-Mbyte/s to 132-Mbyte/s system bus† (see Fig. 1). The system bus is a nonmultiplexed, 32-bit-address, 32-bit-data synchronous bus running at between 25 MHz and 33 MHz. This bus is one of the ports into the memory and system bus controller.[2] Communication on the system bus is always between the memory and system bus controller and another bus entity. No direct communication between other pairs of bus entities is allowed at the present time.

The EISA adapter connects the EISA bus to the system bus. The EISA adapter contains an Intel 82350 EISA chipset along with logic to interface this chipset, which was designed to interface to an Intel486 bus, to the system bus (see Fig. 2). The Intel chipset includes three chips: the 82352 EISA bus buffer, the 82357 integrated system peripheral, and the 82358 EISA bus controller.

The EISA adapter contains byte-swapping logic to overcome some of the problems associated with connecting a big-endian system to a little-endian system.†† The byte-swapping logic ensures that byte arrays are stored in memory in the same order (i.e., the same byte addresses) on both sides of the interface. The EISA adapter also contains an address translation mechanism, known as the I/O map, which maps an EISA page into a physical page in host system memory.

The EISA adapter also contains logic to scramble the portion of the EISA I/O address space that is used by the ISA expansion cards to allow the host's page-based access protection mechanism to be applied to protecting the ISA expansion cards from unauthorized access. This scrambling is done in the EISA address bus buffers.

The Intel 82350 EISA chipset contains all the functionality of a standard PC system. The chipset handles translations between the various EISA communicating entities so that the new EISA cards and the old ISA cards can communicate without knowing each other's protocol. The chipset has shared system resources such as DMA controllers, timer/counters, interrupt controllers, and arbitration controllers.

† The existing memory I/O controller limits throughput to two-thirds of the maximum bus speed.

†† In a little-endian system the LSB has the lowest address. In a big-endian system the MSB has the lowest address.

Host View of EISA Memory and I/O Space

The host CPU can do byte-level accesses to EISA. The system supports arbitrary address alignment of data transfers, but the CPU always does aligned transfers to EISA. This means that the address of data being sent to EISA must be a multiple of the size (in bytes) of the data being transferred.

Fig. 3 shows the addresses for EISA memory and the I/O space as seen from the host. The host uses addresses between FC00 0000 and FFBF FFFF to access EISA and ISA memory and I/O and the internal registers of the EISA adapter. Addresses FC00 0000 through FC07 FFFF are used to access EISA I/O space and the EISA adapters. The remaining addresses (FC08 0000 to FFBF FFFF) are used to access EISA memory. The host uses a portion of its EISA memory address range (FC10 0000 through FC4F FFFF) to access the I/O map entries. These address range restrictions are not a problem since the location of memory on the EISA cards is usually configurable.

Addresses FC00 0000 through FC00 FFFF in the host's EISA I/O space are reserved for EISA expansion cards and EISA system registers. This address range is sufficient for accessing any location in EISA's 16-bit address space, including ISA expansion cards. However, to access ISA cards via this address range is not allowed because the ISA access protection mechanism (described below) would be compromised. The hardware will return garbage on a read operation or ignore the transaction on a write.

The ISA expansion cards must be accessed via the host address range FC02 0000 through FC07 FFFF. This address range is mapped in a special way into the 16-bit addresses 0100 through 03FF of the EISA I/O space for exclusive use by the ISA expansion cards. This special mapping is done to avoid conflict with old ISA cards and to provide card-level access protection for the ISA expansion cards by using the system's page-level (4K bytes per page) access protection scheme.

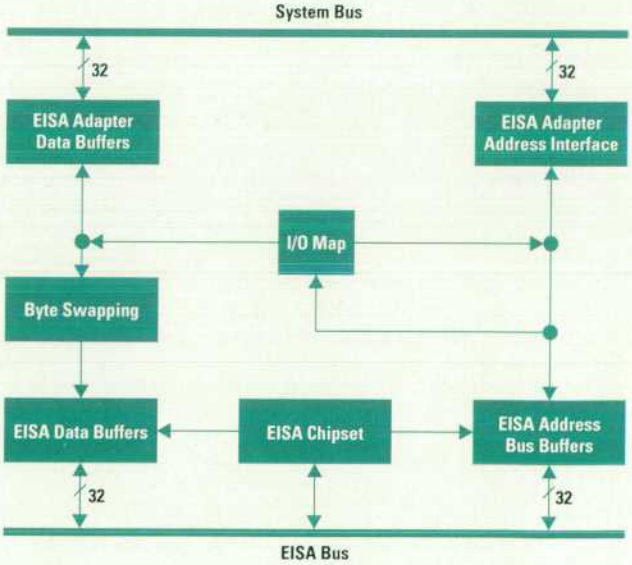


Fig. 2. Block diagram of the EISA adapter.

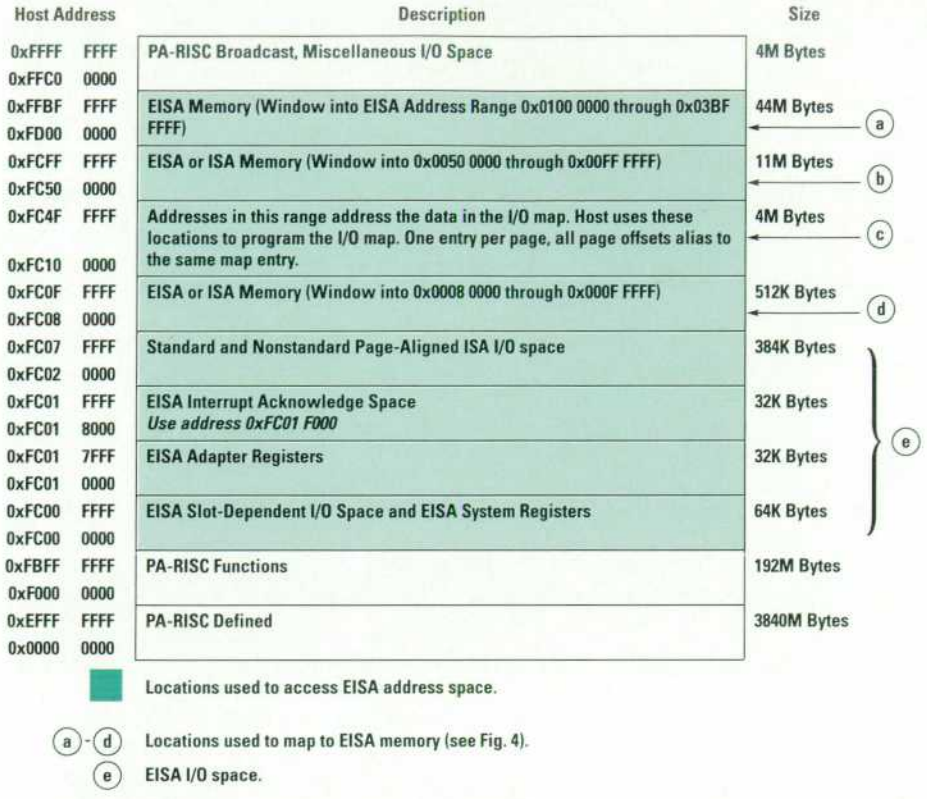


Fig. 3. Host's address map.

The protection scheme works by mapping each of the 96 sets of eight consecutive locations available for an ISA expansion card space to the first eight bytes of a 4K-byte host page. The assumption is that most ISA expansion cards use registers in multiples of eight bytes. If a card needs more than eight registers, then the card will need to be allocated more than one host page. This mapping is not done for the EISA motherboard devices (EISA slot 0) because the registers are scattered all over the address range. This mapping is also not done for the other EISA slot-specific addresses because they naturally fall on page boundaries.

EISA View of the Host Memory

EISA devices can do byte accesses to host memory with arbitrary address alignment. These devices see the host memory through an address translation mechanism known as the I/O map located at the EISA memory address range 0010 0000 through 004F FFFF (see Fig. 4). This map is a 1K array of 20-bit entries. The low-order 12 address bits of the EISA memory address specify the offset within a physical page in the host memory. The next 10 bits select one of 1K entries in the I/O map. Each map entry is a 20-bit address

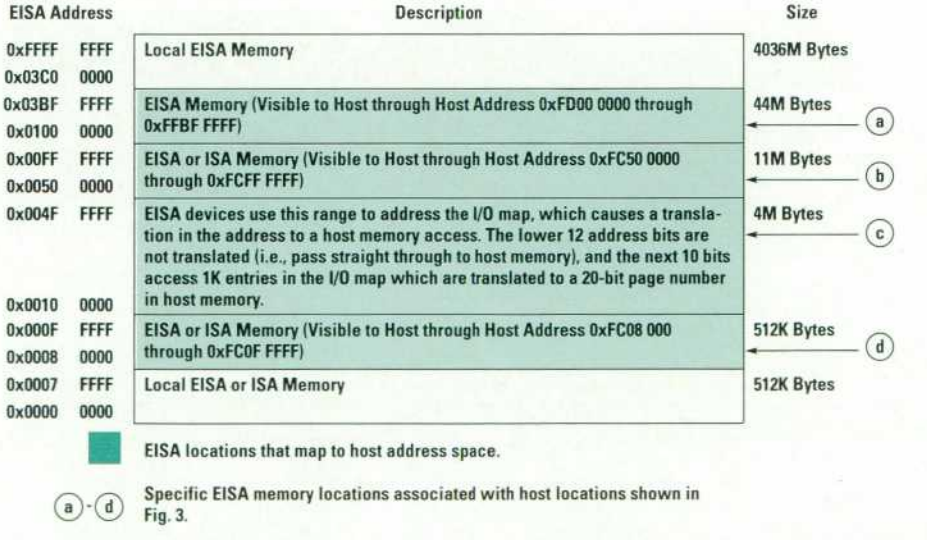


Fig. 4. EISA's address map.

that points to the corresponding physical page number in host memory (see Fig. 5). This mechanism allows scatter/gather accesses to host memory. That is, the driver can map a series of noncontiguous pages in system memory to contiguous pages in EISA memory so that a single EISA DMA transfer can transfer multiple system pages. This eliminates some of the need for DMA chaining (for large transfers) and its associated penalties (interrupt overhead, reprogramming DMA, etc.).

The address ranges shown as local EISA memory in Fig. 4 cannot be seen by the host.

A Note about EISA I/O Space

The ISA bus I/O space has always been 64K bytes (16 address lines), but the I/O address range is arbitrarily limited to 1024 locations. In addition, the first 256 bytes are reserved for devices on the EISA motherboard, with an address range 0000 through 00FF (LA[9:8] both equal to zero).† This leaves 768 bytes (96 sets of 8) for all of the expansion cards on the bus, with an address range 0100 through 03FF. This allows the old ISA expansion cards to ignore (and they do) all but the least-significant 10 bits of the 16-bit address.

To avoid conflict with the old ISA expansion cards, the new EISA expansion cards can only use addresses such that the least-significant 10 bits range between 0000 and 00FF, and LA[9:8] must both be zeroes. Thus, the address range corresponding to each EISA slot is fragmented by the address ranges that alias to ISA addresses. For example, in Fig. 6 in the address range for slot 1 (1000 to 1FFF), four subaddress ranges alias to 0100 to 03FF. The old ISA expansion cards think the EISA slot-dependent I/O addresses are motherboard addresses, since they do not look at the upper address bits.

The EISA expansion cards decode the full 16 address bits, and have available 1024 locations "sparsely populating" each EISA slot. A system can support up to 16 EISA slots.

Byte Swapping

In the PA-RISC architecture of Series 700 workstations, the most-significant byte is at the lower address (big endian), and in EISA, the most-significant byte is at the upper address (little endian).

† LA means latchable address and LA[9:8] are bits 8 and 9 of the LA bus.

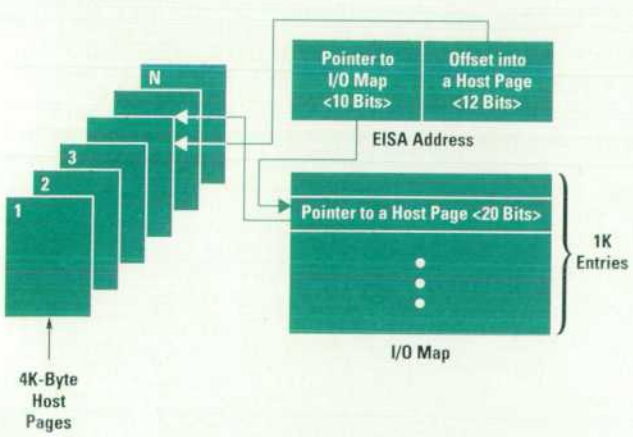


Fig. 5. Address mapping via the I/O map.

	I/O Address Range	I/O Range Reserved for
	0000-00FF	EISA or ISA Adapter
	0100-03FF	ISA Expansion Boards
	•	•
	•	•
	•	•
For EISA Expansion Boards	1000-10FF	Slot 1
	1100-13FF	Alias 0100-03FF
	1400-14FF	Slot 1
	1500-17FF	Alias 0100-03FF
	1800-18FF	Slot 1
	1900-1BFF	Alias 0100-03FF
	1C00-1CFF	Slot 1
	1D00-1FFF	Alias 0100-03FF
	•	•
	•	•
	•	•
	n000-n0FF	Slot n
	n100-n3FF	Alias 0100-03FF
	•	•
	•	•
	•	•
	nC00-nCFF	Slot n
	nD00-nFFF	Alias 0100-03FF

Fig. 6. A portion of the EISA 16-bit I/O address map.

Byte swapping hardware, which is provided on the EISA adapter board, ensures that byte arrays are stored in memory in the same order on both sides of the adapter. This makes it easy for devices like disks to be connected to the built-in SCSI port or the EISA SCSI card, and the data to be interchangeable without needing to know where the disk resided when it was read from or written to.

However, because of this swapping, any multibyte commands written to EISA devices must be preswapped by software. Similarly, any multibyte data structures placed in host memory to communicate with intelligent controllers on EISA must be preswapped.

Arbitration

Two arbiters exist in the system, the EISA arbiter and the system arbiter. The EISA arbiter was designed with the paradigm that it is the sole arbiter and controls all resources in the path to memory. The consequence of this decision is that the EISA arbiter commits irrevocably to its highest-priority client before it knows if it has all the resources it needs to access memory.

If the system arbiter operated the same way we could have a deadlock. The system arbiter could be designed to back off in a potential deadlock situation. However, for simplicity, in this version of the PA-RISC workstation, the system arbiter simply gives the EISA arbiter complete control over all resources in the path to memory and temporarily becomes a client to the EISA arbiter.

The system arbiter uses the CPU slot in the EISA arbiter's arbitration hierarchy. Once the system arbiter gains control it uses its arbitration algorithm to share its slot among its clients, the EISA arbiter being one such client. Thus, only one arbiter is in control of all resources at any given time. The arbiters hand control over to each other amicably.

Conclusion

The expandable architecture described here provides the high-performance characteristics of the EISA standard and a communication link to peripherals using different I/O standards. The article on page 83 describes in detail the hardware design for the current set of EISA I/O cards developed for the HP 9000 Series 700 workstations. The software design for the EISA SCSI card is described on page 97.

Acknowledgments

Thanks to Dave Roberts and Hosein Naaseh for helping us understand the EISA adapter and its role in the Series 700 architecture.

References

1. *Extended Industry Standard Architecture*, Revision 3.10, 1989, BCPR Services, Inc.
2. D. Li and A. Gore, "HP 9000 Series 700 Input/Output Subsystem," *Hewlett-Packard Journal*, Vol. 43, no. 4, August 1992, pp. 26-33.

EISA Cards for the HP 9000 Series 700 Workstations

The EISA specification's high-performance, burst-cycle protocol for data transfer is provided on the Series 700 EISA cards through the implementation of DMA and EISA bus master interfaces.

by David S. Clark, Andrea C. Lantz, Christopher S. Liu, Thomas E. Parker, and Joseph H. Steinmetz

Besides providing EISA capabilities to the HP 9000 Series 700 workstations, the EISA adapter described on page 79 provides the facilities for connecting several EISA cards with different front-end I/O protocols to the Series 700 I/O bus. At the time of system release four HP EISA cards were available for the HP 9000 Series 700: an EISA LAN card, an EISA HP-IB card, an EISA SCSI card, and an EISA PSI (programmable serial interface) card (see Fig. 1).

Each project team working on the EISA cards for the Series 700 had its own project-specific objectives, but the common objectives shared by all were to:

- Provide high-performance add-on I/O (EISA) solutions for the Series 700 workstations
- Design low-cost EISA solutions without compromising quality, reliability, and performance
- Meet all workstation development milestones.

The EISA specification was relatively new at the time we started investigating and proposing different architectures for the four EISA cards described here.

EISA specifies a burst-cycle (8-, 16-, or 32-bit data transfers) protocol for transferring data. There are two primary methods by which an EISA card can take advantage of the burst cycles: through an EISA bus master† or DMA. During our investigation we found very few VLSI and ASIC chips available from vendors that had an EISA bus master or DMA solution integrated into a chip. We looked at all of the available and proposed chips and decided that they did not fit our requirements. The decision was made to implement the EISA bus master and DMA interfaces on our EISA cards with discrete logic using PAL and TTL devices. In addition, the EISA memory and I/O slave interfaces on our cards are implemented with discrete logic.

The EISA LAN and EISA SCSI cards use a technique called a *bus gasket* (described below) to implement an EISA bus master interface. Both cards take advantage of the power of their respective frontplane controller by adding logic between it and the EISA backplane to translate the controller signals

† A bus master transfers data to or from main memory using addresses under its control.

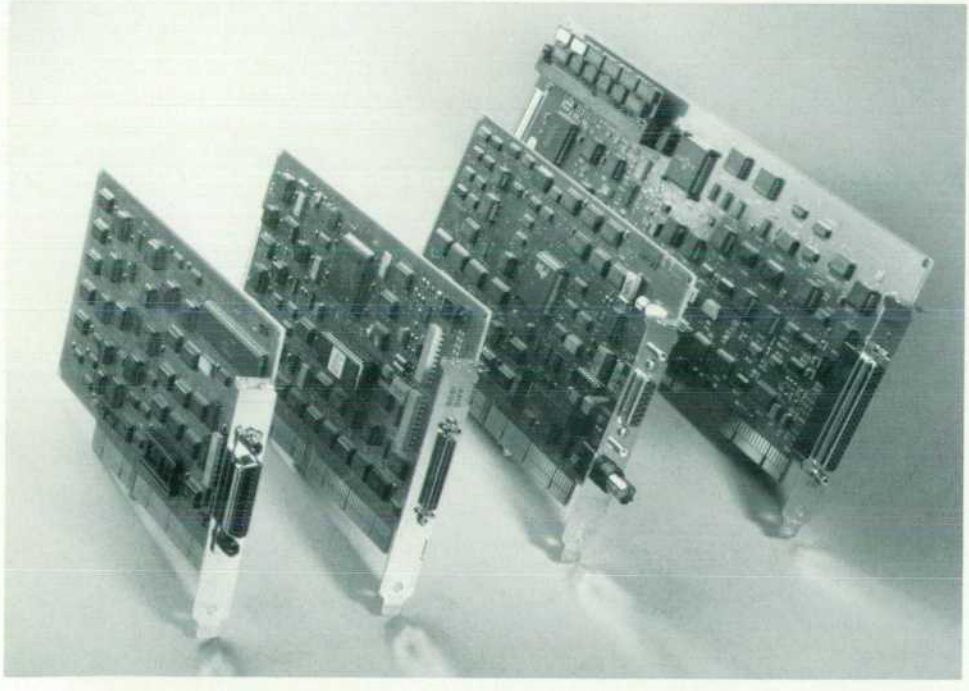


Fig. 1. HP EISA HP-IB, SCSI, LAN, and PSI cards for the HP 9000 Series 700 workstations.

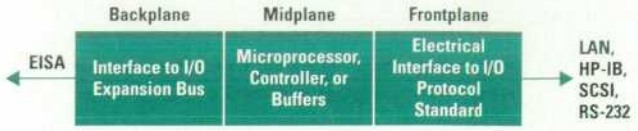


Fig. 2. Typical architecture for an I/O expansion card.

to EISA signals. The incremental cost to implement a bus master gasket circuit for these two cards is low compared to adding a separate VLSI bus master chip.

The EISA HP-IB and EISA PSI cards implement an EISA DMA interface. The EISA HP-IB card has a frontplane controller geared to DMA-type transfers, so naturally the DMA interface is the best method for this card. On the other hand, the EISA PSI card can use either method. The DMA interface was selected for the EISA PSI card because of its relatively simpler circuitry (compared to a bus master) and the nature of the card's protocol data structures.

The Bus Gasket

A bus gasket is an interface that combines a noncompatible processor bus with an expansion I/O bus (e.g., a Motorola 68000 microprocessor with EISA). The bus gasket handles all synchronization, pipelining, and control signal translations between the processor and the expansion I/O bus.

The processor on an add-on I/O card may be a microprocessor or an intelligent link-controller coprocessor. Examples of intelligent link controllers include the Intel 82596CA LAN controller and the NCR 53C710 SCSI controller. From the host system's point of view, a bus gasket interface appears as a standard bus master. From the on-card processor's point of view, the system memory appears as a local memory resource to be used as it would any other memory resource. The objectives of a bus gasket design are reduced complexity, reduced cost, increased performance, and increased utilization of the on-card processor.

The standard architecture of an add-on I/O card consists of frontplane, midplane, and backplane circuitry (see Fig. 2). The frontplane is the link-specific electrical interface and is defined by its standard. The midplane consists of a processor or a buffer that matches the synchronous nature of the backplane to the asynchronous bursty nature of the frontplane. And finally, the backplane is the interface to the expansion I/O bus.

When architecting a card for EISA, one of two data-transfer methods is typically chosen for high-performance cards: bus master or DMA. Many factors determine which to use for a specific card, but if the card needs to source addresses for system memory accesses and if the data transfer is of a scatter/gather nature, the bus master is generally the choice. If the data is more block-oriented and sequential in system memory, DMA is a more practical and low-cost solution.

When the choice is bus master, the cost can be high because of the increased complexity. A non-VLSI implementation of a bus master can be cost and board-space prohibitive, and an off-the-shelf interface chip can be expensive and generally requires a dedicated microprocessor to service it. However, if the card already uses a powerful link-controller coprocessor, its power can be harnessed to provide both cost and performance benefits.

The bus gasket, which is the logic that translates controller signals to EISA signals, provides the following benefits:

- Lower Cost. Bus gaskets can be cheaper than other implementations of a bus master interface. Bus master chips are available to create a "friendly" interface between the card midplane hardware and the bus backplane. However, because they are relatively dumb bus master controllers, these chips require an on-card dedicated microprocessor to control them. The additional complexity and cost of implementing one of these interfaces can be prohibitive in a cost-sensitive application.
- Better Use of the Link Controller. One of the primary benefits of bus gaskets is the ability to harness the power of an on-card link-controller coprocessor. The link-controller coprocessor can operate out of (host) system memory via EISA. On two of the EISA add-on cards, we require a bus master interface because the link-controller coprocessor follows command chains and data buffer chains that reside in system memory. The link-controller coprocessors used in the HP EISA 802.3 LAN and HP EISA SCSI cards are powerful protocol-specific processors. We wanted to make use of the power of these complex processors. If we had isolated them from EISA via a DMA interface or some other simple interface, we would have lost much of the power of the cards to manipulate structures in system memory.
- Less Complexity. The bus gasket reduces the complexity of the card by removing the need to have a microprocessor and associated firmware on the card. Also, since we are already using the bus arbitration and control mechanisms of the on-card link-controller coprocessor, the backplane bus master hardware can be greatly reduced. Some complexity does exist, however, inside the bus gasket itself.

Technical Hurdles

Two primary technical hurdles typically get in the way of implementing a bus gasket. The first problem is the difference in the way in which the link controller and the EISA backplane handle address and data bus cycles. In EISA addresses and data are pipelined—that is, they overlap—because EISA requires a valid (correct) address to be available one-half to one-and-one-half BCLK (clock) cycles before valid data can be captured by the coprocessor (see Fig. 3).

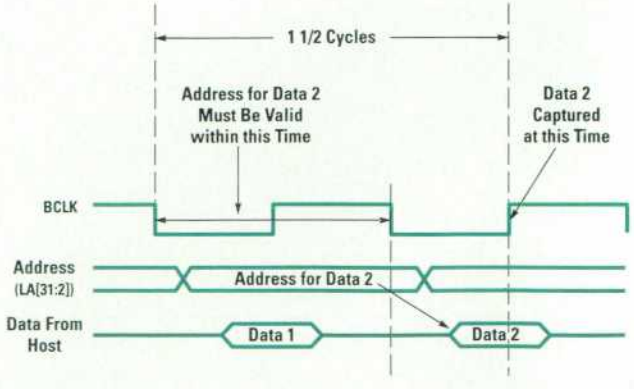


Fig. 3. Simplified timing diagram of EISA pipelining of address and data during read and write burst transfers. Note that the address of data 2 must be valid one-half to one-and-one-half BCLK cycles before data 2 becomes valid for capture.

The second problem is that the link controller's clock might be incompatible with the backplane clock. Typically, the clock rate of the link-controller coprocessor is required to be at least twice the rate of the backplane clock, and in the case of the EISA bus, its clock can be suspended to stretch cycles. This might cause some problems for a coprocessor that requires strict clock timing.

To solve these problems, some logic must be added to the card to synchronize the coprocessor's bus cycles, data, and addresses to the backplane bus. This logic is embodied in the bus gasket.

For the EISA bus gasket designs, pipelining was the major hurdle. The LAN controller used for the EISA LAN card is based on the Intel486 microprocessor which runs straight T1-T2 address/data cycles.† Without some additional logic there could be a deadlock situation when trying to run straight read burst cycles on EISA efficiently. One alternative would be always to insert an extra bus cycle after each read operation. This would give the LAN controller the chance to capture the data presented to it by the system in the current cycle, and allow it time to set up the transaction address for the next transfer. With this method, however, we could not make efficient use of EISA during mastered burst reads†† from memory and perform single-cycle burst transfers. Also, reads from memory would take three cycles.

Two methods can be used to solve this problem. The first is to use the processor's burst (cache-fill) mode. The second method is to implement a way of predicting the next address and verifying its correctness after the next address is presented by the controller. We chose the second method for performance reasons.

The HP EISA LAN Card

The HP EISA LAN card implements a bus master gasket design (Fig. 4). All mastered transfers generated by the card are initiated by the Intel 82596CA LAN controller. The 82596CA performs transfers in the form of Intel486-compatible microprocessor accesses. These accesses are transformed into EISA transactions by the bus master gasket circuits. The card performs all mastered transfers on the EISA bus as 32-bit burst memory accesses.

The 82596CA is driven by a 33.3-MHz oscillator. Since the EISA clock (BCLK) runs at 8.33 MHz, which is four times slower than the 82596CA's clock, a circuit on the card called the master controller synchronizes the data and address buses of the LAN controller and EISA using a signal called Rdy_N.

EISA read and write burst cycles are basically one BCLK period, but as shown in Fig. 3 the address can potentially overlap the previous or next transfer cycle by one-half a BCLK cycle. This is because the address contained on lines LA[31:2] must be valid on the EISA bus relative to the falling edge of BCLK and before the rising edge of the next BCLK cycle that starts the data transfer. Likewise, the subsequent transfer will overlap by one-half cycle as it prepares the LA[31:2] bus for its correct transfer address.

This poses a problem during mastered burst reads because EISA burst timing requires that LA[31:2] for the next read transfer be valid before the data from the current read access is available (read returned data on EISA is not guaranteed to be valid until the rising edge of BCLK). This is not a problem for burst writes because the coprocessor does not have to wait for incoming data.

To handle the problem for mastered reads, a scheme called address prediction is used to compute the addresses the 82596CA will use during burst read transfers. A circuit on the card called the address generator (see Fig. 5) contains logic to predict the next address, drive the address onto the EISA bus, and verify that the address is correct from the previous data transfer.

Fig. 6 shows a snapshot of the timing and signals associated with this address prediction scheme for a burst read. At ⓐ the predicted address 5 for the current transfer is driven onto the LA[31:2] bus lines and the 82596CA LAN controller is

† A data address is available at time T1 followed by data at time T2.

†† Mastered burst reads are 8-, 16-, or 32-bit transfers from host memory when a device is bus master.

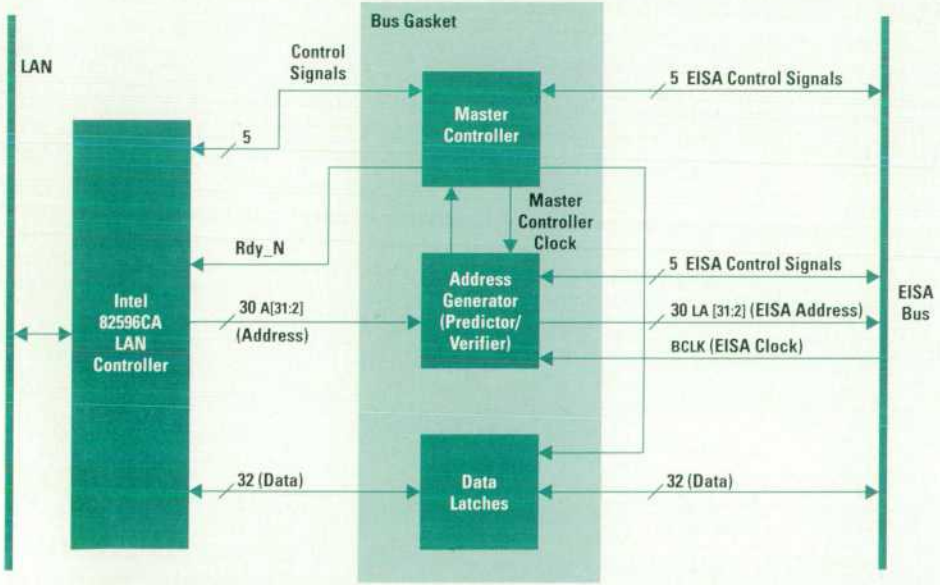


Fig. 4. Block diagram of the HP EISA LAN card.

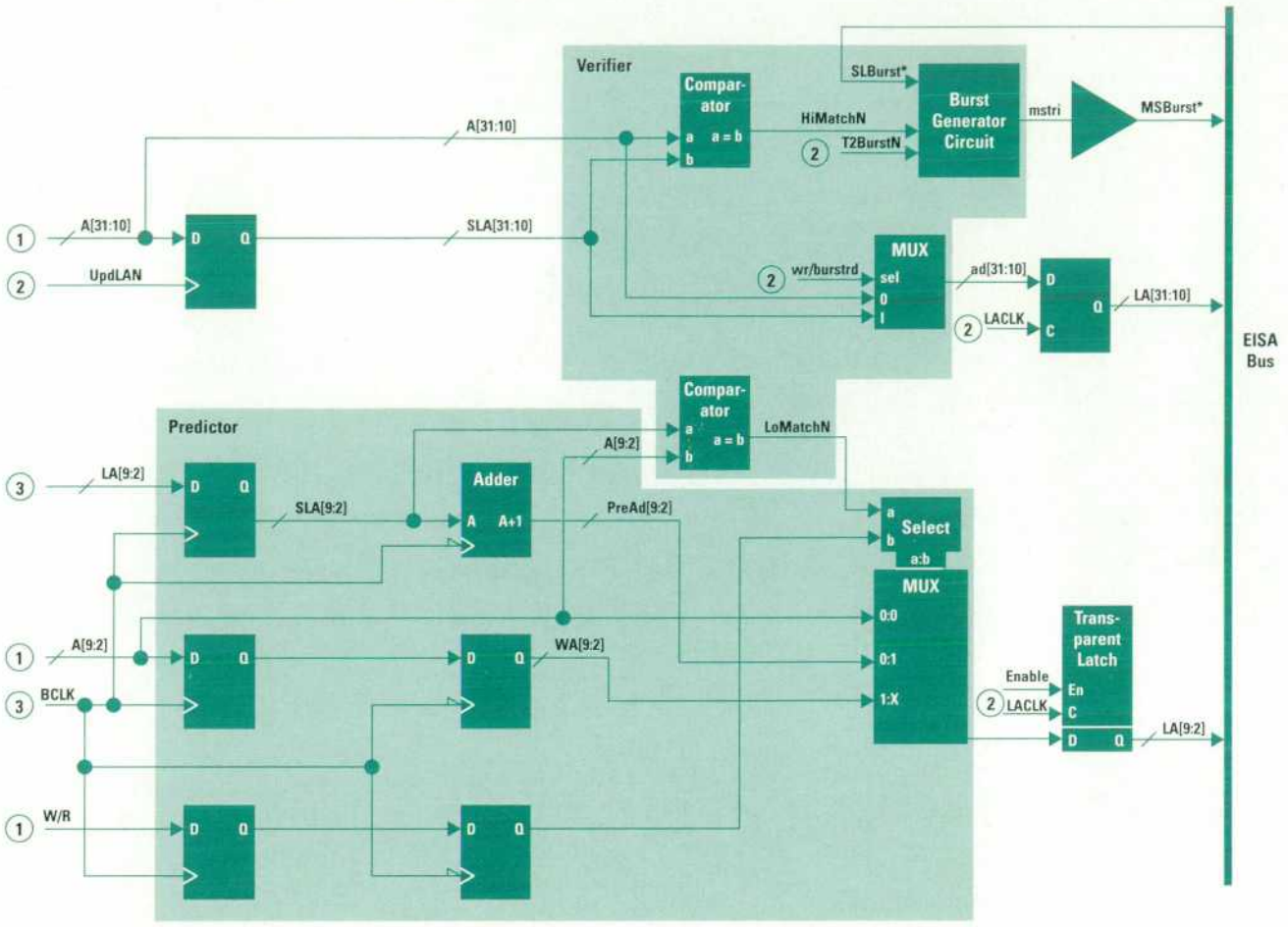


Fig. 5. Address generator for the HP EISA LAN card.

preparing to capture the data from the previous read transfer. At ⓑ, on the rising edge of BCLK, the data from the previous transfer at address 4 is captured by the card. If the predicted address of the previous transfer compares with the address produced by the 82596CA (in this case 4), the 82596CA is stepped by pulsing the Rdy_N signal for 30 ns.

Also during this time the predictor computes the next address (address 5). At ⓒ the 82596CA drives the current transfer address onto its lines. From this point on, the verify, prediction, and Rdy_N operations continue, starting with the verification address being 5 and the predicted address 6.

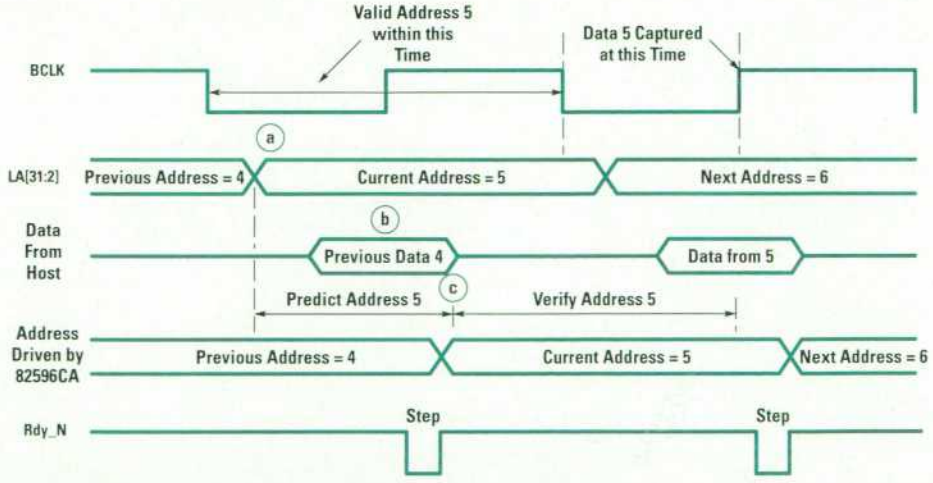


Fig. 6. Timing diagram showing the events associated with address prediction and verification.

If a predicted address does not match the 82596CA address, appropriate action is taken by the master controller and the correct transaction is executed. A predicted address miss can cause the loss of one to three BCLK cycles depending on the types of predictor misses occurring.

Generating Addresses

The address predictor and verifier logic shown in Fig. 5 generates the next address based on the current transfer address during read bursts. The predictor is loaded with the start address at the beginning of a data transfer initiated by the 82596CA LAN controller. Subsequent predicted addresses are generated by the predictor incrementing the current address for each transfer. Each predicted address is placed on the LA address bus before the 82596CA is able to generate the actual address. Standard EISA read transfers and standard and burst write cycles do not require address prediction.

At the start of a standard or burst read or write transfer sequence, the 82596CA loads the predictor with the initial transfer address. The first transfer cycle after being granted the bus is always a standard EISA transfer. A standard transfer must precede all burst transfer sequences. During standard read transfers, the address from the 82596CA (A[31:2]) is latched and driven as LA[31:2] on the EISA bus. Addresses generated during writes are copied directly from the address bus of the 82596CA and latched and not driven onto the EISA bus until the appropriate EISA cycle and the presence of the Rdy_N signal.

After the system has granted the bus to the card and the 82596CA asserts the signal that indicates that the first address and the direction signal (W/R) are valid on its pins, the first address cycle begins. If the 82596CA is starting a write, the predictor and verifier are disabled.

If the direction of transfer is read, the predictor and latching registers are loaded with the address present on A[31:2] and the LA[31:2] bus is actively driven on the falling edge of BCLK just before the EISA Start* signal begins the address cycle. In the next cycle (the data cycle), read data will be valid on the rising edge of BCLK at the end of the cycle. Also during this data cycle, the address for the next transfer must be valid.

After the first transfer, standard transfers and burst writes proceed as normal. However, burst reads require each subsequent address to be predicted because EISA requires a lead time for addresses.

The combined address multiplexer and latch shown in Fig. 5 are used to select one of three address sources to provide the next address to LA[9:2], the address of the transfer within a 1K-byte page.† The three sources are A[9:2] from the 82596CA, the predicted next address PreAd[9:2], and the write address source WA[9:2]. Three sources are required because to provide the correct address on LA[9:2], different values are required depending on the type of transfer being done and the current phase being executed within a transfer.

† EISA restriction to ensure that a memory module can be as small as 1K bytes.

Generating MSBurst*

MSBurst* is a tristate EISA signal driven by a card that wants to perform mastered burst transfers. MSBurst* must be asserted whenever there is a burst cycle address being driven on LA[31:2] and deasserted during the last burst transfer cycle.

MSBurst* is generated in the address predictor/verifier block because the result of the address verification is directly responsible for whether the next cycle can be a burst or not. To start a burst series, the SLBurst* signal from the slave (host memory) is sampled on the rising edge of BCLK at the beginning of the second cycle of a standard transfer. If SLBurst* is asserted, which indicates that the slave can burst, MSBurst* is asserted on the subsequent falling edge of BCLK as the address for the burst cycle is driven onto LA[31:2] (see Fig. 7).

Starting with the first transfer and continuing with each burst cycle, the page address A[31:10] of the next transfer is compared with the page address of the current transfer (SLA[31:10] in Fig. 5). If they do not match, MSBurst* will be deasserted on the falling edge of BCLK during the next burst cycle as the new page address is placed on the LA bus. This will begin a new standard transfer if the card still owns the EISA bus.

If during burst transfers the card is preempted by the system, MSBurst* will be deasserted on the falling edge of BCLK during the last burst transfer. The transfers will cease when the 82596CA deasserts Hold†† in response to the preemption. The master controller will force the deassertion of MSBurst* as it detects the deassertion of Hold. Under these conditions, the deassertion of MSBurst* will not necessarily occur in the current cycle. There is a programmable timeout that will determine how much data the card will continue transferring after preemption.

The HP EISA SCSI Card

The HP EISA SCSI card design is based on the idea of minimizing the component count and using as much of the protocol and features of the SCSI controller as possible. The bus

†† Hold is a signal that requests access to the local bus.

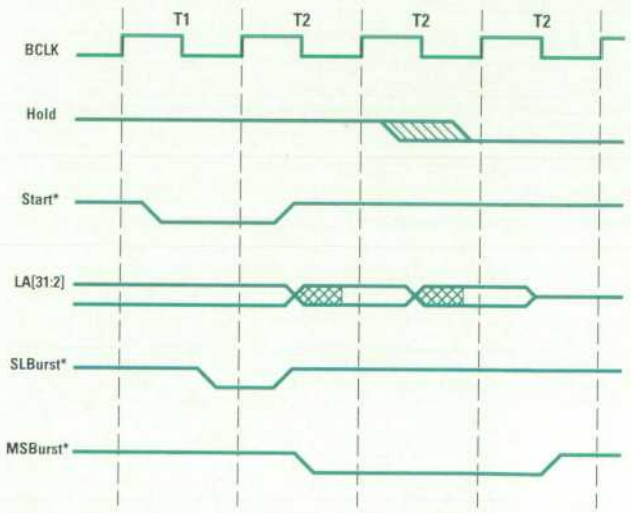


Fig. 7. MSBurst* timing diagram.

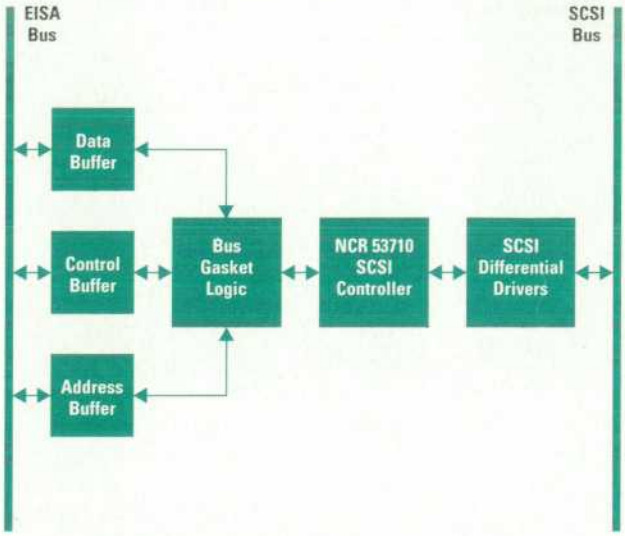


Fig. 8. HP EISA SCSI card block diagram.

master gasket approach described earlier is used to combine two different domains together with logic to translate signals from one to the other while meeting the functional and timing requirements of both.

In this case the two domains are EISA and the bus interface logic of the NCR 53C710 SCSI controller chip, which is configured to function as a 68040 microprocessor synchronous bus (see Fig. 8). Some of the differences between the SCSI controller and EISA that are handled by the bus gasket logic are given in Table I.

Table I
Signal Differences Between EISA and the NCR 53C710 SCSI Controller Chip

EISA	NCR 53C710
A[31:2] – Address bits	A[31:0] – Address bits
BE[3:0] – Indicates which bytes are valid at the current address	SIZ[0:1] – Indicates data width at the current address
Pipelined address	Nonpipelined address
Bus Control Logic	Bus Control Logic
Start* – Indicates address of current I/O transaction	TS – I/O transfer start (address available)
EXRDY – Indicates wait state (I/O transaction is done at the end of the wait state)	TA – Transfer acknowledge (I/O done)
EX32 – Indicates 32-bit-wide data path	

The NCR 53C710 SCSI controller was selected for the EISA SCSI card mainly for software compatibility with the built-in SCSI port of the Series 700 workstations. The EISA SCSI software is described in the article on page 97.

The NCR 53C710 SCSI controller provides several features that made it attractive for our implementation. First of all the controller runs from two clocks: one for the SCSI protocol and one for the host bus interface, with the synchronization between the two domains occurring inside the chip. Working with NCR, we were able to get an internal bus request signal brought out of the chip that deasserts when one transfer is left. Without this signal the EISA burst cycle would not have been possible. Finally, the NCR 53C710 provides a 10-Mbyte/s 8-bit SCSI, which is an added value over the built-in SCSI (5-Mbyte/s) on the workstations.

Clocking. A clock synchronization scheme is primarily determined by the types of interfaces that are being connected together. In this case, the EISA bus control signals and data are synchronous with the EISA bus clock (BCLK), which unfortunately does not have a fixed frequency nor does it have a guaranteed duty cycle from period to period. The 68040-type interface of the 53C710 has all of its signals relative to its clock.

The simplest answer to this problem was to connect the EISA clock BCLK to the host bus interface clock so all signals would be synchronized with the EISA clock. Unfortunately, this would not work because of the different protocols and the need to burst longer on EISA (for performance reasons) than the burst length built into the 53C710.

Another option was to run the controller from a different clock and synchronize it with the EISA clock domain. This would not work because the timing losses in the synchronization would degrade the throughput of the card and not allow EISA to operate optimally. Another scheme considered was to generate a clock from a phase-locked loop circuit to feed into the 53C710. This was not chosen because the EISA clock operates from 6 to 8.33 MHz and can stretch, creating a nonoptimal duty cycle which could create a problem for the phase-locked loop circuit.

The solution chosen was to create a 2× multiple of the EISA clock and present this new clock to the 53C710. This allows the 53C710 to run in nonburst mode (larger transfer size) and be synchronized with the EISA bus.

Address Generation. Sourcing the EISA address was a major area of concern because of the pipelining behavior of the EISA bus in burst mode. To provide addresses for reads from a SCSI peripheral (writes to EISA memory), the 53C710 is allowed to run ahead of the EISA bus while the data is latched. This effectively pipelines the next address from the 53C710. For writes to a SCSI peripheral (reads from EISA memory), the 53C710 cannot be allowed to run ahead because the EISA data is not valid until the end of the clock cycle. For this case, an address predictor (counter) is used to first latch then pipeline the EISA address, which will always be sequential within any single bus tenancy.† The control logic for the address prediction circuit must also know when to increment the address correctly based on inputs from the EISA slave being accessed.

Control Signals. Some of the EISA control signals were not very difficult to handle since they had equivalent signals in the 53C710 domain. For example the EISA Start* and the 53C710 TS (transfer start) signals both indicate the beginning

† Bus tenancy is the length of time a device is in control of and using the EISA bus.

of a cycle (or the address phase). However, the EISA MSBurst* signal, which is asserted when a card wants to perform mastered burst transfers, has to be generated because there is no 53C710 equivalent. In addition, the MSBurst* signal must be deasserted to ensure that the card cannot burst across a 1K-byte boundary (per the EISA specification), burst across the boundary created by the address predictor (memory reads only), or burst when the data being transferred is less than a double word.

The address predictor burst boundary is created by the fact that the counter only loads the first address of a transfer while in the first address phase of the 53C710. From then on, the address is incremented based on the EISA EX32* and EXRDY signals and cannot be reloaded. Therefore, when the count goes to all ones, the 53C710 must back off, release the EISA bus and rearbitrate for the EISA bus.

The less-than-32-bit boundary is created by the 53C710. For example, when the controller is completing a transfer and the last transfer is three bytes long, the transfer will be split into two-byte and one-byte moves. This will create problems for the address prediction logic because it will increment the address between these transfers and place the last byte in the wrong location. In this situation, the 53C710 will back off, release the EISA bus and rearbitrate for the EISA bus.

EISA and DMA

A centralized DMA controller is provided on the EISA adapter board. This architecture simplifies the design of the DMA portion of any add-on DMA card because the complex functionality associated with handling DMA transfers is concentrated on the system end of the bus. The DMA controller on the EISA adapter board is the master of the DMA devices connected to the EISA bus because it services their requests. A DMA device can be configured in a combination of different ways to optimize its performance on EISA.

DMA devices can be programmed for data transfers using one of four DMA cycle types available on EISA. The ISA-compatible cycle allows ISA DMA devices to operate without any hardware or software modifications. These devices will have a higher data transfer rate on EISA than ISA as a result of EISA's efficient bus arbitration. The Type-A and Type-B DMA cycles are EISA modes that allow some ISA DMA devices to achieve even higher performance with special software modifications. The burst Type-C DMA cycle is the highest-performance EISA DMA cycle, and is only available for EISA DMA devices designed specifically to transfer data every clock period.

EISA provides seven ISA-compatible DMA channels. Any DMA device on EISA can be assigned to use one of these channels. The channels available are channel 0 through channel 3 and channel 5 through channel 7. Channel 4 is the cascaded channel and is reserved for system use. Also, each channel can be programmed to support 8-bit, 16-bit, or 32-bit data transfers.

The Intel 82357 integrated system peripheral (ISP) provides the centralized EISA DMA controller capabilities. The ISP resides on the EISA adapter board. The DMA controller, refresh controller, CPU, and bus masters all share the EISA bus. A device requesting use of the bus must assert its bus request signal to the centralized arbitration controller (also contained in the ISP). This arbiter will assert the corresponding bus grant signal when the bus is available.

For a DMA device, the bus request signal is DRQ<x>, where x represents the DMA channel number. All the DRQ<x> signals are fed to the DMA controller, and the DMA controller acts as the intermediary by arbitrating for the bus on behalf of the DMA channels. When the DMA controller has been granted the bus, it asserts the acknowledge signal (DAK*<x>) for the appropriate DMA channel.

Depending on the DMA arbitration sequence selected on EISA, the channel priority can be quite different. In the fixed DMA priority arbitration sequence, each channel has a different priority level, with channel 0 being the highest and channel 7 the lowest. In the rotating DMA priority arbitration sequence, there are two levels of priority. The top level uses a four-way rotation to grant bus access sequentially to channels 5, 6, and 7. The fourth position is channel 4, but this is used as a cascade port for the channels at the lower level. The lower level also uses a four-way rotation to select sequentially from channels 0, 1, 2, and 3.

The DMA controller supports three types of EISA DMA data transfer modes: single, block, and demand modes. In the single mode, a DMA device will perform one transfer for each arbitration cycle. In the block mode, a DMA device will perform a block of transfers for each arbitration cycle. In the demand mode, a DMA device will perform a group of transfers for each arbitration cycle, but it can suspend transfers temporarily by deasserting its DRQ<x> signal. Block and demand transfer modes can be preempted by other devices requesting the bus.

A DMA device is not allowed to add wait states during an EISA DMA data transfer since it is essentially a "third party" on the bus monitoring the transaction taking place between the bus master and the memory slave. In other words, the DMA device must follow the timing of the transfer already negotiated between the bus master and the memory slave. The bus master in this case happens to be the DMA controller, which has initiated and is controlling the transfer.

Two HP-designed EISA add-on cards support the burst Type-C DMA cycle using the demand transfer mode. The EISA HP-IB and EISA PSI cards both use DMA cycles to transfer data because the data structures are more block-oriented. Since the DMA controller is centralized and the DMA devices do not have to source bus addresses, this reduces the complexity of the cards.

The HP EISA HP-IB Card

The HP EISA HP-IB card is divided into three functional blocks: the EISA interface, the FIFO buffers, and the HP-IB interface (see Fig. 9). This card has a 16-bit Type-C DMA module designed to support the demand transfer mode for transferring data between system memory and the HP-proprietary 1TL1 HP-IB controller chip. The EISA HP-IB card can be programmed to use one of seven DMA channels. A 16-bit EISA DMA device can transfer data at rates up to 16.7 Mbytes/s using burst cycles.

The 1TL1 controller chip provides the complete logical HP-IB interface defined by the IEEE 488 (IEC 625) specification. It also has a local-host interface that appears as a set of

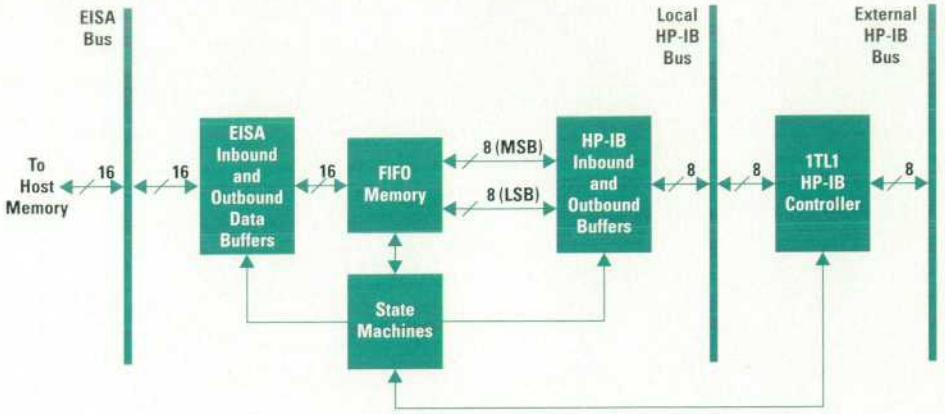


Fig. 9. HP EISA HP-IB card block diagram.

eight addressable registers and provides buffering for outbound and inbound HP-IB data transfers through two internal FIFO queues. The local-host interface of this chip is connected to the EISA interface and controlled by the local DMA state machine on the EISA HP-IB card.

The outbound direction is defined as data movement from system memory to the HP-IB. DMA transfers in this direction are performed with memory read cycles and I/O write cycles to move data directly from the system to the card. Conversely, the inbound direction is defined as data movement from the HP-IB to system memory. DMA transfers in this direction are performed with I/O read cycles and memory write cycles to move data directly from the card to the system.

FIFO Buffer. In the processor-to-peripheral data path, data transfers between components of different speeds can drastically reduce performance. Even with a DMA channel between the system and the peripheral, the processor must be ready to relinquish the bus on short notice, and go to an idle condition. In addition, the data-path reduction from a wide bus to a narrower bus will drop the data throughput by a significant factor.

A buffer helps to offload the burden the peripheral places on the processor during less-than-optimal transactions. The data transferred between the system and the peripheral will be padded in a first-in, first-out (FIFO) manner. The FIFO memory can perform this function.

To smooth out the DMA transfers, the EISA HP-IB card has a set of FIFO memories that act as an elastic data buffer between the faster synchronous EISA interface and the slower asynchronous HP-IB interface (see Fig. 10). Physically, this external FIFO buffer sits between the EISA interface and

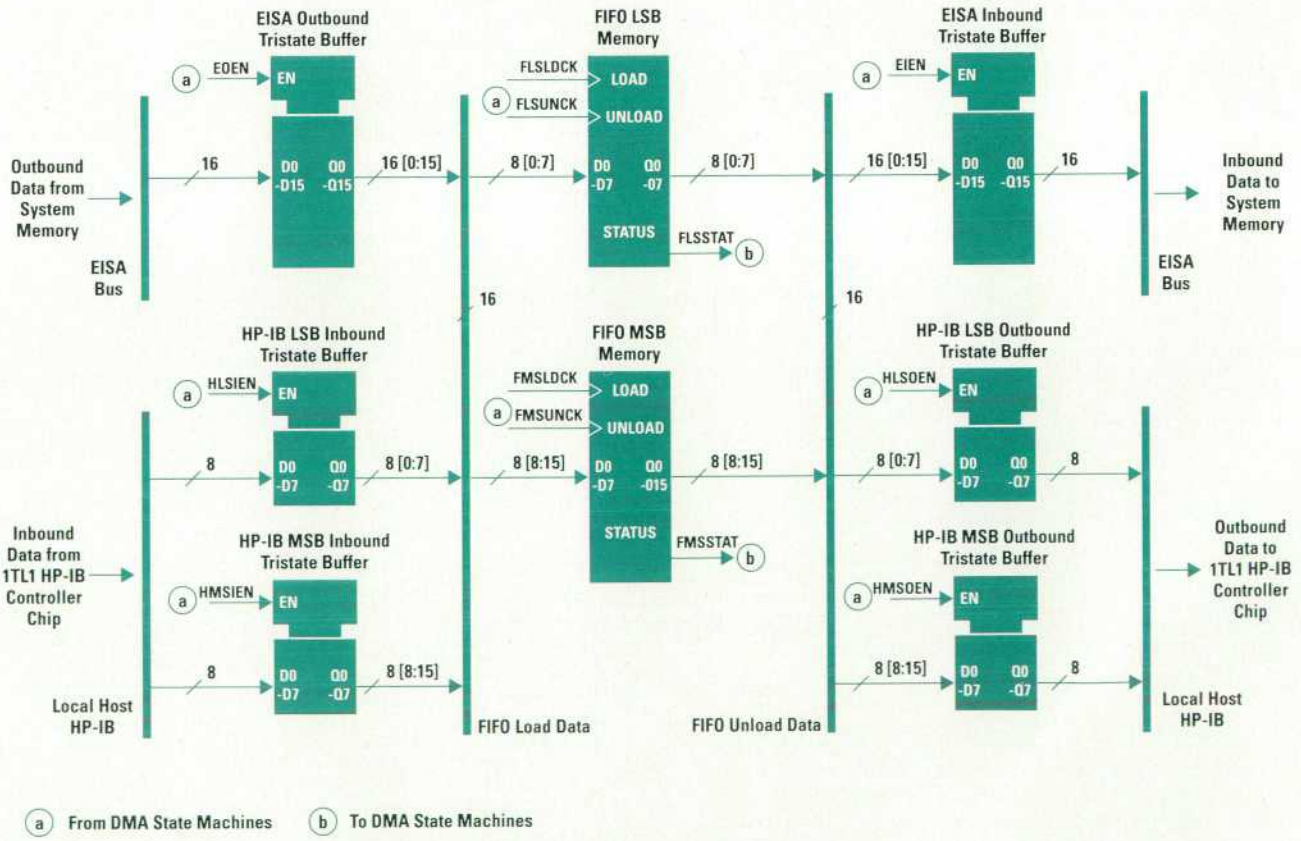


Fig. 10. FIFO buffers for the EISA HP-IB card.

the local-host interface of the 1TL1, and supplements the two eight-deep, byte-wide inbound and outbound FIFO queues in the 1TL1. In addition to the mismatch in bus speeds, there is a mismatch in data widths between the EISA and HP-IB interfaces on the card. The card has a 16-bit data connection to EISA, while the HP-IB interface is only an 8-bit bus. A direct 8-bit HP-IB interface to EISA cuts the EISA data throughput in half, when compared to a 16-bit bus.

During the investigation phase of the EISA HP-IB card project, a 32-bit EISA DMA architecture was proposed. However, the twofold increase in the complexity and cost to implement the FIFO buffer for a 32-bit architecture could not be justified for this card. Thus, the 16-bit architecture was selected, which met the project's performance and cost objectives.

The FIFO buffer has a 16-bit port on the EISA side and an 8-bit port on the HP-IB side. Logically, two 64-deep, byte-wide FIFO memories appear as a single 64-deep, word-wide FIFO buffer to the EISA interface. However, these two FIFO memories look like least- and most-significant byte data banks to the local-host interface. A separate 8-bit bypass port is available that allows the system direct I/O slave access to the 1TL1's control and status registers without having to go through the FIFO memories.

In addition to the local DMA state machine controlling the local DMA process between the FIFO buffer and the local-host interface of the 1TL1, there are two EISA DMA state machines controlling the EISA DMA processes between the EISA interface and the FIFO buffer (see Fig. 11). They are the outbound EISA DMA state machine and inbound EISA DMA state machine. Together, the three DMA state machines can handle the concurrent loading and unloading of the FIFO buffer.

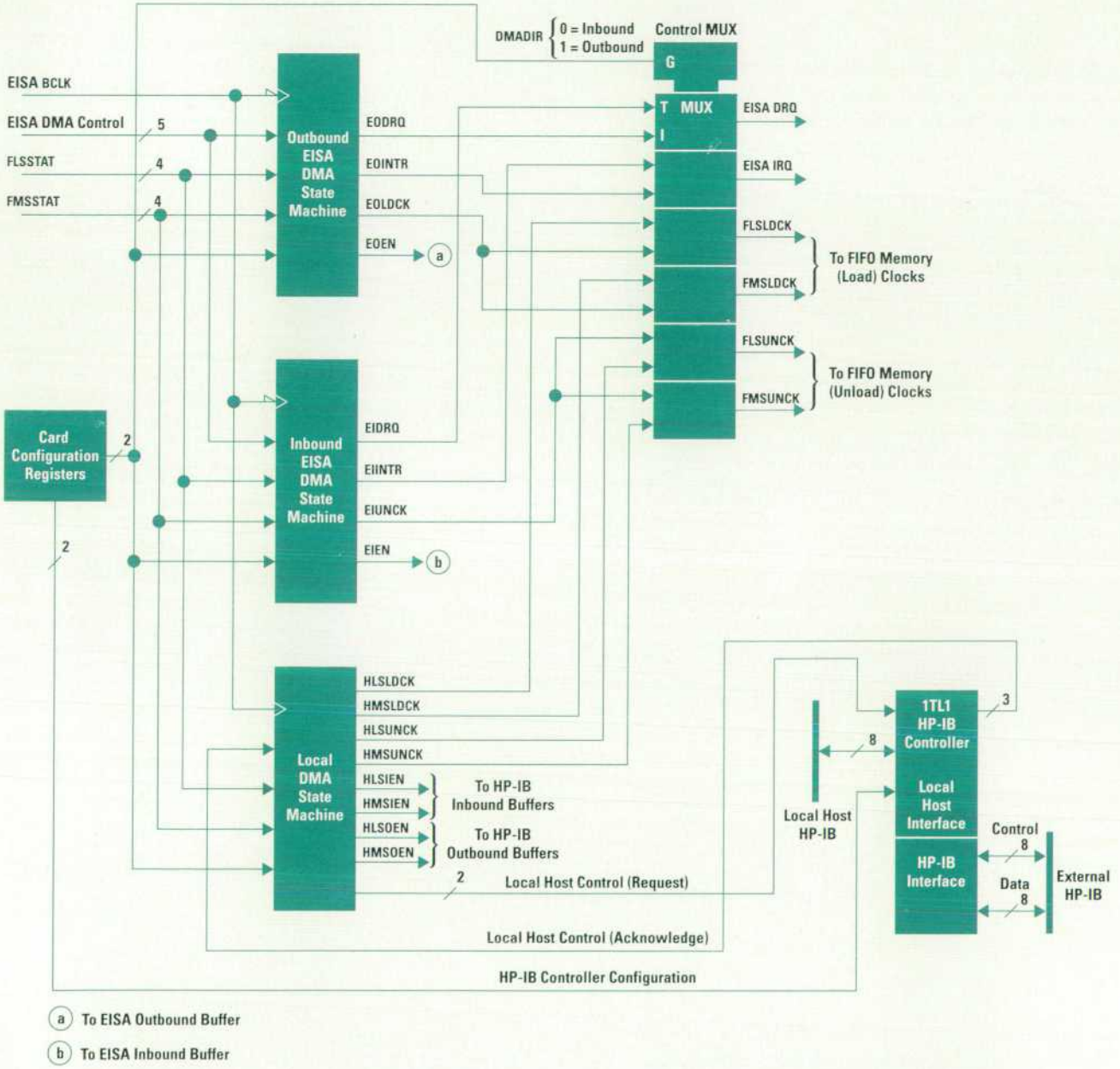


Fig. 11. DMA state machines.

During an outbound DMA transfer, the outbound EISA DMA state machine loads data from the EISA interface into the FIFO buffer, while the local DMA state machine unloads the data from the FIFO buffer to the outbound FIFO queue in the 1TL1. During an inbound DMA transfer, the local DMA state machine loads data from the inbound FIFO queue in the 1TL1 into the FIFO buffer, while the inbound EISA DMA state machine unloads the data from the FIFO buffer to the EISA interface. The data can only be sent in one direction at a time through the FIFO buffer. The appropriate bank of tristate buffers is selected to control the direction of the data flow. In addition, the data width mismatch is resolved by packing the HP-IB data bytes into 16-bit words for EISA on inbound transfers, and unpacking the 16-bit words from EISA into bytes for the HP-IB on outbound transfers. For the byte-packing function, the bytes are arranged so that the first (and subsequent odd) byte received on an inbound transfer is the least-significant byte of the 16-bit word. For the word-unpacking function, the bytes are arranged so that the first (and subsequent odd) byte transmitted on an outbound transfer is the least-significant byte of the 16-bit word.

On the HP-IB side of the FIFO buffer, two octal tristate buffers are used to funnel the outbound data to the 1TL1 HP-IB controller chip. This multiplexer, controlled by the local DMA state machine, separates the 16-bit word into two bytes by selecting one of the two octal buffers sequentially. (The least-significant byte is selected first.) Also, two octal tristate buffers are used as a demultiplexer to move the inbound data bytes from the 1TL1 into the FIFO buffer. One of the two buffers is selected (the least-significant byte first) to assemble a 16-bit word in the FIFO buffer. This demultiplexer is controlled by the local DMA state machine.

On the EISA side of the FIFO buffer, a word-wide tristate buffer is used to direct the outbound 16-bit words from the EISA interface into the FIFO buffer under control of the outbound EISA DMA state machine. Also, a word-wide tristate buffer is used to direct the inbound 16-bit words from the FIFO buffer to the EISA interface under control of the inbound EISA DMA state machine.

DMA Transfer Process. For an outbound DMA transfer, the EISA outbound DMA request signal (EODRQ) is asserted by the outbound EISA DMA state machine when the EISA DMA circuitry is configured and armed on the EISA HP-IB card by its software driver and when the FIFO buffer is empty. The condition of the FIFO buffer being empty is necessary to optimize use and performance of EISA and to maximize use of the FIFO buffer.

The card will interrupt the system following the completion of the outbound DMA transfer when the EISA terminal-count signal (one of the DMA control signals going to the state machines) is asserted by the system (indicating the last EISA DMA cycle) and the FIFO buffer has reached its empty condition after flushing the last data going out to the 1TL1 chip. At this point the card is ready to be serviced and rearmed for another DMA transfer.

For an inbound DMA transfer, the EISA DMA request signal (EIDRQ) is asserted by the inbound EISA DMA state machine when the EISA DMA circuitry is configured and armed on the EISA HP-IB card by its driver and when either the FIFO buffer is partially filled with the last-data flag set, or when the FIFO buffer is full. The condition of the FIFO buffer being full is necessary to optimize the use and performance of EISA and to maximize use of the FIFO buffer. However, a peripheral on the HP-IB may not always send back enough data to the card to fill the FIFO buffer, and the buffered data may sit here for an unexpectedly long period of time if it has to wait for the buffer to fill up completely. To get around the problem of having stale inbound data in the FIFO buffer, an external last-data flag is set by the 1TL1 when it detects the HP-IB end-or-identify (EOI) signal. The HP-IB EOI signal is asserted by the peripheral to signify that the last data byte was sent to the card during an HP-IB data transfer sequence. The flag status and the FIFO buffer status information are used by the inbound EISA DMA state machine. In this case, the state machine will assert the EISA DMA request signal if the last-data flag is set and there is data in the FIFO buffer. This ensures that the last chunk of inbound data will be properly flushed from the partially filled FIFO buffer.

The card will interrupt the system following the completion of the inbound DMA transfer when the EISA terminal-count signal is asserted by the system (indicating the last EISA DMA cycle), or when the last-data flag is set and the FIFO buffer has reached its empty condition after flushing the last data going into system memory. The former situation will occur when the DMA counter in the DMA controller matches the inbound HP-IB data count exactly. A more likely scenario is the latter situation, that is, the DMA transfer ends upon the setting of the last-data flag. When the DMA counter is loaded by the driver with a value greater than the predetermined amount of data to be received from the HP-IB peripheral, the termination of the inbound DMA transfer is controlled by the card and not the DMA controller. The card can "prematurely" end the transfer (with respect to the system) by deasserting the EISA DMA request signal and interrupting the system when the HP-IB EOI signal is detected during a data transfer. The interrupt service routine will handle the residual value in the DMA counter. At this point the card is ready to be serviced and rearmed for another DMA transfer.

The HP EISA PSI Card

The HP EISA programmable serial interface (PSI) card provides the means to support X.25 and SNA on Series 700 workstations. The card is programmable in that the network protocol code is downloaded from the host system into the card's memory for execution by the card's microprocessor. This makes it possible to use the same hardware platform for X.25 or SNA functionality. The EISA PSI card is a 32-bit Type-C EISA DMA module capable of 33-Mbyte/s transfers on EISA using any one of seven DMA channels. Two hardware interfaces are supported on the single frontplane port: RS-232 and V.35. The proper frontplane signals are automatically selected depending on which interface cable is plugged into the frontplane connector. The user does not have to set any switches or jumpers to configure the frontplane.

In general, a card such as the EISA PSI would use a microprocessor to run the networking protocols, a local DMA controller to manage the data transfers on the card, and a serial controller to do the protocol conversions for the frontplane interface. In the past, this would have required three separate chips. The EISA PSI card uses the Motorola

MC68302 integrated multiprotocol processor, which integrates these three functional blocks and other support circuitry into a single chip. The core of the MC68302 is a Motorola M68000 microprocessor. The MC68302 provides three serial communications controllers, each with its own supporting DMA channels. One of these serial communication controllers is used in lieu of a separate serial controller chip on the EISA PSI card. The MC68302 also provides an additional DMA controller (IDMA) which is independent of the DMA channels supporting the serial controllers. The EISA PSI card uses the IDMA instead of a separate DMA controller chip to manage the transfer of data between the card's SRAM and DRAM blocks.

The EISA PSI card requires a large block of onboard memory for the microprocessor's protocol firmware and the data to be processed and transferred across the frontplane. If this memory is also interfaced directly to the EISA bus, the host can transfer data via DMA directly to and from this memory.

One of the requirements of the EISA DMA process is that upon receiving its DMA acknowledge signal, DAK* <x>, the DMA device must be able to provide data on an inbound transfer or receive it on an outbound transfer immediately. The DMA device is not allowed to delay the transfer by inserting wait states. If the DMA acknowledge is received while the card's microprocessor is accessing memory, this requirement would be violated because of the time required for the microprocessor to release memory to the EISA DMA process.

One way to solve this problem would be to keep the microprocessor from accessing memory as soon as an EISA DMA transfer is requested by the card, and not releasing access to the microprocessor until the DMA transaction is complete. This was not an acceptable solution for the EISA PSI card, since the latency between the request of DMA by the card asserting DRQ<x> and the commencement of DMA on receipt of the DMA acknowledge signal is variable because of the arbitration mechanism in EISA. This time added to the EISA DMA transfer time would prevent the microprocessor from accessing memory for too long.

If the MC68302 is blocked from accessing the memory holding its firmware and data buffers for any significant length of time, the microprocessor cannot execute code and the serial frontplane will be starved for transmitted data or overrun with received data.

Therefore, we chose to implement two separate memories on the EISA PSI card: a fast SRAM on the EISA bus to act as a data buffer for DMA transfers, and a large 1M-byte DRAM to provide space for the downloaded protocol firmware and data buffers (see Fig. 12). This approach simplified the card architecture considerably. The SRAM is isolated from the microprocessor data bus by tristate transceivers. This allows EISA DMA transfers to SRAM to execute while the microprocessor has full access to the data buffers in the DRAM memory. The entire EISA DMA transaction can be completed to the SRAM without any interference with the microprocessor. This approach also had other benefits. The microprocessor and DRAM controller use a 16-MHz clock, and with this design, synchronization with the EISA bus clock is not required. Separate memories also helped reduce the materials cost of the board. Since the protocol code requires 1M bytes of memory, an implementation with technology fast enough to support EISA DMA from the same memory would have been much more expensive than the DRAM used.

The basic process for PSI outbound transfers starts with the EISA DMA transferring data from the host system memory to the card's SRAM. Then the MC68302's IDMA transfers the

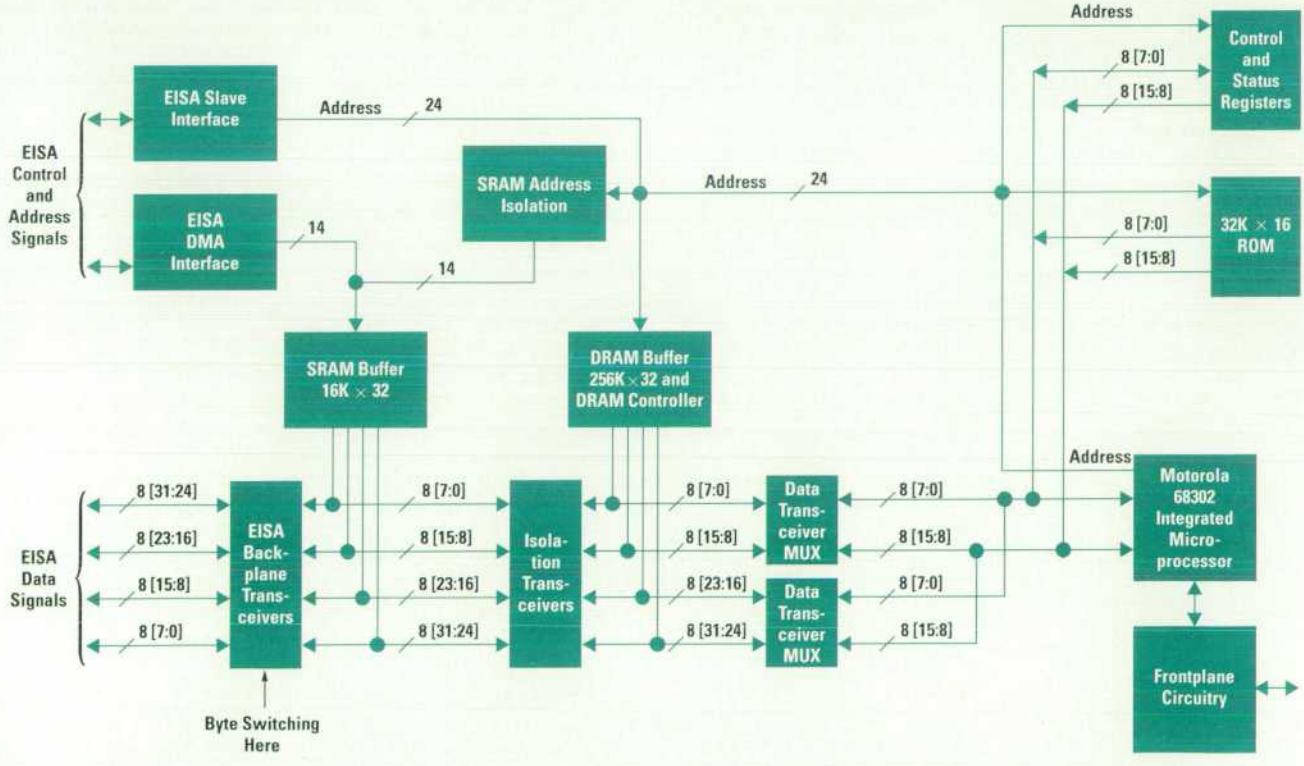


Fig. 12. SRAM and DRAM on the HP EISA PSI card.

Board-Level Simulation of the Series 700 EISA Cards

The complexity and clock speeds of designs are increasing such that traditional "breadboard" debugging techniques are no longer a feasible approach to design verification. It has become increasingly important to verify the design before fabrication. This can lead to a reduction in the number of printed circuit board passes, shorter design cycles, increased product quality, and increased engineer productivity.

The designs of the Series 700 EISA cards were verified through board-level simulation. Aggressive design cycles and recent successes with simulation led to widespread use of board-level simulation on the EISA card projects.

Objectives

The collective objective of board-level simulation was to help ensure first-pass printed circuit board success so that we could deliver defect-free hardware to our software partners, helping to meet the short time-to-market requirement.

The EISA cards were considered to be risky, complex, and difficult to debug—risky because of the various corner cases in timing and function that the Series 700 EISA bus would not exercise. This possibility exists because of the variance in vendor interpretation and implementation of the EISA specification. This interpretation problem is in the area of asynchronous slave transactions and in bus master protocol. Our cards not only had to work in the Series 700 workstations but with other third-party machines. The difficulty in debugging was that the architecture of the designs could not be modified easily if during the debug process a fundamental design flaw was uncovered. Therefore, it was important to prove the architectures of the cards through simulation.

Modeling Techniques

Although simulation libraries can be quite extensive in their model offering, in most cases a number of components do not have model support. Models may not be available for a variety of reasons. These can include new devices, proprietary devices (custom ASICs), and low-volume devices. A design engineer with the task of creating models needs to select appropriate modeling techniques.

Several different modeling techniques are available to the designer. These include gate-level, behavioral, and hardware models. Gate-level models consist of primitive logic functions that are native to the simulator. These include functions such as AND, NOT, NAND, OR, NOR, XOR, and XNOR. From these primitives (and others), the designer can describe most hardware functionality. Behavioral models are software representations of hardware that describe the functionality using some hardware description language (HDL) such as Verilog HDL, VHDL (VHSIC (Very High-Speed Integrated Circuit) HDL, IEEE standard 1076-1987), and GHDL (GenRad HDL). Behavioral models can also be created using programming languages such as C or Pascal. Hardware models are real ICs used directly in the simulation through the use of special hardware and software interfaces.

Gate-level modeling is usually applied from the switch level up to the MSI (medium-scale integration) level. The limited coverage of this method is mainly because of the potential of using a significant amount of memory and being the slowest to process and simulate. It has the advantage of being potentially more accurate than other methods. All of the HP EISA cards used gate-level modeling for most of the standard TTL and PAL components on the cards. The gate-level models for the PALs were created using a custom filter that takes as input an ABEL† list file and outputs gate-level HDL.

Behavioral modeling can be applied from switch level all the way up to system level. The wide coverage of this method makes it a popular choice for engineers. The accuracy, speed, and memory requirements are a function of component complexity and how the code is written. Devices such as the Intel 82596CA LAN Controller and the HP-proprietary 1TL1 HP-IB controller chips were modeled at this level. Only the critical bus interfaces were modeled on the 82596CA. The 1TL1 model not only modeled the local-host interface and the HP-IB interface, but also included two FIFOs which allowed more accurate modeling. Behavioral modeling supports as much complexity as required. In some cases the engineers opted for more skeleton-type models in which timing accuracy and more functionality were required.

Hardware modeling is applied when there is no model available, the component is available in silicon, the level of functional accuracy is important, or the device is sufficiently complex to justify this approach over a behavioral technique. This is usually considered the most accurate approach in terms of functionality; even real device bugs are visible in hardware modeling. The disadvantage of this approach is poor timing accuracy because in some cases the clock cycle time may be limited and only typical times can be represented. We did not use this method.

A final, and perhaps less popular method uses stimulus and assertion capability to model the interface of a particular component. This method is only suited for those devices that serve as bus interfaces for the board such as the backplane bus, the frontplane bus, and the frontplane/backplane interface chip. The HP EISA cards all used vectors to model the bus transactions of EISA rather than a bus exerciser (see below). In addition, the EISA SCSI card project team elected to provide stimulus and assertion checks for the NCR 53C710 SCSI chip using vector modeling rather than a traditional behavioral model.

† ABEL is a software product from DATA I/O that is used to generate JEDEC (Joint Electron Device Engineering Council) files which are used to program PALs.

data from the SRAM to the DRAM, and the MC68302's serial controller transmits the data from the DRAM to the frontplane link. This entire process actually involves several more detailed steps. First, the host must set up the system for EISA DMA by programming the DMA controller in the Intel 82357 on the EISA adapter. Then the EISA PSI driver software must provide the card with some control information by writing to specific registers located in the card's DRAM using EISA slave accesses. The MC68302 will be interrupted when the host has provided this information. To prepare the card for the EISA DMA transfer, the microprocessor loads a starting address into the address counter for the SRAM. EISA DMA does not provide addresses for the DMA device, so the EISA PSI card must create its own addresses for the SRAM to point to where the data should be transferred. The microprocessor then writes to a special control register that serves to isolate the SRAM from the rest of the card by disabling the transceivers between the SRAM and the DRAM. This same control register also activates the state machine on the card that handles the outbound EISA DMA transfer on the card. The outbound state machine asserts the EISA DRQ<x> signal (the specific DMA channel is selected during initialization) and controls strobing data from the EISA bus into the SRAM after the DAK*<x> signal is received.††

When the EISA DMA transfer is complete, the outbound state machine interrupts the MC68302 microprocessor. The microprocessor then sets up the transfer of the data from the SRAM to the DRAM using its IDMA controller. The MC68302 has a 16-bit data bus, but the SRAM and DRAM are 32 bits wide. Two pairs of isolation transceivers are used to select the proper bytes for data transfers involving the microprocessor. After the data has been transferred to the DRAM,

†† The EISA bus is a little endian (MSB is byte 3) and the 68000 bus is a big endian (MSB is byte 0). The endians are switched on the EISA PSI card by the EISA backplane transceivers shown in Fig. 12.

Stimulus

Stimulus, also called vectors, consists of signal patterns or waveforms that are applied to the inputs of a circuit. Stimulus can be generated by a vector generation tool (such as Guide, which is part of the HP internal LogicArchitect tools) or by a command-driven model (bus exerciser).

A bus exerciser is basically a user-generated behavioral model. Its purpose is to issue bus transactions and check for proper circuit responses. It is possible to do away with traditional vector generation entirely through the use of a bus exerciser. The bus exerciser is usually a command-driven device, meaning that it accepts some sort of coded instruction that when decoded instructs the model to initiate some particular bus transaction. In the process of executing this transaction the model can also check for proper circuit response. The ability to provide response checking (also called assertion checks) is a solution to the situation in which the simulator does not have the native capability to perform these checks directly. This method is useful when the bus exerciser can be leveraged to additional developments because of the significant time that this approach requires.

The vector sets for the EISA cards were quite extensive and in one case exceeded 100,000 bus cycles. The reason for the great number of vectors is the numerous permutations of various corner cases in both timing and function. Again, this is because of the loose EISA specification, especially in the area of asynchronous slave transfers.

Assertion Checks

The combination of a circuit netlist, user models, model libraries, and stimulus allows the simulator to generate circuit behavior in the form of signal responses. The circuit response can be verified in two ways; either the simulator checks the specified output signals every clock cycle through strobe statements, or a final response file is generated and the checking is performed postsimulation by an additional tool. GenRad simulators have the ability to perform signal assertion checks cycle by cycle.

In our case, the GenRad simulator checks the particular outputs through the use of strobe statements. These strobe statements are generated by the vector generation tool Guide. The vector generation tool issues strobe statements in response to user requests for assert checks. This is done via a high-level language that is valid within the Guide environment. When the simulator detects a strobe violation (expected logic value different from actual) an error message is issued to the user. A violation can mean that the applied stimulus is in error, the logic level assertion check is incorrect, or the circuit design is not correct.

Process Description

Schematic capture was accomplished using HP DCS (design capture system). The resulting HI-LO3® netlist was generated using HP DVI (design verification interface). HI-LO3 was provided with libraries from two sources: a site-compiled standard TTL library and a local user-built library. Vectors were generated from the LogicArchitect tool Guide. HI-LO3 outputs result and capture files, which are used to interpret the results of the simulation. The capture file contains circuit stimulus and response data that can be viewed graphically using DVI or textually using the GenRad DISPRO tool, which produces the results of a simulation in ASCII form. As bugs are discovered and fixed, the process cycles back to the beginning (see Fig. 1).

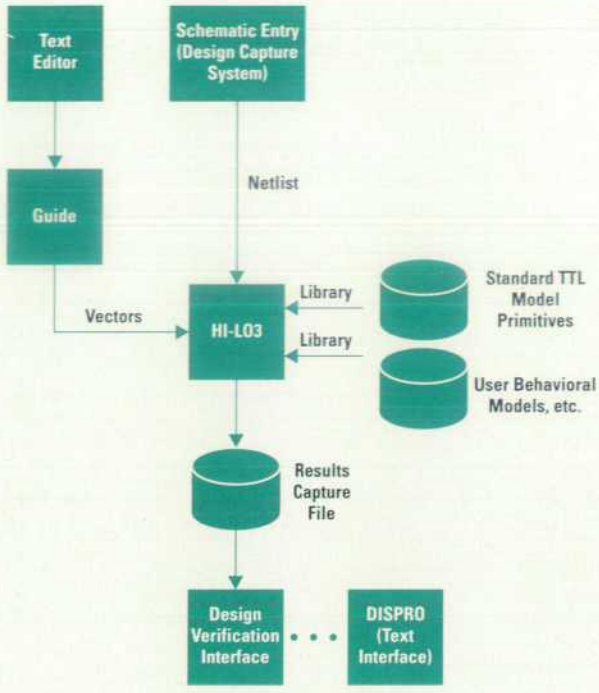


Fig. 1. Design and simulation process.

Results

In all cases the EISA cards were delivered to our software partners with only a single printed circuit board pass. The effort of simulation revealed numerous design flaws before printed circuit boards were created. The process is not perfect. In all cases, the cards had at least one serious bug that was not caught by simulation. However, these bugs were all resolved without jeopardizing the delivery of a first-pass printed circuit board to our software partners on time—our ultimate goal.

HI-LO3 is a registered trademark of GenRad, Inc.

the microprocessor does the processing necessary for whatever network protocol is running on the card. Then the data is transferred by the MC68302's built-in serial controller to the frontplane.

The basic process for inbound transfers is the outbound transfer in reverse—the MC68302's serial controller transfers the received data to the DRAM, then the MC68302's IDMA is used to transfer the data from the DRAM to the SRAM, and EISA DMA is used to transfer the data from the SRAM to the host system memory.

The inbound process begins when the serial controller in the MC68302 transfers the incoming data into a predefined location in the DRAM using its assigned DMA channel built into the MC68302. The microprocessor then executes the necessary protocol processing on the data. Meanwhile, the host sets up the DMA controller on the EISA adapter for the EISA DMA transaction. As with the outbound transfers, the host writes setup information to the registers in the DRAM on the EISA PSI card and then interrupts the MC68302 to signal that the information is available. The microprocessor transfers the data from the DRAM to the SRAM using its IDMA controller. It then loads the starting address into the SRAM address counter and writes to the control register to isolate the SRAM and signal the inbound state machine to request DMA. The inbound state machine asserts the EISA DRQ<x> signal and controls the strobing of the data out of the SRAM onto the EISA bus after the EISA DAK*<x> signal is received.

Memory and I/O Slave Overview

Another type of device that can be placed on the EISA bus is a memory or I/O slave device. EISA supports 8-bit, 16-bit, and 32-bit data transfer cycles for slave devices. This includes ISA-compatible timing for ISA memory and I/O slaves. EISA memory and I/O slaves can support 32-bit data transfers, but burst cycles and full 32-bit addressing are available to EISA memory slaves only.

The slave devices are accessed by the host CPU or bus master. The standard EISA memory and I/O cycle type completes one transfer every two BCLK periods assuming no wait states. Slower slave devices may lengthen a transaction by inserting wait states.

Each EISA card must support some form of I/O slave access. This enables the system to read the EISA identification information from the card during the EISA configuration process. Nonburst memory or I/O transfers are not as efficient as burst DMA transfers for the movement of data, so the slave interface is generally used for sharing control and status information between the system and the EISA card.

For example, the EISA PSI card supports three types of slave interfaces. It provides an 8-bit EISA I/O slave interface for access to the EISA information stored in the architected EISA identification and control registers. The card also provides a 16-bit EISA I/O slave interface for access to its control and status registers. Finally, the card has a 32-bit EISA memory slave interface to allow the system quick and direct access to onboard DRAM without having to go through the EISA DMA process. (The DMA process requires some overhead to set up, which is not very efficient for accessing just a few bytes at a time.) The DRAM is mapped into the system memory space so that it can be a shared resource between the system and the card's onboard microprocessor. This allows efficient access to the stacks of information related to the control of the network protocols running on the card.

Acknowledgments

The authors wish to thank everyone who contributed to the successes of the EISA card projects. Special recognition must go to Mick Jacobs for his EISA expertise during our design reviews, and Bill Martin for his EISA simulations and the development of tools that simplified our simulation tasks. We would also like to acknowledge our project managers Gary Wermuth, Joel Dunning, and Don Ciaglo for their support during the development of our projects.

Software for the HP EISA SCSI Card

Two software architectures, one offline and the other online, are used to provide EISA SCSI support for the HP 9000 Series 700 workstations.

by Bill Thomas, Alan C. Berkema, Eric G. Tausheck, and Brian D. Mahaffy

The SCSI (Small Computer System Interface) bus requires support from many offline and online software modules. Much of the software has a layered structure and runs during the normal run time of the HP-UX* operating system. The SCSI interfaces for the HP 9000 Series 700 include the built-in 5-Mbyte/s single-ended SCSI interface and an optional add-on HP 22525A EISA SCSI interface card with a 10-Mbyte/s differential SCSI bus.† This article will focus on the optional add-on HP EISA SCSI interface card.

The EISA SCSI software falls into three categories: software that runs before the system is completely booted (offline software), software that runs after the HP-UX or MPE system has started running (online software), and special test software that provides workloads and stress conditions not normally achieved by run-time system software.

In a running system a layer of software drivers provides the interface between the user's program and the interface card. Each I/O request may involve different layers depending on the kind of request and type of device. Fig. 1 shows the architecture for the EISA SCSI software drivers. In this architecture the HP-UX kernel, on behalf of a user process, starts an I/O operation by invoking a peripheral device driver through an open, read, write, or close procedure. The peripheral device driver converts the request into one or more SCSI commands and passes them to the transport layer. The peripheral device driver and the transport layer are not dependent on the implementation details of the SCSI interface or the I/O bus hardware. The transport layer determines which SCSI interface the request is for and passes the commands to the appropriate interface driver. The interface driver is aware of the implementation of the SCSI interface and operates accordingly.

Offline software and firmware runs before the HP-UX system is functional. This software includes the processor dependent code (PDC), the I/O dependent code (IODC), the initial system loader, the HP-UX loader, the iomap program, and the SCSI product verification program. All of this software is required so that the EISA SCSI interface can be used to boot the system. Processor dependent code provides a standard uniform access to certain basic operations of the system such as system reset and boot, system information (e.g., time of day), and the ability to load I/O dependent code. Each computer system (e.g., Model 710, 720, or 750) has its own version of processor dependent code stored in ROM on the CPU board. The I/O dependent code contains information about its associated hardware and procedures that are loaded and executed during bootup. These procedures

† Differential SCSI uses differential transceivers for better noise immunity, providing longer cables and higher transfer rates.

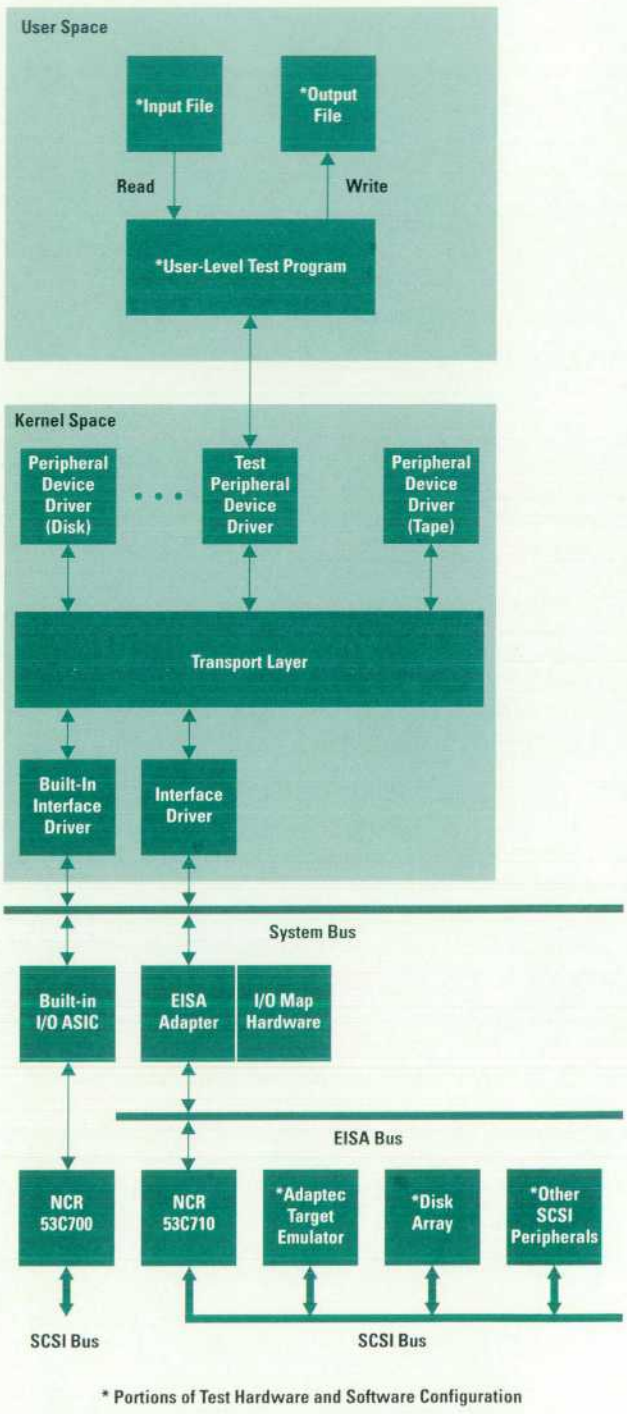


Fig. 1. The online software modules used for communicating with SCSI devices.

test the hardware, detect devices (e.g., disks) connected to it, and read and write data to and from these devices. Each I/O card connected to the EISA I/O bus that might be a potential boot device has a ROM containing I/O dependent code. For built-in interfaces such as the built-in SCSI port, the I/O dependent code is located in the same ROM as the processor dependent code (see Fig. 2).

The iomap program permits the user to examine and test the hardware without having to load the HP-UX operating system. The SCSI product verification program can also be run without the operating system. Because the operating system is not loaded for these programs, tests and other operations can be performed that the operating system would prevent from running.

This article will discuss the implementation of some of these software modules and the test process for the interface driver.

Offline Software Modules

The processor dependent code and I/O dependent code mentioned above are called offline modules because they are only used once, and that is during the boot process.

Booting the System

A system power-on or reset initiates the boot process. The boot process begins with the processor dependent code stored in ROM on the CPU board first checking the processor and memory and then beginning a search for other system hardware with particular interest in finding a device to boot from. The possible locations of boot devices include the server nodes accessible through the built-in LAN interface or other devices connected to the built-in SCSI interface. Boot devices also include devices accessible through the EISA I/O bus such as a SCSI device connected to the EISA SCSI interface. For the configuration shown in Fig. 2, the boot process would proceed as follows:

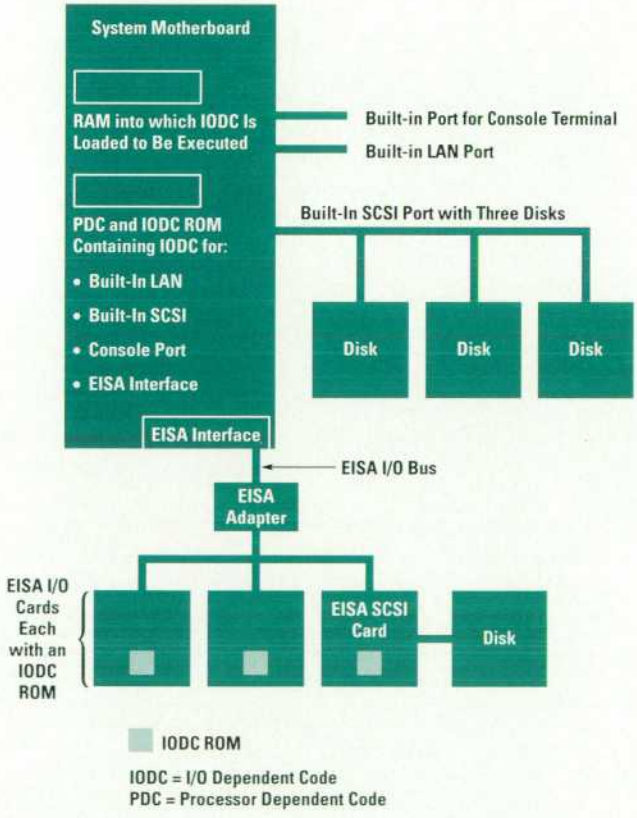


Fig. 2. The locations of the offline software (or firmware) modules used during the system boot process.

1. The reset procedure in the PDC (processor dependent code) is executed.

2. The PDC does a self-test (on the processor) and then searches for other hardware and finds: memory, a built-in serial port, a LAN port, a SCSI port, and an EISA interface.

3. The PDC loads and executes the self-test procedure for each item found in step 2.

4. The PDC finds that the built-in SCSI card can detect more hardware.

5. The PDC loads and executes the built-in SCSI's IODC (I/O dependent code).

6. The built-in SCSI's IODC reports the presence of three disks.

7. The PDC finds that the EISA interface can detect more hardware.

8. The PDC loads and executes the EISA interface's IODC.

9. The EISA interface's IODC procedure reports the presence of three I/O cards with IODC.

10. The PDC loads the IODC for each EISA card and executes each card's self-test procedure.

11. The PDC finds that the EISA SCSI card can detect more hardware.

12. The PDC loads and executes the EISA SCSI card's IODC search procedure.

13. The search procedure reports the presence of one disk.

14. The PDC loads and executes the built-in serial IODC to inform the user that there are four disks and asks the user which one to use. If the user selects the disk on the EISA SCSI card, the PDC loads the EISA SCSI card's IODC and executes the procedure to retrieve the initial system loader stored on the disk.

15. The initial system loader takes over the the process of booting the system by running the hpux program, which is also stored as a file on the disk.

At the end of the boot process and just before a system prompt appears on the system console, the EISA configuration utility (described below) is run by the hpux program.

To detect a piece of hardware, information is retrieved from a reserved set of addresses in system memory and if valid values are returned from these addresses, hardware is present. When a piece of hardware is detected, its IODC is read to determine what kind of hardware it is and what revision. The IODC procedures for the EISA SCSI card and the initial system loader are discussed in more detail later in this article. When an IODC procedure is loaded and executed, it is loaded into system memory.

EISA SCSI Card Configuration

The EISA bootable interface for HP 9000 Series 700 workstations had to overcome the problem of needing a configured interface subsystem before the subsystem could be used to boot the system, while the utility to configure the subsystem could not be run until the host system had booted—a classic system configuration problem. Unlike the predominant EISA bus implementations on MS-DOS® personal computers that have an internal disk drive to boot from, the Series 700 workstations need to provide a basic configuration for booting from many different types of external devices.

Part of the EISA standard, which is intended for usability, specifies that EISA cards have no switches or jumpers. Industry Standard Architecture or ISA cards (the predecessor of EISA) commonly had jumpers and switches for setting the card's basic configuration for I/O transfers. Series 700 EISA cards rely on utilities run at system power-on to read basic configuration information from the EISA EEPROM on the system motherboard to configure the card (set flip-flop states) for I/O transfers. Getting the configuration information into the EISA EEPROM was the problem we had to overcome in designing the Series 700 EISA SCSI card's software and hardware architecture.

In EISA MS-DOS machines, the system boots from the internal disk drive and runs the EISA configuration utility to configure the EISA EEPROM for each of the interfaces installed. The configuration programming information for a given interface is distributed on a flexible disk shipped with the EISA interface card. In a typical Series 700 workstation, an internal flexible drive is not always available (one is available as an option, but it is somewhat rare). Therefore, for the Series 700 workstations the solution was to move the EISA configuration program to the system's boot ROM. In addition, code for configuring (setting the bits) on the interface, which is normally distributed on a flexible disk for an MS-DOS machine, is incorporated into the I/O dependent code on the interface cards.

On MS-DOS systems, EISA configuration information is found in a .CFG file on the flexible disk that comes with MS-DOS EISA cards. By using special execution options, the HP-UX EISA configuration program generates an image of the configuration information found in the .CFG† file and stores it in the IODC ROM when the ROM is programmed.

The process of getting configuration information from the IODC ROM on an EISA card to the EISA EEPROM on the motherboard takes place during the boot process described above. When the PDC is examining each EISA card, it reads the EISA card's identification bytes. If the bytes do not match those found in the EISA EEPROM location for that slot (as is the case when the card is first installed), the PDC's configurator program reads the default (.CFG) configuration information from the card's IODC ROM and places it into a "special location" in the EISA EEPROM. If the identification bytes from the interface card match those found in the EISA EEPROM for that slot, the PDC's configurator program initializes the EISA card (see Fig. 3a).

When the PDC loads and launches the IODC for a particular EISA card, the IODC will examine the state of the card to determine if it had been preinitialized by the PDC's configurator program. If the card has been initialized, the state of the card is not changed. If the card is found to be uninitialized, the card is configured to the default .CFG information in the IODC ROM (see Fig. 3b). For the Series 700 EISA SCSI card, the default configuration includes setting parameters for the SCSI address and parity checking with values commonly used by most systems.

When the initial system loader takes over the HP-UX boot process, one of its tasks is to run the eisa_config (EISA configuration) utility. When eisa_config runs it interrogates the EISA EEPROM for any IODC (default) configuration information that might exist in the special area of the EISA EEPROM. If it finds the default information, eisa_config will enter an automatic configuration mode, validate the default values and move them to the appropriate place in the EISA EEPROM that matches the slot the EISA card is installed into (see Fig. 3c). The next time the system is booted, the appropriate information for configuration will be available. The system administrator can run the eisa_config utility manually and modify the configuration of the interface. In the same way as for the default configuration, the eisa_config utility will save the configuration initialization information in the appropriate location in the EISA EEPROM so it will be available at the next boot time.

HP EISA SCSI I/O Dependent Code

The EISA SCSI I/O dependent code (IODC) is designed according to the PA-RISC outline for bootable interfaces. The IODC ROM contains software modules that can be loaded into host memory and executed by the host processor (see Fig. 4). Two major entry point software functions are defined for the SCSI IODC: ENTRY_INIT and ENTRY_IO. The ENTRY_INIT function call is for initialization of both the EISA SCSI card and devices on the SCSI bus. The ENTRY_IO function call is for I/O data transfer between the workstation and the SCSI devices on the bus. Each of these function entry points has several subfunctions. The ENTRY_INIT function provides testing of the SCSI card, searching for devices on the bus, device initialization, device testing, and testing of data transfer integrity. The ENTRY_IO function has functions for read and write for both character and blocked data type devices.

The first subfunction for ENTRY_INIT is the INITIALIZE function. During the initialization process, care is taken to preserve the card's configuration parameters for the SCSI address, parity checking, and so on if the card has already been configured by the PDC configurator during boot. After the card configuration has been noted the INITIALIZE function resets the card to a known state and performs low-level self-testing of the hardware. The self-test includes a full register check of the NCR 53C710 SCSI controller chip, DMA and FIFO data transfer tests, 53C710 script execution tests, and offline SCSI bus data and parity tests. Upon successful completion of the self-test routines, either the saved card configuration or the default configuration is used, as appropriate, to set the interface card to an initial configured state.

The next function of ENTRY_INIT is the linked pair of calls SEARCH_FIRST and SEARCH_NEXT. These two calls are actually the same subroutines called with slightly different initial parameters. The purpose of these routines is to search the

† For the HP-UX operating system the .CFG file comes as part of the system software.

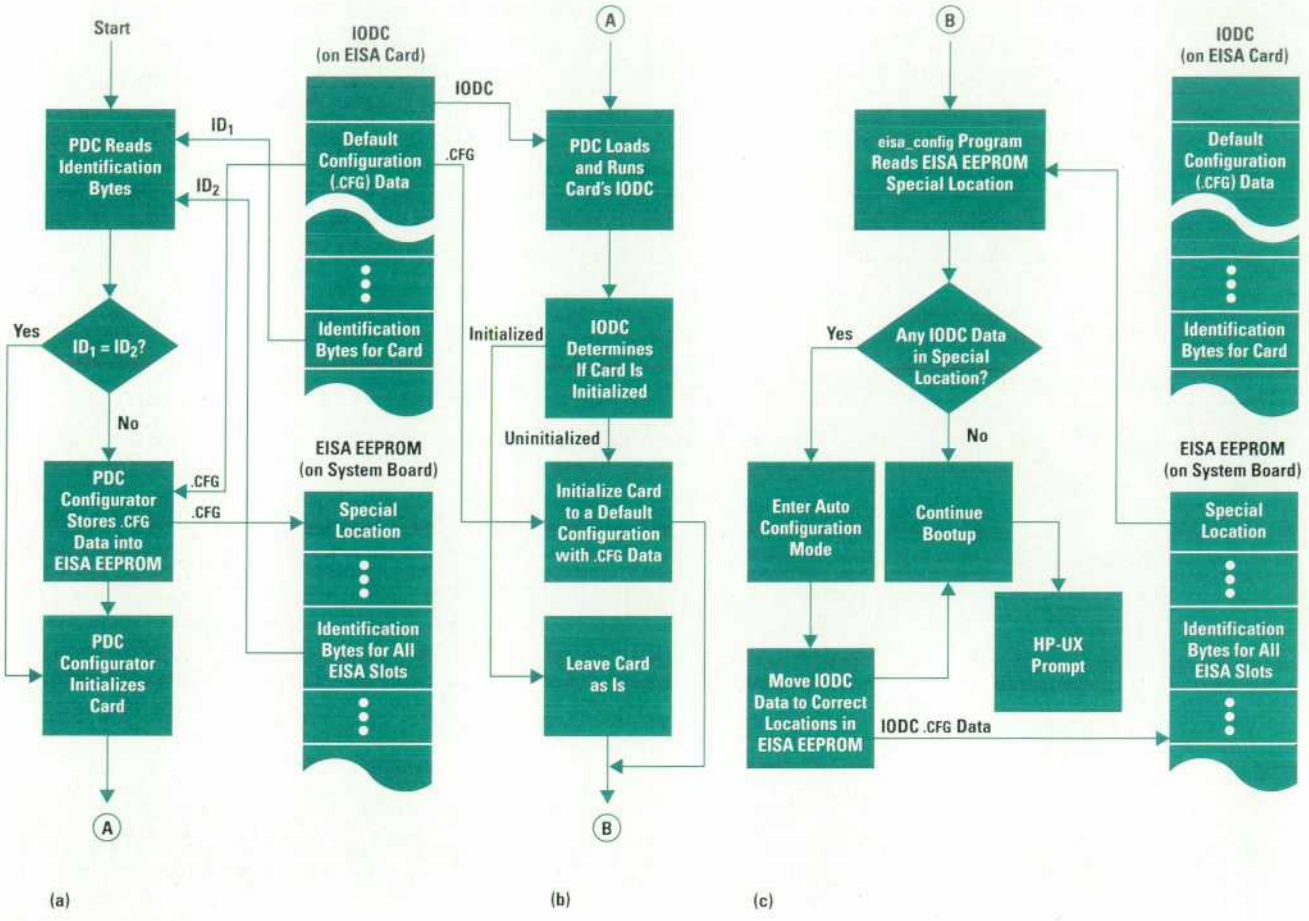


Fig. 3. Different parts of the EISA configuration process. (a) Comparing EISA identification bytes and if necessary storing default configuration information in a "special location" in the EISA EEPROM. (b) Initializing the EISA card. (c) Running the eisa_config utility.

SCSI bus for any devices of a type the IODC knows about. The IODC knows about both character (tapes) and block (disk) type devices, but currently only disks are offered in a differential interface for purposes of booting the system. The SEARCH_FIRST routine is called first with an empty device identifier record passed as a parameter. The routine searches from the highest available SCSI address downward until it either finds a device or runs out of addresses to try. If a device responds to the search, it is interrogated to find out if it is a device the IODC knows how to handle. If the IODC knows the device, the device identifier record will be filled with the device's data and address location. Subsequent searching is done by calling SEARCH_NEXT with the address information maintained in the device identifier record.

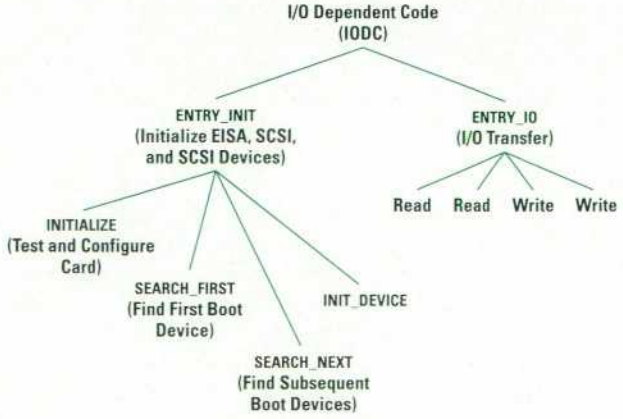


Fig. 4. I/O dependent code subfunctions.

The remaining subfunction for ENTRY_INIT is the INIT_DEVICE function. The device needs to be found on the bus first by the SEARCH_FIRST and SEARCH_NEXT functions before the INIT_DEVICE function is called. Once it is found, the targeted device will be reset to a known state then a series of confidence building steps will be taken to ensure proper communication is possible with the device. First, an internal self-test function that all SCSI devices have built into them is run. When the device responds that it has passed its self-test, the INIT_DEVICE code will then use the device's internal buffer RAM to write a small amount of test data to the drive. The data just written will then be read back from the drive's internal buffer and verified for correctness with what was originally written to the drive. After all theses tests have been successfully completed, control will be returned to the calling routine.

ENTRY_IO has less functionality than the ENTRY_INIT function, but is the function that does the "real work." Four subfunctions are associated with this function, two for data writes and two for data reads. The read and write subfunctions are very similar in design and functionality. One subfunction pair (read and write) is for character-based SCSI devices and the other pair is designed for block transfer devices. Boot support for these devices is present in the IODC if they are ever produced in a differential SCSI version.

Certain types of request validation and parameter checking are common to the four subfunctions in ENTRY_IO. First a check is made to verify that the device being read from or written to has a block size that is an even factor of 2048 bytes. The transfer is next checked to ensure that it is not larger than the maximum size the IODC can handle (because of memory mapping restrictions), and that the transfer size is a multiple of the block size of the SCSI device. The last check of parameters is only for sequential requests. If the data address is zero the tape is rewound first before the write or read is performed. After the parameters have been validated the request is translated to the appropriate SCSI commands and sent out to the SCSI bus.

Initial System Loader

The initial system loader is responsible for getting the hpux program from the boot device and running it. The initial system loader can take instructions from the user or use commands in the AUTO file on the boot device discovered by the IODC's SEARCH functions. If the instructions come from the user, the loader will prompt the user for what to do. The user can command the loader to run hpux to boot the HP-UX system with parameters that are the same as or different from those in the AUTO file. Also, the user can command the initial system loader to run the iomap utility to map and test the system, or run the SCSI Product Verification and Defect Analysis (scsipvda) utility to rigorously test and diagnose the SCSI interface.

The iomap program can search for all components in the system including processors, memory, CPU, bus converters, interface cards, disks, and so on. The iomap program can also invoke the self-test program for components that have self-test in their IODC ROM.

Running self-test programs in the initial system loader (or boot) environment has some advantages over running them in the HP-UX system environment. Diagnostics that run under the HP-UX operating system have many other advantages, but if the system will not boot, these run-time diagnostics cannot be used. The advantages of the boot environment include unrestricted access to system resources such as the SCSI interface configuration registers and no dependency on many system functions such as a functional file system. Also, diagnostics running in the boot environment do not have to contend with concurrent activity on multiple devices and interfaces. The diagnostic has complete control over what activity occurs and when it occurs. To enhance user friendliness, special messages for the user have been included in the iomap program specifically to aid testing of the SCSI interface. These messages are used when the SCSI IODC encounters a failure. These messages inform the user about what action to take, such as to check the cabling or the power-on state of a device.

The verification and diagnostic utility scsipvda also runs under the initial system loader. Like the iomap program, this program runs the IODC of the SCSI interface. The scsipvda program also runs additional tests that cannot run under the HP-UX operating system, such as testing the card with different configurations of the I/O memory mapping hardware. Once the HP-UX system has started, the diagnostics for SCSI devices, disks, and magnetic tapes provide confirmation of the SCSI interface functionality, although not with the completeness possible with scsipvda.

HP-UX Run-Time SCSI I/O Modules

The run-time SCSI modules run in the HP-UX kernel and provide the interface between user-level programs and SCSI devices.

Peripheral Device Drivers

Fig. 1 shows the logical relationship between the run-time modules. Each peripheral device driver is compiled separately and is built when the kernel is created. Each driver has knowledge of how to manage a particular peripheral or set of peripherals and each has a standard driver interface that the operating system uses for virtual memory and file-system management and for providing user applications with an interface to device capabilities.

A peripheral device driver provides open, close, read, write and control (ioctl()) functions to the kernel. The drivers translate these function calls and their parameters into the proper state management of the peripheral through the use of command sequences appropriate to the state of the device being managed and the HP-UX operating system functions requested by a particular device driver.

The SCSI standard specifies the command sets, their functions, and the possible replies for general classes of peripherals (e.g., a random-access class and a sequential-access class). This standardization has made it possible to write one peripheral device driver for a given class to handle a wide variety of peripherals from multiple vendors. For example, the same peripheral device driver for disks can be used for handling flexible, magnetooptical, and hard magnetic disk drives.

For the SCSI driver subsystem, the responsibility of a peripheral device driver is limited to those aspects of the protocol that concern the device the driver is controlling. Details about the physical link to the device and the protocols necessary for I/O processes to communicate between the host interface and the devices is a common service needed by all device drivers and is provided by the interface drivers.

Interface Drivers

Software interface drivers manage the SCSI bus aspects of I/O processes between all the peripheral drivers and a given SCSI interface. An interface driver takes I/O requests coming from a peripheral device driver via the transport layer and presents the requests to the SCSI bus. When an I/O process is done, the interface driver makes a function call back to the appropriate peripheral device driver.

Since the SCSI bus interface can connect to multiple SCSI devices, the interface driver also provides a multiplexing

service for access and control of the SCSI bus on behalf of all of the peripheral device drivers.

The SCSI standard defines another two levels of optional multiplexing with a SCSI device: LUNs (logical unit numbers) and tagged queuing on each LUN (see "Update on the SCSI Standard" on page 103). Peripheral device drivers that use these features need support from the SCSI protocol handled by the interface drivers to support these levels of multiplexing.

Interface Driver Implementation for EISA SCSI

The SCSI standard defines the LUN and tagged queuing features but not the implementation or application of the features. The interface driver provided with our first EISA SCSI release is designed to support only one LUN and does not use SCSI tagged queuing because none of the peripherals supported in the first release use this option or have more than one LUN.

Since the newer disk array products are best used with multiple LUNs and tagged queuing, a special project was commissioned to design and implement an interface driver to support these SCSI features. This required some rethinking about the architecture of the built-in (single-LUN) SCSI interface driver and consideration of methods to minimize interface overhead to meet the increased throughput potential that some of the newer SCSI peripheral devices offer the system on a 10-Mbyte/s differential SCSI bus. Fig. 5 shows the configuration of a system with single and multiple LUN devices.

We learned from an earlier project that the protocol for managing multiple LUNs and tagged queuing is largely software queue management, and that a clear line of functionality can be drawn between the part of the interface driver that controls the hardware-specific aspect of the SCSI bus and protocol and the purely software aspect of queue management for I/O processes.

We went one step farther and defined a simple way to reuse the software queueing portion of the interface driver to provide for the possibility of future SCSI bus interface designs, or perhaps non-SCSI bus links to SCSI bus peripherals. This division of functionality for the new interface driver also allowed a division of labor between the project team that did the software for the hardware dependent portion of the interface driver and the project team that did all the other system software necessary to support disk arrays. The team that worked on the hardware dependent software focused on the task of implementing and proving a full-featured, high-performance, low-latency SCSI-2 standards-compliant solution using the EISA SCSI bus interface card.

The design team tailored the EISA SCSI hardware to make it look enough like the built-in interface so that the system software could be easily modified to support the new SCSI bus.

The HP 25525A EISA SCSI interface design includes a SCSI interface chip (NCR 53C710) with backward compatibility with the chip used for built-in SCSI (NCR 53C700). These interface chips are special-purpose processors that execute an instruction set called SCSI SCRIPTS (from NCR) and use DMA to transfer data between host memory and the SCSI bus. The host CPU builds the instruction stream in system memory that the SCSI processor eventually executes (one SCRIPTS instruction at a time) to make decisions based on SCSI bus activity and to do the appropriate DMA transfers. The instruction stream provides flexible test and control of the SCSI bus while reducing the amount of host interaction and host-induced latency for SCSI bus protocol processing.

The new interface driver provides the functionality required to support disk arrays on the 10-Mbyte/s differential SCSI bus. However, the driver does not support the built-in SCSI interface because it is optimized for the HP 25525A hardware.

We also designed the interface to provide optimized protocol support for the SCSI bus to minimize SCSI bus overhead. This optimization potential results from some additional circuitry and the flexibility of the 53C710 chip used on the HP EISA SCSI card.

(continued on page 104)

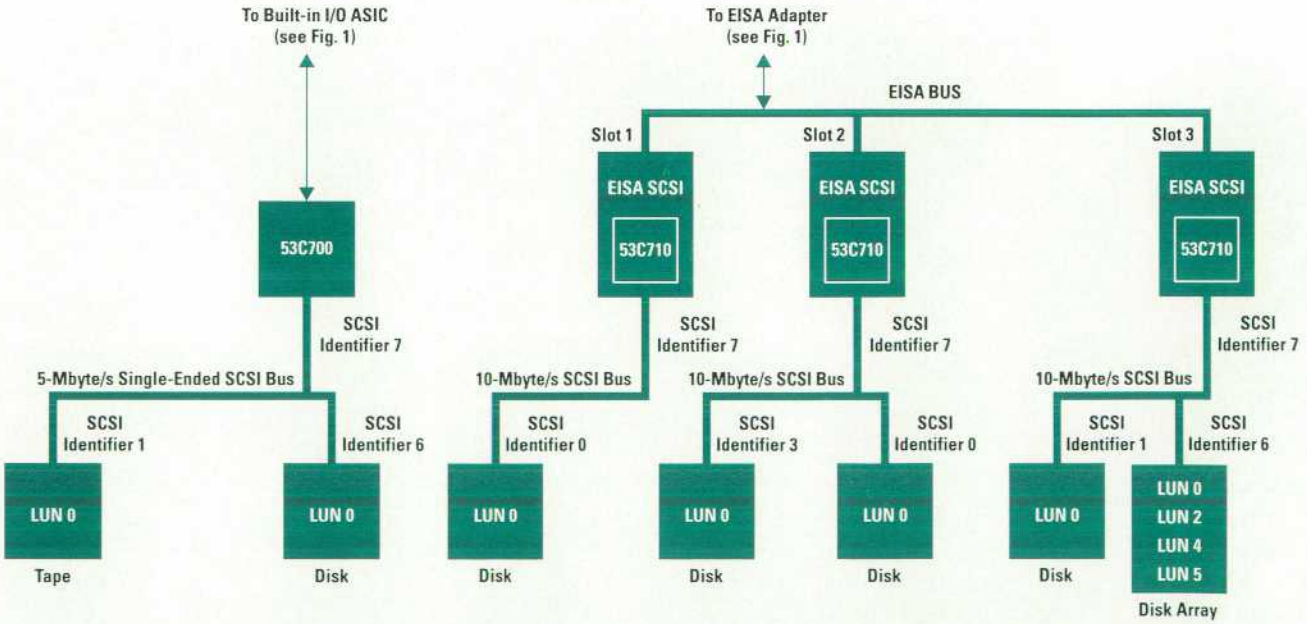


Fig. 5. The configuration of a system with single and multiple LUN SCSI devices.

Update on the SCSI Standard

The SCSI (Small Computer System Interface) bus is an intelligent, general-purpose I/O bus defined in ANSI committee standard X3.131. SCSI-2 is an enhancement to the original SCSI standard. It is designed for higher performance and greater peripheral connectivity. Some of the SCSI-2 features are described below.

I/O Processes and the I_T_L and I_T_L_Q Nexus

The HP EISA SCSI card is an 8-bit SCSI implementation. The 10-Mbyte/s data rates are achieved using the SCSI-2 fast timing specifications and a target SCSI device that also implements fast SCSI-2 timing.

The SCSI bus used in our design allows one host to control up to seven other SCSI device interfaces on each SCSI bus. It is a half-duplex interface bus, which means that information can travel either way between two devices but not at the same time.

Each SCSI device has a unique SCSI identifier. SCSI identifiers are used to resolve contention for the bus through a priority arbitration protocol. Each data line is used to represent a different SCSI identifier during arbitration so that all contending SCSI devices can be determined at the same time. After arbitration, the data lines are used by the arbitration winner to select the SCSI identifier of the SCSI device with which it is reconnecting or establishing an initial connection.

A SCSI initiator is a SCSI device that initiates SCSI I/O processes through initial connections. The initiator is usually represented by the host system and for this discussion is controlled by the interface driver for the HP 25525A EISA SCSI card. A SCSI target responds to initial connections from the initiator and uses the appropriate SCSI protocol steps for each I/O process including reconnecting. Several real or logical peripheral devices may be behind a SCSI target interface.

A SCSI I/O process is the state of a single SCSI command execution starting from the time that the target receives a SCSI command descriptor block from the initiator during initial connection until when the target communicates its completion back to the initiator.

A SCSI nexus, which is a relationship that begins with the establishment of an initial connection and ends with the completion of an I/O process, identifies an I/O process between the initiator and target. A SCSI I_T_L (initiator_target_LUN) nexus is composed of the initiator's SCSI identifier, the target's SCSI identifier, and one of eight possible addresses called logical units or LUNs within the target. These logical units may represent distinct physical or logical peripheral devices. SCSI protocol allows the initiator to send one I/O process at a time per I_T_L nexus.

The I_T_L_Q (initiator_target_LUN_queue) nexus is an optional layer of I/O process identification and protocol that allows concurrent I/O processes to be pending at the logical unit through a mechanism called SCSI tagged queueing.[1] Tagged queuing is a feature of SCSI-2 that allows a peripheral to manage I/O processes more efficiently.

All SCSI-2 devices support I_T_L nexus connections. Through these connections device drivers can use I/O processes to determine for each possible I_T_L nexus of interest which of the eight possible LUNs at a given SCSI identifier exist and which of these, if any, support an I_T_L_Q nexus connection. Typically all SCSI devices support LUN 0. Most of the existing or older devices only support LUN 0 and do not support I_T_L_Q nexus connections. Newer devices, particularly those that integrate multiple devices at one SCSI identifier, are beginning to appear that have multiple LUNs and require I_T_L_Q nexus connections for optimal performance.

The SCSI initiator's responsibilities include starting I/O processes through an initial connection of the appropriate nexus and checking for appropriate SCSI protocol steps for the lifetime of each I/O process. The SCSI target is responsible for receiving the initial connection from the I/O process and taking the appropriate SCSI protocol steps for the lifetime of each I/O process.

During initial connection the peripheral device receives a SCSI command descriptor block that identifies the particular I/O process function to be processed. The command descriptor block implicitly describes the appropriate SCSI protocol steps the initiator will expect the target to follow and what effect the command and data, if any, have on the peripheral device identified within the nexus.

Typically, I/O processes direct a peripheral to locate and move some number of device-specific blocks of data between the device (through its target interface to the SCSI bus) and host memory (through the initiator interface to the SCSI bus). Fig. 1 shows the typical SCSI bus activity for such an I/O process.

A typical I/O process has three protocol steps—command, data, and status—which use the SCSI phases COMMAND, DATA IN and DATA OUT, and STATUS, respectively.[2]

The target is usually given permission (DISCPRV) to release the SCSI bus at any appropriate time during I/O processing. A correctly implemented and configured SCSI target on a SCSI bus with more than one device on the same bus will release the bus when it anticipates long delays between data transfers. These long delays are typically the result of mechanical movements and occur between the COMMAND and DATA phases, during the DATA phase, or between the DATA and STATUS phases. The MESSAGE phase is used to manage SCSI bus multiplexing of different nexuses.

After a cable delay of a few microseconds, the DATA phases operate at full SCSI rates for the duration of the phase. Fig. 1 shows about 30 SCSI bus primitives of overhead needed to transfer the application data for one I/O process. These primitives transfer at a rate limited by the round trip delay between the initiator and target on the SCSI bus, typically 1 Mbyte/s.

The acknowledgment of each primitive may require processing time and shows up as initiator or target SCSI bus overheads. These overheads range from one to several hundred microseconds up to several milliseconds.

Multiplexing the SCSI Bus between Nexuses

A device that disconnects from the SCSI bus during long delays in I/O processing enables other SCSI devices to use the bus. When the bus is relinquished by a given

Initial Connection	
BUS FREE	
ARBITRATION	Initiator (host) arbitrates for the bus
SELECT Target SCSI Identifier	Host establishes contact with target
MESSAGE OUT DISCPRV, LUN	Host identifies target
MESSAGE OUT TAGGED QUEUEING	
MESSAGE OUT TAG #	
COMMAND	Host issues read or write commands
COMMAND	
•	
•	
•	
COMMAND	
COMMAND	
MESSAGE IN DISCONNECT	Target indicates it will disconnect from bus
BUS FREE	
Data Phase (reconnection)	
ARBITRATION	Target reestablishes a link to the host
RESELECT Initiator SCSI Identifier	
MESSAGE IN LUN	Target identifies itself to the host
MESSAGE IN TAGGED QUEUEING	
MESSAGE IN TAG #	
DATA	Data is transferred
•	
•	
•	
DATA	
MESSAGE IN SAVE POINTERS	
MESSAGE IN DISCONNECT	Target indicates it will disconnect from bus
BUS FREE	
Status Phase (reconnection)	
ARBITRATION	Target reestablishes a link to the host
RESELECT Initiator SCSI Identifier	
MESSAGE IN LUN	Target identifies itself to the host
MESSAGE IN TAGGED QUEUEING	
MESSAGE IN TAG #	
STATUS	Target reports on completion status
MESSAGE IN COMMAND COMPLETE	Done.

Fig. 1. Typical protocol steps for one I/O process on an I_T_L_Q nexus.

Multiple targets on a SCSI bus offer a higher aggregate throughput performance when they use the SCSI bus efficiently and only when necessary during I/O processing. Multiple LUNs associated with a target offer the possibility of more concurrent I/O processes, and even though contention for the SCSI bus may increase, aggregate throughput can increase until the SCSI bus is saturated with activity.

SCSI-2 tagged queuing provides even higher SCSI bus throughput by allowing initial connections to be pipelined to the device controllers so that the devices are never left idle waiting for the next command from the initiator. If the peripheral device driver chooses to allow the device controller to reorder the queue-tagged commands within a LUN then the device can further improve utilization if it can reorder I/O process continuation.

References

1. M. Jerbic, "The HP Vectra 486 EISA SCSI Subsystem," *Hewlett-Packard Journal*, Vol. 42, no. 4, October 1991, p. 70.
2. P. Perlmutter, "Small Computer System Interface," *Hewlett-Packard Journal*, Vol. 39, no. 5, October 1988, pp. 39-45.

Performance Advantages

The major performance difference between the EISA SCSI card and the built-in SCSI interface is that the built-in SCSI runs at the standard 5 Mbytes/s whereas the EISA SCSI card implements fast SCSI-2 timing with a 10-Mbyte/s potential (during bulk data transfer).†

In comparing performance between a 5-Mbyte/s SCSI bus and a 10-Mbyte/s high-performance SCSI bus, one would expect a uniform boost in throughput. However, simply doubling the data rate capability of the SCSI bus is not sufficient. The fixed overhead of initial connection and reconnection and the hundreds of microseconds involved in target and initiator†† overhead typically have a serious impact on performance. Fig. 6 shows the effects of different SCSI overhead times for maximum sustained bulk data throughput of different lengths.

Although a higher transfer rate may also have the advantage of a higher I/O process rate through the system, the increased performance levels require a higher rate of the use of limited system resources like I/O bus cycles and CPU execution bandwidth. These factors can also limit improvements implied by a higher transfer rate.

Meeting the high-performance expectations of a 10-Mbyte/s SCSI bus required focused attention on the overhead contributed by the HP 25525A and on the system resources required per I/O process.

Reducing Overhead

Analysis of the graph in Fig. 6 shows that the benefits of a 10-Mbyte/s SCSI bus are not very significant until overheads are reduced from the millisecond range to the hundreds of microseconds range, particularly for the bulk data lengths of 4096 and 8192 bytes.

An analysis of the interface driver's execution path revealed that the following actions could be optimized in the new interface driver to reduce overhead:

- Consume a queue of ready-to-go initial connections by presenting them to the SCSI bus, minimizing the time that the SCSI bus is idle when the queue is not empty
- Service disconnections and their associated reconnections by managing each I/O process state on behalf of the targets as they progress through each of their I/O processes with minimal delay on the SCSI bus
- Transfer bulk data by sustaining the highest transfer rate that the target is capable of
- Produce I/O process completion messages to the host and its peripheral device drivers, minimizing the time from the target's reporting status until the target frees the bus.

To reduce overhead and minimize the impact on system resources during the operation of the interface driver, we focused on minimizing interrupts and maximizing transfer rates. In addition, we had some features added to the NCR 53C710 chip to help us minimize interrupts (see "Adapting the NCR 53C710 to Minimize Interrupt Impact on Performance," on the next page).

† Bulk data is data transferred during the DATA IN or DATA OUT SCSI bus phases.

†† The initiator is the host system that starts an I/O process on the SCSI bus and the target is the SCSI device receiving I/O requests.

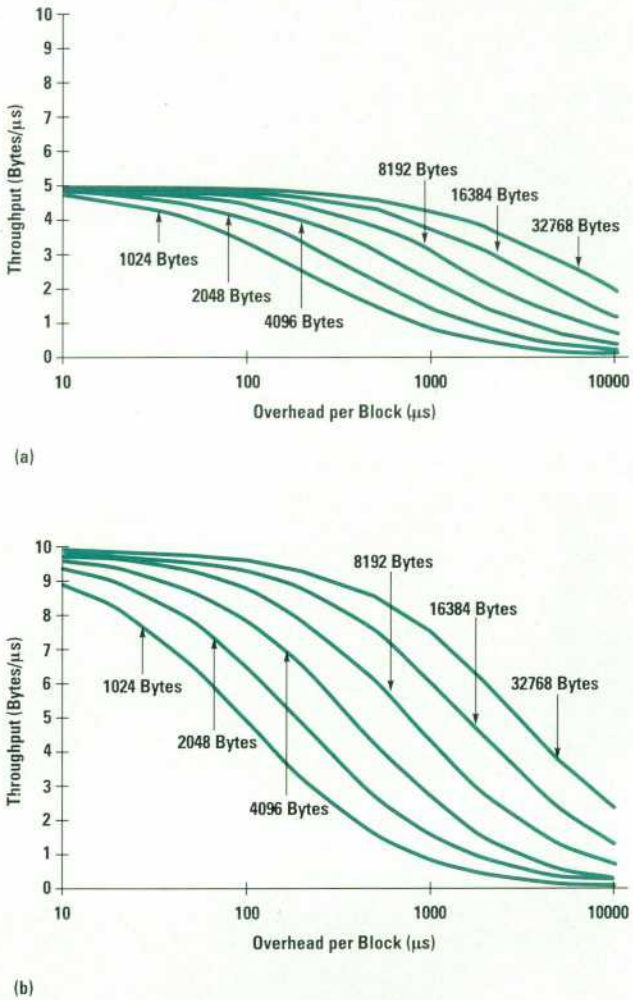


Fig. 6. Estimates of throughput based on different levels of overhead and block sizes for (a) 5-Mbyte/s transfer rate and (b) 10-Mbyte/s transfer rate.

Interrupt Minimization

The most important contribution that the new driver makes toward reducing overhead is the minimization of CPU interrupts per I/O process and the reduction of the performance cost of the remaining interrupts to both the CPU and the SCSI bus.

In our design, the typical paths that a target takes during an I/O process result in one interrupt to the CPU at the end of the I/O process. Perhaps more significantly, for most of the interrupts that do occur in the performance path, the synchronization typically required between a chip and its controlling CPU during a given instance of an interrupt service routine is not required. The CPU and the chip continue on their separate paths while one or the other is servicing an interrupt, or has a pending interrupt.

This independence is accomplished through the use of memory-based messages facilitated by flags and interrupts. The flags are actually parts of SCSI SCRIPTS instructions. This format allows the chip (being much less flexible than the CPU) to participate in this interprocessor communication. Depending on the particular flag, it may be a test condition for a conditional branch instruction or it may be a destination address for a branch instruction. The CPU can set these flags one way, and with an instruction in the NCR 53C710 chip that allows memory-to-memory movement, the chip can set the flags the other way.

The flags can be thought of as part of the messages. These messages change ownership between the CPU and the chip. Interrupts are used to notify the CPU and chip when a flag changes and its message is ready for processing.

The major message types in the design are for starting a new I/O process and notification of completion of an I/O process.

Start of an I/O Process. A message from the CPU to the NCR 53C710 chip to start a new I/O process is initially flagged as owned by the CPU. The CPU fills in the message with SCSI SCRIPTS and SCSI protocol data necessary to start the new I/O process. Then the CPU sets a flag to give ownership of the message to the chip. This handoff is accomplished using the CPU-to-chip interrupt described above. When the chip eventually completes the initial connection phases of the new I/O process, it returns ownership of the message to the CPU by resetting the associated flag.

To minimize CPU interrupts, the chip does not need to interrupt the CPU as part of handing the message buffer back to the CPU. The exception is when the CPU wants to start another I/O process while the chip owns the associated resource message buffer. In this case, the CPU will set another flag to be notified via an interrupt by the chip once the chip finishes with the message buffer.

As the frequency of this condition increases, the number of interrupts per I/O process approaches two. We could have designed out these "extra" interrupts by providing multiple versions of this message buffer. However, we found the frequency too low to justify the cost in design complexity.

I/O Process Completion Interrupt. Initially the flag for the message buffer associated with I/O process completions indicates chip ownership. When the chip encounters an I/O process completion, it will put enough information in the message to indicate which nexus† completed and the completion status. Because of the chip's ability to address itself as the source of a memory-to-memory move instruction, some of the status information is posted to the message buffer by the chip.

Once this information is written to memory, the message-buffer flag is changed to indicate CPU ownership of the message. Since the CPU is anxiously awaiting the completion of I/O processes, the chip interrupts it to notify it of the newly created message. Once the CPU has read the message, it resets the message-buffer flag, giving back ownership of the message to the chip.

In the rare case in which the CPU does not process the message by the time the chip has encountered another I/O process completion, the chip must interrupt and thus halt progress because it only has one message buffer for writing completions to and therefore it doesn't have the ability to remember completion information and continue servicing SCSI bus state changes at the same time.

The process completion interrupt will direct the CPU to process both the pending I/O process completion message and the pending I/O process completion still in the chip. In this case there is only one interrupt for two I/O completions, but the SCSI bus progress is stalled by the rate at which the CPU can acknowledge the message and pending interrupt and then resume chip independence.

A typical target SCSI protocol through the performance path is depicted in Fig. 1 on page 103. Other performance paths follow the same sequencing but the target may choose not to disconnect between the major SCSI I/O process phases of COMMAND, DATA IN or DATA OUT, and STATUS.

Peripherals in the HP-UX operating system environment do not usually disconnect in either the DATA IN or DATA OUT phase

† A SCSI nexus is a relationship that begins with the establishment of an initial connection and ends with the completion of an I/O process. See "Update on the SCSI Standard" on page 103 for more about SCSI nexuses.

Adapting the NCR 53C710 to Minimize Interrupt Impact on Performance

Whenever the NCR 53C710 chip interrupts (by asserting a chip interrupt line) it must first stop executing SCSI SCRIPTS instructions. It resumes executing instructions only after the CPU writes to the appropriate chip register. This is the kind of synchronization interrupt that we wanted to avoid whenever possible so we added some circuitry between the chip and the card's EISA interrupt circuit to allow the NCR 53C710 to assert a general-purpose output pin under SCSI SCRIPTS control which results in interrupting the CPU. In this way the chip can continue executing after interrupting the CPU.

Another problem to solve was how to get the CPU to interrupt the instruction stream of the 53C710 chip without shutting the chip down through a cumbersome abort procedure. We worked with NCR to modify the way some of the 53C710 instructions worked so that for the instructions that wait for new SCSI bus activity to start (the idle state in our design), a bit was added in a register in the chip that the CPU can write to during SCSI SCRIPTS execution. When this bit is set either before or during an idle state, the chip takes an alternate branch from the idle state. This gave us a CPU-generated interrupt to the chip.

These two features gave us the ability to run the CPU and the NCR 53C710 in parallel almost all of the time.

before all of the I/O process' data has been transferred. This is fortunate because, unless the disconnect junctures can be predicted ahead of time, the NCR chip interrupts when this occurs.

It was still a challenge for us to get the NCR 53C710 to be able to process each of the typical target choices through the performance path without CPU intervention once the I/O process is started. However, it was not difficult to implement the case in which the target does not disconnect from the SCSI bus. For the cases in which the target does disconnect, many intervening I/O processes may be started or resumed before the disconnecting process continues. Therefore, the state at which the I/O process disconnects must be remembered.

The I/O process state is remembered by storing it in each I/O process' memory in the form of SCSI SCRIPTS instructions. Initially the chip sets this state as part of the information provided for starting the I/O process.

Once the I/O process is started the chip can access the I/O process state by using what amounts to a computed GOTO instruction in the chip. The actual procedure used is to convert (through SCSI SCRIPTS execution) the SCSI nexus into the address of the SCSI SCRIPTS corresponding to the I/O process next-state actions. When the target reselects a nexus or continues on the same nexus, the associated SCSI SCRIPTS for the next state of the I/O process are executed and modified by the chip to reflect the new next state of the I/O process.

This feature of the design minimizes the time required to react to reconnections, and decouples the response time from CPU interrupt service times. CPU interrupt service times are often just as fast or faster in a lightly loaded system, but as the CPU workload and interrupt rate from other sources increase, the service times increase. However with our design the service time is only dependent on SCSI SCRIPTS memory access times and doesn't contribute to increased CPU workload.

Maximizing Transfer Rate

In our design, the first step in transferring data as fast as the target is capable of handling it is for the EISA data transfer to be able to keep up with a 10-Mbyte/s maximum SCSI transfer rate. The card doesn't buffer EISA data transfers beyond the 64 bytes provided in the chip so it is important for the card to be able to maintain the 10-Mbyte/s rate in real time. This problem was solved in hardware (see the article on page 83).

The I/O map hardware and associated io_services software algorithms are used to map from contiguous virtual memory descriptions of data transfers by user programs, the kernel, and the file system into virtual, contiguous EISA address space. This is done so that the EISA hardware and software aren't burdened with knowledge about PA-RISC page boundaries and their discontinuities in real physical addresses. Although the NCR 53C710 is capable of dealing with non-contiguous addresses by using a separate SCSI SCRIPTS for each physical address range, the cost in performance is such that between SCSI SCRIPTS fetches the SCSI bus is idle and maximum data rates can't be achieved. In the case of our 10-Mbyte/s EISA SCSI interface, the performance cost would have typically been only 1% without using contiguous EISA memory. However, since the io_services functions used for development of the interface drivers performed this optimization, we are able to sustain maximum SCSI burst rates for the duration of bulk data transfer.

Testing the SCSI Interface Driver

Test requirements were recognized as an important aspect of the interface driver development at the very beginning of the project and the software for testing provided special functions and workloads that the regular system software did not normally provide. Development of the test plan and the test software occurred in parallel with the development of the interface driver.

The test plan goals were taken directly from the Roseville Networks Division software life cycle, which include:

- Identifying test cases to verify detailed implementation
- Developing all unit module and integration test code
- Verifying accurate functionality with fully implemented dependencies and hardware.

Software developed to achieve the first two goals included a user-level test program, a test peripheral device driver, and a SCSI target emulator program. The purpose of the test software was to verify conformance to the external and internal specifications by controlling the interfaces above and below the interface driver. The third goal was achieved by using existing stress and busy-system tests.

Implementation

The test environment included an HP 9000 Model 720 workstation with an HP 25525A EISA SCSI host adapter, an HP 1650 logic analyzer with a SCSI bus preprocessor, an HP disk array, several HP 97560 1.3-gigabyte disk drives, and an HP Vectra PC with an Adaptec SCSI target emulator board.

The test software modules and hardware configuration are shown in Fig. 1.

User-Level Test Program. The user-level test program is essentially a test script interpreter. The program reads ASCII data from an input file, invokes the test peripheral device driver, and then writes the test results to an output file.

An input script consists of commands and parameters to execute a test. The basic command is execute an I/O giving the target identifier, the SCSI command descriptor block, the length of the data transfer, and all the other necessary details as parameters. Another useful command is loop last which was used to repeat the previous command a given number of times. With this general mechanism many test cases could be created by just editing the input file and changing the parameters. The test parameters were assembled into a data buffer and passed to the test peripheral device driver through a system control (ioctl()) call.

Additional script commands were used to perform multiple concurrent I/Os, issue a SCSI bus reset, configure the host bus adapter, and provide special instructions to cover corner cases.

On completion of a test script the test title, test number, and results were written to an output file. The title and test

number had a one-to-one correspondence with the test cases in the test plan. The results included all the input parameters as well as any return fields from the driver such as status, SCSI status, and the data transfer residue.

Test Peripheral Device Driver. The test peripheral device driver was developed by starting with a copy of a real peripheral device driver and adding new ioctl() options that correspond to the test commands issued by the user-level test program.

The test peripheral device driver receives a data buffer via the ioctl() call from the user-level test program and is responsible for allocating kernel memory space for I/O control blocks and other data structures, such as the SCSI command descriptor block. The primary function of the test peripheral device driver is to translate the input data buffer into a control block that the SCSI driver can understand. The control block is then passed to the interface driver through the transport layer. After the transaction completes, the test peripheral device driver is revived when its call-back function pointer receives control return parameters.

Although it seems that we could have used a real peripheral device driver for the above functions, we didn't because of timing and control. A good example is the test for exceeding the maximum number of SCSI queue-tagged I/Os. The test peripheral device driver could ensure that the SCSI driver was called with the correct number of I/Os to surpass the SCSI queue-tag limit without HP-UX timing dependencies and we could do this repeatably.

SCSI Target Emulator. The user-level test program and the test peripheral device driver control the type and number of I/Os that the SCSI driver will initiate. On the other end of the SCSI bus we used an Adaptec target emulator test adapter with Adaptec software library functions, which allow great flexibility in manipulating the SCSI bus. Test scenarios using the Adaptec functions were written in the C programming language. The Adaptec functions provided control of each link-level phase of the SCSI bus so that we could perform a wide range of different I/Os and exception conditions. It was necessary to synchronize the number of I/Os between the expected behaviors of the host and the target. However, the target emulator was general enough to read the type of I/O and length of the data transfer out of the SCSI control block.

Test Cases. The test platform described above was used to accomplish over 500 different tests using 48 test suites. These tests were duplicated for the SCSI DATA IN and DATA OUT phases. Each test was also performed for host data alignment boundaries to ensure that data transfers were not page and cache aligned. Some of the major test suites are listed below:

Save Pointers
Restore Pointers
Parity Errors
Short Data Transfers
Overrun Data Transfers
Synchronous Data Transfer Negotiation
Unexpected Disconnect
Unexpected Phase Change
Unexpected Message In
Unexpected Reselector
Message Out Reject
SCSI Bus Reset
I/O Timeout Abort
Selection Timeout

The initial number of test cases for each suite was designed into the original test plan based on experience with earlier host-adapter software. Test cases were chosen to transfer data lengths on boundaries such as 0 bytes, cache line size, and page size. Data transfer lengths were also cycled around these boundaries by counting through plus and minus the SCSI controller FIFO size.

As actual testing progressed the number of different cases in a test suite was increased according to the number of defects discovered. If the initial number of defects for a suite was zero we determined that the tests were adequate. If one or more defects were discovered the number of test cases was expanded. If the new test cases revealed additional defects, the test cases were expanded to approach exhaustion.

One area in which additional testing was required involved a special sequence of data (SCSI DATA IN or OUT) interrupted by messages (SCSI MESSAGE INs). The messages were generated by the Adaptec target device and caused the interface driver to examine or adjust its DMA pointers. The messages included a SCSI Save Pointers or Restore Pointers parameter. These parameters added complexity to the interface driver, which increased the need for extensive testing. Testing revealed that this sequence was further complicated if an unexpected MESSAGE IN was received during the SCSI DATA phase. The interface driver rejects unexpected messages.

White-box analysis of the design showed that receiving two of these messages in a row (separated by data) could also be a problem. From this analysis we concluded that a systematic approach to testing for these sequences was necessary.

The basic test sequence was:

DATA – MESSAGE IN – DATA – MESSAGE IN – DATA – MESSAGE IN – DATA – MESSAGE IN – DATA

The objective was to develop a test matrix that included all possibilities of the basic test sequence. The MESSAGE IN parameters were either Save Pointers, Restore Pointers, or an unexpected MESSAGE IN. Also, one of the parameters could repeat. The following sequences represent three of the 36 test cases we derived.

1. DATA – Save Pointers – DATA – Save Pointers – DATA – Restore Pointers – DATA – Unexpected MESSAGE IN – DATA

2. DATA – Save Pointers – DATA – Save Pointers – DATA – Unexpected MESSAGE IN – DATA – Restore Pointers – DATA

3. DATA – Save Pointers – DATA – Restore Pointers – DATA – Save Pointers – DATA – Unexpected MESSAGE IN – DATA

Examination of available literature on statistics did not provide a convenient formula for choosing four parameters out of three possibilities when one of the choices could repeat. Empirical reasoning led to the following solution. Permutations of three taken three at a time (3!) equals six. Now if we begin each of the six permutations with one of the duplicate choices (as shown in the above sequences) we get 18 (3×6) test cases, and if we end each of the six with a duplicate we end up with a total of 36 test cases. Each of these test cases

was also accomplished with data transfer lengths greater than and less than the number of bytes the host expected.

Regression Testing. Test execution was fully automated allowing the tests to run with no intervention by the tester. First a master results file was created by manually executing and verifying the tests one at a time. The test number was compared to the number in the test plan and the SCSI bus sequence displayed on the logic analyzer was inspected to ensure that it matched the expected sequence. Identical output was written to the display and the results file and these results were examined to be sure that they were correct. This results file then became the master results file and was stored in a separate directory. Differences in subsequent test runs could be observed by comparing the current results file with the master results file.

Results

After each test in the test plan was completed successfully an instruction coverage analysis tool was used to measure test coverage. This tool was used to determine how many lines of code were actually executed. The analysis tool was only used after every test in the test plan had been completed since it is not a good substitute for designing test cases. The tool does identify code that is never executed. The first time the analysis tool was used it reported that we covered approximately 83% of the code. The 17% of code that was not executed fit into roughly three classes: code that should have been covered by the test cases, code that had no corresponding test cases, and code that required catastrophic hardware failure or extreme system conditions.

For the first class we identified the test cases that should have caused execution of the driver code and set breakpoints to determine exactly why the code was missed. For some cases defects were uncovered in the driver and in some cases the target emulator was not doing what it was supposed to be doing. These conditions were corrected.

In the second class, test cases were added to the test plan and implemented to cover the code not executed. Most were simple omissions, but a few were extremely difficult to cover. One case in particular required that the data transfer size of all outstanding I/Os be great enough to exhaust the system's I/O map allocation of resources and the number of outstanding I/Os be large enough to cause the driver to allocate a new page of memory for its data descriptors used for DMA parameters. The simultaneous occurrence of these two events required approximately three days of test effort. The result was worth it because when the block of code in question was finally executed the system went into a panic. The problem was traced to a member of a structure passed as an argument to an io_services routine that was not pointing to the appropriate data structure, and it was only used in this case. In hindsight it is easy to claim that the problem should have been found through inspection. However, this particular block of code was exceptional enough to merit testing.

After adding the cases for the first two classes, the coverage analysis results improved to 93%. The remaining 7% of the code was carefully inspected and consisted of cases for exceptions resulting from abnormal hardware conditions such as an unexpected interrupt from the SCSI controller and a bad REQ/ACK handshake on the SCSI bus. Since recovery from these conditions used the same code as conditions we were able to test, we were satisfied with the quality of the test effort and its results.

Acknowledgments

The EISA SCSI project was accomplished under the incredible time constraints required to get the Series 700 workstations to market in a timely fashion. Special credit should be given to Harry Feit, our section manager, who gave us the power as well as the responsibility to make decisions, Don Ciaglo, the I/O program manager for the Series 700, who made sure no equipment or other resource was lacking, and Joel Dunning, our project manager, who removed all obstacles to our getting our work done. We also wish to thank our many compatriots who shared resources with us while they developed their own cards. At times access to equipment changed every hour as one team completed a test and passed it on to another team. At one point design engineers carried around 1.3-Gbyte disk drives to provide complete versions of their test environments to the next available test station to minimize the setup time. We would also like to thank Mike Jacobson and Randy Matthews from the HP Disk Array Laboratory for their work on the software associated with interfacing disk arrays to our EISA SCSI implementation. Finally, thanks to Rich Testardi from HP's HP-UX Development Laboratory for his work in helping to develop the interface driver architecture and his contribution to performance considerations for the HP EISA SCSI card design.

An Architecture for Migrating to an Open Systems Solution

A process and a model have been developed that provide an easy growth path to a client/server, open systems architecture for information technology applications.

by Michael E. Thompson, Gregson P. Siu, and Jonathan van den Berg

As we move further into the 1990s, information technology is undergoing rapid changes and significant challenges. Business decisions are evolving rapidly to meet competition and satisfy customers. Technology is offering more choices than were available in the past decade. Emerging client/server technologies and competitive business needs are forcing information technology groups to reevaluate how high-quality business solutions can be delivered faster.

HP's Worldwide Support Systems (WSS) organization is an information technology group that develops mission-critical† systems for Hewlett-Packard's worldwide customer support business. For the last decade, WSS has developed software applications using traditional technologies, tools, and processes.

During the late 1980s, WSS embraced client/server technology to meet rapidly changing business needs. Client/server technology is a form of distributed computing in which application processing is divided between a client process and a server process. The interaction is simple: the client process initiates requests to the server and the server fulfills the request and responds to the client. This article discusses our experiences with developing client/server solutions using open systems tools and technologies.

Open systems provides a unified approach for developing and managing systems, networks, and user applications, resulting in the ability to run a single version of a software application on different hardware platforms. Because they communicate using standard protocols, products based on an open systems design can be interconnected and will work together regardless of what company manufactured them. For a user configuring a computer system or network, the benefit of open systems is the freedom to choose the best component for each function from the offerings of many manufacturers.

Our experiences began when the open system concept was in its infancy so we learned along with the rest of the industry. There were no clear leaders for each technology component required to develop a complete client/server solution. We believe our experiences to be extremely valuable to those who are considering client/server development solutions while open systems standards are being defined.

† Mission-critical systems are software applications that cannot be taken offline for an extended period of time without adverse impact (e.g., order management, engineer dispatch, inventory control systems, etc.).

The Early 1980s

During the early 1980s, the primary focus of our use of information technology was to increase productivity in the support community through automation. Most of Hewlett-Packard's customer support business was organized around area and region business units. Each of these business units used HP 3000 systems as the primary, and sometimes only, business computer. Because these business units were decentralized, transaction volume was relatively moderate.

Our applications were designed to run on HP 3000 computers from block mode terminals using the VPLUS/3000 screen handler. Interactive dialogs were rarely used in applications. When communication between applications was required, a batch solution was usually the answer.

Systems were designed with very little effective code reuse. Business edits, database access, and screen I/O were all mixed together into a module or program. Our technology choices were limited to COBOL, Image (DBMS), and the VPLUS/3000 screen handler (Fig. 1).

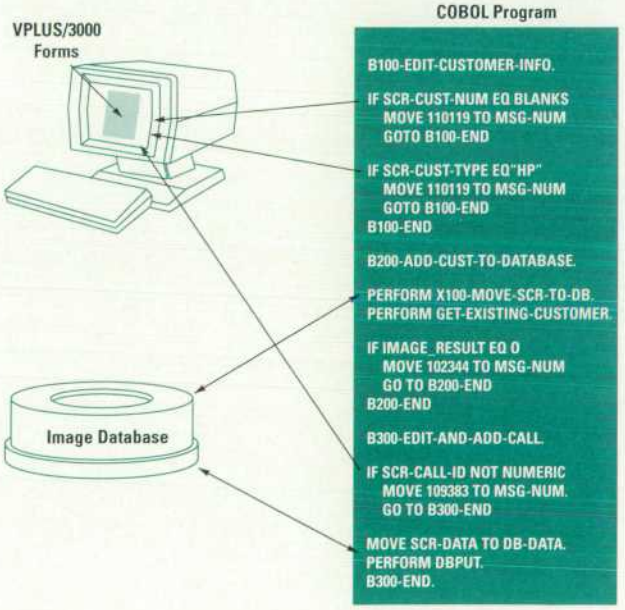


Fig. 1. Traditional architecture for early 1980s support applications. This architecture consisted of a single executable program that handled its own database calls and used a forms package for the user interface.

There were few choices, if any, for software that would run across multiplatform or multivendor hardware or software. Introducing new technology into this type of environment was rarely considered. System developers worked within a limited technology environment where constraints existed because of the HP proprietary hardware selected.

The Late 1980s

Into the mid-to-late 1980s, different technology choices were becoming more available. Business managers were asking if we could deliver our application systems on PCs or HP-UX* workstations. The HP-UX operating system, computer-aided software engineering (CASE) tools, object-oriented languages, and relational database management systems (RDBMS) were emerging as popular alternatives to the traditional technologies and tools we had used over the past ten years.

We were faced with the challenge of developing a strategy to introduce these new technologies into our existing systems without any delay to our customers. This challenge forced us to reevaluate our most fundamental principles of systems development.

Technical Architecture

In the fall of 1989, WSS created a technical architecture to provide a foundation for developing a consistent long-term design strategy for systems development. This architecture provides a common framework for designing modern, integrated client/server systems. These systems are integrated in the sense that they can operate and interact with other applications through common application program interfaces (APIs) using integrated technology tools and components. The components of our current technical architecture are shown in Fig. 2.

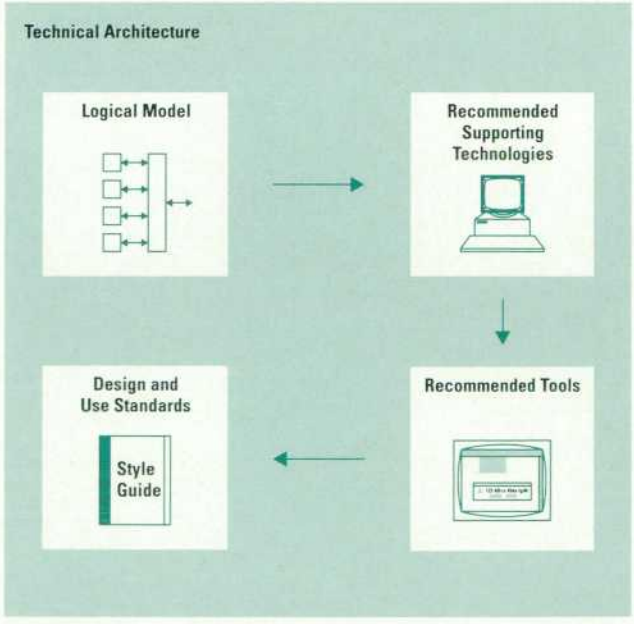


Fig. 2. The new technical architecture.

Logical Model. This portion of the technical architecture defines a template that can be used to create or modify applications that are compatible with the open systems concept. The logical model is described in more detail below.

Recommended Supporting Technologies. Supporting technologies include hardware platforms, operating systems, network services, languages, databases, user interfaces, and development environments that support the logical model.

Recommended Tools. These are tools used by application developers, designers, and maintenance personnel for the creation and support of application software. Tools such as integrated CASE and branch flow analysis are some of the recommendations.

Design and Use Standards. This part of the architecture consists of style guides, interface definitions, and practices. These standards define a common set of terms and a consistent framework. Style guides include items such as guidelines and standards for design and coding. Practices means process standards such as code inspections and testing strategies for client/server implementations.

In summary, this architecture has been adopted as the model for HP information technology, which is an internal HP function responsible for developing and maintaining information systems applications and networks for organizations such as marketing, support, and accounting. The architecture is constantly being revised as new technologies, tools, standards, and other methodologies become available.

The technical architecture enables an application designer to define an application in terms of the logical model components, select the technologies needed to support the application, select specific tools for implementing the design, and follow design standards so that the application will have a common look and feel and be able to interact consistently with other applications.

Logical Model

The logical model shown in Fig. 3 is a definition or template of how an application can be partitioned for an open systems solution. It is a framework that organizes all applications into five components: user interface, user task logic, business transaction logic, data manager, and environment manager. This model is a good basis for researching and recommending open systems strategies for application development. In addition, it defines a framework for strategies relating to application migration and third-party purchases. Finally, this model does not imply any specific technologies or client/server configuration because these decisions are made in other parts of the technical architecture.

User Interface. The user interface provides the user's view into a business application, and manages all communication between the user and the application. It is responsible for painting the "picture" the user sees and collecting user input. It constructs dialogs by presenting data or choices and requesting selection, data input, or other responses.

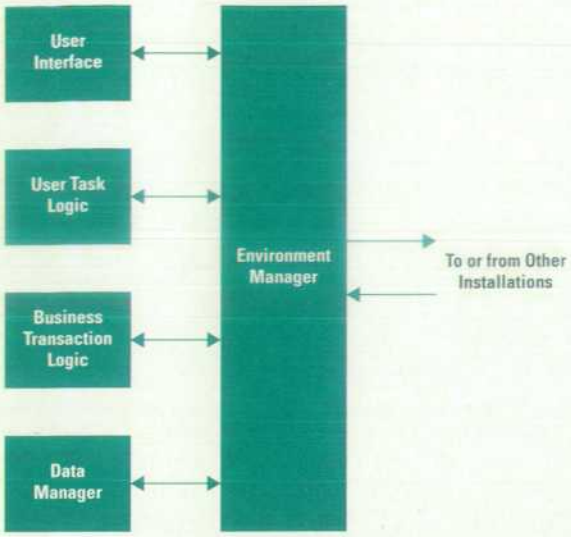


Fig. 3. The logical model for open-systems design.

A good example of a user interface is the automated teller machine, or ATM, which is used for automated banking transactions. The ATM provides an interface that enables a user to perform a bank transaction. The interface itself is mostly idle. However, it will interact with the user by presenting data such as accounts and balances, and requesting selections such as withdrawals or deposits.

User Task Logic. The user task logic drives the user interface and invokes the appropriate business transaction processing. Understanding the results of user requests and communicating them to the user interface (where the results are displayed) are also the responsibilities of the task logic layer. For example, in the ATM example, when the user selects an option (such as withdrawal), the ATM application (client) formulates a request to the bank account management application to perform the withdrawal. The ATM application waits for a response from the account management application. Formulating the request to the bank application and communicating the response to the user interface are task logic functions.

Environment Manager. This component is responsible for managing communication between the other four components of the logical model. Communication responsibilities include connecting two components (dynamic and static connections), overall management of application layer instances, and communication from one client/server installation to another. For example, in the ATM model, the environment manager allows clients to transfer funds from one banking establishment to other banking establishments.

Business Transaction Logic. The business transaction logic creates, deletes, updates, and retrieves business data. It imposes business policy or rules upon the data. Transaction and data access security is managed in this layer to control authentication of the user requesting the transaction.

After the user presses "OK" on the ATM user interface, the task logic will transmit the withdrawal request to the banking application. The requester of the transaction is validated, the withdrawal request is edited, and account balances are checked. These are all functions performed by the business transaction logic.

Data Manager. This layer manages the physical storage of data, and provides read and write access to the data. It manages concurrent access by multiple users (business transactions). It is responsible for ensuring the physical integrity and recovery of data. In the ATM example, the banking application receives requests from many users at one time. The function in the banking application that updates the user's account with the withdrawal information and locks the account until the withdrawal is completed is located in the data manager.

Client/Server Architecture

The five components of the logical model are interconnected through message-based interfaces that allow the components to be separated physically into a client/server configuration. Fig. 4 shows an overview of our client/server architecture. Applications are organized into a client, a server, and a client/server interface. The client, which consists of the user interface and user task logic, deals with presenting and managing data for users. The server, which consists of business transaction logic and the data manager, deals with managing the business data and enforcing business rules and policies. Finally, the environment manger provides the client/server interface, which is a transparent network linkage between client and server.

Migrating Existing Systems

To test the feasibility of the technical architecture, we used it on two existing production applications. The first project consists of 100,000 lines of code and is used to dispatch and manage software-support service calls for HP's customer response center. This project was a step-by-step migration from the traditional application code architecture (shown in Fig. 1), which typically dispersed task and business logic throughout the application, to the client/server structure

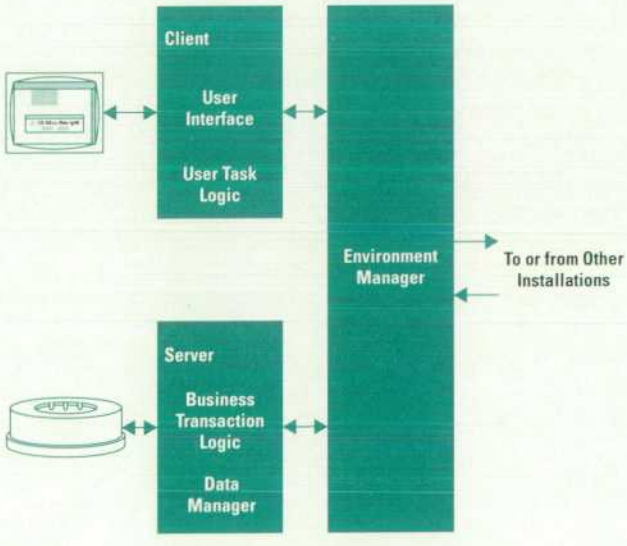


Fig. 4. The client/server logical model.

defined by the technical architecture. The original application ran in an HP 3000 MPE operating system environment and consisted of an Image database, COBOL application code, and VPLUS/3000 online access code. The new client application uses a graphical user interface and runs on HP 9000 Series 300 and 400 machines in an HP-UX operating system environment. The server portion of the application runs in native mode† on an HP 3000 Series 900 machine running in an MPE/iX operating system environment.

One of the objectives of this migration was to salvage as much of the original application as possible while reengineering the technical infrastructure. The existing COBOL code was a mixture of online code and database access operations. We took the following steps in migrating the old application to the new client/server architecture:

1. We evaluated existing code to determine which database access operations should be combined with application edits to ensure the same data integrity and performance.

2. The results from step 1 were encapsulated in a predefined interface which became the single access path for all external processes (clients and other batch applications). This portion of the application is the server in the client/server logical model.

3. We evaluated the existing code again to determine the online application logic that had to be reengineered to run in a new hardware and software environment. This environment included an HP 9000 Series 300 or 400 running the HP-UX operating system and a graphical user interface based on OSF/Motif. This portion of the application is the client in the client/server logical model.

4. The connectivity component that enabled the application component from step 2 to interact with the application component from step 3 was designed and developed. This portion of application and other supporting software represents the environment manager in the client/server logical model.

Fig. 5 shows the components of the new system encapsulated in the client/server model.

The technical architecture helped to describe the division of responsibilities between the client and the server. The biggest impact the architecture had was in determining the location of the network. The logical model does not prohibit an application from introducing a network between any two layers. However, the client/server view of the logical model recommends splitting the application with a network between the user task logic and the business transaction logic.

The second project involved the development of a new client application with a graphical user interface that runs on HP 9000 Series 300 and 400 machines and a server application that runs in native mode on an HP 3000 Series 900 machine. This application configures and prices support products for HP's customer support business. The project consisted of approximately 40,000 lines of code.

The success of these two projects proved that the technical architecture, particularly the logical model, provided us with a framework for migrating our installed base of applications to new hardware and software platforms and developing new applications.

† Native mode implies that a machine is using its own capabilities (capabilities inherent to the machine) as opposed to compatibility mode, in which a machine is emulating another machine.

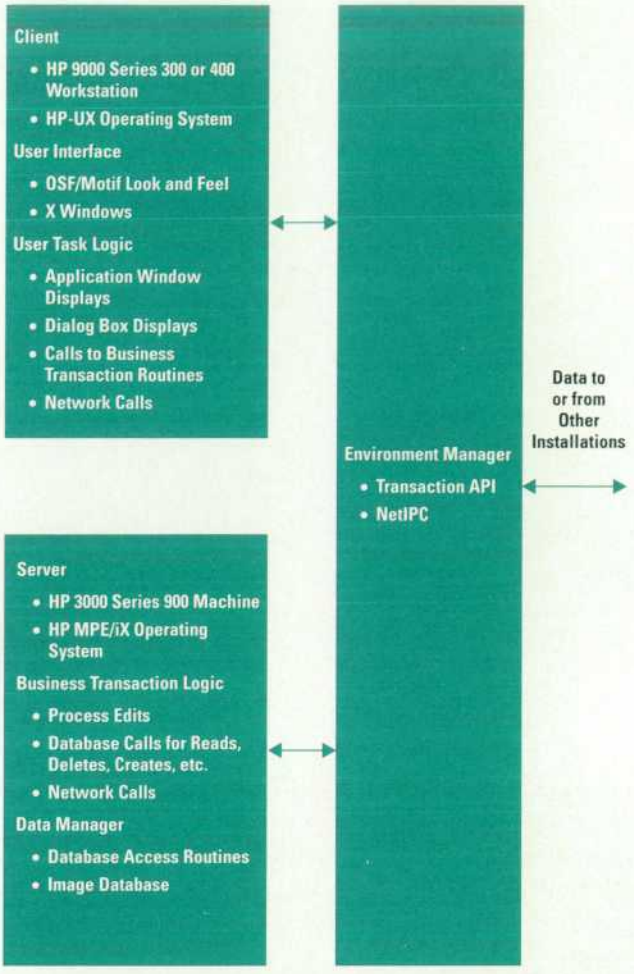


Fig. 5. The customer call tracking system after migration.

Process Learning Experiences

Despite the success of the two projects mentioned above, both projects exposed areas where we needed some improvements in both process and technology.

Business Transaction Design Time. Defining appropriate business transactions requires a lot of work at the beginning of a project. However, investing this extra time helps build a foundation for the rest of the application design. Once these transactions are defined, development teams can work in parallel.

Business transactions provide the interface between clients and servers. If the transactions are designed properly, clients will be able to communicate with any server (aside from security restrictions) using the business transactions defined for it. The design of either the client or the server can change without affecting the other as long as the business transaction rules or the transactions themselves are preserved. New clients can easily be added to any server by simply using the relevant transactions. Therefore, maintenance is decreased and reusability is increased. In addition, if business transactions are defined correctly, network traffic will be reduced.

During the call tracking project, we spent only 10% of our design, code, and test time working on business transaction design. However, with the configuration and pricing project, we invested approximately 50% of our initial work on the transaction definition. These transactions defined our application more concisely, which gave us better results and less rework as the project progressed.

Use of Information Engineering. The technical architecture provides one of the building blocks for application development. However, to meet certain business needs and improve the development process, parallel efforts such as usability engineering, performance engineering, and information engineering are recommended.

We spent a lot of time reworking the transaction definitions and code for the call tracking project because we did not do a thorough analysis of the problem. Using information engineering techniques in the beginning of the project can save time and effort. Information engineering provides a structured methodology and guidelines for discussing the how, what, and where aspects of development. The how relates to the processes in the application, the what is the data the applications use, and the where refers to the location of the solution (in hardware or software).

Use of Data Flow Diagrams. During the configuration and pricing project, we used data flow diagrams (DFDs) and structured analysis and structured design techniques to help represent the flow of data among the modules of the program. This enabled us to address critical elements during transaction definition. Data flow diagramming is the link between the analysis and design phases of information engineering.

Increase Training Time. Migrating the call tracking application was among the first client/server projects for HP information technology. Because the client/server model brought a new environment to HP information technology, users and developers had to become proficient in new skills and a new way of thinking about organizational relationships, applications, processes, and technology. The learning curve for both our developers and our users was substantial. We spent more time training our developers, and did not spend enough time training our users. The amount of time needed to train our users was significantly underestimated. This greatly increased our acceptance risks during the implementation phase of the project.

Consider Performance Needs. We focused our efforts more on network performance than client performance, which later became an issue. Client performance includes the response time from one logical task (such as selecting help, making a menu choice, or editing a dialog box) to the next logical task. The client application is driven by the end user and therefore must perform according to end-user expectations. Thus, the performance needs of users need to be considered so developers can balance new development tools and requirements with the hardware platforms that users demand.

Correct Hardware and Software Infrastructure. The expense of migrating to a client/server environment should be considered. While migrating our applications, we discovered that we did not have the correct hardware and software infrastructure in place. We needed high-speed networks which were not available at all sites. This limited our communication capabilities. In addition, we still used old hardware that had memory limitations. This limited our ability to incorporate new technologies, causing performance problems.

Adopt a Migration Plan. Migrating from a traditional (terminal, host-based applications) architecture to a client/server architecture can be approached in phases. Existing databases should be surrounded with a server shell. Clients can communicate with servers using properly designed business transactions. This client/server separation allows the database to be replaced later without replacing the client. Unmigrated applications can be integrated on the user's workstation, using terminal emulation, until they are migrated to a new platform.

Technology Lessons Learned

Besides the process improvement lessons described above, we also learned some lessons about the technical architecture and our design model.

Adopt a Technical Architecture. Today, internal customers demand multivendor network connectivity providing standardized application services. Open systems will allow selection from a wide range of multivendor solutions. Although open systems standards are still evolving, adopting a technical architecture will help provide guidelines for selecting technologies and tools that support open systems. This reduces the risk of locking application developers into a vendor-specific framework of technologies, tools, and processes. As standards become defined or new technologies are introduced, migrating to open systems will be much easier.

Defining and standardizing interfaces allows application logic to be independent of technology. The API is a standard library of functions that separates the user from the underlying technology. Because an API can be used to encapsulate a technology, changes within the encapsulation can occur without affecting application code. For example, technologies (tools) providing the interface to a database can change without impacting the business transaction logic.

Our migration strategy required interfaces between clients and servers to be standardized, normalized, and portable. For example, HP's response center lab developed a network API called TVAL (tagged values), for interprocess communication between HP-UX and MPE/iX operating systems. This API will protect our investment during migration from NetIPC to an open systems product such as Network Computing System (NCS).

Develop a Business and Data Model. Implement a model to provide a description of the business that is going to be supported. This model will document what processes the business needs to meet its goals, and what information is needed for each process. Once this information is determined, develop a data model to provide consistent definition and interpretation of the data wherever it is used, and of business rules that determine its integrity. This will allow different applications to be consistent and to interact successfully. The objective is to manage data so that it is defined only once, although it may be used by multiple processes or applications (clients) and physically stored in several locations (servers).

Develop Data Security. As we migrate to an open systems environment, customers from outside HP will be able to access our systems. In our current environment, systems only check for valid identification and authorization. Standards groups are working to establish a Distributed Computing Environment (DCE) that includes security services. DCE is a comprehensive, integrated set of services developed and sponsored by the Open Software Foundation (OSF) which supports the development, use, and maintenance of distributed applications. DCE supports transparent connection and communication of any object (client or server) within a distributed networked environment.

Benefits

While developing and migrating our applications, we also discovered some unexpected benefits.

Development Team Independence. Splitting applications into client, server, and communication components encourages application development across organizational boundaries. Thus, applications can be developed in parallel, which decreases the time to market. The call tracking project was developed by two different development teams in HP's support organization. This allowed the teams to be managed separately while the application was being developed simultaneously. The configuration and pricing project teams were also split between client, server, and communication logic. After defining the various APIs (communication, error handling, and data buffer) between the client and the server, each team was able to develop and test its part separately. The teams worked independently and did not need to be concerned with the internals of the other components. The three parts were brought together successfully during integration testing.

Lower Maintenance and Higher Quality. Both projects greatly simplified maintenance. We noted that most service requests associated with the the initial release of two products were for changes to client behavior, not business logic. Because the user task logic (client) is separate from the business transaction logic (server), the amount of code to be reviewed before a change can be implemented is greatly reduced. This allowed us to estimate schedules easier and have faster turnaround for application change requests. Lastly, the development teams became familiar with the application quicker because they did not have to learn both the user task logic and the business logic.

Since their initial release both projects have exhibited a very high quality. In addition, the application teams consisted of eight to ten people for the initial release and only one to two people during the subsequent releases.

Improved Application Testing. Server testing was automated by defining a single interface to the server. We developed a test driver that could send predefined test cases, capture server responses, and compare them with expected results. We developed 2500 reusable server test cases. This accomplished about 90% of our unit testing. In addition, the complete test for the server takes only two hours, allowing a full regression test suite to be run whenever the server is changed. This allows the team to perform verifiable regression tests quickly as needed.

Conclusion

Developing a client/server solution using an open systems design strategy provided several learning experiences. These experiences included leveraging current investments of hardware, software, and personnel, choosing the right products and standards to integrate software, avoiding software and hardware lock-in, and incorporating new technologies.

The technical architecture provides a framework for deciding which technology and tools to use for client/server development. Once this architecture is defined, migrating and developing applications to the client/server paradigm can be accomplished without completely rewriting existing applications.

In addition, developing client/server applications using the open systems concept means that a development environment can be tailored to meet specific business needs. This minimizes the risk of locking developers into specific hardware and software platforms and processes. Resources can be spent on designing the best combination of existing and new components.

Finally, we have found that the technical architecture described in this paper will work because it provides an open systems solution, offering a simple migration path to a client/server architecture.

Acknowledgments

Gratitude to Skip Ross and his staff—including the development team and production application teams—for their great efforts in following this project through. Special thanks to Tobin Cudd and Greg Spray who made significant contributions to the design and development of the technical architecture.

Authors

December 1992

6 DesignJet Plotter

Robert A. Boeller

Bob Boeller is an R&D section manager at HP's San Diego Technical Graphics Division. A native of Long Island City, New York, he attended the Polytechnic Institute of Brooklyn, graduating with a BSEE degree in 1971 and an MSEE degree in 1972. He joined HP's San Diego Division in 1972 and contributed to the design of the HP 3969A tape recorder. He then served as project manager for the HP 7310A thermal printer and the HP 17623A graphics tablet, and was program manager for the HP 7586A plotter, the DraftMaster large-format pen plotter, and the DesignJet plotter. He is named as an inventor in three patents related to large-format plotters and serves on the board of directors of the Executive Programs of the University of California at San Diego. Bob is married and has two children. He is an avocado farmer and raises show ducks.

Samuel A. Stodder

Development engineer Sam Stodder received his BSME degree from the University of California at Irvine and joined HP's San Diego Division in 1984. He contributed to the design of the paper axis mechanics and worked on print quality issues for the DesignJet plotter. He is named as an inventor in two patents on paper positioning systems. He is married, has a daughter, and enjoys boardsailing, surfing, and sailing.

John F. Meyer

As a manufacturing development engineer, John Meyer was responsible for media development for pen and thermal inkjet plotters, including the DesignJet. Now an R&D engineer at HP's San Diego Technical Graphics Division, he received his BS degree in chemical engineering from the University of Detroit in 1967. Serving in the U.S. Air Force for five years, he did photographic R&D, authored a series of seven reports on photochemistry image quality, and attained the rank of captain. Later, with Snook Corp., he developed photoprocessing equipment. In 1973, he joined the HP Stanford Park Division's Microwave Technology Center and served as a facility, safety, and environmental engineer and as facility engineering manager. John is a native of Roanoke, Virginia. He is married and has one child. His interests include woodworking, photography, and racewalking.

Victor T. Escobedo

Development Engineer Victor Escobedo joined HP's San Diego Division in 1983. He has contributed to the design of disposable plotter pens, various DraftMaster and DraftPro plotter models, and the DesignJet plotter, including the media stacker. He is named as a coinventor in a patent on a device for stacking cut sheets. A native of Mexicali, Baja California, he received his BS degree in mechanical engineering from California State University at San Diego in 1985. Victor is involved with education, including the Choices program and universities in Mexico. His interests include soccer, diving, and travel.

16 DesignJet Electronics

Alfred Holt Mebane IV

Holt Mebane was the lead firmware designer for the DesignJet plotter. Born in Greensboro, North Carolina, he received his BSEE degree from the Georgia Institute of Technology and joined HP's San Diego Division in 1980. He has designed firmware for the HP 7570A, 7575A, 7576A, and DraftMaster plotters and is named as an inventor in a servo control patent. Holt's interests include golf, amateur radio (N4HR), skiing, and home improvement. He is married and has two children.

James R. Schmedake

A hardware development engineer at HP's San Diego Technical Graphics Division, Jim Schmedake was responsible for the design and development of the processor board for the DesignJet plotter. He joined HP in 1984 with six years of experience in embedded processor design, and has had a variety of assignments including ASIC design and inkjet pen development. He received his BSEE degree from the University of California at Santa Barbara in 1978 and his MSEE degree from California State University at San Diego in 1984. Jim was born in California but grew up in Oregon. He is married, has three children, and serves as a Cub Scout leader. His leisure activities include running, backpacking, and softball.

Iue-Shuenn Chen

Now a principal engineer at HP's Asia Peripherals Division in Singapore, Iue-Shuenn Chen was with HP's San Diego Technical Graphics Division from 1984 until this year. As a manufacturing development engineer for four years, he designed custom production testers and controllers for the HP 7550A, DraftPro, DraftMaster, and ColorPro plotters. Moving to R&D, he was responsible for the DesignJet ASIC architecture and development methodology, the design of the pen interface ASIC and the carriage ASIC emulator, and RFI and ESD testing of the ASICs. He is named as an inventor in a patent on the ASIC architecture. He received his BSEE degree in 1984 from the State University of New York at Buffalo and his MSEE degree in 1987 from Stanford University. Iue-Shuenn was born in Taiwan. He is married and has one child. He is interested in church activities and music.

Anne Park Kadonaga

Anne Kadonaga is a hardware engineer/scientist at HP's San Diego Technical Graphics Division, specializing in ASIC design and computer-aided design methodologies. A graduate of the Massachusetts Institute of Technology, she received her SB degree in electrical engineering in 1982 and her SM degree in electrical engineering in 1984. At MIT she worked as a research assistant and as a teaching assistant in VLSI design. She first joined the HP Data Terminals Division in 1980 as a co-op student, then rejoined the Systems Technology Division in 1984. She designed I/O boards for data terminals and burn-in system hardware for the HP 2700A color graphics workstation. She also worked on a CPU, a cache controller, and a system interface unit for two generations of high-end PA-RISC chipsets for HP 9000 and HP 3000 computers. She designed the processor support ASIC for the DesignJet plotter and the main board for the DesignJet 600 plotter, and worked on print resolution enhancement for the DesignJet 600. She is named as an inventor in several pending patents related to the DesignJet 600. Anne was born in Seoul, Korea and grew up in California's Silicon Valley. She is married and has a son.

24 Pen Alignment

Robert D. Haselby

With HP since 1973, Bob Haselby is a member of the technical staff at HP's San Diego Printer Division. He has contributed to the design of the HP 9872A and 7221A plotters, was electronic project leader for the HP 7580A plotter, and has designed inkjet test systems. For the DesignJet plotter, he designed the automatic pen alignment software and optical systems and the drop detector system. He is the author of a paper on stepper motor control and is named as an inventor in six patents and several pending patents, mostly related to optical alignment systems and DesignJet alignment. His current professional interest is sensor systems. Bob was raised on a farm in Indiana and served in the U.S. Navy submarine service for six years. His BSEE and MSEE degrees are from Purdue University (1972 and 1973). He is married, has two children, and enjoys sailing.

28 DesignJet Chassis

Timothy A. Longust

Mechanical design engineer Tim Longust joined HP's San Diego Division in 1979. He has contributed to the design of the HP 7580A, 7585A, DraftPro, and DesignJet plotters and the PaintJet printer, and is named as a coinventor in patents on the DraftPro ornamental design and the DesignJet chassis assembly. He's now at the Vancouver Division working on DeskJet printers. Born in East St. Louis, Illinois, he received his BSME degree from the University of Illinois in 1979. In 1989 he received an MBA degree from the University of San Diego. Tim is married, has a son, and enjoys hiking, camping, and basketball.

32 Architecture Development

David M. Petersen

For the DesignJet plotter, Dave Petersen served as a mechanical architect and designer and as a manufacturing support engineer. Previously, he was a designer for the HP 7470A and 7475A plotters, mechanical project leader for the DraftPro plotter, and manufacturing engineering supervisor for the DraftPro. He is named as an inventor in two pending patents on the DesignJet chassis and media cutter. With HP's San Diego Division since 1978, Dave recently transferred to the Barcelona Peripherals Operation, where he is a member of the technical staff and a technical contributor. A native of Palo Alto, California, he is married and has two sons.

Chuong Ta

Chuong Ta was project manager for the DesignJet plotter at HP's San Diego Technical Graphics Division. A native of Ha Noi, Vietnam, he holds a BSME degree from the National Technical University of Vietnam, a BSME degree from the University of Minnesota, and an MSME degree from California State University at San Diego. With HP since 1979, he has also been a project leader for the PaintJet printer. Four patents have resulted from his design work. Chuong is married, has two children, and enjoys volleyball, skiing, camping, and hiking.

35 DraftMaster Plus Plotter

Robert W. Beauchamp

Robert Beauchamp is a design engineer at HP's San Diego Technical Graphics Division. He joined HP in San Diego in 1980 and has contributed to the design of the PaintJet and DraftPro plotters and the SurePlot pens and optical line sensor for the DraftMaster Plus plotter. Three patents have resulted from his work on these projects. A graduate of California Polytechnic State University at Pomona, he received a BS degree in mathematics in 1977 and an MS degree in engineering in 1980. Before joining HP he was with the Jet Propulsion Laboratory, where he worked on wideband error correction schemes for Voyager transmissions. He is a member of the ASME and a licensed professional mechanical engineer. Robert was born in Warwick, Virginia. He works with local school children as a Choices volunteer and says that he enjoys running, bicycling, surfing, and "long walks along the beach with my wife and daughter."

Josep Giralt Adroher

R&D engineer Josep Giralt has been with HP's Barcelona Peripherals Operation since 1989. A specialist in computer graphics and firmware, he has contributed to the design of the DraftMaster BX/MX user interface, developed line sensor algorithms for the DraftMaster Plus plotter, and designed and implemented firmware for the DraftMaster Plus. Currently, he is implementing a software development environment and defect tracking system. He holds a degree in electrical engineering from the Polytechnical University of Catalonia and is a member of the ACM and Eurographics. Before joining HP he was with the computer science department of the same university and later with the Siemens software development center. He was born in Barcelona, is married, and enjoys skiing and soccer.

Joan Uroz

As a quality engineer with HP's Barcelona Peripherals Operation, Joan Uroz was responsible for performance testing and competitive analysis in the development of the DraftMaster Plus plotter. Since joining HP in 1989, he has done similar work for earlier DraftMaster plotters. He is a graduate of the Universidad Politecnica de Barcelona and has an Ingeniero Superior de Telecomunicacion degree with specialization in electronics. He is particularly interested in applications of machine vision to print quality evaluation. Before joining HP he developed flexible workcells for Lucas CAV and fluids measurement equipment for Schlumberger. He is named as an inventor in a patent on a remote reading system for home gas meters. Joan is a native of Guadix, Granada. He is married and has two daughters. His interests

include skiing, climbing, and other activities in contact with nature.

Isidre Rosello

Isidre Rosello was R&D project manager for the DraftMaster Plus plotter at HP's Barcelona Peripherals Operation. He is named as an inventor in a pending patent on the SurePlot drawing system. Now an R&D program manager, he is particularly interested in process improvement for new-product programs. A native of Barcelona, he received his BS degree in electrical engineering from the Polytechnical University of Catalonia in 1981 and his MS degree in computer science from Stanford University in 1985. Before joining HP in 1988, he worked in new product development for a small company and served as assistant professor of digital systems at the Polytechnical University of Catalonia. He's a member of the ACM. Isidre is married and has a son, and says that in addition to life, friends, and family, he also enjoys reading and bicycling.

42 Media Cutter

Ventura Caamaño Agrafojo

Ventura Caamaño is a manufacturing engineer at HP's Barcelona Peripherals Operation (BPO). He received his BSME degree from the Universidad Metropolitana in Caracas in 1980 and his Master of Engineering degree in mechanical engineering from Stevens Institute of Technology in 1983. He joined HP in 1988, helped build the mechanical infrastructure for the BPO lab, and did feasibility studies of DraftMaster plotter enhancements. He was responsible for the cutter design for the DraftMaster Plus plotter and is named as an inventor in a pending patent on the cutter design. He is an associate member of the ASME. Before joining HP he was involved in the mechanical design of inertial navigation systems with Ceselsa S.A. Ventura was born in Noia, La Coruna, Spain. He is married and enjoys mountaineering, scuba diving, and mountain bicycling.

David Perez

Quality engineer David Perez set the reliability goals and defined the reliability assurance plan for the DraftMaster Plus plotter. A Barcelona native, he joined HP's Barcelona Peripherals Operation in 1989. He has a degree in industrial engineering from the Polytechnical University of Catalunya in Barcelona and is a member of the Collegi Oficial d'Enginyers Industrials de Catalunya. His leisure interests include mountain biking, skiing, scuba diving, tennis, and reading.

Josep Abella

Until recently a metrology engineer in the Barcelona Peripherals Operation quality department, Josep Abella did the metrology engineering for the DraftMaster Plus plotter. Currently a materials engineer specializing in packaging, He joined HP in 1986. He received his degree in industrial engineering in 1985 from Escola Tecnica Superior d'Enginyers Industrials de Terrassa. Before coming to HP he developed fire protection systems with Guardian Iberica, was an industrial engineer with Torredemer, and served a year in the military engineers. He was born in Terrassa, Barcelona, is married, and has a daughter. In addition to spending time with his family, he likes to travel, jog, ski, and play soccer and is a fan of Formula I racing.

49 Plotter User Interface

Jordi Gonzalez

R&D software engineer Jordi Gonzalez joined HP's Barcelona Peripherals Operation in 1988. He has developed HP-GL/2 and pen plotter firmware, most recently for the DraftMaster Plus plotter, and has professional interests in operating systems and distributed processing. He is a member of the IEEE and has a degree in telecommunications engineering (1982) from the Universidad Politécnica de Barcelona. Before coming to HP, he was with Siemens, where he developed computer communication software systems and applications. He is married and is a native of Barcelona.

Jaume Ayats Ardite

Jaume Ayats is a quality engineer with HP's Barcelona Peripherals operation and served in that capacity for the DraftMaster Plus plotter project. He has a degree in industrial engineering from the Polytechnical University of Catalonia (1987) and is interested in quality planning as a tool for project tracking. Before coming to HP in 1989 he worked on automation and process control at the university's Instituto de Cibernetica. Born in Barcelona, he is married and just welcomed his first child.

Carles Castellsague Pique

Carles Castellsague joined HP's Barcelona Peripherals Operation in 1988. An R&D software engineer, he has worked on implementation of the HP-GL/2 graphics language for pen plotters and on the design and development of the DraftMaster Plus user interface. Carles received his BS degree in computer science in 1981 from Barcelona Autonomous University and his MS degree in computer science in 1988 from George Washington University. Before joining HP he was a software developer for the Generalitat Health Department and the World Bank. He's a member of the ACM and is interested in computer graphics. Carles is married and has two daughters. He was born in Granollers, Spain and enjoys mountain biking.

56 Multiprocessor HP-UX

Kyle A. Polychronis

Kyle Polychronis is a project manager at HP's Open Systems Software Division. He joined the Information Networks Division in 1983 as a software engineer, working on the TurboImage profiler and then the HP-UX kernel. He then served as project manager for the HP-UX kernel and the multiprocessor adaptation of HP-UX for release 8.06. He was a consulting project manager for the HP corporate engineering software initiative, and is now project manager for HP-UX open systems I/O. Kyle's professional interests are operating systems, software engineering techniques, and database systems. He received his BS degree in computer science from the University of Utah in 1979 and his MS degree in computer science from the University of California at Berkeley in 1980. Before coming to HP he developed database-oriented UNIX applications for the telecom industry at Bell Laboratories. A native of Salt Lake City, Utah, he is married and enjoys hiking, camping, music, and gourmet food and wine.

Douglas V. Larson

Software development engineer Doug Larson joined HP's Data Systems Division in 1979. He worked on the RTE operating system until 1983, then joined the HP-UX kernel lab, now part of the Open Systems Software Division. For the multiprocessor HP-UX 8.06 operating system, he was responsible for process management and system performance. Born in Minneapolis, Minnesota, he served in the U.S. Navy for six years, attaining the rank of petty officer first class, and attended the University of Minnesota Institute of Technology, receiving a BS degree in computer science in 1978. Before joining HP, he did scientific programming for the University of Minnesota and the U.S. Bureau of Mines. Doug's interests include go (he's a three-dan player), live theater, and ballroom dancing.

62 Demountable TAB

Farid Matta

Since joining HP in 1981, Farid Matta has been involved in various phases of advanced integrated-circuit processing, testing, and packaging. He served as a process engineering manager for NMOS and CMOS IC manufacturing, as a project leader developing the membrane probe test technology, and as a project manager for the development of the DTAB packaging technology. Currently, Farid is a project manager with HP Laboratories, working on the development of new measurement instruments. He is an author or coauthor of over 30 technical publications and is coauthor of a book on electrical contacts and interconnects published in the USSR in 1974. He is named as an inventor in five patents and is a member of the IEEE and the IEEE Components, Hybrids, and Manufacturing Technology Group. Before joining HP, he worked on bipolar IC fabrication at Advanced Micro Devices, and was an assistant professor at the Institute of Electronics in Menouf, Egypt. He received his BSEE degree in 1965 and his PhD degree in microelectronics in 1972 from the Leningrad Institute of Electrical Engineering. Farid was born in El-Menya, Egypt. He is married, has a son, and enjoys painting, art history, and hiking.

78 EISA for HP 9000

Vicente V. Cavanna

Vince Cavanna is an engineer/scientist at HP's Systems Technology Division, specializing in high-speed digital design, synchronization, and electromagnetic compatibility (EMC). He worked on the architecture for the HP EISA cards and consulted on the implementation of the EISA subsystem for the HP 9000 Series 700 workstations. Born in Manila, Philippines, Vince received his BSEECS degree (1974) and MSEECS degree (1976) from the University of California at Berkeley. He joined HP's Automatic Measurements Division in 1976. The projects he has worked on include the HP 2240/50 measurement and control processor, HP Fiber Link, and various backplane interface chips for channel I/O and HP precision bus backplanes. His work has resulted in a patent on an LED driver for a high-speed fiber optic transceiver and a joint patent on a FIFO-based synchronizer that is used in various HP products. Vince is married and has four children. He is a table tennis tournament player, as well as a soccer player and coach, and he enjoys reading, gardening, and woodworking.

Christopher S. Liu

Chris Liu is a hardware development engineer at HP's Roseville Networks Division. He joined HP in 1984 after receiving his BSEE degree from the University of the Pacific that same year. At present he is working on network interfaces for printers and plotters. For the HP EISA project he worked on the hardware development of the HP EISA HP-IB card. Before the EISA project, he worked on HP precision bus development tools and before that he worked as a manufacturing development engineer on HP 9000 I/O interfaces. Before coming to HP, Chris was a co-op student at NASA Ames Research Center where he worked on the control system for a laser-based velocity measurement system. His work at HP has resulted in a pending patent for dynamically configurable interface cards with variable memory size. Born in Lodi, California, he is married, and his interests include bicycling, skiing, softball, and golf. He also enjoys carpentry and reading.

82 EISA I/O Cards

David S. Clark

After receiving a BSEE degree from California State University at Sacramento in 1986, David Clark joined HP's Roseville Networks Division. He is currently a hardware engineer with HP's Systems Technology Division and is working on high-speed networks, as well as completing his MSEE degree at California State University at Sacramento. Since joining HP, he has worked on HP precision bus I/O interfaces and the HP EISA programmable serial interface card. He is listed as an inventor on a pending patent for the HP EISA bus master. Born in Tokyo, Japan, David enjoys gem cutting (faceting), cabinetmaking, running, and learning to play the piano.

Andrea C. Lantz

A graduate of the University of Delaware with a BSEE degree (1985) and the University of California at Davis with an MSEE degree (1988), Andrea Lantz joined HP's Roseville Networks Division in 1988. As a hardware development engineer, she designed hardware and application software for HP precision bus hardware tools that are used in PA-RISC machines. She also worked on the hardware design and development for the HP EISA programmable serial interface card. At present, she is a manufacturing development engineer responsible for HP's JetDirect products. She has published four papers on LAN access protocols. As a career guidance chair for the Sierra Foothills section of the Society of Women Engineers she works at encouraging high school and college students to become engineers. Born in Wilmington, Delaware, Andrea is married and enjoys music, hiking, softball, and trying to play golf. She is also on the adjunct faculty of the National University in Sacramento, California, where she teaches classes in the computer science department.

Christopher S. Liu

Author's biography appears elsewhere in this section.

Thomas E. Parker

Tom Parker came to HP's Computer Support Division as a summer intern in 1981. Starting as a production technician, he later became a production engineer. He received his BSEE degree from California State University at Sacramento in 1983 and joined HP's Roseville Networks Division as a development engineer. He was the hardware design engineer for the HP EISA SCSI card. A native of Sacramento, California, Tom is married and has three children. He enjoys playing golf and soccer and contributes time as a youth soccer coach.

Joseph H. Steinmetz

A graduate of California State University at Chico with a BS degree (1988) in computer engineering, hardware engineer Joe Steinmetz joined HP's Roseville Networks Division in 1988. Joe contributed to the development of the HP EISA SCSI card for the HP 9000 Series 700 workstations. Some of his previous assignments have included developing SCSI I/O cards for HP precision bus and channel I/O SCSI, and backplane ASICs for other HP 9000 I/O interfaces. He is currently listed as a coinventor in a patent application for a coprocessor supporting architecture for a processor that does not natively support coprocessing. Born in Gilroy, California, he enjoys mountaineering, rock climbing, running, and biking when not pursuing his professional interests in ASIC design tools, simulation, and synthesis.

97 HP EISA SCSI Card

Bill Thomas

A member of the technical staff at HP's Systems Technology Division, Bill Thomas worked on the design and implementation of the test software for the HP EISA SCSI card for the HP 9000 Series 700 workstations. After receiving his BSEE degree from the University of California at Berkeley in 1969, he joined HP's Loveland Instrument Division in Loveland, Colorado, and completed his MSEE degree work at Colorado State University in 1972. During his career with HP, he has worked on software and hardware design for IC test systems, test system software for the HP 1000 Automatic Test System, thick-film substrates, IC design for the HP 9866A thermal printer, design and implementation of the HP-IB channel I/O interface, and diagnostic and self-test software for the SCSI channel I/O interface. He

is a member of the IEEE. His professional interests include software, hardware, and firmware interfaces between host systems and peripherals, as well as software reuse and object-oriented programming. Bill is married and has three children. He is an active member of two amateur radio emergency groups in Sacramento, California, and is a judge for California applicants seeking national science scholarship grants. His hobbies include cabinetmaking and camping.

Alan C. Berkema

Software development engineer Alan Berkema attended California State University at Sacramento where he received his BS degree in computer science and systems software in 1984. The next year he joined HP's Roseville Networks Division. Currently, Alan is working on a high-speed mass storage I/O adapter. For the HP EISA SCSI project, he helped design and edit the interface specification and all test software for the interface driver. Before the EISA SCSI project, he worked on the SCSI channel I/O host adapter firmware. Born in Djakarta, Indonesia, Alan served four years in the United States Air Force as a computer technician, attaining the rank of sergeant. He is married, has one daughter and enjoys jogging, fishing, and camping.

Eric G. Tausheck

Eric Tausheck is a development engineer at HP's Systems Technology Division. He joined HP's Roseville Networks Division in 1980 after receiving a BS degree in information and computer science from the University of California at Irvine that same year. Presently, he is working on high-speed adapter design and the associated interface driver. He designed and implemented the hardware-specific portion of an interface driver for the HP EISA SCSI card. Other projects he has worked on include firmware implementation for the HP-CIO (channel I/O) HP-IB and HP-FL adapters, the HP precision bus and interface driver designs, and specifications for HP designs based on the NCR 53C700 chip family. His main professional interest is high-performance I/O system design. Born in San Lorenzo, California, Eric enjoys time with his wife and two children.

Brian D. Mahaffy

Brian Mahaffy, a development engineer at HP's Roseville Networks Division, joined HP in 1980. He received his AS degree in electronics technology from American River College in 1981 and later received his BS degree in computer engineering from California State University at Sacramento in 1989. He worked on implementing the I/O dependent code for the HP EISA SCSI project. Currently, he is working on hard-copy interconnect network interfaces. Past projects include SCSI channel I/O, HP 3000 CPUs, and I/O interfaces for HP 1000 computer systems. A native of Sacramento, California, Brian is an active member of the Sacramento County Radio Amateur Civil Emergency Services (RACES), which provides service during natural disasters and other emergencies as part of the U.S. Civil Defense network. Brian is married and has one child (another is expected soon). His hobbies include ham radio and radio-controlled aircraft.

Michael E. Thompson

Mike Thompson is a productivity and quality manager at HP's Worldwide Customer Support Operation (WCSO) and has served as a project manager for various software support applications developed at WCSO. He presently manages technical architecture implementors and quality engineers in the worldwide support systems group in WCSO. A native of Las Cruces, New Mexico, Mike received his BBA degree in management information systems (1983) and his MBA degree (1985) from New Mexico State University. Before coming to HP in 1985, he was a consultant for Arthur Andersen & Co.

Gregson P. Siu

Gregson Siu is a technology quality section manager at HP's Worldwide Customer Support Operation. He joined HP in 1982. He was the project manager for the server portion of the customer order tracking system described in the article. In the past he has worked as a development engineer for HP's order management system, a project manager for a field resource management system, and a productivity manager for the worldwide support services group in WCSO. He was born in Honolulu, Hawaii, and received his BBA degree in management information systems from the University of Hawaii in 1982.

Jonathan van den Berg

Jon van den Berg is an information technology architect in HP's Worldwide Customer Support Operation (WCSO). He served as a technology architecture manager for the client/server projects described in his article. After receiving a BBA in management information systems from California Polytechnic Institute at San Luis Obispo in 1984, Jon came to HP's Computer Support Division that same year as a software engineer. He has worked as a development engineer, project manager, and technical architect for various projects at WCSO. Before coming to HP, Jon was an MIS analyst for IBM Corporation. Born in Annapolis, Maryland, he is married and has one child. His professional interests are object-oriented analysis and design and integrated development environments. His hobbies and interests include U.S. amateur soccer, British sports cars, and California open-space parks.

HEWLETT-PACKARD
JOURNAL
INDEX

Volume 43 January 1992 through December 1992

Hewlett-Packard Company, P.O. Box 51827, Palo Alto, California 94303-0724 U.S.A.
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan

Part 1: Chronological Index

Deferred Binding, Relocation, and Initialization of Shared Library Data

Integrating an Electronic Dictionary into a Natural Language Processing System, *Diana C. Roberts*

Application of Spatial Frequency Methods to Evaluation of Printed Images, *Dale D. Russell*

Parallel Raytraced Image Generation, *Susan S. Spach and Ronald W. Pulleyblank*

August 1992

Midrange PA-RISC Workstations with Price/Performance Leadership, *Andrew J. DeBaets and Kathleen M. Wheeler*

HP 9000 Series 700 Workstation Firmware

VLSI Circuits for Low-End and Midrange PA-RISC Computers, *Craig A. Gleason, Leith Johnson, Steven T. Mangelsdorf, Thomas O. Meyer, and Mark A. Forsyth*

PA-RISC Performance Modeling and Simulation

ECL Clocks for High-Performance RISC Workstations, *Frank J. Lettang*

HP 9000 Series 700 Input/Output Subsystem, *Daniel Li and Audrey B. Gore*

Design Verification of the HP 9000 Series 700 PA-RISC Workstations, *Ali M. Ahi, Gregory D. Burroughs, Audrey B. Gore, Steve W. LaMar, Chi-Yen R. Lin, and Alan L. Wiemann*

HP Standard PA-RISC Test Programs

Simulation Toolset

Debugging Tools

Metrics

Mechanical Design of the HP 9000 Models 720 and 730 Workstations, *Arlen L. Roesner and John P. Hoppal*

Meeting Manufacturing Challenges for PA-RISC Workstations, *Spencer M. Ure, Kevin W. Allen, Anna M. Hargis, Samuel K. Hammel, and Paul Roeber*

High-Performance Designs for the Low-Cost PA-RISC Desktop, *Craig R. Frink, Robert J. Hammond, John A. Dykstal, and Don C. Soltis, Jr.*

Low-Cost Plain-Paper Color Inkjet Printing, *Daniel A. Kearl and Michael S. Ard*

Thermal Inkjet Review, or How Do Dots Get from the Pen to the Page?

Ink and Print Cartridge Development for the HP DeskJet 500C/DeskWriter C Printer Family, *Craig Maze, Loren E. Johnson, Daniel A. Kearl, and James P. Shields*

Color Science in Three-Color Inkjet Print Cartridge Development

Making HP Print Cartridges Safe for Consumers Around the World

Automated Assembly of the HP DeskJet 500C/DeskWriter C Color Print Cartridge, *Lee S. Mason and Mark C. Huth*

Color Inkjet Print Cartridge Ink Manifold Design

Adhesive Material and Equipment Selection for the HP DeskJet 500C/DeskWriter C Color Print Cartridge, *Douglas J. Reed and Terry M. Lambright*

Machine Vision in Color Print Cartridge Production, *Michael J. Monroe*

HP DeskWriter C Printer Driver Development, *William J. Allen, Toni D. Courville, and Steven O. Miller*

An Interactive User Interface for Material Requirements Planning, *Alvina Y. Nishimoto, William J. Gray, and Barbara J. Williams*

HP MRP Action Manager Project Management

October 1992

The HP Network Advisor: A Portable Test Tool for Protocol Analysis, *Edmund G. Moore*

Network Advisor Product Enhancement Philosophy

Embedding Artificial Intelligence in a LAN Test Instrument, *Scott Godlew, Rod Unverrich, and Stephen Witt*

The User Interface for the HP 4980 Network Advisor Protocol Analyzer, *Thomas A. Doumas*

Object-Oriented Design and Smalltalk

The Forth Interpreter

The Network Advisor Analysis and Real-Time Environment, *Sunil Bhat*

Network Advisor Protocol Analysis: Decodes, *Rona J. Prufer*

Mechanical Design of the HP 4980 Network Advisor, *Kenneth R. Krebs*

The Microwave Transition Analyzer: A New Instrument Architecture for Component and Signal Analysis, *David J. Ballo and John A. Wendler*

Frequency Translation as Convolution

Design Considerations in the Microwave Transition Analyzer, *Michael Dethlefsen and John A. Wendler*

A Visual Engineering Environment for Test Software Development, *Douglas C. Beethe and William L. Hunt*

Object-Oriented Programming in a Large System

Developing an Advanced User Interface for HP VEE, *William L. Hunt*

HP VEE: A Dataflow Architecture, *Douglas C. Beethe*

A Performance Monitoring System for Digital Telecommunications Networks, *Giovanni Nieddu, Fernando M. Secco, and Alberto Vallerini*

G-Link: A Chipset for Gigabit-Rate Data Communication, *Chu-Sun Yen, Richard C. Walker, Patrick T. Petruno, Cheryl Stout, Benny W.H. Lai, and William J. McFarland*

Bang-Bang Loop Analysis

December 1992

A Large-Format Thermal Inkjet Drafting Plotter, *Robert A. Boeller, Samuel A. Stodder, John F. Meyer, and Victor T. Escobedo*

DesignJet Plotter User Interface Design: Learning the Hard Way about Human Interaction

Electronic and Firmware Design of the HP DesignJet Drafting Plotter, *Alfred Holt Mebane IV, James R. Schmedake, Iue-Shuenn Chen, and Anne P. Kadonaga*

Pen Alignment in a Two-Pen, Large-Format, Inkjet Drafting Plotter, *Robert D. Haselby*

DesignJet Plotter Chassis Design: A Concurrent Engineering Challenge, *Timothy A. Longust*

DesignJet Plotter End Covers Produced by Coinjection

DesignJet Plotter Mechanical Architecture Development Process, *David M. Petersen and Chuong Ta*

Improved Drawing Reliability for Drafting Plotters, *Robert W. Beauchamp, Josep Giralt Adroher, Joan Uroz, and Isidre Rosello*

Average User Plot

Acceptable Quality Level Index

An Automatic Media Cutter for a Drafting Plotter, *Ventura Caamaño Agrafojo, David Perez, and Josep Abella*

Definitions and Measurement Procedures for Cut Quality Parameters

Reengineering of a User Interface for a Drafting Plotter, *Jordi Gonzalez, Jaume Ayats Ardite, and Carles Castellsague Pique*

A Multiprocessor HP-UX Operating System for HP 9000 Computers, *Douglas V. Larson and Kyle A. Polychronis*

Next-Generation Multiprocessor HP-UX

Advances in Integrated Circuit Packaging: Demountable TAB, *Farid Matta*

The EISA Standard for the HP 9000 Series 700 Workstations, *Vicente V. Cavanna and Christopher S. Liu*

EISA Cards for the HP 9000 Series 700 Workstations, *David S. Clark, Andrea C. Lantz, Christopher S. Liu, Thomas E. Parker, and Joseph H. Steinmetz*

Board-Level Simulation of the Series 700 EISA Cards

Software for the HP EISA SCSI Card, *Bill Thomas, Alan C. Berkema, Eric G. Tausheck, and Brian D. Mahaffy*

Update on the SCSI Standard

Adapting the NCR 53C710 to Minimize Interrupt Impact on Performance

An Architecture for Migrating to an Open Systems Solution, *Michael E. Thompson, Gregson P. Siu, and Jonathan van den Berg*

Part 2: Subject Index

G

H

I

K

L

M

N

O

P

Q

R

Part 3: Product Index

Part 4: Author Index

HEWLETT-PACKARD
JOURNAL

December 1992 Volume 43 • Number 6

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Company, P.O. Box 51827
Palo Alto, California, 94303-0724 U.S.A.

Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan

HEWLETT PACKARD

5091-6002E