ports. This allows DSP boot code (not to be confused with DSP application code) to be downloaded from the host system. This provides additional flexibility and eliminates the cost and board-space limitations associated with external ROMs.

XBAR has several asynchronous control signals that are connected to downstream hardware, including reset lines for the DSP subsystems and hook control and telephone status (such as the ring indicator) signals to and from the data access arrangement chips.

The biggest challenge in XBAR's design was purely logistical because we had a lot of different data types (e.g., stereo data and telephone command and status data) to handle and very little time to implement them in XBAR. There are no less than 52 separate data types that XBAR must recognize and generate for the two DSP serial interfaces alone, with a slightly smaller number required for the system I/O interface. To provide these types, each transaction between XBAR and a DSP consists of 16 bits, with the upper eight bits providing type information and the lower eight bits providing the associated data. To prevent data overruns, XBAR requires a data acknowledge (Ack) word back from the appropriate DSP for every transaction.

Audio data samples in HP TeleShare are 16 bits long. Since XBAR sends eight bits of data at a time, audio samples must be broken into two pieces: an upper half, or most-significant byte (MSB) and a lower half, or least-significant byte (LSB). Using this model, it requires two transfers from XBAR and two Acks back to send one sample of audio data to a DSP. Sending one set of stereo system and line-in audio samples to a DSP requires eight output transfers (four transfers for the system sample and four transfers for the line-in sample), with an Ack back after each transfer. The DSP will then send mixed audio samples back for the system and the CODEC, requiring an additional eight transfers, for a total of 24 transfers per sample. This has to happen at an 8-kHz sample rate (once every 125 microseconds). Fortunately, XBAR can handle these transactions, but order must be maintained exactly or audio quality will suffer. Other data types, such as AT commands and responses, are given lower priority during audio frames and are queued until audio transfer is finished.

Digital Signal Processor. The DSP used by TeleShare is an Analog Devices ADSP2101. This is a programmable single-chip microcomputer optimized for digital signal processing, and operates at 16.67 MHz. The 2101 operates on 16-bit data and uses a 24-bit instruction word. It has 1024 words of data RAM and 2048 words of program RAM on the chip. The part has two data address generators and a program sequencer, which allows program and data accesses to occur simultaneously in a single cycle. Dual data operand fetches can also occur in a single cycle since program memory can also be used to store data. The part can address up to 16K words of data and 16K words of program memory, both of which are supplied on the HP TeleShare board in the form of six external SRAMs.

The DSP has two independent serial ports, SPORT0 and SPORT1, which support multiple data formats and frame rates and are fully programmable. In the HP TeleShare design, SPORT1 on each DSP is dedicated to communication with XBAR, while SPORT0 is dedicated to communication with an AD28msp01 telephone line CODEC. Each full transfer to or from one of the SPORT lines triggers an associated interrupt in the 2101, allowing programs to act on the incoming data as it arrives.

Analog Devices supplies a complete set of software development tools for the 2100 microprocessor family, including a C compiler with a DSP function library.

CODEC. The Analog Devices AD28msp01 provides HP TeleShare with a multiple-sample-rate CODEC specifically designed for use in modem designs. This device supports sample rates of 7.2 kHz, 8.0 kHz, and 9.6 kHz, and has an 8/7 mode* for sampling at 8.23 kHz, 9.14 kHz and 10.97 kHz. For voice mode operation, 7.2 kHz and 8.0 kHz are all that is required, but the sophisticated algorithms used by modern modem standards often require the other rates. The CODEC uses 16-bit sigma-delta conversion technology and includes resampling and interpolation filtering along with transmit and receive phase adjustments.

Each CODEC has one serial port which is connected directly to SPORT0 on the associated DSP. This port operates in free-running mode once it is properly initialized and continually sends 16-bit data samples from the telephone line to the DSP. All transfers to the DSP consist of a serial output frame sync followed by a 16-bit address word, then a second frame sync followed by a 16-bit data word. These address and data pairs are transmitted at the selected sample rate and trigger SPORT0 receive interrupts in the DSP. The DSP transfers data to the CODEC using the same mechanism as just described (in the other direction, of course). The address portion of each transfer coming from the DSP identifies the data as a control word (for programming the part) or as a data word to be sent through the on-chip digital-to-analog converter (DAC) and transmitted to the data access arrangement chip. Data from the CODEC to the DSP is identified as either a control word or as a data word from the on-chip analog-to-digital converter (ADC).

The AD28msp01 CODEC is attached to the telephone line through the data access arrangement chip. Transmit data outputs are differential for noise reduction, while the receive data input is single-ended.

Data Access Arrangement. HP TeleShare uses the TDK 73M9002 data access arrangement chip as its telephone line interface. This part provides all the necessary line monitoring, filtering, isolation, protection, and signal conversion functions for connection of high-performance analog modem designs to the PSTN (public switched telephone network) in the United States, Canada, and Japan. The 73M9002 incorporates, on a two-to-four-wire hybrid, ring detection circuitry, off-hook relay, and on-hook line monitoring for caller-ID support (see "Caller-ID" on page 72).

The 73M9002 comes with FCC (Federal Communications Commission) part 68 DOC CS-03 and JATE (Japan Approvals Institute for Telecommunications Equipment) protection circuitry built in, and is compliant with UL 1459 2nd Edition

* The 8/7 mode is a capability required by some modem applications. It simply adds some sampling bandwidth. For example, in 8/7 mode the normal 8-kHz sample rate becomes 9.14 kHz (8 × 8/7).

© Copr. 1949-1998 Hewlett-Packard Co.

Caller-ID

Caller-ID information is sent between the first and second power ringing signals. The data is sent a minimum of 500 milliseconds after the first ring and ends at least 200 milliseconds before the second ring begins. This leaves 2.9 to 3.7 seconds of time for data transmission. The data is sent at 1200 baud using frequency shift keying (FSK) modulation. All data is 8-bit ASCII.

Two standard formats exist for Caller-ID information: single message format and multiple message format. In general, both formats can be described using Fig. 1.

The message type is 0x4 (hexadecimal 4) for single message format. The message length is variable and indicates the number of message words in the message body. The final word is a checksum word, used for error checking. Single message format provides the receiver with date, time, and calling number data.

The message type is 0x80 (hexadecimal 80) for multiple message format. The message length is variable as before, but provides the receiver with date, time, calling number, and calling name data if available. In the absence of calling name data, a P indicating private or an O indicating out of area or unavailable will be sent.

Caller-ID detection requires on-hook line monitoring, which the HP TeleShare data access arrangement chip fully supports. HP TeleShare can detect and display both message formats.

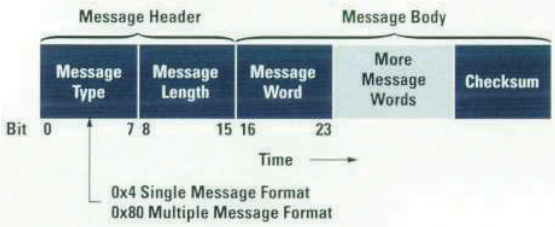


Fig. 1. Caller-ID message format.

with the addition of an external slow-blow fuse. The off-hook relay is controlled by a TTL-level input from XBAR. This relay determines when the phone is off the hook and can be pulsed for use as a pulse dialer. Another TTL-level input from XBAR is used to enable on-hook line monitoring.

The ring detection circuitry is capable of detecting ringing signals that comply with Ringing Type B from the FCC Part 68 regulations. The detected ring signal appears at a pair of differential outputs which are also connected to XBAR.

The 73M9002 provides telephone connectivity to the DSP subsystem through the CODEC's analog receive and transmit lines and is attached to the telephone line through a standard RJ-14 connector.

Operating Modes

HP TeleShare is capable of operating in three modes: voice, fax, and data. The modes are selected through a graphical user interface by the workstation user. The mode application software is downloaded through XBAR to a DSP as needed and runs continually until a reset of that DSP is performed. The voice mode code was developed solely by HP and can be run on one or both lines simultaneously. The fax and data modem code was developed with a third party and because of licensing restrictions, only one line can be configured as a fax or data modem at a time. Combinations of voice and fax or data are fully supported.

Voice Mode Operation. When configured in the voice mode, HP TeleShare essentially operates like an enhanced telephone. Digital mixing of microphone, line-in, telephone, and recorded audio (from system disk) is supported for both playback and recording. This capability allows numerous interesting audio configurations including placing a line on hold with music, recording conversations, playing back recorded audio over the phone, and so on. While in voice mode, HP TeleShare provides the user with caller-ID information if it is available. In addition, DTMF (dual-tone multi-frequency) tone and pulse dialing are supported, along with DTMF tone detection for unattended phone functions like answering machines or voicemail (see "Call Progress, DTMF Tones, and Tone Detection" on page 73).

Dialing and hook manipulation actions are performed through the GUI (graphical user interface), but at the lowest level these actions are sent to the DSP as standard AT commands like ATDT (attention dial tone) and ATH n (n = 0 is on-hook or hang up, and n = 1 is take telephone off hook). Special functions like audio mixing are also controlled with low-level AT-type commands, but are manipulated using sliders in the GUI.

The voice mode application firmware is driven primarily by DSP SPORT interrupts. Every incoming 16-bit SPORT0 word from XBAR triggers an interrupt, which in turn causes the SPORT1 interrupt service routine to execute. Likewise, every 16-bit SPORT0 word from the CODEC causes the SPORT0 interrupt service routine to execute. The SPORT1 interrupt service routine is responsible for audio I/O with XBAR and queueing AT commands as they arrive. Commands arrive asynchronously, that is, they can arrive at any time, while audio arrives in 8-piece bundles every 125 microseconds (one frame) as described earlier. Normally, every piece of data received by SPORT1 causes an interrupt, but the firmware disables these interrupts for the rest of a frame once it recognizes the first piece of audio data. Otherwise, at least eight context switches would occur every frame, which would render the system useless. Once the SPORT1 interrupt service routine has received all of the audio samples, it is responsible for transmitting the new audio back to XBAR for routing to the workstation (i.e., headphones and/or disk).

The SPORT0 interrupt service routine is responsible for receiving and transmitting telephone-line audio and mixing all audio data, including DTMF tones. Before mixing can occur in the DSP, all of the LSBs must be appended to the MSBs. Remember that each 16-bit sample transferred between XBAR and the DSP is divided so that the most-significant byte contains the data type and the least-significant byte contains the data. Thus, all the data from XBAR is put back into 16-bit linear format before transfer to the CODEC.

The audio input and output amplitude matrices, built by the user via the GUI, are used to determine what the final mix will sound like. The DSP firmware processes each output in sequence by adding together any inputs that are on to create a total value for each output. Any gain adjustments are made at this time as well. When this is completed for all outputs, the resulting 16-bit values are broken into MSBs and LSBs, if required.

Audio data that is meant for XBAR is transmitted during the next XBAR audio frame. Audio data meant for output to the

© Copr. 1949-1998 Hewlett-Packard Co.

Call Progress, DTMF Tones, and Tone Detection

HP TeleShare's voice mode firmware has the ability to detect a number of tones used commonly in telephone communications, including DTMF tones and call progress tones like busy, ringback (the ringing sound you hear when you call someone), and dial tone.

DTMF Tones

Dual-tone multifrequency (DTMF) tones are made up of two separate tones, as the name suggests, and can be accurately generated using easily understood principles. The DTMF standard specifies two sets of distinct tones, called row frequencies and column frequencies (see Fig. 1). The row frequencies correspond to the horizontal rows on a standard telephone touchpad. The column frequencies correspond to the vertical columns on the touchpad, plus an additional column to the right of the last touchpad column.

This makes eight separate frequencies, which combine for a total of sixteen DTMF tones (see Fig. 2).

Generation of a DTMF tone is accomplished by creating a sinusoid for each of the two frequencies, row and column, and then adding the results. In a digital implementation, the sinusoids are computed and added on a sample-by-sample basis. HP TeleShare uses a five-coefficient Taylor series approximation for the sinusoid generation. The sinusoid samples are updated and added at 8 kHz, or every 125 microseconds, and the sum of the sinusoid samples is used as the current DTMF sample.

Tone Detection

Tone detection is accomplished through the use of a 512-point fast Fourier transform (FFT), which is implemented in the ADSP2101 C-language run-time library. The FFT, when given a set of samples of an input signal over some time interval, returns the frequency spectrum of the signal during the interval. This can be done in almost real time with a DSP, making it very useful for detecting incoming tones. The following important rules and relationships should be noted concerning sample rate, input points, output points, time, frequency, and the FFT in general:

- The FFT requires complex (real and imaginary) data for input (two arrays).
- The imaginary input array may be filled with zeros if unused.
- The output data is complex (two arrays).
- The frequency spectrum returned covers half of the sampling frequency.
- Only the first half of output data is used, and the other half is a mirror image.
- The output frequency resolution is equal to (sampling rate)/(number of input points).

Using an 8-kHz sampling rate and 512 points causes the FFT to return a spectrum from 0 to 4 kHz, with 512 complex output points. The second 256 output points can be ignored since they are the mirror image of the first 256. The output will have a resolution of 15.625 Hz per point, using the formula above. These output points will be referred to as bins since they include spectral data on either side of each point.

HP TeleShare calculates magnitude-squared values for each bin by squaring the real and imaginary values at each point and adding them. The magnitude-squared values correspond roughly to the power of the signal in each bin. Once the powers are known for each bin in the spectrum, they can be analyzed to see if any DTMF or call progress tones are present.

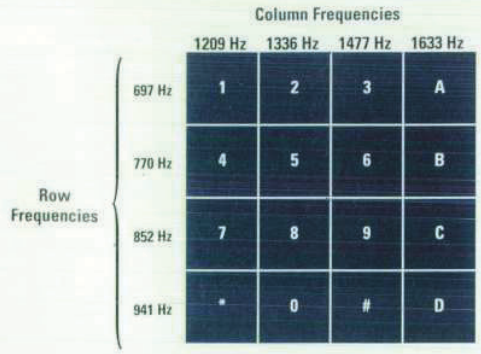


Fig. 1. Dual-tone-multifrequency digits and the frequencies associated with them.

	Tone	Frequency (Hz)	512-Point FFT Index
Call Progress Tones	Dial Tone	350 + 440	22.4 + 28.16
	Busy	480 + 620	30.72 + 39.68
	Ringback	440 + 480	28.16 + 30.72
16 DTMF Tones	DTMF 1	697 + 1209	44.61 + 77.38
	DTMF 2	697 + 1336	44.61 + 85.5
	DTMF 3	697 + 1477	44.61 + 94.53
	DTMF 4	770 + 1209	49.28 + 77.38
	DTMF 5	770 + 1336	49.28 + 85.5
	DTMF 6	770 + 1477	49.28 + 94.53
	DTMF 7	852 + 1209	54.53 + 77.38
	DTMF 8	852 + 1336	54.53 + 85.5
	DTMF 9	852 + 1477	54.53 + 94.53
	DTMF *	941 + 1209	60.22 + 77.38
	DTMF 0	941 + 1336	60.22 + 85.5
	DTMF #	941 + 1477	60.22 + 94.53
	DTMF A	697 + 1633	44.61 + 104.51
	DTMF B	770 + 1633	49.28 + 104.51
	DTMF C	852 + 1633	54.53 + 104.51
	DTMF D	941 + 1633	60.22 + 104.51

Fig. 2. Call progress and DTMF tones recognized by HP TeleShare.

As an example, suppose the telephone has been taken off-hook in preparation for dialing and HP TeleShare is configured to check for dial tone. 512 samples of the input signal would be stored in the real input array, while the imaginary array is filled with zeros. Next, the FFT function is called, returning the real and imaginary arrays. The magnitude-squared values of the first 256 bins are computed using the two output arrays. The two frequencies that make up a dial tone are 350 and 440 Hz (see Fig. 2), so FFT indexes (or bin numbers) must be computed for these frequencies:

$$350/15.625 = 22.4 \qquad 440/15.625 = 28.16$$

An effective method of checking for the existence of a particular frequency is to compare the power present at that frequency with the total power of the spectrum. This is done quite easily with magnitude-squared values since they represent power in each bin already. Total power is simply the sum of all the magnitude-squared values for the first 256 FFT return values. Divide this into the power of the frequency being checked for, and the result is the percentage of total power for that frequency. For example, when checking for 350 Hz, compute the sum of the power values for bins 22 and 23 since the real index (22.4) falls between them, and then divide by the total power. The result is the percentage of the total power present around 350 Hz. The same can be done for 440 Hz, using bins 28 and 29.

Once the percentage of total power is calculated, a comparison can be made to see if the power in each frequency meets *match criteria*. The HP TeleShare firmware typically uses 35% of total power as a match condition. In other words, if the power present at the desired frequencies is 35% or more of the total power, dial tone has been detected. Otherwise, no dial tone is found.

The number of bins used in the comparison and the match criteria can be fine-tuned for a particular application. The match criteria can include other tests and can be relaxed or tightened as needed. The number of bins used can be influenced by the total number of points in the FFT and by a preprocessing tool that does windowing. Windowing is used to create a finite-length sequence from a continuous sequence. It is basically a digital filter that truncates an infinite-length input sequence while preserving its frequency characteristics. Since we are grabbing finite pieces (sequences) of data, we need to window the data.

© Copr. 1949-1998 Hewlett-Packard Co.

telephone line is immediately sent to the CODEC without being split in half. Since both interrupt service routines run at 8 kHz, there is no need to worry about sample rate changes. DTMF audio data is only available for mixing when a tone is being generated. A new DTMF sample is generated during every SPORT0 interrupt and is based on the sample rate (always 8 kHz) and the time elapsed since the tone began.

All of these interrupts and audio manipulations require almost all of a DSP's processing bandwidth and can effect some areas of system performance. Because of DSP bandwidth limitations, DTMF detection can have a slight, but noticeable effect on the audio quality heard by the user. However, in unattended modes like answering machines or voicemail (where DTMF detection could be used for such things as navigation), this should not be a concern. The default configuration has DTMF detection disabled, since the typical user will never use it, and the current GUI does not support it.

Fax and Data Modem Operation. The fax and data modem functionality was codeveloped by HP and Digicom Systems Incorporated and uses their SoftModem technology. The fax mode allows transfers up to 14,400 bits/s and covers Group 3 Class II and all fallbacks. Data mode supports transfers up to 14,400 bits/s (V.32bis) and can reach peak rates of 57,600 bits/s with compression.

Conclusion

HP TeleShare effectively combines telephone communications capability with a low-cost computer workstation. Context switches between the display and the telephone are minimized by integrating the telephone into the computer system and providing an easy-to-use graphical user interface. Voice, fax, and high-speed data modes are supported using flexible digital signal processing technology.

References

1. D. Garfinkel, B. Welti, and T. Yip, "HP SharedX: A Tool for Real-Time Collaboration," *Hewlett-Packard Journal*, Vol. 45, no. 2, April 1994, pp. 23-36.

© Copr. 1949-1998 Hewlett-Packard Co.

Product Design of the Model 712 Workstation and External Peripherals

A product design without fasteners and the use of environmentally friendly materials and low-cost parts with integrated functions provides excellent manufacturability, customer ease of use, and product stewardship.

by Arlen L. Roesner

The HP 9000 Model 712 workstation and and the three new peripherals that go with the product are an excellent example of computer integration and simplicity. The new workstation, while providing a new class of performance with HP's new PA-RISC PA 7100LC processor, pushed the envelope of product design by using relatively few and inexpensive parts. In addition to simplicity and low cost, the product promotes good product stewardship by making parts easy to identify and recycle. Customers find the hardware easy to manage because there are no fasteners to deal with, and all the components snap or drop into place. The main workstation product is a small compact size that fits easily under a monitor or stands vertically on the desk, and the external peripherals can be positioned on the desktop where they are most convenient to the user. Fig. 1 shows the Model 712 workstation and its three peripherals.

Outward Simplicity

Several assemblies of the Model 712 workstation products have high levels of functional integration. This functional integration tends to make components more complex, but yields an outer simplicity by reducing the number of physical parts and the methods necessary to work with them. Once configured, the only accessible components of the Model 712 workstation include the chassis, system board, option boards (including memory), disk drive, flexible disk drive, and top cover. All of these components are accessed through quick removal of the cover and the manipulation of a few snap or drop-in fits, which require a minimum of time and effort. Fig. 2 shows the workstation and one of the peripherals with their covers removed. Benefits of this resulting simplicity include better manufacturability, easier customer use and configuration, and serviceability.

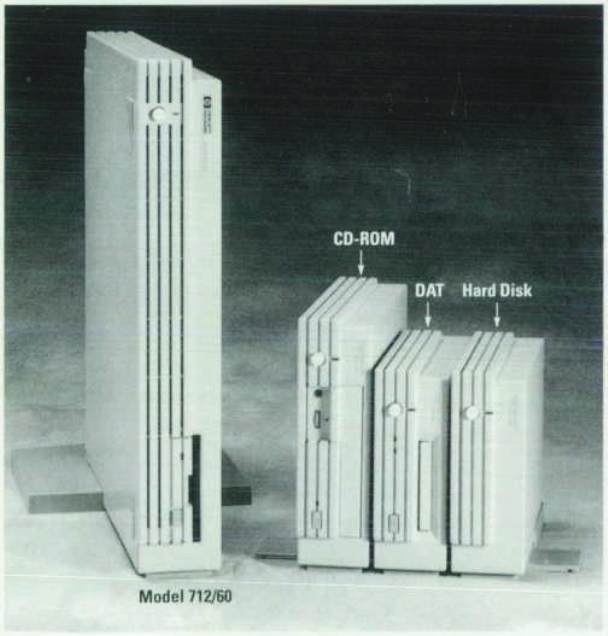


Fig. 1. HP 9000 Model 712 workstation and related external peripherals.



Fig. 2. The Model 712 workstation and hard disk peripheral with top covers disassembled.

© Copr. 1949-1998 Hewlett-Packard Co.

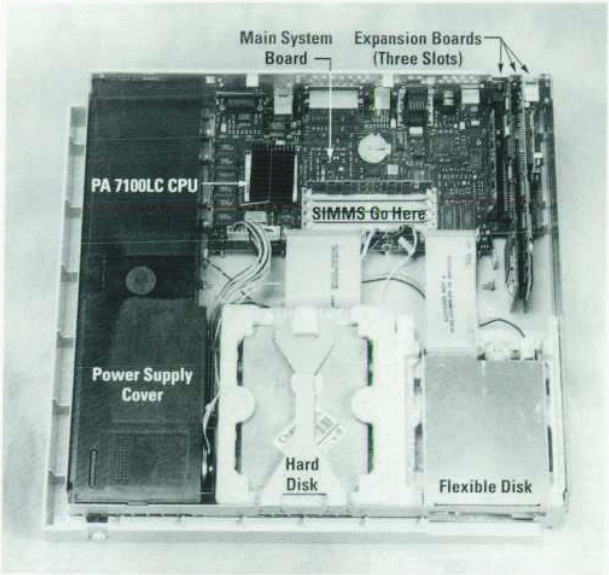


Fig. 3. Top view of the Model 712 without top cover.

Electronics

The system electronics is the place where integration is most likely to be first noticed in the Model 712 product. Electronic assemblies consist of one main system board, a power supply, three optional circuit boards, and up to four memory SIMMs. The main system board is relatively small, and all of the core electronics is incorporated onto this board through integration of functionality into relatively few VLSI components. (Fig. 3 in the article on page 9 shows the main system board). The main system board uses dual-sided surface mount construction, with I/O connector space being provided mostly by double-high (stacked) bulkhead connectors. Optional boards are provided for telephony, extra I/O, and high-resolution graphics. Compared to today's personal computers, the Model 712's system board functions are usually found on a personal computer's motherboard, backplane (if any), and two to three expansion boards. This level of integration on the Model 712 exceeds the density of personal computer functionality, while providing current workstation performance.

Chassis

The chassis assembly consists of a plastic base, a metal chassis, a metal liner for EMI containment of the rear I/O connectors, and a plastic rear dress panel (see Fig. 4). The dress panel includes silkscreened graphics to identify the connectors and state necessary regulatory information, eliminating the need for information labels. The chassis has a variety of holes and embossments to assist in joining the plastic parts to it. The plastic base provides outer air venting and cosmetic appeal to the product while also containing several snaps and guides for mating parts. The metal liner provides EMI finger contact to all connectors in one part, whereas previous products often required many different clips for such functionality. Held together via plastic heat stakes, the plastic base, the metal chassis, the metal liner, and the plastic dress panel make up the main assembly chamber of the product. The main circuit board, power supply and cover, disk brackets, and top cover all snap or drop into this chassis. Option boards are also easily installed into the chassis on top of the main system board, with integral bulkheads that mate vertically to chassis cutouts (also without fasteners).

Power Supply Cover

The power supply cover is another example of integration. Many parts were "designed out" by this single plastic part that performs six functions. The main function is to protect end users from dangerous voltages by shrouding the exposed power supply. The cover snaps into the chassis from front to rear and is removable only by using a screwdriver to disengage the snap that holds it in place. In addition to shrouding the power supply, the cover secures the power supply board in place, houses the fan and speaker, channels air flow, and provides structural support for the monitor. The fan simply snaps down inside the cover and seals to the sides and top of the cover. The speaker slides down and press fits into a simple pocket, which provides acoustic baffling. After the cover is installed, cables from these devices are routed to the main system board for electrical connection.

HP-PAC Disk Brackets

The disk brackets are made of HP's newly patented HP-PAC material.[1] This material is made of expanded polypropylene beads, and is used most often to produce shipping carton cushions for many types of products. Instead of placing this material around a finished product to cushion it in a shipping carton environment, it is instead formed to fit inside a product with integral recesses to embed internal components. For the Model 712 workstation, the HP-PAC material is used to hold the hard disk and flexible disk mechanisms in place. The HP-PAC used in the workstation consists of three parts: a bottom shell which provides a recess for both flexible and hard disk, and two separate top pieces for covering each disk mechanism (see bottom portion of Fig. 4). Because of the cushioning properties of the HP-PAC material, the disk drive mechanisms benefit from reduced shock and vibration levels. The HP-PAC material also provides integral air channels for inlet air to be drawn across hot areas of the disk drive mechanisms. The interesting feature of HP-PAC is that no screws are needed to install the mechanisms. The devices simply drop into recesses inside of the cushioning material, and cables can be connected directly to the embedded mechanisms. Once in place, the chassis enclosure then retains the top and bottom shells of HP-PAC around each device.

Top Cover

The top cover includes a configurable bezel for the flexible disk area, a plastic top shell, and a thin metal liner to complete the EMI enclosure. The liner is held to the cover via plastic heat stakes and has a series of fingers on each side of the cover to contact the chassis and contain EMI radiation. The flexible disk bezel is designed to snap into the front of the cover, which then configures the frontal appearance of the product. The cover assembly drops vertically onto the chassis and then slides rearward until alignment hooks and snaps in the cover engage to hold the cover in place.

© Copr. 1949-1998 Hewlett-Packard Co.

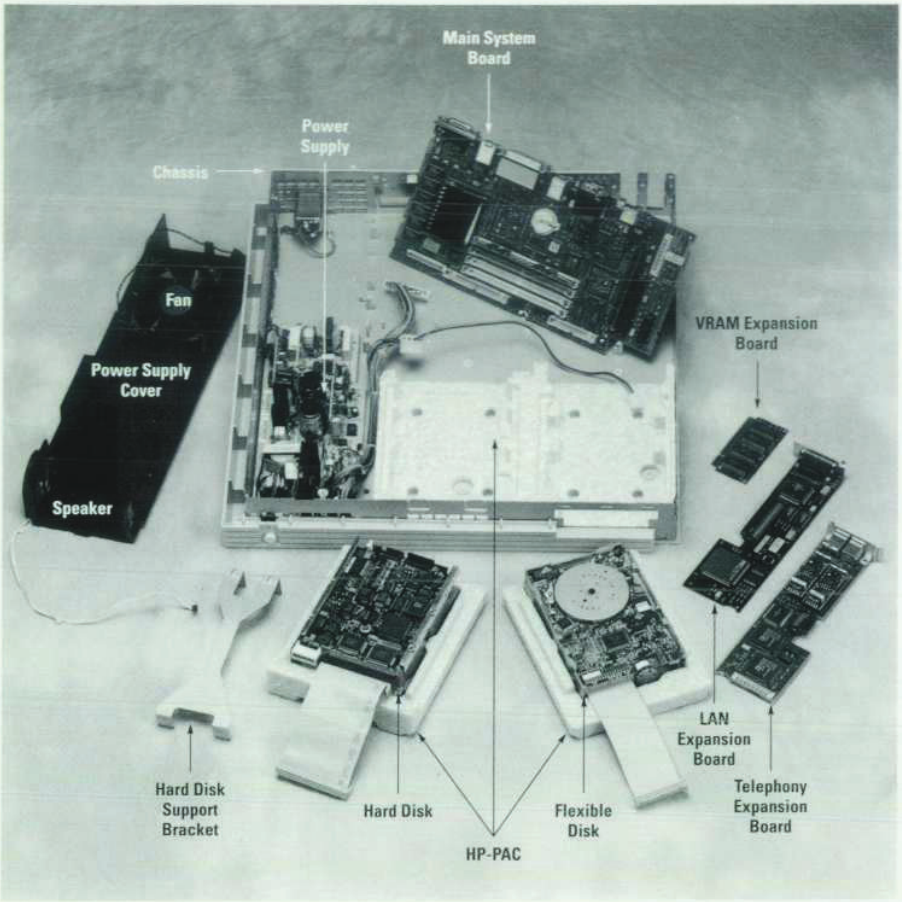


Fig. 4. The Model 712 workstation showing components disassembled from the chassis.

External Peripheral Products

The product design of the three external peripherals also includes a large degree of functional integration. Each of these boxes is designed as a miniature Model 712 workstation, with HP-PAC cushions providing location and support for the drive mechanism, a printed circuit board (for power conversion), power switch plunger, and cabling. The plastic cover for each product includes any necessary doors, light pipes, and buttons. The chassis assembly of each product integrates a plastic base, metal chassis, spring clip, dress panel, and SCSI signal cable (attached with screws by the vendor). Thus, final assembly parts involved in the manufacturing of the box include only the chassis assembly, internal power cable, printed circuit board, plunger rod, HP-PAC, disk mechanism, and top cover. Like the workstation, there are no fasteners for manufacturing or the customer to deal with, and the top cover snaps into place to retain all parts inside.

Low Cost for Entry-Level Pricing

To command lower material costs for mechanical components, all custom plastic and sheet-metal parts were hard-tooled for mass production. The chassis of each product was designed with a minimum of folded features to reduce part complexity and the cost associated with that complexity. All major sheet-metal parts use progressive tooling for the lowest price.

To reduce the amount of final assembly time (and labor costs) involved in the product, components were designed with a high degree of functional integration. Integrated components (such as chassis or top cover assemblies) are assembled by vendors, placing the burden of labor on these non-HP processes and thus achieving lower pricing of the final product. This functional integration of components also lowers cost by reducing part count and related inventory management.

Because of the no-fastener design, final assembly takes under four minutes for the workstation product and comparable times are achieved for the external peripherals. This ease of manufacturing lowers manufacturing costs because of reduced assembly time and overhead costs. It also makes the product much better suited to indirect market channels, which prefer to configure products themselves and often do this at the last possible moment before shipment.

Environmentally Friendly

The Model 712 workstation and peripherals also conform to HP's new guidelines for product stewardship. Virtually every component of the workstation and peripheral products can be easily disassembled, identified, and recycled. Each plastic part contains engraved information that identifies the type of plastic used, and only four different types of plastic are used within the entire family of products. To assist the disassembly process, the products use plastic heat staking to join parts together, which can easily be cut away during the disassembly process. The new HP-PAC material can be recycled as well, either by grinding to pellet size and reusing in other shipping cushion parts, or by melting the material

© Copr. 1949-1998 Hewlett-Packard Co.

down to solid plastic. And again, because there are virtually no fasteners to deal with, disassembly is quick and thus more parts are given to recycling. Materials with bromide compositions have been avoided, except for the HP-PAC parts, which require a bromide flame-retardant treatment to meet safety requirements.

Other product stewardship features include:

- No painted components (all plastics with molded colors)
- No plated plastics
- No adhesives
- Required labels can be recycled along with plastic base material
- Reusable aftermarket components (flexible and hard disk, power supply, CPU, and fan)
- Bulk packaging of final assembly components implemented on larger parts (reduces manufacturing waste)
- Printed circuit boards built in approved non-ODS (ozone-depleting substance) processes
- Embedded fan (low acoustic noise).

Reference

1. J. Mahn, et al, "HP-PAC: A New Chassis and Housing Concept for Electronic Equipment," *Hewlett-Packard Journal*, Vol. 45, no. 4, August 1994, pp. 23-28.

© Copr. 1949-1998 Hewlett-Packard Co.

Development of a Low-Cost, High-Performance, Multiuser Business Server System

Using leveraged technology, an aggressive system team, and clearly emphasized priorities, several versions of low-end multiuser systems were developed in record time while dramatically improving the product's availability to customers.

by Dennis A. Bowers, Gerard M. Enkerlin, and Karen L. Murillo

The HP 9000 Series 800 Models E25, E35, E45, and E55 (Ex5) and the HP 3000 Series 908, 918, 928, and 938 (9x8) business servers were developed as low-cost, performance-enhanced replacements for the HP 9000 F Series and low-end G Series and the HP 3000 Series 917, 927, 937, and 947. The development of the PA-RISC PA 7100LC processor chip and the LASI (LAN/SCSI) I/O interface and the evolution of DRAMs for main memory enabled the development of these low-end servers. The PA 7100LC and the LASI I/O interface are described in the articles on pages 12 and 36 respectively.

The priorities for the Models Ex5 and Series 9x8 server project were short time to market, low cost, and improved performance. The functionality and quality of the new servers were to be as good as the products they were replacing, if not better. The challenge was to get these new servers to market as soon as possible so that HP could continue to be competitive in the business server market and our customers could benefit from better performance at a lower price. We were able to get the first versions of these systems completed, released, and shipping on time with all new VLSI components.

Low-Cost, Higher-Performance Features

The principlal reason for achieving high integration and low cost for the Model Ex5 and Series 9x8 servers was the development of the PA 7100LC processor chip, which was being developed at the same time as our servers. Integrating the floating-point unit, the 1K bytes of internal instruction cache, the external cache interface, the TLB (translation lookaside buffer), the memory controller, and the general system connect (GSC) I/O interface inside the PA 7100LC processor chip allowed the Model Ex5 and Series 9x8 designers to condense the CPU and main memory onto the same board.

Also, at the same time as our new servers were being developed, DRAM densities doubled (in some cases quadrupled) to allow more memory to be put into a smaller space. The Model Ex5 and Series 9x8 servers use the same industry-standard ECC (error correction coded) SIMM modules used in the HP 9000 Model 712 and other HP workstations. The Model Ex5 and Series 9x8 servers use 16M- and 32M-byte SIMMS which must be inserted in pairs to provide 32M to 256M bytes of main memory. ECC memory was chosen because it carries two additional address lines making it possible to put four times the memory capacity on one SIMM while staying compatible with industry-standard modules. The 64M-byte SIMM was designed several months after first introduction of the new low-end servers to boost their maximum memory to 512M bytes. This larger SIMM is not available as an industry standard.

Four versions of the Model Ex5 and Series 9x8 processor have been developed, differentiated by clock speed, cache size, and cost. Each version is fully contained on the system board (which also contains cache, main memory, processor dependent hardware and firmware, and 802.3 LAN connect) and is easily installable and upgradable. Table I lists the technical specifications for the different Model Ex5 systems and summarizes the HP-UX* performance characterizations. The Series 9x8 MPE/iX systems have equivalent CPU hardware, and their specifications are close to those given in Table I.

Table I
Technical Specifications for HP 9000 Model Ex5 Systems Running the HP-UX Operating System

Processor Performance	Models			
	E25	E35	E45	E55
Clock (MHz)	48	64	80	96
SPECint92	44	65	80	104
SPECfp92	66	98	120	156
OLTP Transactions/s	80	125	155	180
Standard memory/cache (M bytes)	16	16	16	16
Maximum memory (M bytes)	512	512	512	512
Cache size (K bytes)	64	256	256	1000
Cache SRAM speed (ns)	15	12	10	7.5

© Copr. 1949-1998 Hewlett-Packard Co.

Architecture

Fig. 1 shows a block diagram for the Model Ex5 and Series 9x8 servers.

The general system connect (GSC) bus was designed as a new, more powerful system bus for higher performance. The Model Ex5 and Series 9x8 servers only use the GSC bus for the processor, main memory, and 802.3 LAN through the LASI chip. The midrange and high-end server systems also support the GSC bus as their high-performance I/O bus. All PA-RISC systems support the HP-PB[1] (HP precision bus) as the common I/O bus because multiple functionality (hardware and drivers) currently exist for this bus. The interface from the GSC bus to the HP-PB is accomplished in a chip called the HP-PB bus converter.

The HP-PB bus converter chip is a performance-improved version of the bus converter that was used in the HP 9000 F and G Series and HP 3000 Series 9x7 machines. This chip allows the Model Ex5 and Series 9x8 servers to leverage HP-PB I/O functionality from the systems they are replacing.

The HP-PB bus converter implements transaction buffering† as an HP-PB slave, gaining performance improvements of 10% to 28% over its predecessor. The chip supports GSC to HP-PB clock ratios ranging from 3:1 to 5:1 in synchronous mode when the GSC bus is operating under 32 MHz. It switches to asynchronous mode when the GSC bus operates in the 32-to-40-MHz range. These ratios and the asynchronous feature of the HP-PB bus converter allow fair flexibility in CPU and GSC operating frequencies while maintaining a constant 8-MHz HP-PB frequency. The bus converter also provides an interface to the access port used for remote support, and the control signals used for the chassis display and status registers. The chip is designed for the HP CMOS26B process and comes packaged in a 208-pin MQFP (metal quad flat pack).

The other key VLSI chip used in the I/O structure for the Model Ex5 and Series 9x8 servers is the LASI chip. The LASI

† With transaction buffering, during reads from disk, data is buffered so that HP-PB transactions can continue at maximum pace.

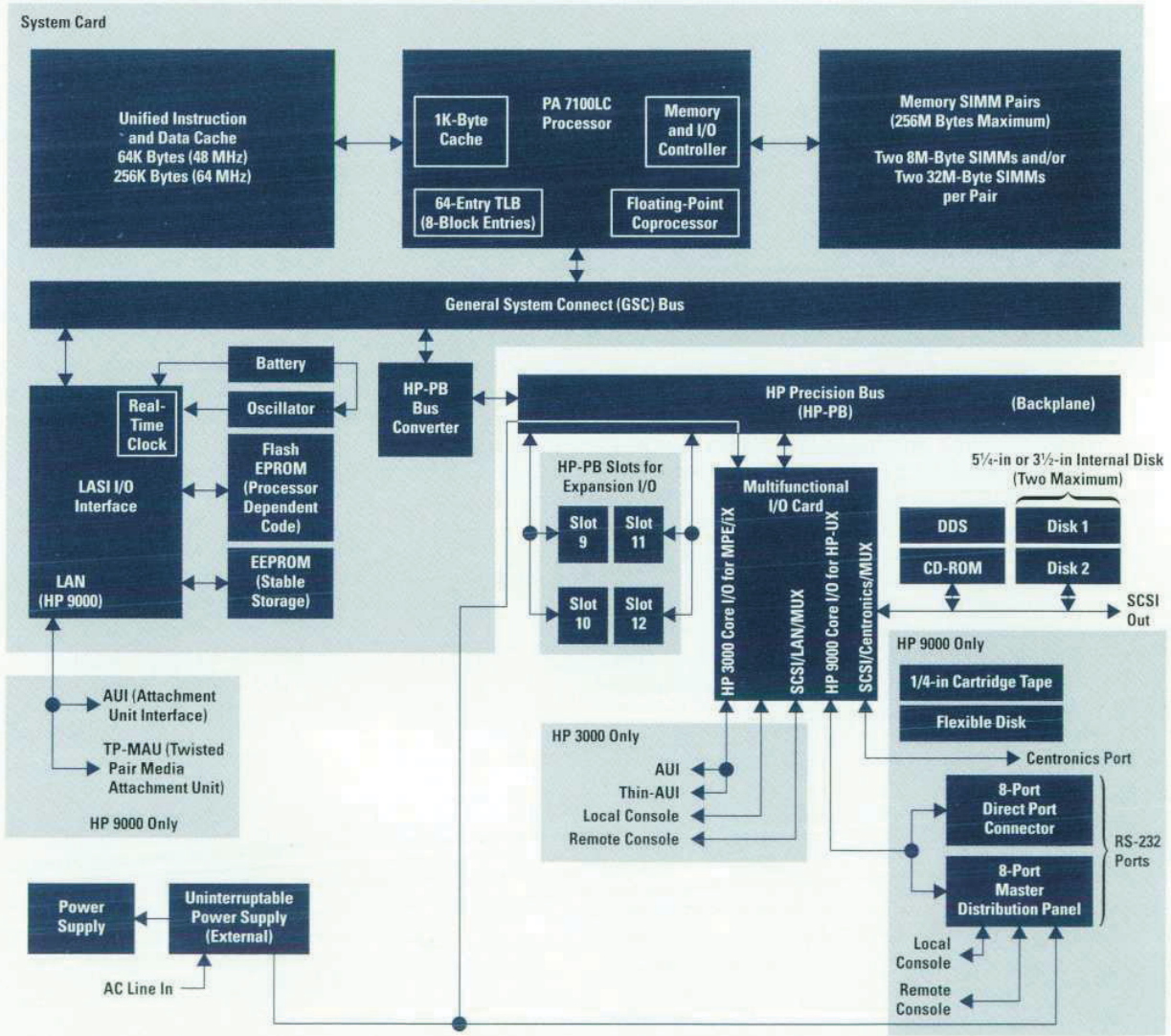


Fig. 1. Block diagram for the HP 9000 Series 800 Models Ex5 and the HP 3000 Series 9x8 business servers.

© Copr. 1949-1998 Hewlett-Packard Co.

chip is designed to have the same integration impact on core I/O as the PA 7100LC had on the CPU and GSC bus interface. The workstation products are able to take advantage of this (see article on page 6), but the multiuser server systems were not able to take advantage of LASI functionality.

LASI functionality includes interfaces to IEEE 802.3 LAN, SCSI, processor dependent code, Centronics, RS-232, audio, keyboard, flexible disk, and GSC bus arbitration logic and the real-time clock. Because HP-UX and MPE/iX software drivers could not be made available in time for our release, only a small subset of LASI functionality could be used on the new servers. Thus, the decision was made to continue using the core I/O card from the previous versions of low-end servers because it provides all the functionality needed.

For the 96-MHz version of the Model Ex5 and Series 9x8 servers, a chip with a subset of the functionality of LASI was used. This was developed as a cost reduction for those applications that use only the LAN, GSC bus arbitration, and processor dependent code path. The 96-MHz version had to add a real-time clock on the system board to have equivalent functionality to what was needed from LASI.

In addition to the above VLSI chips and printed circuit boards, the Model Ex5 and Series 9x8 servers have the internal capacity for two disk drives (4G bytes), two removable media devices, and up to four I/O slots. The packaging and power supplies for the new servers are highly leveraged from the previous low-end server systems.

Meeting Fast Time-to-Market Goals

Meeting deadlines for any program is always a challenge. Too often it is believed that a few extra hours a week is all that is needed to keep the project on track. But many well-intentioned programs soon lose time with unexpected delays even when the project team is made up of industrious folks willing to do whatever it takes to stay on schedule.

At large corporations like HP, where releasing a product to market may span several divisions, the task is even more daunting. With our lab's mission of providing world-class low-end commercial business systems and servers, time to market is always expected to be a key objective. In the case of the Model Ex5 and Series 9x8 program, it was the primary objective. Additionally, we were challenged to keep cost projections in line with the set goals, and to meet or exceed the quality of the versions of the low-end servers that we were replacing. Quality is consistently a key objective on all HP products.

The main challenge for the Model Ex5 and Series 9x8 program was to achieve (on schedule) an order fulfillment cycle time† of 10 or fewer days for the entire product family. With the existing product family averaging order fulfillment cycle times four to five times larger than our 10 or fewer days goal, it was evident that for the new servers a well-orchestrated program that involved the entire system team was necessary to meet this challenge.

Fig. 2 shows a spider chart of the overall metrics for the Model Ex5 and Series 9x8 program. Note that the program achieved or exceeded all planned manufacturing release goals. Even the factory cost goal was exceeded, which was

† Order fulfillment cycle time is measured from when HP receives a customer's order to the time when the order is delivered at the customer's dock.

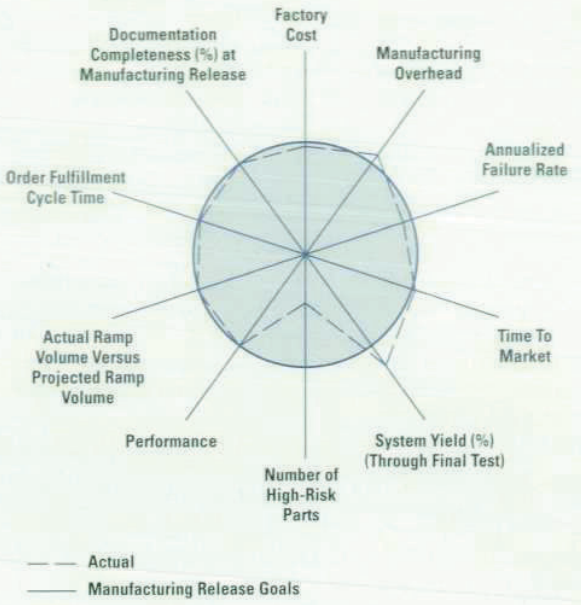


Fig. 2. Spider diagram showing how well the project team for the Models Ex5 and Series 9x8 servers met their project goals.

at risk when the hardware team added existing material into the design over less expensive functionality, reducing the software development schedule. The order fulfillment cycle time objective was not only achieved but exceeded! For the first three months of production, order fulfillment cycle time averaged under nine days.

The following sections summarize the reasons we met or exceeded our cost, quality, time-to-market, and manufacturing release goals.

Consolidation of Project Team. When the Model Ex5 and Series 9x8 program was in its early stages of design the development team was dispersed in two different geographic locations. The remote organization was eliminated and the project development and management were consolidated in one location under one manager. With this organization, technical decisions regarding system requirements could be made quickly and effectively.

Ownership of Issues. A system team composed of representatives from the different organizations involved in the development of the Models Ex5 and Series 9x8 servers was organized. Weekly one-hour meetings were held with the main focus on issues or concerns that impacted the project schedule. Communication was expected to be limited to discussions that affected everyone. Issues were captured and assigned an owner with a date assigned for resolution of the issue. Representatives at the meeting were expected to own the issues that were presented to their organization. No issue was closed until the team agreed upon it. This ensured that technical problems did not "bounce" around looking for an owner.

Interdivisional Communication. Effective interdivisional teams establish good working relationships to ensure timely response to actions and issues. An example was the decision to change the core I/O functionality. While the hardware team improved their factory cost by incorporating new, less costly hardware, the software team would have realized a

© Copr. 1949-1998 Hewlett-Packard Co.

longer schedule to provide new software features to support the new hardware. After reviewing the plans, the hardware team, aware of the critical time-to-market objective, recommended a return to the existing I/O feature implementation at an impact to factory cost for the sake of the software development team's ability to improve their schedule. The end result was that the hardware team still achieved their factory cost goals (by making adjustments elsewhere), and the software development team achieved their schedule goals.

Leverage Design Where Possible. When time to market was established as the key objective for the project, the development teams realized that leveraging from as many existing products as possible would greatly benefit achieving this goal. The following components were leveraged from new or existing products:

- Product package. Sheet metal was leveraged from the existing low-end business servers with minor changes to accommodate new peripherals and a different processor and main memory partitioning scheme. Plastic changes were kept to a minimum in an effort to use tools already established. (Only one new tool was required.)
- Base system configuration. The base system was established using the I/O printed circuit boards and several peripherals available on the existing low-end servers.
- Memory. The memory design was leveraged from the memory configuration used in the HP 9000 Model 712 workstation, which uses SIMM modules for the base memory system. Higher-density memory was designed specifically for the Model Ex5 and Series 9x8 servers after first release to increase their maximum memory capacity.
- Power supply. The power supply was leveraged from the existing servers.
- Printed circuit boards. The core I/O boards from the existing servers were used with only minor firmware changes to the HP-UX version. The processor board and backplane were new designs based on ideas shared with the Model 712 development team.
- VLSI. The PA 7100LC processor chip and the LASI core I/O chip were leveraged from the Model 712 workstation, which was being designed at the same time as our server systems.
- Firmware. Some of the firmware and I/O dependent code was codeveloped with the Model 712 development team.

Fast Time to Manufacturing Release. The use of concurrent engineering played a key role in reducing the back-end schedule. The back end of the schedule consists largely of manufacturing activities (including final test and qualification) aimed at achieving a release of the product for volume shipment. In the case of the Model Ex5 and Series 9x8 servers, with the individual boards being built in two geographically different manufacturing facilities, it was imperative that communication between these entities receive ample attention.

To facilitate this communication, a coordination team consisting of new product introduction engineers and new product buyers and logistics people were located in close proximity with the R&D development team. Everyone attended the system team meetings, which were led by the hardware lab, to ensure that the most current information was applied to the overall system schedule. In addition, production build meetings were held before, during, and after each prototype run to discuss build results. Ensuring that all manufacturing personnel realized that these systems were engineering prototypes, with a high potential for problems, was a difficult task. Most people were not used to seeing lab prototypes being built in a production process. Since the line was shared with currently shipping products, it was extremely important to ensure that building the prototypes did not impede shipping other products.

Prototype Management. Two operating system environments were required for the new servers, the HP-UX operating system release 9.04 and the MPE/iX operating system version 4.0. Since these environments were under development at the same time as our products, it was essential that hardware prototypes be delivered efficiently and be of sufficient quality to ensure expedient use by the software development groups. Thus, three key objectives were considered essential by the development groups. First, units had to be of the highest quality. Second, delivery of the units had to be on time. Finally, downtime because of hardware problems had to be minimized.

To accomplish the first goal, all prototypes were built using the entire production process. No prototypes were handcrafted in the lab. This ensured that units were built with the same quality standards as are applied to released systems. Additionally, each customer was assured of receiving the latest revision of materials released to production. Even new parts not covered under manufacturing release criteria were guaranteed to be of the same revision level. All revision levels were tracked on each unit for the life of the project.

For the second objective, a customer priority list was generated based on customer orders and needs. After the orders were submitted to the manufacturing systems, build priorities were set based on the critical needs being supplied first. From functional prototypes to production prototypes, upgrade kits were structured and made available. In cases where a new system was not required, customers had the option of moving immediately to an upgrade. Also, performance upgrades were designed to require a swap of the processor card only.

Tracking the revision level of all hardware was essential to achieving the third objective of minimizing downtime because of hardware. Another key point was being able to react to a customer's problem quickly. We used a prerelease support team at another HP division to ensure timely response. Spare material was purchased by the support team and defective parts were returned to the lab for analysis.

Using all these methods, we were able to achieve the goal of having all operational prototype units upgraded to manufacturing release equivalence before manufacturing release. This guaranteed test partners use of the machines for future development without the "not-quite-final-product" concerns.

We were not without our share of problems in terms of effectively managing the prototypes. For instance, several units were placed inside an environmental test chamber for weekend testing. During the early morning hours on a

© Copr. 1949-1998 Hewlett-Packard Co.

Sunday, the temperature controller of the chamber went out of control, ramping the temperature to beyond 70 °C. The additional heat caused the fire sprinkler system in the chamber to turn on, flooding the chamber at a rate estimated at 10 gallons per minute. The units were standing in four feet of water, but with the disk drives external to the chamber, the test continued. When the chamber was finally shut down, the water mopped up, and the results checked, it was discovered that two of the seven units,which were on the top rack, out of the standing water, continued to operate without failure throughout the test. This test was affectionately named the "bathtub test."

Time-to-Market Focus. Establishing the time to market as the key objective for the program was not enough to ensure its success. The teams involved required constant reminders to stay focused on this objective and make trade-offs accordingly. Once the schedule was confirmed and accepted, it was important to acknowledge the progress. Any activities that appeared in danger of jeopardizing the schedule were reviewed and tackled accordingly.

However, the project team realized that in the past changes to system requirements had a big impact on meeting project schedules. Changes to system requirements to modify or include a feature that might improve sales or could be easily implemented at the cost of another metric might result in significant changes to the hardware or operating system design. In the case of the Model Ex5 and Series 9x8 servers, the system team implemented a process that was also used by the software development teams to control design changes. This process is called *change control*, which requires the change requester to provide a specific level of information to determine whether a particular change is viable. While this is not a new idea, the Model Ex5 and Series 9x8 development team elected to make one additional rule change. Each change request submitted would be briefly looked at to determine how the change would affect the base system. In other words, we wanted to ensure that a change was critical enough that it needed to be added to the products planned for the first release.

The hardware system team put on hold all change requests that were determined not to be required for the first release. To avoid causing lots of changes to the software after first release, some of the critical enhancements that were considered crucial to future sales were briefly reviewed and included in the initial software release. In some cases this meant no changes were required after the first software release. However, there were some instances of patches required for full functionality.

Customer Order Fulfillment Cycle Time

For the Model Ex5 and Series 9x8 servers to stay competitive, cost and performance were not the only items that played an important role. During 1993, it was clear that HP had an order fulfillment cycle time problem, which of course made our customers unhappy and affected our competitiveness. A task force was formed to address HP's order fulfillment cycle time problems. We found out that results from this task force would not arrive in time to help us with our new products. Thus, we formed a team seven months before introduction to ensure that the reduced order fulfillment cycle time process for the Model Ex5 and Series 9x8 servers was in place when the products were ready to be shipped to customers.

Our goal was to reduce the time between the receipt of a customer purchase order for a system and the time when the system is delivered to the customer site. We wanted to reduce this time by 75% of what it was for our existing servers. To accomplish this goal, the following changes were made before product introduction:

- The product structure was made much simpler and it includes fewer line items.
- Product offerings to distributors were unbundled.
- Product numbering for distributors' orders had a single SKU (stock keeping unit) for ease of ordering.
- The rules for our factory configuration system and field configuration system were mirrored.
- Early and proactive material stocking was performed before introduction to ensure that plenty of material was on hand to meet customer demand immediately.
- Factory acknowledgments were automated for clean orders.
- Intensive training was given to order processing personnel in the field and the factory about the Model Ex5 and Series 9x8 servers two months before introduction.
- Consignment, demonstration, and distributor units were stocked before introduction.
- More capacity was added to the factory, and assembly processes were streamlined.
- All new processes were tested intensively before introduction.

With these steps we were able to meet and exceed our order fulfillment goal.

Conclusion

The real success of the Model Ex5 and Series 9x8 server program was that the goals for fast time to market and reduced order fulfillment cycle time were achieved. These were major accomplishments considering the events that took place throughout the whole project including the development of a major VLSI component, consolidation of the design team from different divisions and locations, communication between different manufacturing entities, and a stream of last-minute catastrophes such as flooding prototypes in the environmental test ovens and several eleventh-hour VLSI bugs that had to be fixed.

Acknowledgments

Although one of our key success factors was the small dedicated team, many key people and teams contributed to the development of the Model Ex5 and Series 9x8 low-end servers. We would like to acknowledge the team in Germany (both lab and manufacturing) led by Bernd Buggerman that began and nurtured the initial idea for the new servers. They brought forth many of the design and process improvements that made the products successful. Once the project moved to Roseville, the assembled design team took hold quickly and stepped up to the challenge. We would like to thank John Spofford for his tenacity and Scott Stallard and John Adelsbach for their support. Karen Murillo would like to thank her lab team and especially Loren Koehler, Matthias Meier, and Don Zimmer for providing material for this article. In addition to these three, Robert Dobbs, Delfina Mooney, Ken Tang, Linda VanderWegan, Darrell Walker, and Charles Zacky were key to the success of the the Model Ex5 and Series 9x8 servers' timely implementation, as were John

© Copr. 1949-1998 Hewlett-Packard Co.

Connolly and his team who developed the HP-PB bus converter chip. We would also like to thank the teams responsible for the PA 7100LC and LASI chips, including Craig Simpson, Tom Meyer, and Tom Spencer. Dennis Bowers would like to thank the members of his system team, Wolfgang Boehm, Hank Cureton, Mark Driscoll, Gerard Enkerlin, Phil Knutson, Barb McGillivary, Lisa Mitchell, Marty Nichols, Craig Nunes, Kevin Smith, and Dave Snow (and their teams) as they were consistent links to our program partners and allowed us to push harder on our goals than anyone thought achievable. A special thanks goes to Elaine Calomeni for keeping our prototypes up and hardware flowing to and from our remote testing partners.

Thanks also go to the order fulfillment team of Carol Talley, Mauricio Cori, Mary Marcus, Cheryl Marshman, Doug Anderson, John Duff, Tom Hack, Jordan Levin, Wil Pomeroy, Ray Barrett, Dawn Williams, Don Leigh, Steve Wehr, Oanh Phuong, Joe Barrington, Marty Rothenberg, Bob Littlejohn, Kyle Christensen, Bill Schaefer, Laura McMullen, and many other individuals who helped make this program successful.

References

1. T. Alexander, et al, "Corporate Business Servers: An Alternative to Mainframes for Business Computing," *Hewlett-Packard Journal*, Vol. 45, no. 3, pp. 8-30.

2. *Ibid*, p. 17.

HP-UX is based on and is compatible with Novell's UNIX® operating system. It also complies with X/Open's* XPG4, POSIX 1003.1, 1003.2, FIPS 151-1, and SVID2 interface specifications.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open is a trademark of X/Open Company Limited in the UK and other countries.

© Copr. 1949-1998 Hewlett-Packard Co.

HP Distributed Smalltalk: A Tool for Developing Distributed Applications

An easy-to-use object-oriented development environment is provided that facilitates the rapid development and deployment of multiuser, enterprise-wide distributed applications.

by Eileen Keremitsis and Ian J. Fuller

HP Distributed Smalltalk is an integrated set of frameworks that provides an advanced object-oriented environment for rapid development and deployment of multiuser, enterprise-wide distributed applications. Introduced in early 1993, and now in its fourth major release, HP Distributed Smalltalk leverages the ParcPlace Smalltalk language and the VisualWorks development environment. Together, HP Distributed Smalltalk and VisualWorks enable rapid prototyping, development, and deployment of CORBA-compliant applications.†

In the global marketplace, corporate information technology needs are increasingly demanding because worldwide competition requires geographically dispersed operations, changing markets require agility to remain competitive, pressure to improve return on investment requires strong cost controls, timely access to complete information is crucial for business success, and finally, corporate users require access to both legacy and newly developed information sources and applications.

HP Distributed Smalltalk helps answer these business needs by supporting:

- Easy on-demand access to information and services across the enterprise
- Dynamic interaction of distributed people and resources
- Greater application flexibility and ease of use
- Insulation from differences in operating environments
- An architecture that supports an evolutionary approach including legacy system integration
- Industry standards that will allow application interoperability across languages, high productivity, and code reuse.

Customers can take advantage of HP Distributed Smalltalk's easy-to-use development environment to create distributed solutions to compete effectively in the global marketplace. For example, with HP Distributed Smalltalk, customers might build on the sample Forum application (described later) so that their geographically dispersed users can simultaneously annotate a shared document. Also, customers might use HP Distributed Smalltalk to create three-tiered database access applications that extend the advantages of existing client-server architectures for better isolation between user interfaces, data manipulation models, and legacy and new data.

† CORBA, or Common Object Request Broker Architecture, defines a mechanism that enables objects to make and receive requests and responses. HP Distributed Smalltalk's implementation of this architecture is described later in this article.

Three-tiered applications are the most efficient and scalable form of software design for building complex applications. They carefully separate the user interface (tier one) from the business rules governing the application (tier two) and the persistent storage for the information in a database (tier three). Each tier can reside on a different machine in a network, making best use of the network resources. HP Distributed Smalltalk contains objects that enable the straightforward construction of these applications.

Using HP Distributed Smalltalk

An application written in HP Distributed Smalltalk is able to respond to service requests from remote systems. Remote entities that request services of an application do not have to be written in HP Distributed Smalltalk as long as they are in a system that implements the standard ORB (object request broker) and common object services from the Object Management Group (OMG). See "Object Management Group" on page 86 for a description of these items.

In many cases an HP Distributed Smalltalk application's component objects are distributed across several systems. These distributed objects can interact seamlessly so that end users are unaware of where the objects are located.

An overview of the process of running an HP Distributed Smalltalk application is shown in Fig. 1. For incoming requests to the service provider, the ORB translates requests from the implementation-neutral Interface Definition Language (IDL) to the local language (ParcPlace Smalltalk) and forwards them to the correct local object for processing. To complete the request, the service provider's ORB takes return values, translates them to IDL and forwards them to the remote ORB from which the request was received.

Not only does HP Distributed Smalltalk support distributed application delivery but it also provides an environment for distributed application development, which includes:

- A complete implementation of the Object Management Group's latest standards
- A rich suite of tools for application development and administration including simulated remote test support, a remote debugger, and an IDL interface browser and generator
- A user interface environment and sample applications that developers can reuse or extend, or simply use to become familiar with the system.

© Copr. 1949-1998 Hewlett-Packard Co.

Object Management Group

The Object Management Group, or OMG, is a nonprofit international corporation made up of a team of dedicated computer industry professionals from different corporations working on the development of industry guidelines and object management specifications to provide a common framework for distributed application development.

OMG publishes industry guidelines for commercially available object-oriented systems, focusing on areas of remote object network access, encapsulation of existing applications, and object database interfaces. By encouraging industrywide adoption of these guidelines, OMG fosters the development of software tools that support open architecture, enabling multivendor systems to work together.

To define the framework for fulfilling its mission, in 1992 OMG published its Object Management Architecture Guide. This guide provides a foundation for the development of detailed interfaces that will connect to the elemental components of the architecture. Fig. 1 shows the four main components of this architecture:

- The object request broker (ORB) enables objects to make and receive requests and responses in a distributed object-oriented environment.
- Object services is a collection of services with object interfaces that provide basic functions for creating and maintaining objects.
- Common facilities is a collection of classes and objects that provide general-purpose capabilities useful in many applications.
- Application objects are specific to particular end-user applications.

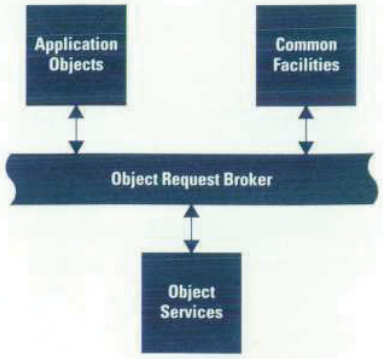


Fig. 1. The object management architecture.

The application objects, object services, and common facilities represent groupings of objects that can send and receive messages. The software components in each of these primary components have application programming interfaces that permit their participation in any computing environment that is based on an object technology framework.

In addition, because HP Distributed Smalltalk is an extension of VisualWorks, developers are able to do their programming in a language they already know (ParcPlace Smalltalk) using the VisualWorks application builder.

VisualWorks is an implementation of the Smalltalk programming language and environment. It provides an excellent environment for building standalone and simple client/server applications that are 100% portable between many of the major computing platforms and operating systems. HP saw an opportunity to enhance the capabilities of VisualWorks to be the basis for next-generation applications by adding objects that enable VisualWorks systems to communicate directly using a standardized set of communications facilities.

Framework

The HP Distributed Smalltalk framework is an environment that encompasses everything from communication with other systems through database access to the object-oriented ParcPlace Smalltalk language and a rich suite of developer's tools, all seamlessly integrated to facilitate distributed application development.

The major components of HP Distributed Smalltalk are shown in Fig. 2 and briefly defined below:

- HP Distributed Smalltalk ORB. This is a full implementation of the Object Management Group's Common Object Request Broker Architecture (CORBA).
- Remote Procedure Call (RPC) communication. This component supports efficient and reliable transfer of messages between systems.
- HP Distributed Smalltalk object services. This includes all standard object services required by distributed systems, as well as support for creating and maintaining objects and the relationships between them.

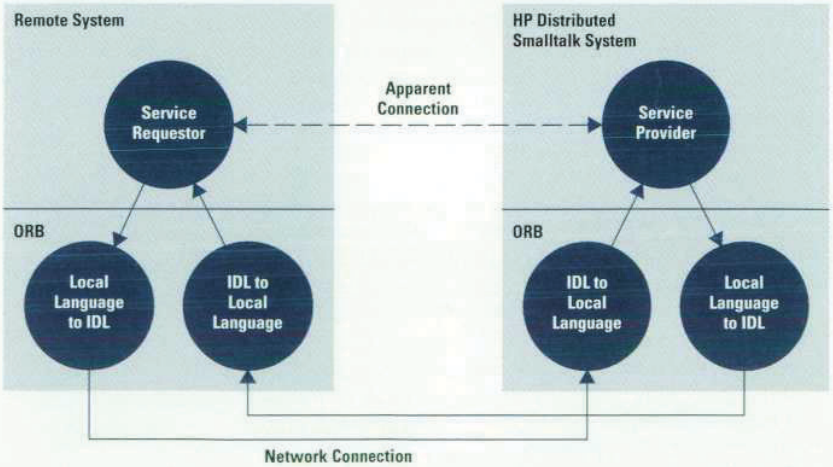


Fig. 1. Overview of the HP Distributed Smalltalk process.

© Copr. 1949-1998 Hewlett-Packard Co.

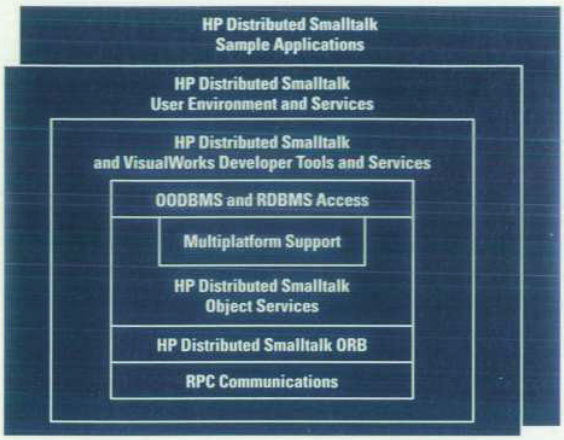


Fig. 2. The major components of HP Distributed Smalltalk.

- Multiplatform support. HP Distributed Smalltalk applications that run on one platform (hardware and operating system combination) can run, without porting, on any other supported platform.
- OODBMS and RDBMS access. HP Distributed Smalltalk provides database access directly to HP's Odapter† and Servio's GemStone as well as to Sybase and Oracle (via VisualWorks). HP Odapter can be used to provide access to a variety of other database systems.
- HP Distributed Smalltalk developer tools and services. This level of the framework provides support specifically designed for developing, testing, tuning, and delivering distributed applications. HP Distributed Smalltalk incorporates a rich development environment, application builder support, and the ParcPlace Smalltalk language.
- HP Distributed Smalltalk user environment and services. These services include a reusable demonstration user interface and desktop environment support for users' work sessions and normal desktop activity.
- HP Distributed Smalltalk sample application objects. These objects provide developers with example code that can be reused or extended, or can provide a source of ideas for developing alternate applications.

The following sections provide more detailed descriptions of the components that make up HP Distributed Smalltalk.

HP Distributed Smalltalk Object Request Broker

HP Distributed Smalltalk is a complete implementation of CORBA, the Object Management Group's specification of an object request broker. HP Distributed Smalltalk's compliance provides the basis for object and application interoperability.

CORBA specifies core services that are required of an object request broker to support interoperable distributed computing. The CORBA specification includes the following core services.

Interface Definition Language Compiler. OMG has defined the Interface Definition Language, or IDL, to be independent of other programming languages. Interfaces for objects that can provide distributed services are written in IDL so that they are accessible to service requesters that might be written in Smalltalk, C, C++, or another language.

OMG recently approved the IDL-to-Smalltalk language binding proposed by HP and IBM. This is important because it allows users to build distributed systems using multiple languages where appropriate, allowing a Smalltalk object to be able to request services of a C++ object or vice versa.

Interface Repository. This service provides a registry of distributable object interfaces for a given system. Any object that remote objects can access has an interface in the interface repository. For example, when objects on two or more systems at different locations collaborate in an application, they interact by sending messages to their interfaces. Since external clients have access to an object's services only through the object's interface, the implementation of the object is private. This privacy provides a variety of benefits, including security, language independence, and freedom to modify the implementation of how a service is performed without external repercussions.

HP Distributed Smalltalk ORB Support. The object request broker (ORB) is the key to providing support for distributed objects. By providing an ORB on each system, HP Distributed Smalltalk makes the location of any object transparent to clients requesting services from the object.

When a message is sent to a local object, the activity is handled normally. When a message is sent to a remote object, the remote object's local surrogate (created automatically by the ORB) intercepts the message, then uses the ORB to locate the remote object and communicate with it (see Fig. 3). Results returned to the calling object appear exactly the same, whether the message went to a local or remote object.

An ORB's responsibilities include:

- Marshalling and unmarshalling messages (translating objects to and from byte streams for network transmission)
- Locating objects in other images or systems
- Routing messages between surrogates and the objects they represent.

While a request is active, both client and server ORBs exchange packet information to track the course of the request

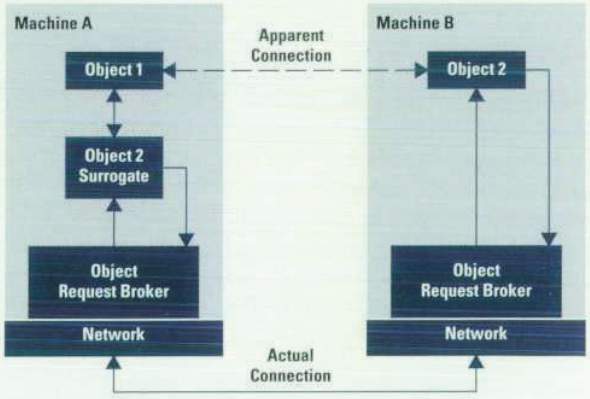


Fig. 3. HP Distributed Smalltalk handles remote access so that a request to a remote object appears the same as a request to a local object.

† HP Odapter is a complementary product from Hewlett-Packard that provides an efficient and scalable link between objects implemented in an object-oriented language such as Smalltalk or C++ and the entities in an Oracle relational database.

© Copr. 1949-1998 Hewlett-Packard Co.

and resolve any network or transmission errors that might occur.

Object Services and Policies

Object services extend the core ORB services to support more advanced object interaction. HP Distributed Smalltalk implements OMG's Common Object Services Specification (COSS), which extends CORBA to provide protocols for common operations like creating objects, exporting and destroying objects (life cycle), locating objects (naming), and asynchronous event notification. Additional object services and policies provide efficient interaction between finer-grained distributable objects.

Naming.† There is a standard for assigning each object a unique user-visible name. Names are used to identify and locate both local and remote objects.

Event Notification.† This is a service that allows objects to notify each other of an interesting occurrence using an agreed protocol and set of objects.

Basic† and Compound Life Cycle. There are standard ways for objects to implement activities such as create and initialize, delete, copy, and move both simple and compound objects, externalize (prepare for transmission to remote systems), and internalize (accept objects transmitted from remote systems). Compound objects, built from simple objects, can include application components, anything that appears on a user's desktop (such as a document, a mail handler, or a graphics toolbox), complete applications, and so on.

Relationships: Containment and Links. Links allow networked relationships among objects. Objects can be linked together with various levels of referential integrity (determining how to handle situations when one of the parties to the link is deleted), and in one-to-one, one-to-many, and many-to-many relationships.

Together with links, containment establishes and maintains relationships between objects. Each object has a specific location within some container. Containers are related hierarchically. HP Distributed Smalltalk provides objects that implement a generic distributed container. Programmers can use these objects to build specific implementations such as an electronic mail envelope (containing components of a message) or a bill of sale (containing information about items in a shipment) with minimal extra programming.

Properties and Property Management. Properties are part of an object's external interface (owner, creation date, modification date, version, access control list, and so on). They are a dynamic version of attributes.

Application Objects and their Assistants. Application objects are relatively large-grained compound objects that end users deal with (e.g., a file folder or an order entry form). Application assistants are lightweight objects that implement most of the policies and participate in most of the services that desktop objects need to participate in. Application assistants function as the developer's ambassador into the object services. Application assistants can be stored and activated efficiently and provide the basis for future transaction support.

† This service is specified in COSS 1.0.

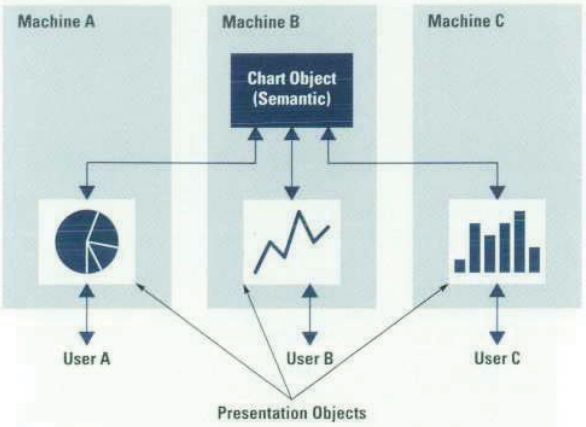


Fig. 4. The bulk of user interaction is with local presentation objects, minimizing and condensing the need to propagate semantically relevant changes over the network. Here for example, a user might choose to look at a chart (semantic object) as a pie, line, or bar chart presentation object.

Presentation/Semantic Split. A logical split between distributed objects, the presentation/semantic split provides an efficient architecture for distributed applications. Local presentation objects handle the bulk of user interaction, while a semantic object (which can be anywhere on the network) holds a shared persistent state of the object (see Fig. 4).

By using the presentation/semantic split, the designer can choose what part of the application should be shared and what should be unique to each user. Applications that might use the presentation/semantic split include a team whiteboard where all behavior is shared but each user can write comments, or a common document with pages that are unique to each user so that all users can read at their own pace. A variety of sample applications included with HP Distributed Smalltalk provide illustrations of how to use the presentation/semantic split.

While use of the presentation/semantic split is optional, it facilitates and optimizes distributed application development and execution. Advantages of using the presentation/semantic split include:

- Acceptable performance levels even over wide area networks
- Association of a single semantic object with multiple presentation objects, a critical feature in distributed computing environments where it is common for many users to work with the same application
- Application access independent of local windowing systems
- Better code reusability.

The HP Software Solution Broker described on page 93 is a good example of using the presentation/semantic split in an application.

Developer Services

HP Distributed Smalltalk also extends VisualWorks with services that support development and test of distributed applications.

© Copr. 1949-1998 Hewlett-Packard Co.

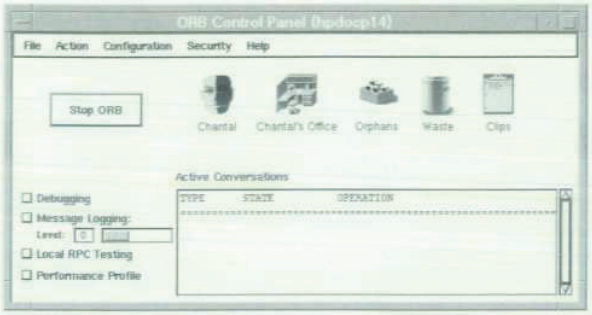


Fig. 5. The control panel provides an easy-to-use interface to administrative and developer services.

Control Panel. The technical user interface to HP Distributed Smalltalk for administrators and developers is invaluable for testing and maintenance (Fig. 5). The control panel provides:

- Controls to start and stop the system cleanly
- Support for local RPC testing (simulated distribution)
- Tracing facilities to log network conversations between objects
- Performance monitoring.

Interface Repository Browser and Editor. The interface repository browser provides an iconic view of the contents of the interface repository where publicly available interfaces are specified (see Fig. 6). It is organized hierarchically so that developers can explore and edit interfaces and construct requests to use the interfaces.

Shared Interface Repository. In HP Distributed Smalltalk, users can share an interface repository on a remote system so they do not have the overhead of keeping a copy of all of the interfaces on every system. The product also supports version management of interfaces, which is very important in large-scale, evolving distributed systems.

Remote Context Inspector and Debugger. This service is an extension that allows debugging on remote images when appropriate. It supports object inspection and debugging for

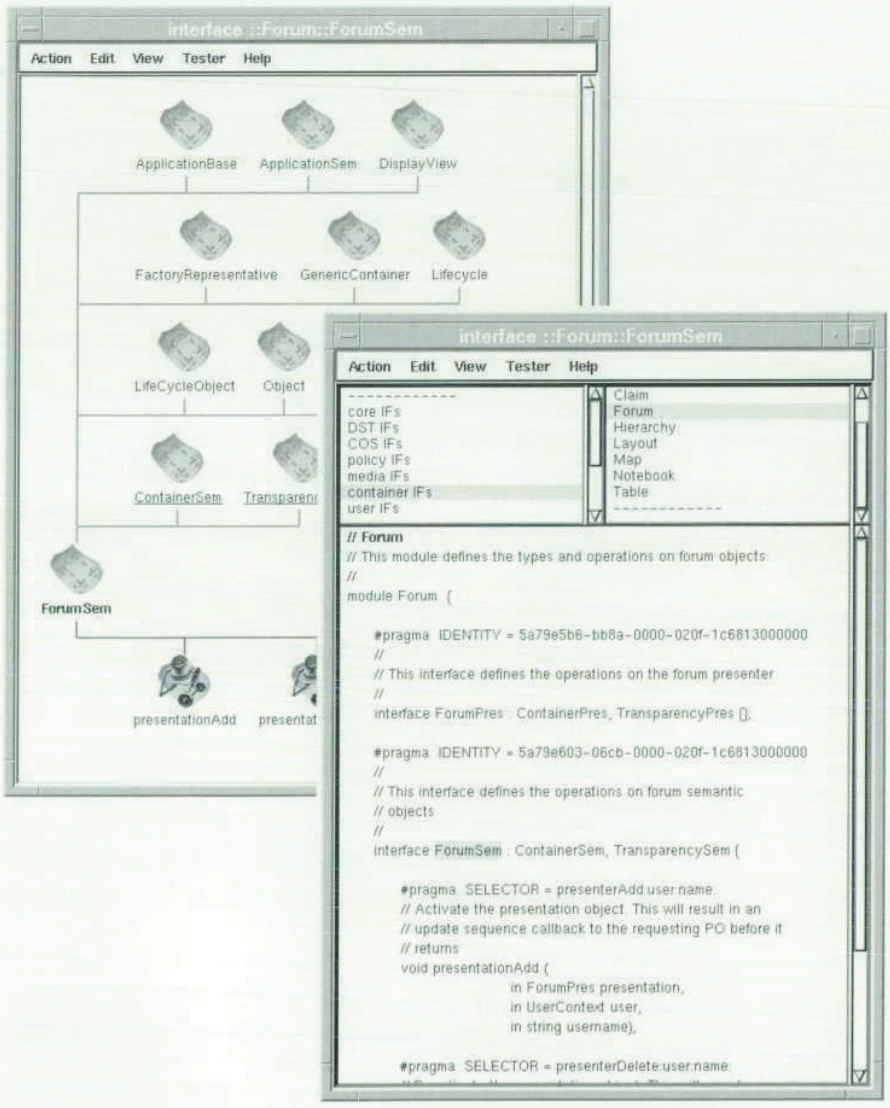


Fig. 6. The interface repository browser can be used to view or edit interfaces that remote clients can use to call local objects.

© Copr. 1949-1998 Hewlett-Packard Co.

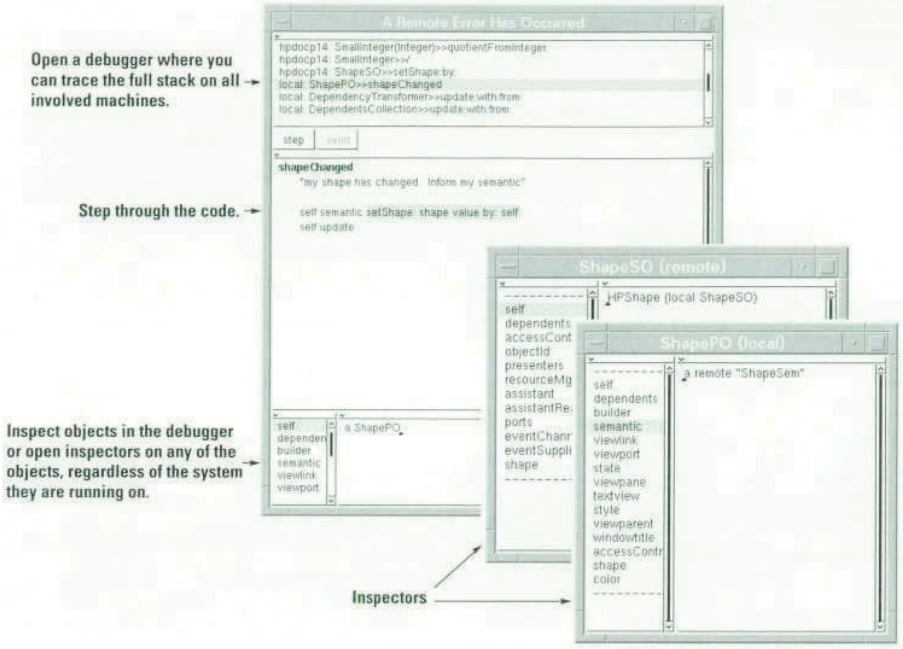


Fig. 7. Screens associated with a remote debugger.

the entire distributed execution context, including communication between images. Fig. 7 shows using the debugger to step through code and inspect objects that might be located anywhere in a distributed environment.

Stripping Tool. To prepare an application for delivery, developers use the HP Distributed Smalltalk stripping tool to remove unneeded classes and interfaces and seal source code when application development is complete. The stripping tool's user interface suggests likely items for removal (see Fig. 8).

User Services

User services allow developers to build a desktop or office environment and control activities during a session.

System Objects. HP Distributed Smalltalk supports a variety of system objects: user, session, clipboard, wastebasket, and orphanage.

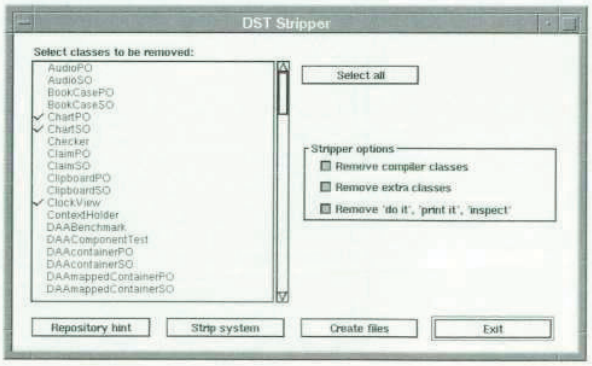


Fig. 8. Interface for the HP Distributed Smalltalk stripping tool.

- User object. This object contains information held about end users of the system including who they are, how to contact them, and so on. User objects may be included by reference in other objects. For example, a user might include a business card in a memo that would enable the receiver to get in touch with the sender.
- Session object. All the information required about the state of a user's environment, including user login, preferences, layout, and so on are contained in a session object. The session object also supports the notion of workspaces, with the potential for developing richer workspace environments. It has no icon on the desktop but it interacts with and supports other application objects.
- Clipboard. This is a container for objects that are being cut, moved, or copied from one location to another.
- Wastebasket. This container receives objects that users throw away. The wastebasket can be cleared when it gets too full.
- Orphanage. This is a container for holding objects that are no longer needed.

Security. Developers can use or extend HP Distributed Smalltalk's access control services in the applications they build, setting controls for host systems, users, or both. Host-system access control lets developers determine whether an image can receive messages from another system. User-level access control lets a developer determine whether a given user has any one of several kinds of privileges (e.g., read or write privilege) for a given object.

Developers can administer access control programmatically or from the default user interface.

Example Code

While all HP Distributed Smalltalk code is available to read, reuse, or extend, the default user interface and certain sample applications may be the best place to start.

© Copr. 1949-1998 Hewlett-Packard Co.



Fig. 9. The screen for presenting the office metaphor and some typical objects in an office.

User Interface. HP Distributed Smalltalk uses and provides support for a user interface based on an office metaphor which is designed for easy use and understanding. In the default user interface, all the objects a user works with locally (folders, file cabinets, documents, and so on) are contained in an office. All offices on the same system are in the same building. Users can navigate between buildings to access objects in other offices. Fig. 9 shows a typical office and some of the objects available in an office.

Sample Applications. Sample applications illustrate the use of distributed objects. For example, the Forum (Fig. 10) provides a shared window in which several users can view and annotate a picture or document. The Notebook is a place to store both local and remote objects on a desktop.

Users can also build their own objects from any of the simple objects available, including a table, chart, input field, picture, and text window (see Fig. 11). The sample applications can be extended and customized to create a variety of simple distributed applications.

Creating Applications

HP Distributed Smalltalk allows VisualWorks programmers to create distributed applications quickly and easily. Building on the benefits of Smalltalk and VisualWorks, HP Distributed Smalltalk users can build CORBA-compliant applications either from scratch or by modifying existing applications. Like any Smalltalk application, the distributed development process is iterative and designed for dynamic refinement.

Development. Distributed application development is a four-step process.

1. Design and test the application objects locally.

2. Define the object interfaces and register them in the interface repository.

3. Use HP Distributed Smalltalk's simulated remote testing tools (which actually use the ORB to marshall and unmarshall object requests) to verify the interfaces specified in the interface repository.

4. Track messages and tune performance.

Distribution. Once an application is developed, tested, and tuned locally, it is easy to set it up for distributed use.

5. Copy the application classes to the Smalltalk images they will run on.

6. Update the interface repositories in these images.

The application can then run in the fully distributed environment without further change. Except for actual packet transfer, the distributed application is identical to the simulated remote application developed, tuned, and tested during development.

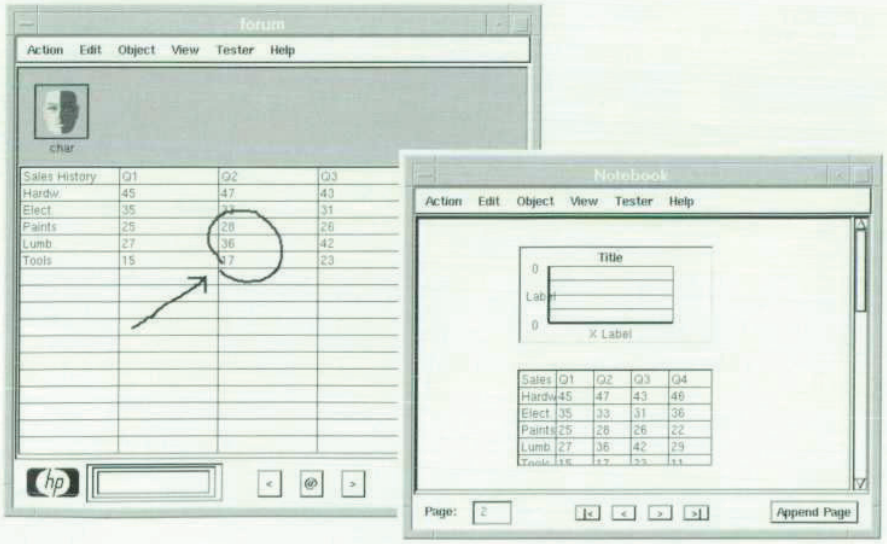


Fig. 10. User interfaces to the sample applications Forum and Notebook.

© Copr. 1949-1998 Hewlett-Packard Co.

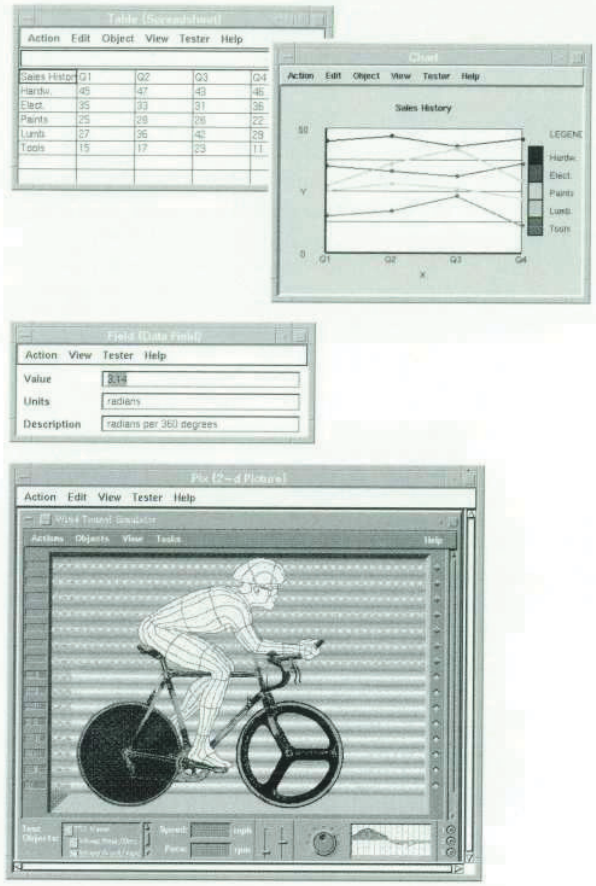


Fig. 11. Sample objects provided with HP Distributed Smalltalk.

Delivery. Once the application is tested, developers can deliver it to their users by stripping the environment of unneeded objects and tools. Once stripped, the application looks exactly the same as applications developed in other languages and can be executed on any supported platform, including: HP-UX,* SunOS/Solaris, IBM AIX, Microsoft® Windows, Microsoft Windows NT, or IBM OS/2. Support for these platforms is available under a run-time license from Hewlett-Packard.

Acknowledgements

We would like to thank the program team manager for HP Distributed Smalltalk, William Woo, for his unflagging support and encouragement. Dr. Jeff Eastman deserves immense credit for designing and building the first implementation of the project. We would also like to thank the HP Distributed Smalltalk program team in Cupertino, California and Fort Collins, Colorado for all their contributions: Lynn Abbott, Jerry Boortz, Jim Borchert, Kevin Chesney, Jürgen Failenschmid, Warren Greving, Robert Larson, Lynn Rowley, Terry Rudd, and Brent Wilkins.

HP-UX is based on and is compatible with Novell's UNIX® operating system. It also complies with X/Open's* XPG4, POSIX 1003.1, 1003.2, FIPS 151-1, and SVID2 interface specifications.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open is a trademark of X/Open Company Limited in the UK and other countries.

Microsoft is a U.S. registered trademark of Microsoft Corporation.

Windows is a U.S. trademark of Microsoft Corporation.

© Copr. 1949-1998 Hewlett-Packard Co.

A Software Solution Broker for Technical Consultants

A distributed client-server system gives HP's worldwide technical consultants easy access to the latest HP and non-HP software products and tools for customer demonstrations and prototyping.

by Manny Yousefi, Adel Ghoneimy, and Wulf Rehder

On a typical working day an HP consultant, one of thousands worldwide, sits down with a customer to solve a business problem. The challenge, the customer may tell the consultant, is to move sales data to headquarters more quickly so that management can make timely strategic decisions. For a solution, the consultant might propose a decision support system that integrates the customer's older legacy system where the sales data has been stored traditionally with a faster "warehouse" database and easy access tools that present the information in just the form needed, right on the customer's desktop. "Let me show you what I mean," the consultant says, turning on a laptop computer (which had previously been connected to a LAN or telephone socket). Navigating through the windows on the screen, the consultant invites the customer to look through a virtual shelf filled with databases and access tools, all represented by icons, together with middleware and application development toolkits (see page 98). The consultant clicks on an icon and the tool becomes immediately available for browsing or for self-paced learning. From here the consultant may show one of the demos that are included, or navigate the customer through a hypertext document to more information, alternate products, additional options, and prefabricated software building blocks. No wonder that this virtual software laboratory is called by HP consultants, "the software sandbox." This consultant is actually building—from the tool and product portfolio in front of them—a prototypical decision support system for this customer. How much of this is fantasy and how much reality?

The answer is that it is all reality now. The software sandbox that the consultant was starting to "play in" is called the HP Software Solution Broker (or Broker, for short) and is available now to HP consultants. Defining and creating a decision support system is, of course, not play but serious work. However, the ease and immediacy of the Broker, the ample choices, and many helpful hints make even urgent business problem solving an experimental sport. Best of all, the consultant receives these products and tools, together with support and on-line documentation, free of charge. For this convenience, substantial research efforts had to be poured into building such a virtual software depot, using HP's own hardware platform and the most advanced object technology. Before explaining this implementation more systematically, it is useful to watch our technical consultant and the customer at work.

Using the Software Solution Broker

To get a feeling for how the Software Solution Broker is used we will briefly watch the technical consultant show the customer how to build a prototypical decision support system.

After clicking on the icon in the ORB control panel, which starts the object request broker (an action that in effect opens the lid covering the sandbox), the consultant activates the Software Solution Broker icon. Another window opens offering the Broker's classification of products, either by vendor, by technology, or by product name. (Alternative paths into the Software Solution Broker, such as a classification by business problem, are under development.) Choosing the **i**(nformation request) button for technology, the consultant asks whether the customer wants to see database information first or options for the user interface. As an executive, the customer is eager to see or build a nice GUI. Clicking on the graphic user interface **i** button brings up several choices of which three are shown in Fig. 1. Having heard about VisualWorks the customer selects it and is presented with the VisualWorks Showcase.

The consultant then shows a VisualWorks demonstration to explore with the customer what kind of data display windows, control buttons, analysis tools, and other features would be appropriate. After jotting down these initial requirements the consultant is ready to build a first prototype. The help button launches a palette of GUI building tools, and it takes only minutes to draw an example of a transaction entry tool for the transactions underlying the decision support system the customer wants built (see Fig. 2). Here the customer interrupts and requests that the data be shown in spreadsheet form as well as graphically. They agree on bar chart and pie chart presentations for a first cut and proceed to discuss the requirements for the underlying database. The Software Solution Broker has a "virtual shelf" of relational databases that work with VisualWorks, and among these the customer may have a favorite system, or an already installed legacy database. They again discuss the pros and cons while viewing various product demonstrations.

We meet the customer and the consultant again after another hour or so. By then the VisualWorks front-end tool displays some real data pulled from a database (Fig. 3). At this point we leave the executive's office and describe how the Software Solution Broker is constructed.

© Copr. 1949-1998 Hewlett-Packard Co.

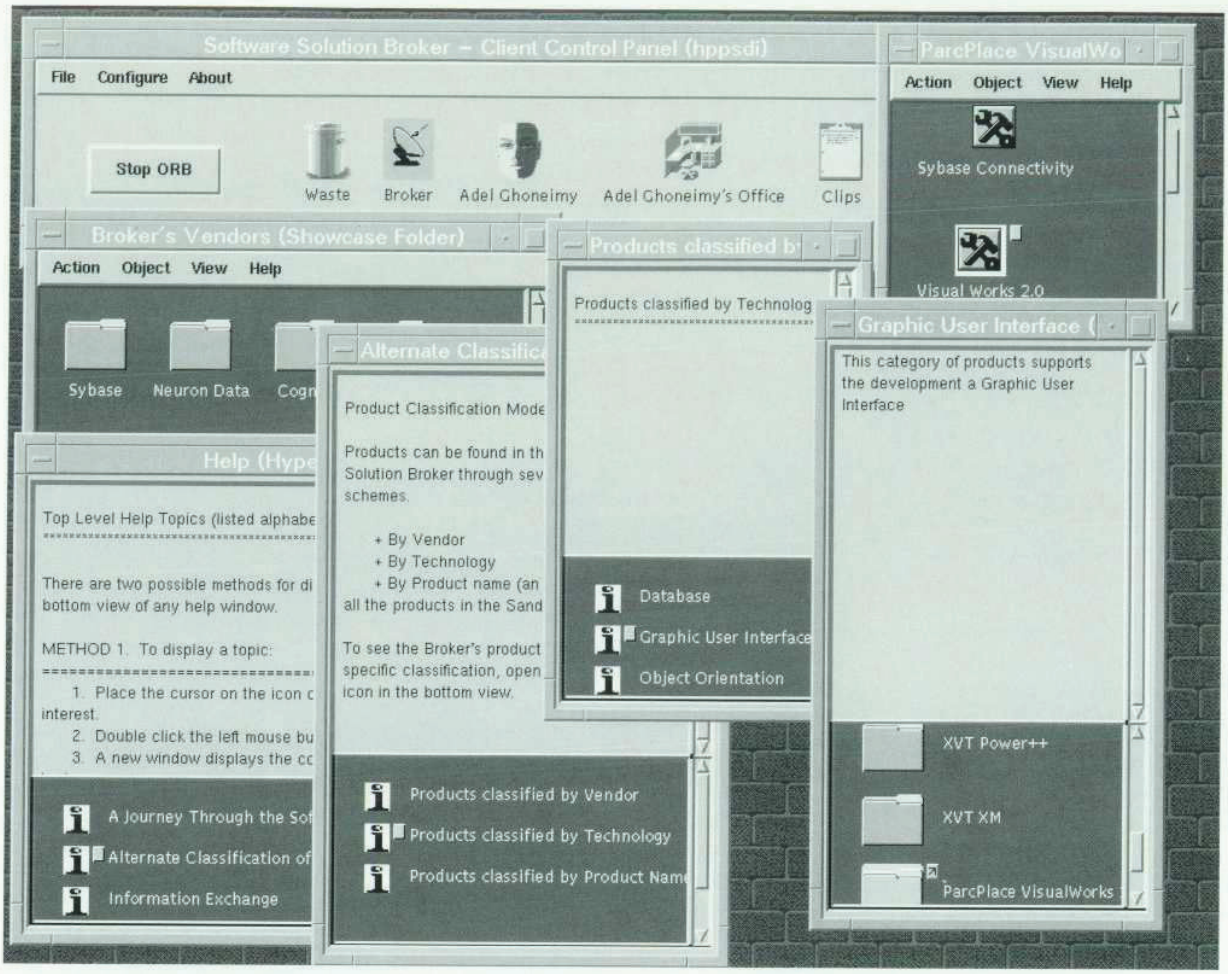


Fig. 1. Software Solution Broker user interface.

Constructing the Software Solution Broker

Two considerations determined the architecture and consequently the implementation of the Software Solution Broker. First, since the products on the Broker have to be accessible worldwide but will be updated and maintained locally, the global partitioning between distributed users and a central server functionality called for a client/server implementation on a wide area network (WAN). Secondly, the need to accommodate many different types of clients and to be able to encapsulate many different products in the software server strongly suggested vendor independence (openness) and adherence to certain industry standards such as the Common Object Request Broker Architecture (CORBA).

Software Substrate

Here we will not focus on the WAN implementation but instead will concentrate on the software substrate on which the Software Solution Broker is built. In the software substrate (see Fig. 4) we include the entire software kit composed of server and client development tools, tools for building the client/server interaction components of the system, and repository tools. Repository tools are essential for the construction of a depot that contains the information in the system, including the logic for accessing this information. After a careful technical analysis of five alternative complete substrate kits, VisualWorks from ParcPlace Systems was chosen as the development software for the PC, UNIX® client, and UNIX server, while HP's Distributed Smalltalk (see article, page 85), which also works with VisualWorks, was the tool of choice to build and manage the client/server interaction. All system information (e.g., documentation) at this time of writing (release 2.0) still resides with the products and a central repository has not yet been chosen. Tools such as Object Lens (working with VisualWorks) or HP Odapter make relational databases look like object databases, so we know that the selection of a repository can be made very quickly when needed.

VisualWorks was the easy winner because it provides a complete environment for the development of true graphic applications that run unchanged on UNIX-system-based, PC, and Macintosh computers under their native windowing systems. Three of VisualWorks' features made it especially appropriate for the Software Solution Broker:

- VisualWorks is built on Smalltalk, a pure object-oriented language designed for fast modular design.
- VisualWorks possesses a tested set of development tools, including browsers for object classes, a thread-safe debugger, and a change manager to track modifications to the code, as well as an inspector for use in testing.

© Copr. 1949-1998 Hewlett-Packard Co.

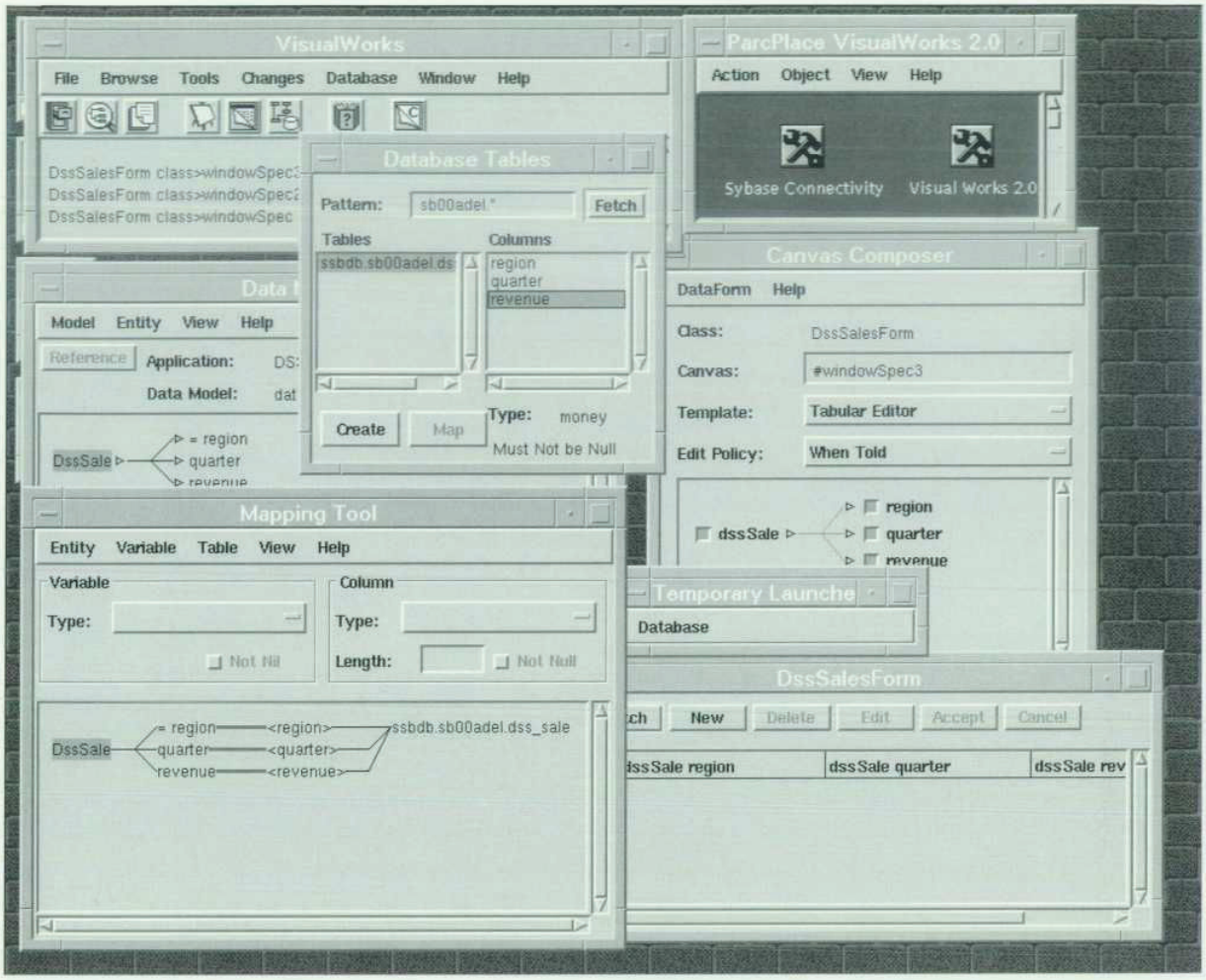


Fig. 2. A window within the Software Solution Broker showing VisualWorks tools for prototyping a customer application.

- VisualWorks has a large class library of more than 350 types of portable objects. These include a rich user interface development toolkit suitable for all major windowing systems.

HP Distributed Smalltalk extends VisualWorks' capability for developing standalone systems into an environment for creating distributed object systems (see Fig. 5) by adding the following:

- A full implementation of the Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) core services
- Common Object Services for life cycle operations such as creating objects and the relationships between them
- Sample application objects, for example for the modular partitioning of client/server functionality into semantic and presentation objects.

These objects and services for building distributed applications are portable to all platforms supported by VisualWorks. Furthermore, they are compatible with the OMG CORBA standards. HP's Distributed Smalltalk provides seamless support of client/server interactions between VisualWorks images. CORBA compliance makes our Software Solution Broker implementation open and capable of interoperability, for instance with C++ CORBA-compliant applications, and as soon as HP Distributed Smalltalk is OSF DCE-compliant,* also with DCE remote procedure calls (RPCs). For the current release, TCP/IP or HP Sockets are being used.

* OSF DCE is the Open Software Foundation's Distributed Computing Environment.

Product Encapsulation

Everybody who has worked with spreadsheets, word processors, or CAD systems knows that similar or identical functionality does not mean that the user interfaces and more generally the visual, iconic, and mental models are comparable. For the Software Solution Broker, too, each product has its own artifacts and idiosyncrasies, its own look and personality by which we can identify it when we see it in use or on the shelf of a vendor. This unavoidable fact poses challenges for the "virtual shelf" of the Broker. Without wanting to blot out the individuality of a vendor's offering it was the objective of the development team to minimize the effort needed for the user to get accustomed to this diversity. Generally speaking, the variety has to be hidden behind a simple and consistent, product independent mode of access with uniform and intuitive graphical symbolism. A particular example is the double click used consistently to launch an application.

© Copr. 1949-1998 Hewlett-Packard Co.

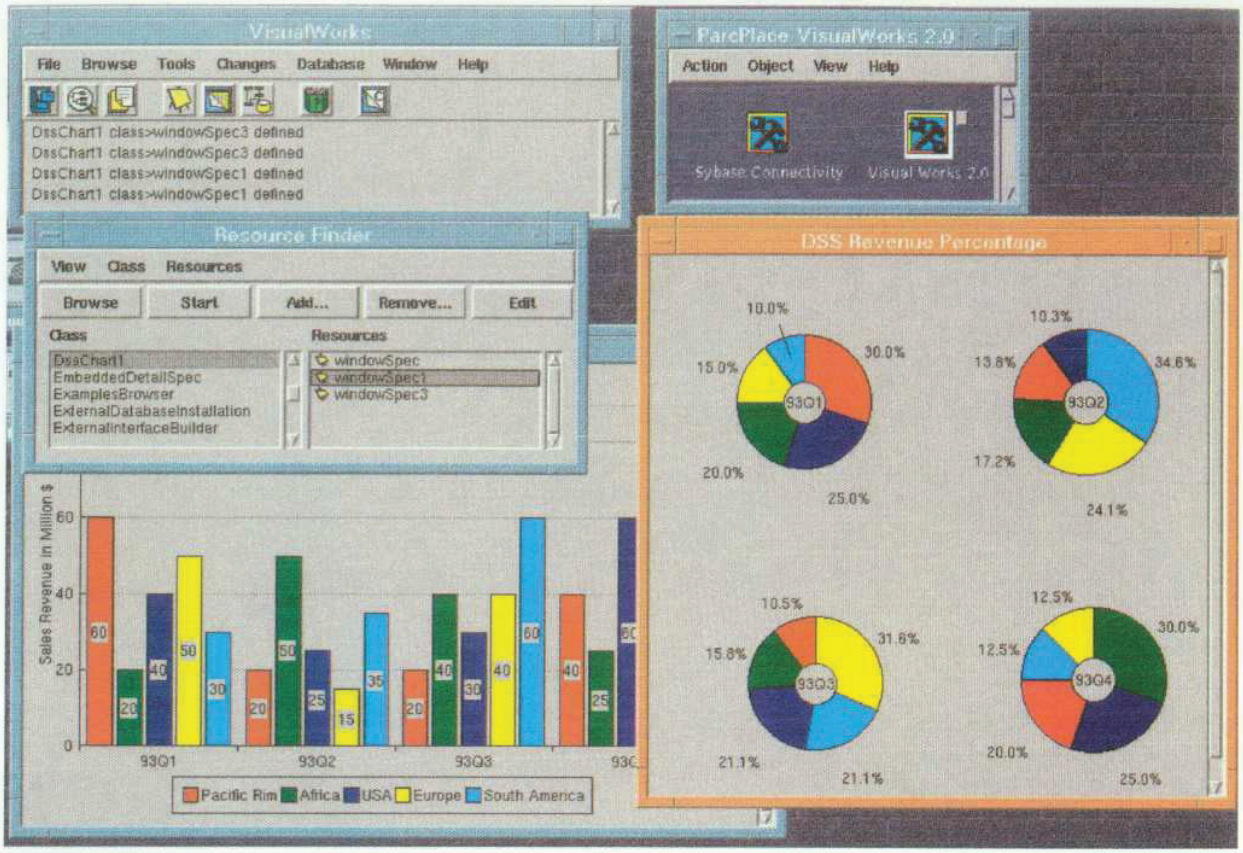


Fig. 3. Prototype display for a customer application constructed using the Software Solution Broker to select user interface and database software.

Encapsulation, in the context of the Software Solution Broker, describes a body of activities and software mechanisms that have two purposes: to integrate each product within the overall product portfolio so that the consultant can use it in its native mode, and to provide a uniform way to access the products, their associated tools, and other artifacts. This accessibility, it should be noted, is restricted to the features and artifacts that are relevant to consulting work with the customer. This means the consultant can access editors, executable code, and documentation, but isn't able to change the internal product configuration, the way it is stored and administered in folders, or the source code. Because of the intrinsic symmetry between Software Solution Broker servers and clients (see Fig. 4) the encapsulation can be done either on the server side or on the client side, provided the

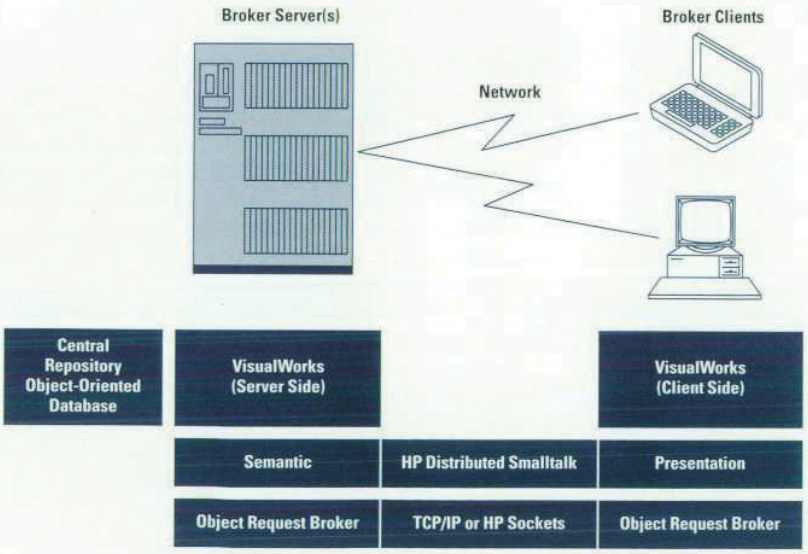


Fig. 4. Software Solution Broker software substrate, showing the client/server architecture, the user interface engine (VisualWorks), and the client/server framework (HP Distributed Smalltalk).

© Copr. 1949-1998 Hewlett-Packard Co.

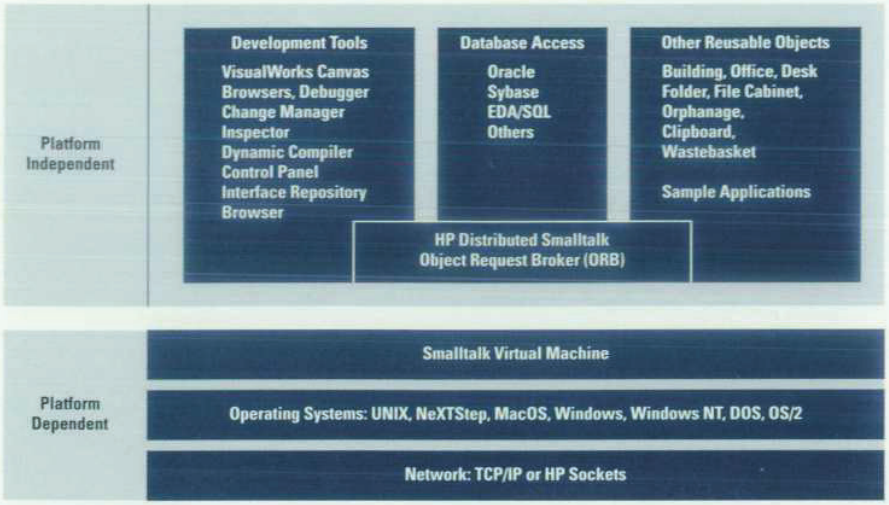


Fig. 5. HP Distributed Smalltalk and VisualWorks provide a full development and run-time environment for distributed computing.

classes FolderPlusPO and EncapsulationDialog are present in the client. These two classes will be discussed below.

It is the already mentioned semantic/presentation split, together with object-oriented features such as *inheritance* and *polymorphism* that make the encapsulation effortless. The semantic/presentation object distribution model is HP Distributed Smalltalk's implementation of a distributed client/server architecture. In this model, classes always appear in logical pairs, one representing the server semantics, the other their presentation in the client. Consequently, the class instances or objects also come in pairs. Take for instance the window object. Every window is composed of two logical parts: its shared (semantic) properties such as its rectangular shape, and its local and personal (presentation) attributes such as color. In general, a semantic object often has (and controls) many different presentation objects, which in the case of the Software Solution Broker handle the remote user interactions, thus reducing network traffic. For instance, one semantic data display object creates and controls different presentations of the data as a bar chart and a pie chart in a decision support system. HP Distributed Smalltalk allows various modes of collaboration between the semantic and presentation objects, including messages that are handled by the object request broker. (For a simple but complete example see the *HP Distributed Smalltalk User's Guide*, chapter 10.)

After this abstract introduction of HP Distributed Smalltalk's semantic/presentation split architecture we will describe in more concrete terms how it works for the encapsulation procedure. As stated above, encapsulation must achieve two goals: it has to present a graphical representation of the artifact (product, tools, demos, documentation) in its native mode to the remote client, and it must allow the remote user to launch the artifact at the server side through this representation. HP Distributed Smalltalk has a pair of classes, MediaSO and MediaPO, that accomplish exactly this. (The suffixes SO and PO imply that semantic and presentation objects, respectively, are spawned by these classes). Tracing the interaction diagram between two objects of these classes we found that there exists a ready-made method called updatePresenter, visible in the MediaSO class, that creates the remote presentation object of a product or other artifact in the server. To customize the generic MediaSO and MediaPO classes and the method updatePresenter for the encapsulation of specific artifacts we first created the narrower subclasses ArtifactSO and ArtifactPO. Then we augmented ArtifactSO with the attributes of artifacts such as vendor and product names. Finally, using overloading, we extended the method updatePresenter to include, among several other administrative tasks, the crucial behavior required for launching the artifacts while exporting their display to the client platform.

Concurrent with this architectural design of the classes and methods that bring about encapsulation in the Software Solution Broker, a few product dependent steps must also be taken. This is done at the instance or object level of every concrete artifact (such as a product) so that it will behave in its expected, native mode. This is a simple matter of inserting the right environment variables and parameters in an encapsulation dialog window. The required information can easily be gleaned from the installation manual of the particular product that is being encapsulated. Finally, products, tools, and other components are put into folders and the encapsulation is done.

Use of Object Technology

The design and building of the Software Solution Broker were characterized by a short development time, a minimal amount of new coding, and a high degree of reuse. The major reason is the application of object-oriented technology. The object-oriented use is pervasive throughout the design, as indicated above, but it is helpful to point to specific examples. We'll give two examples for the object-oriented features inheritance and polymorphism in the context of encapsulation.

One of the examples has just been described: the subclass ArtifactSO of the class MediaSO inherited the method updatePresenter, which in turn, through the feature of polymorphism, was overloaded (that is, extended to include additional functional behavior).

The encapsulation dialog window provides another example. As an administrative tool, it is not available to the user. It is an object built from a subclass of the existing HP Distributed Smalltalk class called SimpleDialog. From this class, the window inherits characteristics such as its property to pop up in front of other windows (it's not obscured), its basic layout

© Copr. 1949-1998 Hewlett-Packard Co.

HP Software Solution Broker Accessible Products

Vendors

- Cognos Corp.
- ParcPlace System
- XVT Software
- Itasca System
- Informix
- Neuron Data
- Sybase
- Unison Software
- ProtoSoft
- Oracle
- Dynasty
- NetLabs

Tools

HP-UX*

- Cognos Corp.
  - PowerHouse 4GL 7.23
- ParcPlace System
  - VisualWorks 2.0
  - VisualWorks with Sybase connectivity
- XVT Software
  - XVT-Design (C Developer Kits)
  - XVT/XM (C Developer Kits)
  - XVT-Power++
  - XVT/XM (C ++ Developer Kits)
  - XVT-PowerObject Pak I
- Itasca System
  - ODBMS Server
  - Developer Tool Suite
  - C Interface
  - Lisp Interface
  - API Libraries (C++, CLOS, Ada)
- Informix
  - Informix Online R4GL
  - Informix WingZ
  - Informix SE R4GL
  - Informix SE ISQL
  - Informix Hyperscript Tools
  - Informix Online ISQL
- Neuron Data
  - Smart Elements (Nexpert object)
  - Smart Elements (Openedit)
  - Open Interface Elements (Open edit)
  - CS Elements (Openedit)
- Sybase
  - SA Companion (client & server)
  - SQL Monitor (client & server)
  - SQL Debugger inspector
  - SQL Debugger console
  - SQL Data Workbench
  - SQL APT Edit
  - SQR Workbench (Easy SQR)
  - Open Client/Server
  - ISQL/SQL Server
- Unison Software
  - Maestro
  - Load Balancer
  - Express
  - RoadRunner
- ProtoSoft
  - Paradigm Plus
- Oracle
- NetLabs
  - Net Labs/AssetManager
  - NetLabs/Vision
  - NetLabs/Assist
  - NetLabs/NerveCenter
  - NetLabs/Manager
  - NetLabs/OverLord Manager
  - NetLabs/Discovery

MS Windows

- Cognos Corp.
  - PowerHouse Windows 1.2E
  - Axiant
  - Impromptu
  - PowerPlay
- ParcPlace System
  - VisualWorks 2.0
  - VisualWorks with Sybase connectivity
- XVT Software
  - XVT-Design (C Developer Kits)
  - XVT/Win (C Developer Kits)
  - XVT-Power++
  - XVT/Win (C ++ Developer Kits)
  - XVT-PowerObject Pak I for MS Windows
- Itasca System
  - ODBMS Server
  - API Libraries (C++)
- Informix
  - New Era
- Neuron Data
  - Smart Elements (Nexpert object)
  - Smart Elements (Openedit)
  - Open Interface Elements (Open edit)
  - CS Elements (Openedit)
- Sybase
  - Net-Library
  - Open Client /C
  - SQL Monitor Client
  - SQR Workbench
  - APT Execute
- ProtoSoft
  - Paradigm Plus
- Oracle
- Dynasty Technologies
  - Dynasty
- NetLabs
  - NetLabs/Vision DeskTop

Artifacts

- Cognos Corporation
  - QUICK Application
  - QUIZ Application
  - PDL Application
  - QDESIGN Application
  - QTP Application
  - QUTIL Application
  - PDL And Utilities Reference Manual
  - PowerHouse for UNIX – Primer
- ParcPlace System
  - Product Overview
- XVT Software
  - Product Overview
  - XVT Design Tutorial
  - XVT Database Demo
  - XVT-Power++ Overview
  - XVT-Power++ Demo Guide
  - XVT Power ++ Earth Demo
- Itasca System
- Informix
  - Product Overview
  - Informix R4GL Demo
  - Six demos with source codes
  - Informix ISQL Demo
  - Informix Hyperscript Demo
- Neuron Data
  - Product Overview
  - Notepad Widget example with source files
  - Pack example with source files
  - Print widget example with source files
  - Resize widget example with source files
  - Resource Picker example with source files
  - Scripting example with source files
  - Scroll area usage example with source files
  - Scroll bar usage with source files
  - Sliders usage with source files
  - Special widget example with source files
  - String search example with source files
  - Text edit validation example with source files
  - Windows MD example with source files
  - Alert Windows example with source files
  - Browsex example with source files
  - Browsinc example with source files
  - Cbox example with source files
  - Chart example with source files
  - Clock Widget example with source files
  - C++ Notepad widget example with source files
  - Drag drop example with source files
  - Draw example with source files
  - DropDown pale example with source files
  - File manager example with source files
  - File name translator example with source files
  - File Picker example with source files
  - Floating window example with source files
  - Cantt chart example with source files
  - Help engine example with source files
  - Help viewer example with source files
  - ICON generator example with source files
  - List Box example with source files
  - Local drag drop example with source files
  - Menu example with source files
  - Multiple font text example and source code
  - Notepad example with source files
- Sybase
  - SyBooks
  - APT Demo
  - Compute example with source files
  - Csr_disp example with source files
  - i18n example with source files
  - blktxt example with source files
  - Five other examples with source files
- Unison Software
- ProtoSoft
- Oracle
- Dynasty
- NetLabs

HP-UX is based on and is compatible with Novell's UNIX® operating system. It also complies with X/Open's* XPG4, POSIX 1003.1, 1003.2, FIPS 151-1, and SVID2 interface specifications.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open is a trademark of X/Open Company Limited in the UK and other countries.

MS Windows is a U.S. trademark of Microsoft Corporation.

© Copr. 1949-1998 Hewlett-Packard Co.

with a text box and O.K. and cancel buttons, and its link to a value holder that holds the environmental variables, names, and other information needed for the encapsulation. The only method needed in addition to inherited ones is the one requesting the encapsulation parameters mentioned above.

The same procedure, that is, the use of predefined classes and thus minimal coding, applies to HP Distributed Smalltalk's folders containing the encapsulated product with its tools and other artifacts. The HP Distributed Smalltalk class FolderPO (PO indicates it is the folder class spawning presentation objects) has a method windowMenu, which creates a window with several pop-up menus that have labels such as Action, Edit, and so on. For a subclass of FolderPO called FolderPlusPO, these properties of windowMenu are inherited, but windowMenu is also changed (while keeping the same name), by the addition of a method artifactCreate and its label in one of the pop-up menus of windowMenu. The method artifactCreate is responsible for the inner workings of the encapsulation dialog window mentioned above.

Development Methodology

Funding for the Software Solution Broker project was subject to the condition that the development team find, justify, and implement a design that brings the tools to the consultants in the fastest possible way with the least amount of resources, including development, maintenance, and support resources. At the same time, every released version, even the very first one, had to find immediate user acceptance. Based on these stipulations the team chose a development method that is a hybrid of *iterative prototyping* and the *Fusion method.*

Our reasons for favoring iterative prototyping over a classical software design paradigm that starts with a complete specification (such as the so-called *waterfall model*) were:

- Time constraints. There are never enough engineer-months to write a complete specification, implement and test it into production strength.[1]
- Constraints imposed by the intrinsic nature of the Software Solution Broker tool we were building, that is:
  - Client-side usability. The GUI that was eventually chosen is the result of repeated testing by potential users to achieve maximum ease of use and intuitiveness, and this amount of trial-and-error cannot be specified in advance.
  - Tool accessibility. The different products on the virtual shelf have different behaviors and their own requirements for resources and administration, and creating the encapsulation process again requires much experimentation and gradual maturation based on experience that cannot be specified a priori.
  - Using the object paradigm. The software substrate chosen (HP Distributed Smalltalk with VisualWorks) is well-suited for the rapid development of GUI and client/server applications.

Based on these considerations, our overall approach was that of evolutionary prototyping, in which a fully functional prototype is ushered through repeated refinement steps into a production-strength end product. We realize that often a prototype leads only to an executable specification or a validated model, not a high-quality, stable product. However, in our case the sophisticated framework of HP Distributed Smalltalk with its semantics/presentation split and VisualWorks with its Model View Controller ensured full functionality and high quality at each refinement step because we reused the existing, high-quality code (including the library of classes) and very sparingly added new, thoroughly tested code, preferably as instances (objects) of the existing class library.

Fusion Method

While iterative prototyping can be seen as a software development philosophy that is primarily dictated by business requirements such as time to market, break-even time, or optimal return on investment, the Fusion method[2] was developed with the goal of creating a language independent, comprehensive, software project management method. Being a systematic object-oriented development method, it blends well with our software substrate, which we chose based on openness, compliance with industry standards, ease of use, and the ability to separate the server (semantics) from the remote clients (presentation). The Fusion method emphasizes a modular design process in clearly demarcated phases, so it synchronizes well with the iterative prototyping approach, which requires the repetition and refinement of certain development stages without impacting others. Furthermore, the Fusion method insists that a software development process of the complexity encountered today must cover the entire software development life cycle. The Fusion method's phased development process served as the blueprint for the Software Solution Broker. It can be summarized as follows[2] (our italics):

Starting from a *requirements document*, the *analysis phase* produces a set of models that provide a declarative description of the required system behavior. The analysis models provide high-level constraints from which the design models are developed. The *design phase* produces a set of models that realize the system behavior as a collection of interacting objects. The *implementation phase* shows how to map the design models onto implementation-language constructs.

In our hybrid approach we take an early, loosely defined functional prototype as our initial requirements definition (an executable specification), to be modified and refined in subsequent iterations through the three phases of analysis, design, and implementation. After each of these phases a review of the phase outputs is conducted by the development team in conjunction with users. The results of this audit are prioritized and, if deemed important, incorporated into the prototype which, through several of such review loops, evolves after a full cycle into the production product. (For details about the outputs mentioned and the complete Fusion process breakdown see reference 2, especially Appendix A.)

In summary, the two complementary methods of iterative prototyping and Fusion serve two main purposes. First, at the end of each prototyping cycle a fully functional production-strength product is released. Second, the three Fusion phases—analysis, design, and implementation—of every cycle are independent of the phases in another cycle. Therefore, we are in effect working towards several releases at the same time (see Fig. 6).

© Copr. 1949-1998 Hewlett-Packard Co.

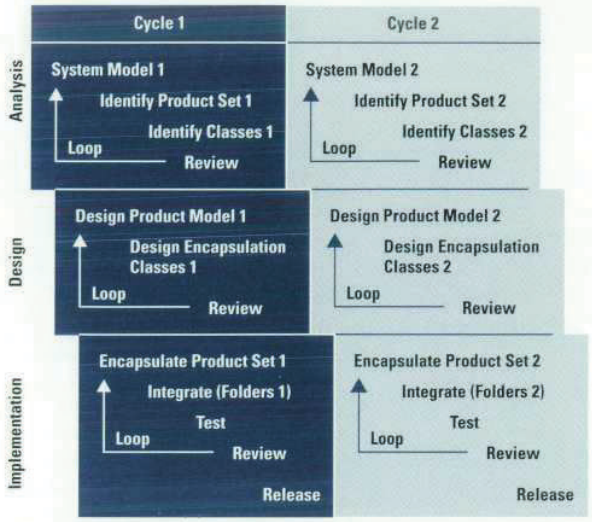


Fig. 6. Software Solution Broker development used iterative prototyping and the Fusion method, resulting in parallel development cycles.

Customizing the Software Solution Broker

In addition to being a productivity tool and a hub of product expertise for HP's technical consultants, the Broker can be customized to meet the business needs of end customers as well. To sketch how such a customization can be done using the object-oriented framework of HP Distributed Smalltalk, imagine a vendor of CAD (computer-aided design) software. Rather than offering shrink-wrapped software packages on the shelves of the store the retailer wants to offer customers an environment where they can, by navigating through virtual shelves, choose interesting products and "test drive" them in the store before deciding what to buy.

For an end customer such as the CAD software vendor, the Broker can be customized by mapping the particular customer requirements into several levels of design complexity. These levels describe in technical terms what level of intervention into the framework of HP Distributed Smalltalk is needed to alter and customize the existing classes and methods. On the lowest level, the requirements fit the HP Distributed Smalltalk framework exactly, and the system can be built from existing classes without change. A higher level of intervention would be needed to construct the Software Solution Broker for the CAD software vendor. Slight modifications of core services (relating to containment and life cycle semantics), in addition to class augmentation and overloading of methods, would be recommended. Working with predefined, well-documented levels of intervention that are necessary to meet a customer's requirements has the advantage of communicating to the customer in advance, during the analysis and before system design begins, how much reuse of the framework is possible, and how much nonframework augmentation is necessary. Intervention levels are thus not only technical assessments but also indicators of the final costs for the system.

Conclusion

The Software Solution Broker was not a typical client/server application development project. We were not primarily concerned about two-tier or three-tier architectures, about objects per se, about the one "right" programming language, or about coding. In fact, we went the opposite route. Based on the working requirements of HP's technical consultants and our own analysis of how consultants work with customers, we resolved to translate these requirements into a system built from distributed objects. The building, however, consisted mainly in the skillful choice of existing classes and the exploitation of HP Distributed Smalltalk's framework. The novelty in our approach lies not in the coding of new structures, but in the extensive application of reuse. In fact, whenever new code seemed required, we took it as a warning that further analysis was needed to look for prefabricated code within the framework of HP Distributed Smalltalk. This simple principle, essential for a fast time to market, also guaranteed a short turnaround time and high quality.

Through its first two releases, 1.0 and 2.0, the Software Solution Broker can be viewed as a distributed productivity tool offering three overlapping types of services. These three types can be described metaphorically as a virtual software shop for the display of individual products, a consultative workbench or simulated classroom for studying and experimenting with several collaborating products, and a virtual demo center with remote satellite offices where the technical consultants can build prototypes and create demos for a customer. Looked at from a broader perspective, however, the Software Solution Broker architecture and implementation are, with small customization, also ideal for other, related applications that require one (or a few) persistent centers and many locally distributed and individually presented clients. One example is software distribution. Another is the establishment of a worldwide software application development lab where each satellite group can develop its own part locally, check it in with a central repository where it is available to the other satellites, and participate remotely in the integration of the parts into a system. Furthermore, object technology, with its concept of containers, makes available compound documents (text, picture, voice, video, etc.) that can be employed also on the nontechnical side of business as vehicles for elaborate project proposals and other communication with business customers—for instance, to propose a solution by showing a video of a prior, successful installation (this would take the place of a paper document of reference sites). In this role, the Software Solution Broker can be a *worldwide business solutions exhibit* and a convenient repository for a portfolio of repeatable solutions from which the customer, advised by a consultant, can select products the way we now choose from mail-order catalogs.

Acknowledgments

A project survives by the benevolent patience of its sponsors, the enthusiasm of the project team, and the acceptance of the receiving customers. In the case of the Software Solution Broker we've tried to justify this benevolence by turning our research ideas into a useful tool in the shortest possible time.

© Copr. 1949-1998 Hewlett-Packard Co.

This feat was only possible with the support of our managers David Kirkland and Sherry Harvey. Special thanks are due Chu Chang, general manager of the Professional Services Division, for his encouragement. We are also very grateful for the contributions and for the heated (but object-ive) discussions with friends and partners from many HP's entities. Nothing, however, would have happened without the quick and thorough feedback from our customers, the technical consultants in the field. The Software Solution Broker is dedicated to them.

References

1. F.P. Brooks, *The Mythical Man-Month: Essays in Software Engineering*, Yourdon Press, Englewood Cliffs, 1982.

2. D. Coleman, et al, *Object-Oriented Development—The Fusion Method*, Prentice Hall, 1994.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open* Company Limited.

X/Open is a trademark of X/Open Company Limited in the UK and other countries.

© Copr. 1949-1998 Hewlett-Packard Co.

Bugs in Black and White: Imaging IC Logic Levels with Voltage Contrast

Voltage contrast imaging allows visual tracking of logical level problems to their source on operating integrated circuits, using a scanning electron microscope. This paper presents an overview of voltage contrast and the methods developed to image the failure of dynamic circuits in the floating-point coprocessor circuitry of the HP PA 7100LC processor chip.

by Jack D. Benzel

As pressure for higher performance and higher integration drives integrated circuit design towards increasing complexity, IC designers need an ever-broadening set of analysis and debugging tools and methodologies for tracking down functional bugs and electrical margin issues in their designs.

In developing the new HP PA 7100LC PA-RISC microprocessor chip, the floating-point arithmetic logic unit (FPALU) megacell used design techniques based on the PA 7100 design.[1] The FPALU design is implemented with mostly mousetrap-style dynamic logic[2] with significant use of single-ended dynamic logic in the last pipeline stage.

Past experience in debugging electrical problems in mousetrap designs has shown these problems to be very difficult to find.[3] A failure mechanism that emerged in prototypes of gate-biased PA 7100LC FPALUs proved highly challenging and evasive and required a large engineering effort to get from detection to the root cause identification. The voltage contrast imaging methodology proved useful in analyzing and later confirming the root cause of the failure mechanism. Results from the analysis allowed us to correct the design and verify its quality.

The Wall

The FPALU failure mechanism was named "the wall" because of its appearance on a frequency-versus-voltage shmoo plot depicting regions of passing and failing vectors (see Fig. 1).

Considerable engineering resources were applied toward finding the root cause of the wall using many of the techniques that had proved successful on previous design projects, including but not limited to shmoo plots, failing vector/opcode analysis, clock phase stretching, focused ion beam (FIB) experiments, and simulations of probable circuit failures.[3] These techniques were not providing enough information, and a new methodology was clearly needed.

Why Voltage Contrast?

Another HP design team had recently had success in using an electron-beam prober[4] to track down the root cause of a noise problem on the same CPU chip.

Previous experience with another project several years ago provided insights into a methodology similar to electron-beam probing called voltage contrast, using a scanning electron microscope (SEM). After considering the various trade-offs it was decided to proceed with the voltage contrast imaging while keeping open the option of going to electron-beam probing if further analysis was required.

SEM Fundamentals

The SEM displays objects by sensing and imaging the release of secondary electrons from the surface of a sample which is held in a very high vacuum. A finely focused beam of electrons accelerated from an electron gun with a thousand-volt potential is swept over the surface of the sample in much the same way that a television screen is scanned. As the high-energy electrons in the beam strike the sample, several valence electrons will be "knocked loose" from the sample as the impinging electrons lose energy. These now-free electrons, or secondary electrons, find their way to the surface of the sample and are released from the surface. A highly biased metal screen situated near the sample collects escaping secondary electrons into a detector which generates a signal proportional to the number of electrons collected. The signal from the detector is amplified and displayed on a CRT screen which is scanned in synchronization with the electron beam sweeping the sample.

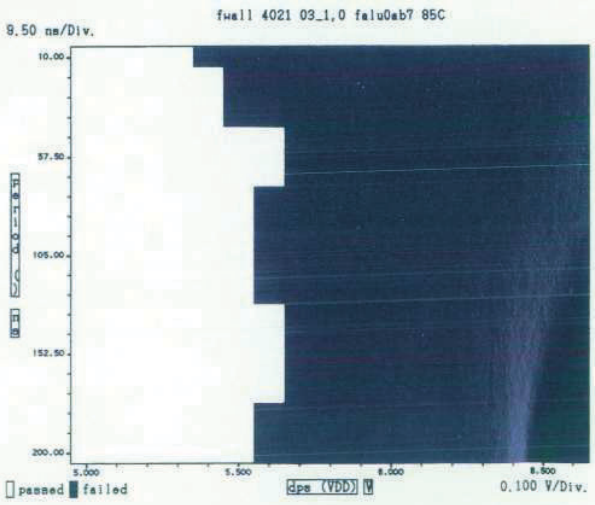


Fig. 1. Shmoo plot of "the wall."

© Copr. 1949-1998 Hewlett-Packard Co.

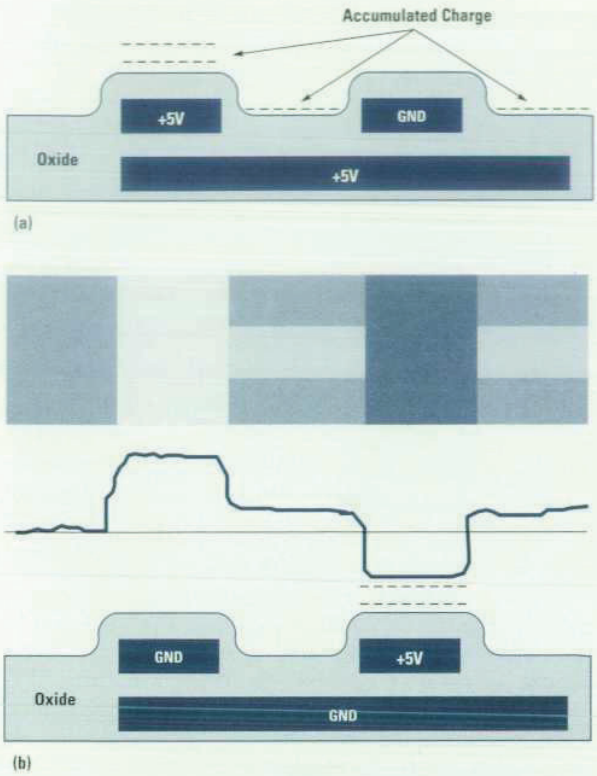


Fig. 2. (a) First pass "charge" of IC surface with secondary electrons. (b) Second pass "read" of charged surface (bottom), resulting video signal (middle), and 2D video image (top).

Voltage Contrast Imaging

Voltage contrast imaging uses the electrical nature of the SEM to view voltage potentials on a sample changing with time. Figs. 2a and 2b show a cross section of the top two metal signal layers of an IC with the metal lines insulated by an oxide.

The imaging is done in two stages: charging and reading. Fig. 2a shows the state of the IC at the end of the charging stage. The positive potential of the buried metal lines attracts and holds the generated secondary electrons on the surface of the oxide above the metal lines. These charges will remain on the surface for long periods of time, basically acting like a capacitor.

Fig. 2b shows the state of the IC at the end of the read stage with the voltage potentials of the metal lines now changed. The resulting detector signal level and the CRT image generated from it are also shown above the cross section. As the electron beam sweeps the surface of the sample, the electrons that were once held by the positive charge of the upper-left and lower metal lines (Fig. 2a) are knocked off the surface and are collected into the detector, generating a bright signal on the CRT. On the other hand, the upper-right metal line is now more positive, and the surface above it will release fewer secondary electrons as the surface capacitively charges, corresponding to a lower number of electrons collected and thus a darker signal on the CRT.

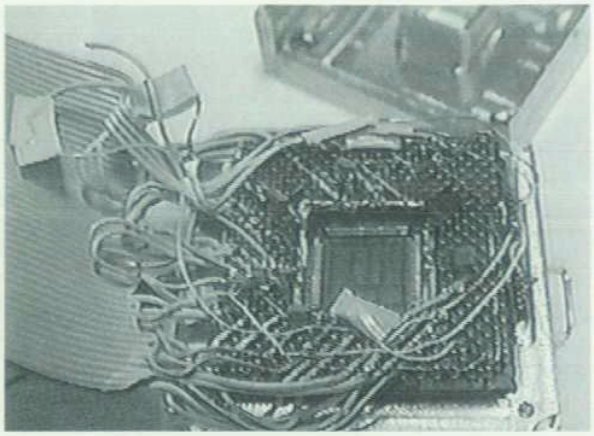


Fig. 3. Video image of DUT fixture for voltage contrast setup with top shield removed.

DUT Preparation

Preparing the IC for the SEM environment required careful attention to several details as follows:

- *Clean Power Environment.* Some previous experiments indicated that the wall was somewhat remedied by a power environment that restricted the V_{DD} current supply. Therefore, careful attention was paid to provide adequate low-inductance power feeds with adequate decoupling capacitance.
- *Simple Vector Stimulus.* Restricted cabling into the SEM chamber and easy portability between two different SEM facilities required a simple method for executing a wall-sensitive floating-point operation (FLOP). A successful method was developed to launch and step through the phases of a FLOP using the JTAG†-conforming serial test port and a serial test board.
- *Image Capture Synchronization.* The capture and imaging of events on the SEM system requires a synchronizing signal generated by the device under test (DUT). Several small surface mount ICs were mounted on the PA 7100LC package to decode the clock signals and derive another synchronizing signal to provide the SEM with an accurate sync pulse that identified the leading clock edge at the starting phase of the failing FLOP.
- *Minimize Outgassing.* To achieve an adequate vacuum in the SEM system, materials that had minimal outgassing were required. This prevented the use of heatshrink tubing and quick-cure epoxies and required careful cleaning of the DUT.
- *Packaging.* The packaging fixture containing the CPU (see Fig. 3) met several requirements. The wall was a high-temperature phenomenon and required heating the part inside of the SEM with large resistors mounted inside the fixture. The metal enclosure shielded all but the die surface from the electron beam, since the beam will positively charge plastics (wiring, capacitors). The shield also prevented electrical signals in the DUT wiring from interfering with the beam's trajectory. The last requirement filled by the fixturing was a compact size to fit inside the small SEM chamber.

† JTAG is the Joint Test Action Group, which developed IEEE standard 1149.1, *IEEE Test Access Port and Boundary-Scan Architecture.*

© Copr. 1949-1998 Hewlett-Packard Co.

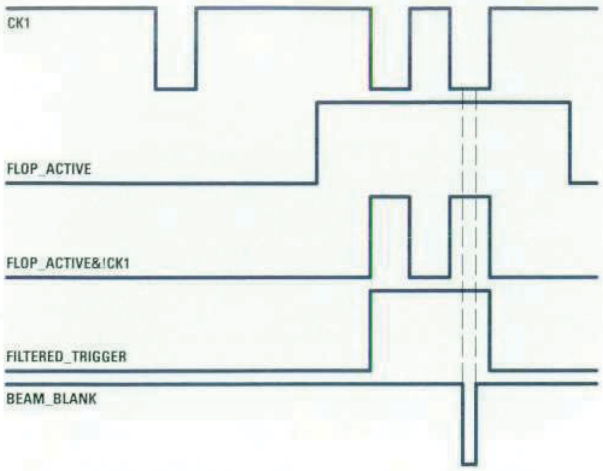


Fig. 4. Beam blanking and synchronization signal generation.

Imaging Dynamic Signals

The electron beam scan is synchronized with the scan of the video display tube and consequently has a slow refresh rate of 1/60 second. This slow refresh rate works well for stationary objects and static electrical signals, but the signals of interest involved in the wall failure are dynamic, typical of mousetrap designs. The imaging of dynamic signals required the development of a new process.

Synchronization and Beam Blanking

The slowest rate at which the DUT could be clocked with reliable operation of the scan path driven through the JTAG port was 2.5 MHz, giving a 200-ns phase or period during which dynamic signals would be active. Connecting a pulse generator to the DUT's sync pulse allowed the generation of a variable-width, variable-delay pulse (see Fig. 4) which was used to blank the electron beam scanning the DUT. Using this blanking signal, the SEM could be controlled to charge or read the IC only during the time of interest when the wall-related signals were active. A 100-ns sample window was chosen for the blank signal, which was centered in the clock phase to reduce possible overlap into adjoining phases.

Once the beam was properly synchronized and blanked, the apparent lack of information in the video image shown in Fig. 5 gave a strong indication that more development was needed.

Image Capture

The next problem to resolve was imaging the brief 100-ns video information successfully. Several ideas were evaluated and tried before an acceptable method was found:

- *Photographic Film Integration.* The SEM focuses the light from a secondary CRT onto the film plane of a Polaroid camera over a period of several minutes while exercising the DUT. This method resulted in either completely black or very indistinct images of the IC.
- *Two-Dimensional Scan.* The SEM can operate with basically a zero-frequency vertical scan rate. This provides an image of a single horizontal slice of the IC surface while improving the refresh rate. Changes in beam intensity were undiscernible in this mode.
- *Two-Dimensional Scan in Oscilloscope Mode.* Using the same two-dimensional scan mode as above, the intensity vector of the SEM's display can be used to drive the vertical component of the video signal. The resulting image is reminiscent of an oscilloscope display showing intensity on the y axis. No discernible changes in intensity were visible in this mode as well.
- *Two-Step Charge/Read.* Instead of trying to charge and read on each or every other FLOP, the process was broken into two steps. The first step involved turning the beam on only during the phase of interest while the part was executing wall FLOPs over a period of three minutes. A long integration time was required because each time the beam turned on it only charged a tiny area of the field of view. At the end of the integration time, the beam was turned off, the IC powered down, and the beam blank removed from the SEM. The IC now had a surface charge that reflected the state of the metal lines during the phase of interest. The second step was to turn the beam on with no blanking to read the surface charge in its first pass over the IC. The resulting video image was clear but brief (one video frame). This process produced an image in which metal lines with a positive voltage were white and metal lines at ground were black. Another small variation in this process was not to power down the part before the read step. The resulting image took a little more thought to interpret because only the metal lines that changed state from the previous step were black or white.
- *Two-Step Charge/Read with VCR Frame Capture.* By adding a VCR to the setup, the resulting video image fed to the CRT could be captured on tape and then freeze-framed for viewing. The purchase of a VCR with a forward and reverse single-frame jog shuttle control greatly aided in isolating the image captured on a single frame. It was apparent from the videotape that the majority of the IC's surface charge was removed in the first sweep of the beam across the die area. This last methodology was used successfully for imaging the dynamic signals in the FPALU.

Results

Once the methodology was established, over 120 images were captured and catalogued on video tape over a four-week period. Several days were spent at the outset trying to understand why an active clock line in the imaged phase

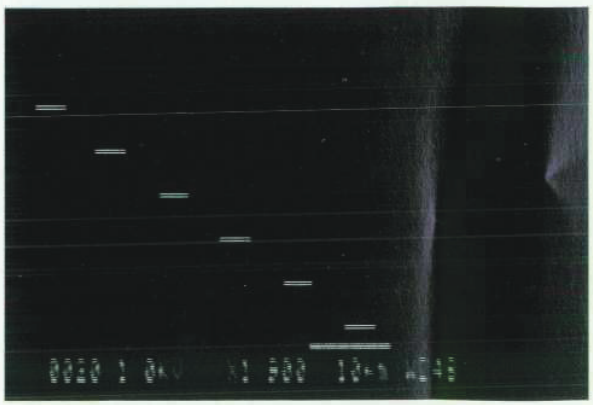


Fig. 5. Video image of the first-pass imaging attempts.

© Copr. 1949-1998 Hewlett-Packard Co.

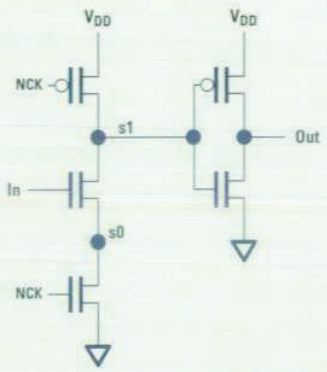
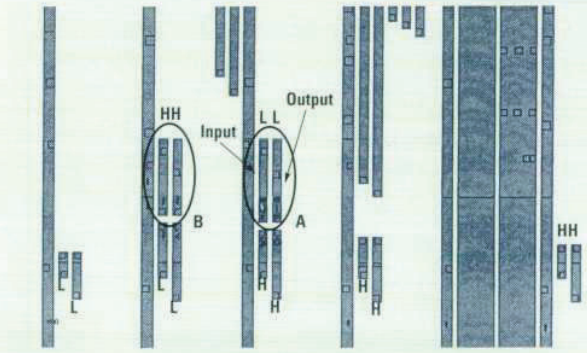


Fig. 6. Zero_topH mousetrap buffer.

was not showing activity, a key indicator that the proper phase of the FLOP was being captured. This issue was never satisfactorily resolved, yet phase-by-phase clock gating in the FPALU ensured that the signals would only be active and thus visible in the phase of interest.

Figs. 6, 7a, and 7b show the schematic, artwork, and voltage contrast image of probably the clearest failure identified. The circuit in Fig. 6 shows a mousetrap buffer whose storage node, s1, was somehow being compromised, possibly through a ground differential problem or a noise spike on the input.



(a)



(b)

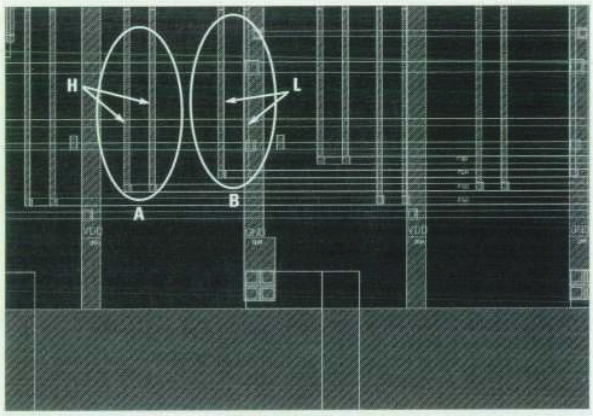
Fig. 7. (a) Metal 3 plot of Zero_topH buffer with failing input/output pair A. (b) Voltage contrast image of victimized buffer with failing input/output pair A.

Circle A in Fig. 7a identifies the buffer's input on the left and the output on the right. The expected value of each metal 3 line is indicated above the lines (L=Low, H=High).

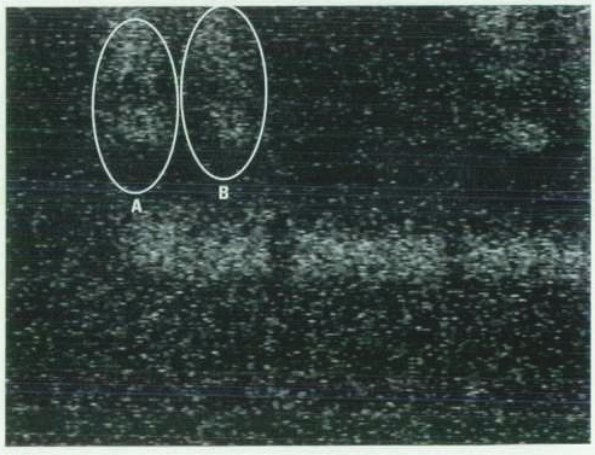
Fig. 7b shows a voltage contrast image captured from the videotape showing the failure of the buffer. The image clearly shows a low level on the input (black) and a high level (white) on the output of the buffer in circle A. Note the difference between circle A and circle B which identifies the input and output of an identical buffer with no failures. It became clear from this picture that the electrical event that caused the buffer to output a high level was transitory in nature and not a static event. The read step of the image was taken with the IC powered down.

Metal 1 and even metal 2 lines can be difficult to image unless they are well-isolated from other metal structures. Fig. 8a shows the artwork and expected values where several metal 1 lines were imaged. The vertical metal 1 route in circle A should have a high or white level, and the route to the right of it in circle B should have a low or black level.

Fig. 8b is the voltage contrast image showing the logical misfiring (high/white) of the metal 1 route in circle B. This



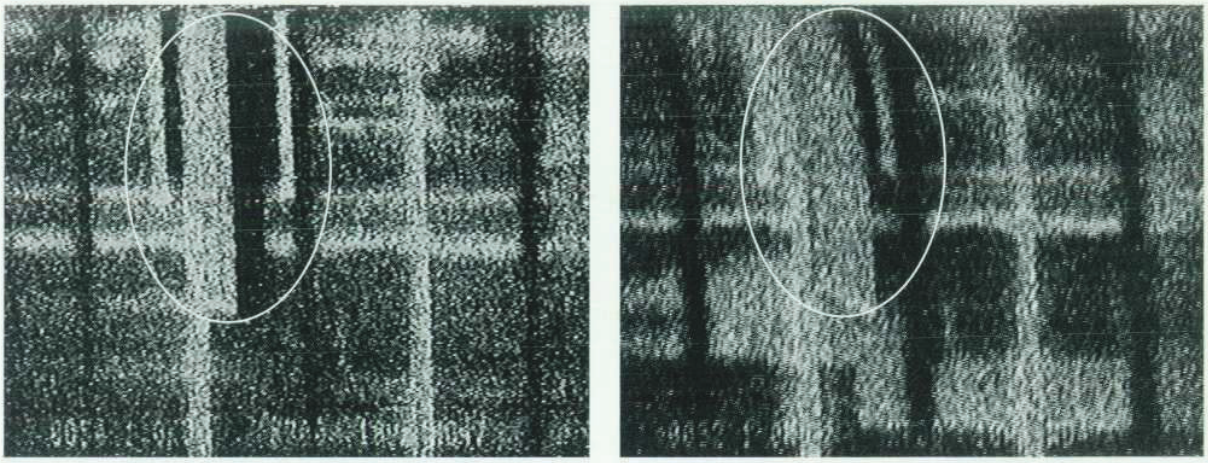
(a)



(b)

Fig. 8. (a) FS[ABCD] bus artwork. (b) Voltage contrast image of FS[ABCD] bus in metal 1 showing correct firing of the lines in circle A and the incorrect firing of lines in circle B.

© Copr. 1949-1998 Hewlett-Packard Co.



(a) (b)

Fig. 9. (a) Metal 3 structure(vertical routing) in passing state at nominal voltage. Horizontal routes are metal 2. (b) Metal 3 structure in failing state at high voltage with wall failure.

failure was not seen until the root cause of the wall was identified and the proper FLOP for arming the failure was identified.

The logical states of individual lines of dense bus structures in lower metal levels can be difficult to discern, yet differences between two states can often be readily identified.

Figs. 9a and 9b illustrate the differencing technique with an example of a metal 3 structure in both a passing and failing state (note the differences in the vertically routed lines in the top-center of the figures). The bend or distortion in Fig. 9b is the result of poor synchronization between the SEM and the VCR that recorded the images. Note also the changes in the horizontally routed metal 2 lines.

One technique that greatly aided the interpretation of the captured images was to plot the artwork of the areas being imaged and annotate the plots with the expected logical levels as derived from a simulator.

Improvements and Future Use

It is difficult to determine if E-beam probing would have provided quicker, more pertinent information than voltage contrast. Each tool has its own benefits and drawbacks that the IC designer must weigh in light of the problem to be solved.

Additional IC physical structures and layouts could make new designs more amenable to voltage contrast imaging as well as E-beam probing and FIB experiments. These features could provide regular, systematic, top-level-metal access to control and data path signals throughout the design. Top-level-metal access could be provided through directed routing or through "via stacks" to top layers from lower-level metal routes. The efficiency of such features in terms of improved accessibility versus increased layout area is unknown.

The image quality obtained from the SEM for voltage contrast work could be improved by changing the electron gun filament from tungsten to a crystalline element. The crystalline filament would increase the beam current and thus effectively provide a brighter image without increasing the beam energy which reduces resolution.

Conclusions

The use of voltage contrast imaging proved to be a useful tool for analyzing and verifying the FPALU margin failure known as the wall. Although the information gleaned from the process did not lead directly to the discovery of the root cause of the failure, the voltage contrast process functioned well as a clue generator as suggested in reference 3 and provided important confirmation of the root cause hypothesis.

Acknowledgments

The author would like to thank Phil Fangman and Chris Anderson for their expertise and operation of the SEM, Rick Butler for his insight and probing questions which helped keep the project on track, and Craig Heikes and Bob Miller for their insight and musings on mousetrap and dynamic logic design and debug.

References

1. C. Heikes, et al, "The Design of 200 MFLOP Floating-Point Megacells for PA-RISC 7100," *Proceedings of the 1992 HP Design Technology Conference.*
2. B. Miller, "Mousetrap Logic—Its Uses and Abuses," *Proceedings of the 1992 HP Design Technology Conference.*
3. B. Miller, et al, "Turn-on and Debugging of MouseTrap Logic Designs," *Proceedings of the 1993 HP Design Technology Conference.*
4. S. Ferrier, et al, "Electron Beam Probing at HP: Contributions and Improvements," *Proceedings of the 1993 HP Design Technology Conference.*

© Copr. 1949-1998 Hewlett-Packard Co.

Component and System Level Design-for-Testability Features Implemented in a Family of Workstation Products

Faced with testing over twenty new ASIC components going into four different workstation and multiuser computer models, designers formed a team that developed a common system-level design-for-testability (DFT) architecture so that subsystem parts could be shared without affecting the manufacturing test flow.

by Bulent I. Dervisoglu and Michael Ricchetti

Members of the latest-generation family of HP workstation and multiuser computer products use the same system architecture and differ mostly in their I/O subsystem architecture and configuration. From a system development point of view an important characteristic of these products is their use of a new high-speed system bus architecture and a large number (over 20) of new ASIC components that were developed to implement all of the various different configurations of the product line. Furthermore, all components that interface with each other via the system bus are required to operate with the same high-frequency system clock.

A further difficulty was that four different models, ranging from a single-user desktop workstation to a multiuser computer, were being developed by different design teams that were both organizationally and geographically separated from each other. This made it necessary to develop a common system-level design-for-testability (DFT) architecture to be used throughout the system and across the different computer models so that subsystem parts could be shared among the different computer models without affecting the manufacturing test flow.

To address these difficulties a DFT core team was formed at the very early stages of the project. Because of the large number of different ASIC teams involved, it was decided that all ASIC teams at the same site would be represented by a single representative on the DFT core team. This team has been instrumental in achieving goal congruence among the different design teams and manufacturing organizations. Furthermore, the presence of the DFT core team made it possible to develop and implement a DFT methodology that was used by all of the ASIC teams, although the level of adherence varied. The DFT core team also collected data and performed DFT design reviews for some of the ASICs.

ASIC DFT Design Rules and Guidelines

One of the first activities of the DFT core team was to develop a set of design rules and guidelines to be followed by the ASIC design teams to ensure that DFT features would be common among the various components. This made it possible to share efforts and results and to access the different DFT features in the ASICs during prototype system bring-up. The following is a summary of these rules.[1]

1. *All (functional) system clocks must be directly controllable from the chip pins and must not be used for any other function.* All systems use a common ASIC component (the system clock controller ASIC) to drive their clock terminals on the system board. This ASIC has control pins through which it can be programmed for different clock generation schemes as well as for starting and halting the system clocks. Thus, not only the individual ASICs but also the entire system board has directly controllable clocks.

2. *All scan and test clocks must be directly controllable from the component pins, which must not be used for any other purpose.* On the system board all test clocks are tied together and controlled from a single test point.

3. *For each ASIC there is a specific reset state which is entered when the component's* ARESET_L *signal is asserted.* On the system board, the power-on condition is detected and is used to reset the ASICs to a known starting state. Next, the memory controller ASIC generates an SRESET_L signal to all other components on the system bus. Additional reset signals are generated by other ASICs for use locally.

4. *All ASICs must implement a dedicated boundary scan register and its associated test access port (TAP) as specified in IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture.*[2] Serial scan-in and scan-out ports of all ASICs in the system (including the PA 7200 processor, which is on a separate module) are connected to form a single serial scan chain.

5. *Access to each ASIC's on-chip test functions must be provided using the IEEE 1149.1 test access port (TAP) protocol.* The same TAP controller design[3] is used or heavily leveraged in many ASICs. This way, test features implemented in this controller as an extension to the IEEE 1149.1 standard were easily leveraged across different ASICs. For example, the DRIVE_INHIBIT/DRIVE_ENABLE instructions and the OUT_OFF bit in the boundary scan register (see "TAP/SAP Controller," below) are duplicated in different ASICs in this way.

© Copr. 1949-1998 Hewlett-Packard Co.

6. *All ASICs shall be designed to support I_{DDQ} testing, whenever this is not prevented by the technology used.* In most cases this requirement did not present any further design constraints or changes. In a few cases, an internal I_{DDQ} enable signal had to be used to disable active pull-up and pull-down circuits. However, because of schedule and cost considerations the PA 7200 processor chip does not support I_{DDQ} testing.

7. *All ASICs shall implement internal scan for testing. The percentage of internal nodes that are scannable shall be kept as high as possible without sacrificing major chip area or otherwise affecting the design methodology.* For most practical purposes all ASICs have implemented internal scan for 100% or nearly 100% of all internal flip-flops. However, because of design style and technology differences, some portions of the PA 7200 processor chip are not scannable.

8. *There shall be no asynchronous logic implemented in the ASICs.* Lack of asynchronous logic is an important requirement for many CAD tools for generating test vectors. Furthermore, this rule is intended to prevent side effects caused by changing the internal and external signals in arbitrary sequence. The only exception to this rule is granted for the reset signals, which are implemented to follow a carefully planned system reset strategy.

The following sections describe some of the DFT features that have been implemented in the ASICs. Not all features are implemented in all ASICs. Among the various ASICs, the memory controller stands out as the chip with the most extensive DFT features.

TAP/SAP Controller

Access to all on-chip DFT features is implemented through a test controller block called the *test access port/scan access port* (TAP/SAP). The test controller implements all of the required instructions for the IEEE 1149.1 TAP controller as well as an extensive set of public and private instructions which are targeted mostly for internal testing of the ASIC. Table I lists all of the TAP instructions that are implemented. Among the public instructions that have been implemented are the DRIVE_INHIBIT and DRIVE_ENABLE instructions which are used to set and clear a latch in the system logic domain (not considered part of the test logic).

System logic for all ASICs has been designed such that for normal system operation (i.e., when test logic is not controlling the I/O pins) the ASIC can drive out only if the DRIVE_INHIBIT latch is cleared. Each ASIC uses its ARESET_L input to clear the DRIVE_INHIBIT latch during power-up. Whereas ARESET_L controls the DRIVE_INHIBIT latch only if the TAP is in a reset state, explicit TAP instructions can be used at other times to set or clear this latch. This scheme allows in-circuit ATE programs to set the DRIVE_INHIBIT latch before they terminate and reset the TAP without creating possible board-level bus contention before removing electric power from the board. Whereas the DRIVE_INHIBIT latch is considered part of the on-chip system logic, it is implemented as part of the TAP controller design so that ASIC designers implementing normal system functions do not have to deal with any of the issues surrounding the DRIVE_INHIBIT and DRIVE_ENABLE operations.

Table I
TAP Instructions

Instruction	Drive I/O Pads	Scan Register
EXTEST	Boundary Register	Boundary
BYPASS	System Logic	Bypass
SAMPLE/PRELOAD	System Logic	Boundary
IDCODE	System Logic	ID Code
HI_Z	High-Impedance	Bypass
DRIVE_INHIBIT	Boundary Register	Bypass
DRIVE_ENABLE	System Logic	Bypass
SCAN_INTERNAL	System Logic	f(Mode)
CHIPTEST	High-Impedance	f(Mode)
INTEST	Boundary Register	Boundary
DR_SCAN	System Logic	f(Mode)
SELECT_MODE	Boundary Register	Mode
SET_MODE_BIT	Boundary Register	Mode
CLR_MODE_BIT	Boundary Register	Mode
ISAMPLE	System Logic	Bypass
ESAMPLE	System Logic	Bypass
DS_DRIVE	Boundary Register	Boundary
DS_RECEIVE	System Logic	Boundary

Other TAP instructions are used to set and clear bits of the mode register to provide access to additional test features such as I_{DDQ} testing, double-strobe, and so on. It is also possible to speed up internal scan operations by switching on the parallel scan bit in the mode register. This feature enables multiplexing of the chip's I/O pins to perform serial scan-in and scan-out of the internal scan register by breaking it into three independent sections which are scanned in parallel together with the boundary register, which is always scanned using the test data in and test data out pins of the TAP.

CHIPTEST Instruction

One of the major difficulties in implementing DFT in the ASICs used for this project has resulted from a common leveraged I/O pad design that contains nonscannable latches. Furthermore, the bidirectional I/O cell implements an internal bypass path to feed into the chip the same value that is being driven onto the I/O pad by that chip. In effect, I/O pads contain nonscannable pipeline stages that control both the direction and the value of data on the I/O pad. Following a recommendation from the DFT core team the basic I/O cell design was modified to allow data values received by the on-chip system logic to be set up using the dedicated boundary scan register. In addition, system logic output values can be captured into the boundary scan register using the system clock. These design changes were coupled with features provided by the CHIPTEST instruction in the TAP controller to streamline the internal testing of the ASICs. For example, all internal logic of the memory subsystem ASICs (memory controller, slave memory controller, and data multiplexer) is tested by the following sequence:

1. Load the CHIPTEST opcode into the ASIC.
2. Use test clocks to perform a parallel scan of the ASIC internal nodes and the boundary register. At the end of the scan-in process the newly scanned-in values are automatically moved from the boundary register to the nonscannable latches in the I/O-cells.

© Copr. 1949-1998 Hewlett-Packard Co.

3. Apply a single system clock to capture test results in internal nodes and system logic output values in the boundary register.
4. Repeat steps 2 and 3 for each new vector, overlapping the scan-in and scan-out operations.

Since the CHIPTEST instruction drives the I/O pins to a high-impedance state it is possible (indeed it is intended) to execute these tests on a populated system board without fear of creating board-level bus clashes during such testing.

BIST Implementation

The memory controller ASIC incorporates several wide and shallow register files that are used for queueing operations within the data paths. The total number of storage elements in the register files is quite large, so it was not practical to make these storage elements scannable. Therefore, a built-in storage test (BIST) approach was chosen to test the memory controller data path register files.

The memory controller BIST implementation was developed with the following objectives:

- Provide high coverage and short test times.
- Provide at-speed testing of the register file structures to ensure that the memory controller ASIC works at the required system clock frequency.
- Provide flexibility and programmability in the BIST logic to allow alteration of the test sequence for debug and unforeseen failure modes. In particular, the system bring-up and debug plans provide a means for system-level scan access to the state within the ASICs. Providing these features allows read/write access to the nonscannable queue states for prototype system debug.
- Provide for testability of the logic surrounding the register files through added observation and control points at the inputs and outputs of the register file blocks. This is intended to support automatic test pattern generation (ATPG) tools used to generate test vectors for the memory controller and thus ensure high coverage of the standard cell control logic for the queues.

The design of the BIST logic in the memory controller data paths is based on previous work that was done for the PA 7100-based HP 9000 Model 710 workstation. For that product, a structure independent RAM BIST architecture that uses a pseudoexhaustive test algorithm and signature analysis was developed and was implemented in the I/O controller ASIC.[4] The structure independent, pseudoexhaustive test algorithm provides 99.9% fault coverage of typical RAM faults and can provide 80% to 99.9% coverage of neighborhood pattern-sensitive faults. It also allows the test time (number of read/write accesses per memory address) to be varied according to the desired fault coverage. BIST architectures for both the present memory controller ASIC and the previous I/O controller ASIC use a test algorithm similar to that described by Ritter and Schwair.[5] Using the system clock for BIST execution, the RAM structure can be tested at the normal system clock rate, thus providing at-speed testing of the RAM.

A dual-port write/single-port read register file from the present memory controller data path, with test structures that provide both BIST and ATPG support similar to the previous I/O controller BIST architecture, is shown in Fig. 1. The two write ports, A and B, can both be addressed and written independently. The single read port can also be addressed and read independently of the A and B write ports. Thus, two write operations and one read operation can all occur simultaneously for one to three register locations, depending on the A, B, and read port addresses.

Given the dual-ported design of the memory controller register files, it was necessary to extend the previous I/O controller BIST architecture to test a dual-ported RAM. This meant that the memory controller BIST implementation should be able to test not only the simultaneous dual-write operations but also the various combinations of A/B write and read operations to verify that the port interactions are working correctly. For the dual-port register files in the memory controller such interactions include an internal bypass when the read address is the same as either of the A or B write addresses and a B-port dominant write when the A and B write addresses are equal. This dual-write BIST algorithm is described in reference 6.

For the register file shown in Fig. 1 each of the BIST structures—LFSR (linear feedback shift register), SHIFT, COUNT, and MISR (multi-input signature register)—is dedicated to BIST. Each register file also has its own dedicated programmable BIST control queue for sequencing the BIST algorithm. The BIST_MODE signal enables the BIST functions and can be

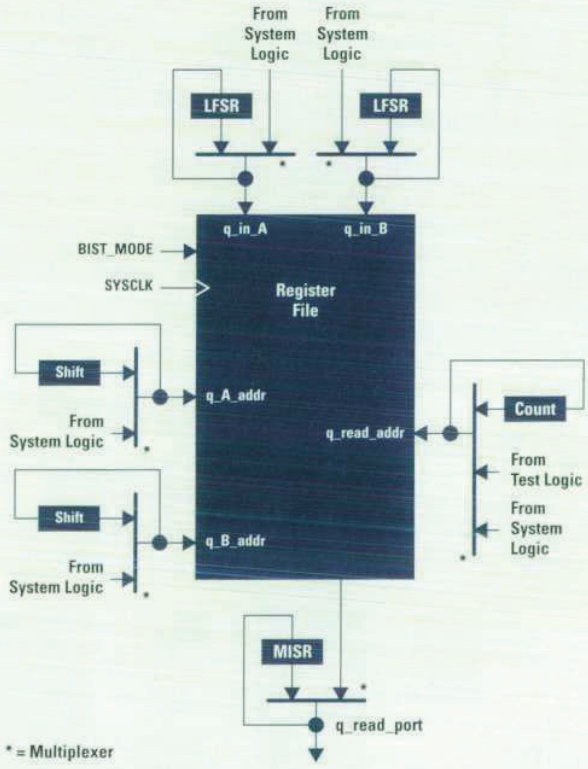


Fig. 1. Dual-port register file with built-in storage test (BIST) and automatic test pattern generation (ATPG) features. The inputs to the central, embedded RAM structure are provided by multiplexing between the normal system value and a BIST register, which is implemented as a linear feedback shift register (LFSR). The output multiplexer makes it possible to capture the outputs into a multi-input signature register (MISR) and to send either the RAM outputs or the MISR contents to the rest of the system.

© Copr. 1949-1998 Hewlett-Packard Co.

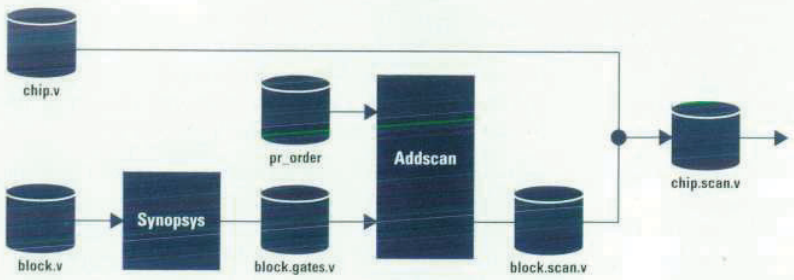


Fig. 2. Addscan tool flow. Addscan is an in-house software tool for scan insertion. Synopsys is an automatic design synthesis tool from Synopsys, Inc.

controlled either by a pin on the chip or through other test access logic such as an IEEE 1149.1 TAP controller or P1149.2 SAP controller.[2,7] All of the BIST registers are implemented in standard cell blocks separate from the data path register files. A detailed description of the memory controller BIST implementation and operation, along with hardware overhead and test coverage, can be found in reference 6.

Test Tools

The following sections describe the tools and tests that were used and developed to test the three memory subsystem ASICs: the memory controller, the slave memory controller, and the data multiplexer.

Addscan. Fig. 2 shows the flow for scan synthesis. All three memory subsystem ASICs were designed using a structured-custom design method.[8] Each standard cell block used the in-house Addscan tool[9] for scan insertion and the scan links between blocks were connected by hand in the top-level netlist of the chip. The internal scan order of each block is based on post place and route information.

Test Vector Generation. The test vector generation flow is shown in Fig. 3. The ATPG tools from Crosscheck, Inc. were used to generate most scan-based vectors. Some vectors were hand-generated. A gate-level netlist of the chip, prior to Addscan scan insertion, is used to create an ATPG database for vector generation. A similar data base is used by designers to do Timver static timing analysis. Timver, a timing analysis tool from Aida, Inc., is used as a part of the test methodology for two purposes. First, it allows the design to be checked for hold violations on all paths to guarantee that there will be no timing violations even if ATPG vectors exercise nonfunctional paths in the design. Secondly, Timver critical paths can be fed back into ATPG in the form of a vector.tcp file to generate double-strobe path delay vectors.

The following test vector sets were created for each of the memory subsystem ASICs:

- Continuity. Checks for opens and shorts among the ESD protection diodes. Prepared manually.
- Ringtest. Uses serial "flush" speed (total scan path delay) through the boundary scan register as a measure of the IC process and verifies that the part is within the six-sigma range. Generated manually in the form of a Cadence Verilog body file.
- Dc. These tests use the boundary scan ring to drive out all ones or zeros for dc parametric testing. Generated manually in the form of a Verilog body file.
- Leakage and tristate testing. Places the ASIC into a high-impedance state to allow testing the I/O pads for leakage. Generated manually in the form of a Verilog body file.
- I_{DDQ}. These vectors are generated by ATPG and are used to perform static I_{DDQ} test and measurement.
- TAP Tests. These are tests targeted at functional testing of the TAP controller. Generated manually in the form of a Verilog body file.
- Chiptest. These vectors are generated by ATPG to test the core chip logic in from and out to the boundary scan ring using the TAP CHIPTEST instruction. I/O pad logic is not fully tested by chiptest vectors.
- Pintest: These vectors are generated by ATPG and will test the remaining faults (primarily in the I/O pad logic) that are not covered by the chiptest.
- Bus Holder. Further testing of the electrical characteristics of the bidirectional I/O cells. Generated manually in the form of a Verilog body file.
- BIST. BIST vectors are only generated on the memory controller. These tests require only two scan vectors, one each to set up the initialization and test passes for BIST. After that, a burst of system clocks is applied to test the target

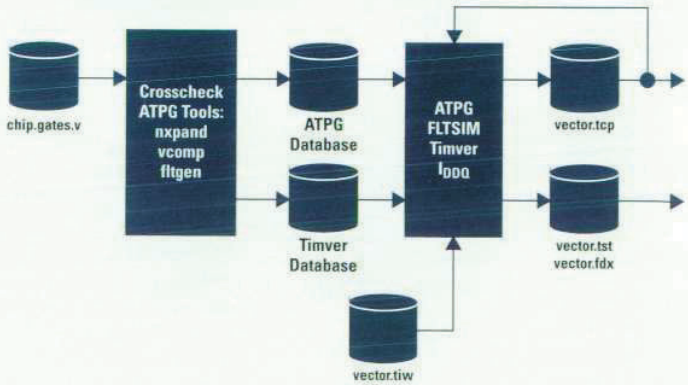


Fig. 3. ATPG (automatic test pattern generation) and timing tools flow. Timver is a timing analysis tool from Aida, Inc. FLTSIM is an in-house fault simulator for test verification. The I_{DDQ} vectors are generated by ATPG from Crosscheck Corp.

© Copr. 1949-1998 Hewlett-Packard Co.

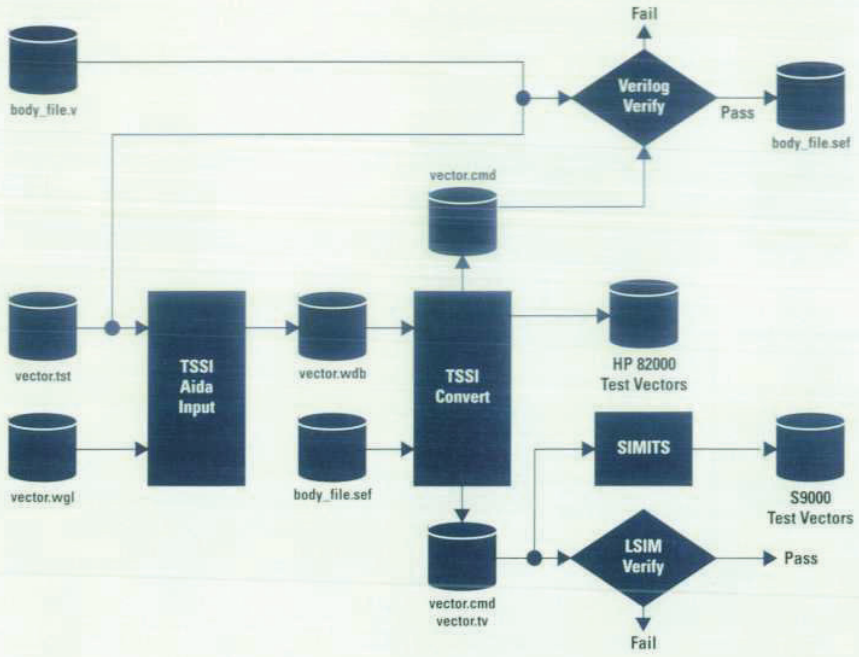


Fig. 4. Vector verification and tester translation tools flow. TSSI is a test program generation tool from TSSI, Inc. Aida represents a suite of test tools from Aida, Inc. LSIM is a special FET-level simulator. SIMITS is a format conversion tool from Schlumberger.

blocks at speed. These vectors are generated using a Perl script to produce a .tst vector file.

- Double-Strobe. These vectors are generated by ATPG based on Timver critical paths and are used to provide at-speed testing of the ASIC.[10,11]
- Ac testing of I/O Paths. These are functional tests that test the speed characteristics of critical I/O paths. Generated manually only if testing, design review, and chip characterization results indicate a concern.
- Process, Voltage, and Temperature (PVT) Block Test. Generated manually, this group of tests applies only to the slave memory controller chip which uses a unique PVT block to compensate for process, voltage, and temperature variations in a particular I/O cell.

Vector and Test Logic Verification. Fig. 4 shows the flow for test vector verification using Verilog or LSIM (a special FET-level simulator). Test vectors from ATPG can be directly converted using TSSI (a tool for test program generation from TSSI, Inc.) into a command file format and verified against a gate-level netlist in Verilog or a FET-level netlist using LSIM. Alternatively, test vectors can be simulated using a Verilog body file. A body file is a wrapper or test jig that can either be a test vector set itself (hand-generated functional tests) or can run scan-based ATPG vectors using a scan and clock sequence.

The AT&T Tapdance tool was used for further verification of the TAP logic before tape release of the ASICs. Tapdance generates a set of IEEE 1149.1 compliance tests to verify standard TAP functionality. The Tapdance vectors were converted using Perl scripts† into a Verilog force file and simulated on a gate-level netlist.

Tester Format Translation. Fig. 4 also shows the flow for translation of vectors into a tester format. Using TSSI, vectors were formatted directly to HP 82000 tester format. To get to the Schlumberger S9000 tester, vectors were first formatted to LSIM and then passed through SIMITS, a format converter from Schlumberger.

Using a Verilog programming language interface that outputs a TSSI simulation event format file dump, vectors can also be translated from body files to one of the testers.

System DFT Features

The new systems have been designed to provide a method to access ASIC scan paths, both boundary and internal, at the system level. This has two major purposes. First, it provides a means of accessing the internal state of complex VLSI components. This provides additional hardware state information to designers that would typically be inaccessible and can aid traditional prototype bring-up and debug methods. Second, it provides the ability to do scan-based testing of board and system interconnect and internal scan testing of ASICs.

The following test and debug features are provided by system scan access:

- Ability to halt the system clocks and interrogate the internal scan state of the ASICs.
- Single-cycle debug of the system core by halting the system clocks, interactive scanning of the internal state, and then starting or cycling the system clocks.
- Board-level and system-level interconnect testing and interactive debug using boundary scan. This includes testing connectors between two boards where boundary scannable buses cross the connector.
- Ability to test an ASIC while it is on the board using boundary and internal scan. This may include double-strobe tests and running on-chip BIST, if supported by the ASIC under test.

As part of the overall DFT requirements, all ASICs implement the IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture. This provides support for system-level scan access. In addition, key debug support features are

† Perl is a high-level programming language.

© Copr. 1949-1998 Hewlett-Packard Co.

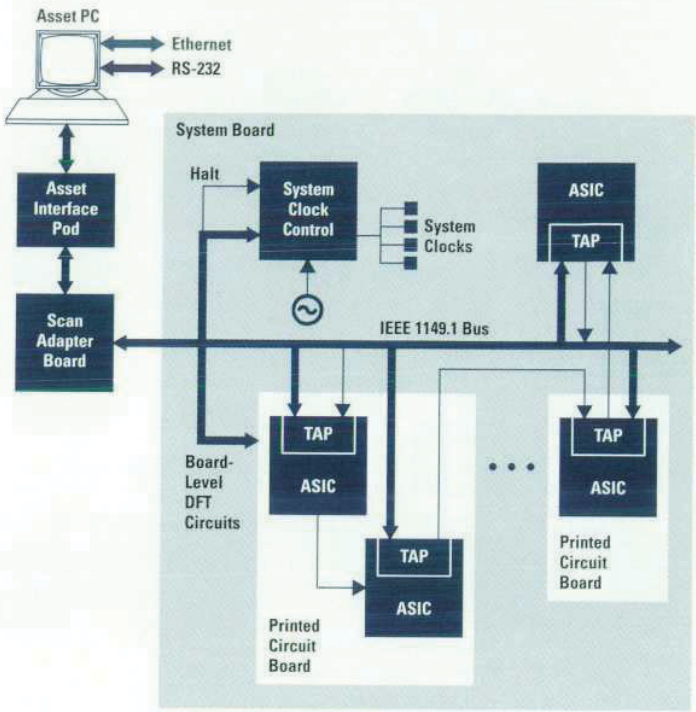


Fig. 5. System-level scan access. Asset is a set of scan tools from Texas Instruments, Inc.

incorporated into the system clock controller chip to allow for halting and controlling the system clocks. Further information on system clock controller features can be found in reference 12.

Fig. 5 shows a diagram of the system-level scan access hardware. The Texas Instruments PC-based Asset toolset is used as the interface to system scan. The Asset PC is connected to a scan adapter board via the Asset interface pod. The scan adapter board then plugs onto the system board and provides control of the system clock controller features, the TAP controller and system logic reset, the clock halt triggers, and the I/O device clock halt from the Asset software. The scan paths in the system are configured as a single serial scan chain with optional system boards implemented as dynamic scan paths that can be configured in Asset.

The Asset scan tools provide the following capabilities for system scan access:

- Interactive control of scan path data and TAP controller instructions with scan-bit name mapping and packing and unpacking of scan data.
- Macro scripting capabilities for combining several interactive operations into a single macro command. Asset also accepts serial vector format scan vectors for user-developed tests.
- Specification of system scan path configuration for dynamic scan paths and optional boards, such as CPUs, memory extender boards, and I/O.
- Scan path integrity testing and boundary scan interconnect testing of intraboard and interboard nets.
- Control of system clock halt, single-cycle stepping, and system and TAP reset.

Results and Conclusions

The DFT techniques described above, which were championed by the DFT core team, were implemented in several different ASICs with varying degrees of adherence to the DFT rules and methodology. In general, results obtained during prototype chip debug have shown a direct correlation between the level of DFT implementation and the rapidity of test development, chip characterization, and root-cause analysis. For example, while the three memory subsystem ASICs were the last to reach tape release, these chips were the first to reach the operational test release (OTR) and release to manufacturing test (RTPT) checkpoints. The availability of high-quality and comprehensive test sets for these chips enabled chip characterization efforts to be started right away. Furthermore, success in reaching the OTR checkpoint made it possible to transfer the task of testing prototype chips (which are used in the prototype systems) to the manufacturing engineers. This had a very positive effect on resources available to perform chip characterization. In turn, successful completion of this step coupled with efforts of the R&D engineers to improve test coverage enabled the team to reach the RTPT milestone well before any of the other ASICs had reached their OTR checkpoints.

The Asset tool and its customized extensions provided a low-cost system scan access solution with flexible functionality and ease of use. As a commercial tool solution it cut down on development and maintenance costs compared to developing a proprietary toolset and can be reused for future projects.

© Copr. 1949-1998 Hewlett-Packard Co.

Acknowledgments

The authors gratefully acknowledge contributions from many members of the Chelmsford DFT group, the DFT core team, and several ASIC designers. In particular, contributions from Ruya Atac, Steven Zedeck, Tom Dickson, Shawn Clayton, Joy Han, Jim Williams, Matt Harline, Dave Malicoat, Ken Pomaranski, and Mick Tegethoff have been key to the successful implementation of DFT in this project.

References

1. B. Dervisoglu, *ASIC DFT Design Rules*, Hewlett-Packard ASD/CSL, November 1992.

2. IEEE Test Technology Technical Committee, *Standard Test Access Port and Boundary Scan Architecture*, IEEE Standard 1149.1/1990, Institute of Electrical and Electronic Engineers, Inc.

3. B. Dervisoglu, *PA 7200 Memory Subsystem TAP/SAP Design*, Hewlett-Packard ASD/CSL, January 1994.

4. M. Ricchetti, "A Structure Independent Built-In Test Architecture for Random Access Memories," *Proceedings of the 1992 Hewlett-Packard Design Technology Conference*, pp. 417-424.

5. H.C. Ritter and T.M. Schwair, "A Universal Test Algorithm for the Self-Test of Parametrizable Random Access Memories," *Proceedings of the Second European Test Conference*, April 1991, pp. 53-59.

6. M. Ricchetti, "Built-In Self-Test of the Memory Controller Data Paths," *Proceedings of the 1993 Hewlett-Packard Design Technology Conference*, pp. 303-312.

7. IEEE Test Technology Technical Committee, *Extended Digital Serial Subset*, IEEE Draft Standard P1149.2/D0.2, Institute of Electrical and Electronic Engineers, Inc.

8. S. Clayton, D. Chou, T. Dickson, D. Montrone, R. Ryan, and D. Schumacher, "The Automation of Standard Cell Block Design for High-Performance Structured Custom Integrated Circuits," *Proceedings of the 1993 Hewlett-Packard Design Technology Conference*, pp. 349-356.

9. S.A. Markinson, "The Application of Scan Synthesis in System ASIC Design," *Proceedings of the 1992 Hewlett-Packard Design Technology Conference*, pp. 449-458.

10. B.I. Dervisoglu and G.E. Stong, "Test and Design-for-Test Standards for the Kodiak Project," *Proceedings of the 1990 Hewlett-Packard Design Technology Conference*, pp. 564-571.

11. B.I. Dervisoglu and G.E. Stong, "Design for Testability: Using Scanpath Techniques for Path-Delay Test and Measurement," *Proceedings of the International Test Conference*, October 1991, pp. 364-374.

12. D. Malicoat, *System Clock Controller ASIC: External Reference Specification, Version 0.91*, Hewlett-Packard Systems Technology Division, November 30, 1992.

© Copr. 1949-1998 Hewlett-Packard Co.

Authors

April 1995

6 PA-RISC Multimedia Workstation

Roger A. Pearson



Born in Minneapolis, Minnesota, Roger Pearson is a member of the technical staff at HP's Fort Collins systems laboratory. He came to HP in 1979 at the HP Desktop Computer Division and is presently responsible for printed circuit board design of the CPU assembly for a new workstation. He was responsible for the printed circuit board design of the Model 712 system assemblies including schematic capture and layout. Roger received a BSEE degree from the University of Minnesota in 1979. Past HP achievements include R&D and production engineering for the power supply used in the HP 9845B desktop computer, production engineering for several assemblies used in the HP 9845C and 9845B computers, new product manufacturing engineering for various graphics cards and products, and work on the 68000-based workstation products. He is married, has two daughters, and enjoys playing the piano, playing hockey with an adult league, and coaching a youth hockey team.

12 A Case Study of IC Design Decisions

Mick Bass



Mick Bass has been a member of the technical staff at the Systems Technology Division since 1989 when he joined HP. He is currently working on the development of the next generation of low-end microprocessors. He participated in the development of the memory and I/O controller, presilicon verification, and postsilicon verification for the PA 7100LC processor. He has worked on core VLSI circuit design and electrical verification for HP 9000 Series 700 workstations. He received a BS degree in computer engineering from the University of Illinois at Champaign/Urbana in 1989 and is a member of the IEEE. He has coauthored two papers about the PA 7100LC processor, one that appeared in the April 1994 issue of IEEE Micro and another in the February 1993 Compcon Proceedings. In his leisure time he enjoys sailing, cycling, skiing, and ice hockey.

Patrick Knebel



Patrick Knebel is an R&D project manager at the Systems Technology Division where he is currently responsible for managing the core design and verification group for the next generation PA-RISC processor. Pat helped to define the PA 7100LC feature set and implemented the control algorithms for the core CPU. His previous accomplishments at HP include design and verification on the PCX processor (HP's first CMOS CPU) and the HARP1 processor (Hitachi's first PA-RISC CPU). Pat received a BSCEE degree from Purdue University in 1986 and an MSEE from Stanford University in 1992. He is a member of the IEEE Computer Society and is interested in RISC computer architectures and verification techniques. His work has resulted in several patents pending for design features on the PCX and PA 7100LC processors. He presented a paper on the PA 7100LC processor at the 1993 Spring Compcon conference, which was published in the Compcon Proceedings. Born in Jasper, Indiana, he is married, and enjoys snow skiing, hiking, motorcycle riding, and flying stunt kites.

David W. Quint



David Quint is a technical contributor at the Integrated Circuits Business Division where he is responsible for the package design for the next-generation CPU. Since he joined HP in 1979 he has worked on IC process engineering and package design for CPU chips. He has designed ceramic PGAs, multichip modules, TAB, and most recently ceramic solder column grid arrays using C4 chip attach. His work has resulted in two patents and has been published in the Journal of Applied Physics and the HP Journal. Dave attended the University of Wisconsin at Madison, where he received a BSEE degree in 1973 and an MSEE in 1976. He continued at MIT, earning a PhD in 1980. Between degrees at the University of Wisconsin, he worked at Westinghouse on reactors for Nimitz-class aircraft carriers. Dave is married and has three children.

William L. Walker



Born in Houston, Texas, Will Walker attended Rice University from which he received a BSEE degree in 1985. He later went to Stanford University where he earned an MSEE in 1992. He came to HP in 1985 at the Systems Technology Division where he was a member of the technical staff. Will designed the superscalar execution controller and the floating-point controller for the PA 7100LC. He also worked on the floating-point controller design for the PA 7100 processor and earlier models of the HP 9000 Series 700 computers. He has published two papers on superscalar PA-RISC processors, one in the 1992 Compcon Proceedings and the other in the 1993 Compcon Proceedings. Will currently works for Cyrix Corporation as a senior engineer involved with CPU functional verification. He is married, has two daughters and enjoys basketball.

23 Design Methodologies

Mick Bass

Author's biography appears elsewhere in this section.

Terry W. Blanchard



An R&D manager at HP's Systems Technology Division, Terry Blanchard is currently responsible for core CPU control design and verification, cache control design, and the physical implementation of the cache data path for a new PA-RISC CPU chip. His primary responsibilities since joining HP in 1989 have been in the areas of presilicon and postsilicon verification, chip modeling, simulation, emulation, and tools liason with outside vendors. He has contributed to several PA-RISC CPU chip efforts, including the PCX, PCX-S, PA 7100, PA 7100LC, and subsequent processors. He attended Brigham Young University from which he received a BSECE degree in 1989. Terry is married, has three young daughters, and enjoys classical and broadway music, camping, landscaping, woodworking, and spending time with his family. He also leads a local Boy Scouts troop.

© Copr. 1949-1998 Hewlett-Packard Co.

D. Douglas Josephson



Doug Josephson is an technical contributor with the Systems Technology Division where he is presently responsible for VLSI circuit design and test methodology for future HP processors. He was responsible for circuit design, test methodology, introduction of new test methods, and test and electrical characterization for the PA 7100LC processor. Previous work at HP includes test engineering and electrical characterization for the PCX and PCX-S processors, which were the first CMOS PA-RISC designs created at HP. He came to HP in 1989 at the Integrated Circuits Business Division. Born in Monmouth, Illinois, he attended the University of Iowa, where he received his BSEE degree in 1988. Before coming to HP, he was a summer intern at Lattice Semiconductor where he worked on electrical characterization of electrically erasable programmable logic devices. Doug is a member of the IEEE and is interested in custom VLSI circuit design and design for testability. He has published a paper on PA 7100LC features and an article on microprocessor I_{DDQ} testing. His work has resulted in a patent pending for the sample-on-the-fly technique described in this issue. He enjoys backpacking, rock climbing, and high-altitude mountaineering.

Duncan Weir



Duncan Weir has been a member of the technical staff at Systems Technology Division since he joined HP in 1986. Born at Clark Air Force Base in the Philippines, he attended Washington University from which he received BSEE and BSCS degrees in 1986. Previous HP accomplishments include verification of the PCX, PCX-S, and PCXT processors. He is currently responsible for verification of the chip that is the follow-on to the PA 7100LC processor. He worked on postsilicon verification for the PA 7100LC processor. His work has resulted in a patent pending for the hardware TLB miss handler on the PCXT processor. He is interested in the use of random code generation to test PA-RISC processors. He is married and says he is kept pretty busy entertaining and exercising his two dogs.

Daniel L. Halperin



A native of Oak Ridge, Tennessee, Dan Halperin attended the University of Tennessee from which he received a BSEE degree in 1978. He continued his studies at the University of Illinois, earning an MSEE in 1981 and a PhD in 1984. He is a member of the technical staff at the Systems Technology Division and is presently responsible for the physical implementation of control blocks similar to the ones described in this issue. Previous contributions at HP include work on five different designs of PA-RISC VLSI hardware. Before coming to HP, he served as a research and teaching assistant at the University of Tennessee and the University of Illinois. He is a member of the IEEE and is interested in computer architecture, digital circuits, and VLSI. Dan is married, has two children and enjoys hiking, bicycling, skating, and electronic keyboards.

36 An I/O System on a Chip

Thomas V. Spencer



Born in Louisville, Kentucky and raised in southern Indiana, Tom Spencer attended Purdue University from which he received a BSEE degree in 1980 and an MSEE in 1982. He came to HP's Desktop Computer Division in 1982 and is now an R&D project manager with the Fort Collins system laboratory of the Systems Technology Division. Previous accomplishments at HP include design engineering for the HP 9000 Models 825 and 845 computer systems. He was also project manager for the LASI chip used in the HP 9000 Models 712, 715/64, 715/80, and 715/100. He is currently working on the I/O chip for the next generation workstation. He is married and the new father of triplets born in November 1994. Before the triplets his hobbies included mountain biking, tennis, basketball, and skiing. His new hobbies include changing diapers and feeding babies.

Frank J. Lettang



A development engineer with the Systems Technology Division, Frank Lettang has been with HP since 1980. He holds a BSEECS degree from the University of California at Berkeley and an MSEE from Stanford University. He worked on HP 9000 Series 800 computer systems, the HP 9000 Series 700 workstations, and the general system connect (GSC) and SCSI interfaces in the LASI chip. He is presently working on a new family of workstations. He is a member of the IEEE and a member of the PCI electrical working group. Frank is interested in high-speed digital design, RF engineering, and signal processing, and has authored an HP Journal article about clocks for high-speed workstations. Born in Lynnwood, California, he is married and has two children. He enjoys gardening, hiking, cycling, playing with his children, and participating in elementary school volunteer work.

Curtis R. McAllister



A member of the technical staff at HP's Systems Technology Division, Curtis McAllister joined HP in 1988 after receiving a BSEE degree from Iowa State University in that same year. He is currently responsible for work on a bus converter, which includes work on the bus interface design and CAD tool support. Previously, he worked on the HP 9000 Models 715/64, 715/80, and 715/100 workstation products. On the LASI chip he provided CAD tool support for several ASIC tools and designed the parallel (Centronics) interface and the interface from the LASI internal bus to the Intel 82596 megacell. Before the LASI chip he worked on the graphics ASIC used on the CRX24 graphics adapter and in the Model 710 workstation. He also designed an HP-300 physical DMA interface board for the TurboVRX graphics accelerator. Before joining HP, he did summer internships at McDonnell Douglas and Texas Instruments, working on various hardware design and software programming projects. His work has resulted in a patent on using buffer pointers for direct memory access. He has published an article on frame buffer design. Curtis is interested in VLSI design, computer architecture, and CAD tools, and is a member of the IEEE. He was born in Mt. Pleasant, Iowa and enjoys skiing, basketball, and day hikes.

Anthony L. Riccio



Anthony Riccio came to HP in 1976 at HP's Calculator Products Division. He is now a member of the technical staff at the Fort Collins systems laboratory. He was born in Norwalk, Connecticut, and attended the Rochester Institute of Technology from which he received a BSEE degree in 1975. He is presently working on the processor board for the next generation of desktop workstations. Previous HP contributions include hardware design on an E-beam project at HP Laboratories, VLSI design for PA-RISC CPU, cache, and memory chips, processor design for the HP 9000 Models 750, 712, and 715 workstations, and design work on the LASI chip. His work has resulted in a patent related to a programmable maximum/minimum counter for performance analysis of computer systems and he is a member of the IEEE. Tony is married and has two children. He enjoys family outings, playing and coaching hockey, coaching soccer, playing tennis, hiking, biking, sailing, photography, and reading. He also participates as a Christian hockey camp counselor.

© Copr. 1949-1998 Hewlett-Packard Co.

Joseph F. Orth



A member of the technical staff at HP's Systems Technology Division, Joe Orth joined HP in 1992. He was born in Jerome, Idaho and attended the University of Idaho from which he received a BSEE degree in 1992. He is presently responsible for environmental testing of the next generation CPU chip. He worked on the LASI chip's PS/2 interface for the project reported in this issue. He is interested in digital design and enjoys rock climbing, hockey, basketball, softball, and hiking.

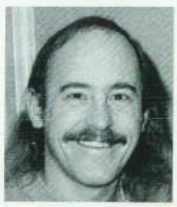
Brian K. Arnold



Brian Arnold was born in Canon City, Colorado and attended Colorado State University from which he received a BSEE degree in 1987. He is a member of the technical staff at the Systems Technology Division and has been with HP since 1989. He was a graphics hardware engineer for two different products and wrote low-level graphics software for the Model 425E workstation. He is currently working as a design engineer on the next generation PA-RISC processor, which includes responsibility for three standard cell libraries, Synopsys and Cell3 tools support, and block design. He was responsible for the audio and telephony interface for the LASI chip. Before coming to HP, he researched multilevel logic, fast arithmetic algorithms, and design engineering on an MIMD super computer. His work at HP has resulted in a patent related to telephony interface design. He is interested in computer architecture as well as design and IC tools. He has published several articles on the benefits of a signed-binary number system and also on algorithms that perform fast multiplication and division using a signed-binary representation of numbers. He is married, has a four-month-old son, enjoys outdoor sports and recreation, and leads a marriage ministry at his church.

43 Graphics Chip

Paul Martin



Paul Martin was born in Salt Lake City, Utah. He received a BSEE degree from the University of Colorado (1982) and an MSEE from Stanford University (1984). He has worked on diagnostic software development for Bell Laboratories and mapping software for the National Oceanic and Atmospheric Administration before joining HP's Fort Collins System Division in 1985. He worked as a designer for the SRX and CHX graphics systems and as a manager for microcode development for the TurboSRX graphics subsystem. He was also a manager for graphics development for the Model 425e workstation. He worked on the development of the graphics chip for the Model 712 workstation. As an R&D manager in the graphics hardware laboratory at HP's Workstation Systems Division, he is presently working on developing the next generation single-chip graphics controller. He has coauthored a paper on the TurboSRX graphics subsystem. He enjoys mountain biking, skiing, and playing the guitar.

51 HP Color Recovery

Anthony C. Barkans



Tony Barkans has been a member of the technical staff in the graphics hardware laboratory at the Systems Technology Division since he joined HP in 1988. His past HP contributions include work on scan conversion chips used in the Turbo-VRX and HCRX48Z graphics subsystems and various other chips used in the CRX and HCRX graphics subsystems. He is currently responsible for investigating graphics features for possible inclusion in future HP workstations. Before coming to HP Tony worked on designing hardware for computer graphics workstations while at Applicon Incorporated. His work has resulted in five U.S. patents, four U.S. patents pending, and two international patents pending, all relating to computer graphics hardware, including color recovery. He is a member of the IEEE and has published several technical articles related to computer graphics. He is married, has a son and a daughter and enjoys everything from short weekend family trips to the Colorado mountains to longer journeys for snorkeling in the warm Caribbean waters.

60 Real-Time MPEG Decoding

Ruby B. Lee



Ruby Lee serves as chief architect for HP's interdivisional multimedia architecture team and senior technical contributor in the computer systems architecture group in HP's Computer Systems Organization. She is currently leading the HP-Intel multimedia task force and continues to work on architecture evolution in the PA-RISC extensions team. She is also a consulting associate professor of electrical engineering at Stanford University. She joined HP Laboratories in 1981 and was one of the original architects of PA-RISC. She was the lead designer of the first CMOS PA-RISC processor chip and manager of subsequent microprocessor designs. Ruby attended Cornell University, receiving a BA degree with distinction in computer science and comparative literature. She continued her studies at Stanford University, earning an MS in computer science and a PhD in electrical engineering in 1980. Before joining HP she served as assistant professor of electrical engineering at Stanford University. Ruby designed the PA-RISC multimedia enhancements and led the multimedia architecture team that coordinated work across four HP sites involved with real-time MPEG decoding in software. Her work has resulted in seven patents and several more pending in the areas of processor architecture, pipeline implementations, branch optimizations, cache hints, assists architecture, and multimedia. She has authored or coauthored approximately thirty papers on subjects related to PA-RISC architecture, multimedia, parallel processing, VLSI, and computer performance. She is a member of the IEEE and the ACM. Ruby is married, has two children, and enjoys skiing, aerobics, windsurfing, and classical music.

John P. Beck

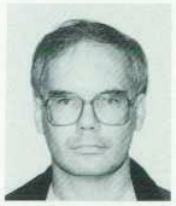
John Beck is an engineer/scientist with the Workstation Technology Division. He joined HP when Apollo was acquired by HP in 1989. He has worked on digital video conferencing and the CRX graphics system. Before coming to HP he was with Digital Equipment Corporation and Apollo where he worked on high-performance CPUs and workstations. His work has resulted in two patents pending related to parallel subword averaging and digital video compression. He was a key contributor to the software MPEG project, and he did the initial software optimizations.

Joel Lamb



Joel Lamb has been a member of the technical staff at HP's Systems Technology Division since 1985 when he first joined HP. Born in McCook, Nebraska, he attended the University of Nebraska where he received his BSEE degree in 1983 and his MSEE in 1985. He is currently working on a new PA-RISC processor. He was responsible for implementing the multimedia instructions for the PA 7100LC processor. His previous work at HP includes work on the circuit design for several PA-RISC processors and design of an integer ALU, integer shifter, and integer data path. He is a member of the IEEE and his work has resulted in four patents related to VLSI circuit designs. Joel is married and has two sons.

Kenneth E. Severson



A project manager with the Workstation Systems Division, Ken Severson is presently responsible for imaging software at the graphics software laboratory in Ft. Collins, Colorado. He came to HP in 1989 as an Apollo employee working on 3D graphics. He was responsible for leading the group that worked on digital video for the MPEG decompression project in the multimedia lab in Chelmsford, Massachusetts. Born in Grand Forks, North Dakota, he attended the University of North Dakota where he earned a BS degree in physics and mathematics in 1974 and an MS in physics in 1977. Before working at Apollo or HP, he worked on software development in computer-aided design and manufacturing at Si-

© Copr. 1949-1998 Hewlett-Packard Co.

korsky Aircraft and Applicon Inc.. Ken is a member of the ACM and SIGGRAPH and is interested in graphics and multimedia. He was a principal developer of two ray-traced computer animations that debuted at SIGGRAPH. He is married, has two children, and enjoys cross-country skiing, ice skating, and bicycling.

69 HP TeleShare

S. Paul Tucker



Paul Tucker is a member of the technical staff in the graphics hardware laboratory in HP's Workstation Systems Division. He joined HP's Graphics Technology Division in 1988 and is currently responsible for work on HP's next generation of 3D graphics products. He received a BSEE degree (1986) and an MSEE (1987) from Oklahoma State University, and his work has resulted in a patent related to graphics memory organization. His professional interests include graphics and digital signal processing. Born in Tulsa, Oklahoma, he is married and enjoys music, scuba diving, cycling, skiing, golf, personal computers and brewing beer.

75 Product Design Model 712 Workstation

Arlen L. Roesner



Arlen Roesner was born in Glendive, Montana and grew up in Montana's Glacier Park area. He attended Montana State University, earning a BS degree in mechanical engineering with a minor in physics in 1984. He is a product design engineer at the Fort Collins system laboratory of the Systems Technology Division where he is responsible for the mechanical design of the next generation of workstations. With HP since 1984, his previous work includes mechanical design for several HP 9000 Series 700 workstations. He was responsible for the mechanical and thermal design of the Model 712 workstation described in this issue. His work has resulted in two patents related to an EISA/ISA card mounting concept for rear-instrument access, and a mass-storage tray concept for rear-instrument access of multiple drives. He is the author of a previous Journal article on the mechanical design of the HP 9000 Models 720 and 730 workstations. He is married, has two children, and enjoys fishing, woodworking, and computers. He is an active participant in the Fort Collins Vineyard Christian Fellowship and the suicide task force of Larimer County.

79 Multiuser Business Server System

Dennis A. Bowers



A systems manager in the general systems laboratory at the HP Systems Technology Division, Dennis Bowers has been with HP since 1980 when he began work at the HP Computer Support Division in Roseville, California. He received his BSEET degree from California State Polytechnic University at Pomona in 1978 and an MSECE (electrical and computer engineering) from the University of California at Davis in 1985. Currently responsible for midrange HP 3000 and HP 9000 platforms, Dennis provided project leadership for the low-end business servers described in this issue. Previous contributions at HP include production engineering at both the Computer Support Division and the Computer Systems Division on the HP 1000 and the HP 3000 Series 37 computer systems. He has also worked in different technical development capacities on several HP 3000 and HP 9000 computer systems. Before joining HP he worked at Odetics Inc. in Anaheim, California as a lab technician, Computer Automation's Automatic Test Division in Irvine, California as a field support engineer, and Alpha Microsystems in Irvine, California as an R&D engineer. He is a member of the IEEE and the ACM. Born in Reading, Pennsylvania, he is married and has two sons. His favorite hobby is to tinker in his garage (usually accompanied by both sons). He enjoys auto restoration (currently working on a 1934 Ford 5-window coupe), is a licensed ham radio operator (KB6HLC), and is active in his church and involved with several bible study groups. He also serves as an assistant scout master for the Boy Scouts of America.

Gerard M. Enkerlin



Gerard Enkerlin was born in San Luis Potosi, Mexico and attended the University of North Dakota where he received a BS degree in engineering management and electrical engineering in 1986. He continued his studies at the University of Texas, earning an MS in manufacturing engineering in 1989, joining HP's Network Computer Manufacturing Operations that same year. He is presently a manufacturing program engineer responsible for new product introduction for new low-end servers. Previous experience at HP includes work as a product engineer, process engineer, and program manager for various business server projects. Before coming to HP he worked for Delco Products of General Motors as a master scheduler and manufacturing engineer. He is interested in manufacturing and distribution modeling and manufacturing optimization analysis. Gerard is married and enjoys golf, soccer, skiing, traveling, reading, and movies. He is also fluent in Spanish, German, and English.

Karen L. Murillo



An R&D project manager at HP's System Technology Division, Karen Murillo joined HP in 1979 at the Computer Systems Division. She was a design engineer on the HP 3000 Series 64 memory subsystem, HP 3000 Series 37 hardware simulation, and the HP 3000 Series 48/58 computer systems. She was also a project manager for the HP 3000 Series 922 to 958 and the HP 9000 Series 822 to 852 computer systems. Her responsibilities for the project reported in this issue were as project manager for the HP 3000 Series 8x7 and HP 9000 Series Ex5 computer systems. Karen is currently a project manager for future low-end multiuser server systems. She received a BSEE degree in computer engineering from Iowa State University in 1979. She authored an HP Journal article on testability in 1982. Born in Iowa, she is married and has a son and two golden retrievers. She enjoys woodworking, playing soccer and basketball, and coaching her son's soccer team. Karen also participates in bringing computer and electronics technology into her son's third-grade classroom.

85 HP Distributed Smalltalk

Eileen Keremitsis



Eileen Keremitsis is a publications consultant with Rhinoceros Consulting of San Francisco. She received a BA degree in Latin American studies from the University of California at Berkeley in 1973, an MA in history and economic development from Columbia University in 1976, and a PhD (Fullbright Scholar) in Latin American history from Columbia University in 1982. She worked with the HP Distributed Smalltalk team from 1993 to 1994 creating all user documentation for the first three major releases and providing all learning products support. She has worked with object-oriented software for eight years and has been involved with technical publications for ten years. She previously taught history at the University of Maine and the University of North Carolina at Charlotte. She is a senior member of the Society for Technical Communication and a member of the ACM and the IEEE Computer Society.

Ian J. Fuller



Ian Fuller is a marketing program manager at the Software Engineering Systems Division. He joined HP in 1980 with the Office Systems Division in Pinewood near Reading, England. He received a BS degree in physical and computer

© Copr. 1949-1998 Hewlett-Packard Co.

sciences from Oxford Polytechnic in 1978. As a product manager, Ian is presently engaged in consulting and marketing for HP Distributed Smalltalk. Previous contributions at HP include project management for HP DeskManager, software engineering for the HP Cooperative Computing Center, and project management for HP NewWave. He was also a software architect for the Information Architecture Group, marketing product manager at the Open Systems Software Division, and the HP representative on the Object Management Group's technical committee. Before joining HP Ian was with ITT Business Systems in London, England where he programmed computer message switch software. He is interested in distributed systems and object-oriented systems for business applications and has published two previous articles in the HP Journal concerning HP DeskManager and NewWave. He was born in Gosport, Hampshire, England, is married, and enjoys travel (especially to Asia) and photography.

93 Software Solution Broker

Manny Yousefi



Manny Yousefi came to HP's Manufacturing Productivity Division in 1988 and is now a project manager with the Professional Services Division in Mountain View, California. He was the project manager for development of the Software Solution Broker reported in this issue and is one of its coarchitects. He was also responsible for the negotiation and procurement of third-party product licenses needed for various Software Solution Broker product features. He received a BS degree in computer science from the California Polytechnic State University in 1983. His previous accomplishments at HP include work as an architecture team member for an object-oriented manufacturing application and work on an object-oriented text management system. Before coming to HP, Manny was with U.S. Sprint as a senior software engineer and project manager and with Memorex Corporation as a software engineer. He is interested in object-oriented development and methodologies and distributed client/server applications. In his free time, he enjoys gardening.

Adel Ghoneimy



Adel Ghoneimy is a freelance software consultant and has been working with HP's Professional Services Division since 1993. He is responsible for the architecture and the ongoing development of the Software Solution Broker reported in this issue. He holds a BS degree in computer science and automatic control from Alexandria University, Egypt and has an MS degree in computer science from the University of Minnesota. Before becoming an HP contractor, Adel was with Digital Equipment Corporation where he led the development of a workflow system and with Honeywell Corporation where he managed large software development efforts and contributed to several projects in the area of distributed systems applied research and advanced development. At Honeywell, he worked on problems of systems engineering for distributed automation and control systems. He has published applied research articles in the areas of workflow, object management, and instrumentation for distributed applications. His work has produced two patents concerning long-lived transactions and workflow management systems. He is interested in business process engineering, object-oriented development and methodologies, distributed client/server applications, and transactional workflow management systems. He was born in Cairo, Egypt and enjoys ballroom dancing.

Wulf Rehder



Wulf Rehder came to HP's Systems Technology Division in 1986 and is now the curriculum manager for object technology with HP's Professional Services Division. He received a BS degree in mathematics and physics from Hamburg University in 1969, an MS in mathematics and statistics from Dortmund University in 1972, and a PhD in mathematics from Berlin Technical University in 1978. He is currently responsible for the development and deployment of customer education courses in object technology. He participated in the discussions of the development team for the Software Solution Broker reported in this issue. His previous contributions at HP include simulation and modeling for PA-RISC computers, work as a project manager at the HP Laboratories Pisa Science Center in Italy, and the creation of test algorithms for field programmable gate arrays at HP Laboratories. Before joining HP, he was a performance manager at Metaphor Computer Systems and a mathematics professor at the California State University at San Jose. A prolific author, Wulf has published 36 scientific papers on mathematics, physics, and engineering, another 50 articles for various magazines and newspapers, and a book. He is interested in object-oriented methods, mathematical tools, and "making difficult things appear simple." He was born in a small village in northern Germany, is married, and has two children. He enjoys writing for several literary magazines and is a contributing editor for the Bloomsbury Review and a contributing writer for the Boston Book Review.

102 Voltage Contrast Imaging

Jack D. Benzel



Jack Benzel is a member of the technical staff at the Integrated Circuits Business Division (ICBD) in Fort Collins, Colorado. He is responsible for structured-custom IC design for floating-point math coprocessors for PA-RISC processors. He joined the company in 1990. Beginning in test process engineering in ICBD manufacturing, he developed an off-line wafer inking system before moving to the ICBD design lab three and a half years ago. Jack holds a BSEE degree from Colorado State University (1985) and was with IBM at Boulder, Colorado for four years before joining HP. He served as a design engineer for the voltage contrast imaging project reported in this issue. He is married, has three young children, and enjoys time with his family, camping, and music and drama performance at church.

107 Design for Testability

Bulent I. Dervisoglu



Before leaving HP recently to join Silicon Graphics, Bulent Dervisoglu was a senior consulting engineer at HP's Advanced Systems Division. He received his BSEE degree from the Middle East Technical University in Ankara, Turkey, in 1969 and his PhD degree in computer science from the University of Edinburgh, Scotland, in 1973. He taught and conducted research at several universities in Europe and the U.S.A. before joining the Sperry Corporate Research Center in Massachusetts in 1980. Later he joined the MIT Lincoln Laboratories in Lexington, Massachusetts, where his research interests included built-in self-test and design verification. He joined Apollo Computer in 1989 and became an HP employee with HP's 1989 acquisition of Apollo. Bulent has published extensively in various IEEE technical journals and conferences and is the chairman of the IEEE P1149.2 working group for developing a flexible and high-performance-oriented version of boundary-scan architecture.

Michael Ricchetti



Mike Ricchetti is a member of the consulting staff at Cadence Design Systems where he is responsible for design for testability (DFT) methodology and tools. He earned a BS degree in computer engineering at Ohio State University in 1982. Previously, he was with Hewlett-Packard and Apollo, where he worked on design for testability and chip test development with the Workstation Systems Division. He was with HP-Apollo for 10 years and was the DFT engineer for the memory controller ASIC reported in this issue. Before joining Apollo in 1985, he was a diagnostics engineer at Digital Equipment Corporation. Mike is an IEEE member and participates on the IEEE P1149.2 Testability Standards Committee working group.

© Copr. 1949-1998 Hewlett-Packard Co.

HEWLETT-PACKARD
JOURNAL

April 1995 Volume 46 • Number 2

Technical Information from the Laboratories of
Hewlett-Packard Company

Hewlett-Packard Company, P.O. Box 51827
Palo Alto, California, 94303-0724 U.S.A.

HEWLETT® PACKARD

5963-5387E

© Copr. 1949-1998 Hewlett-Packard Co.