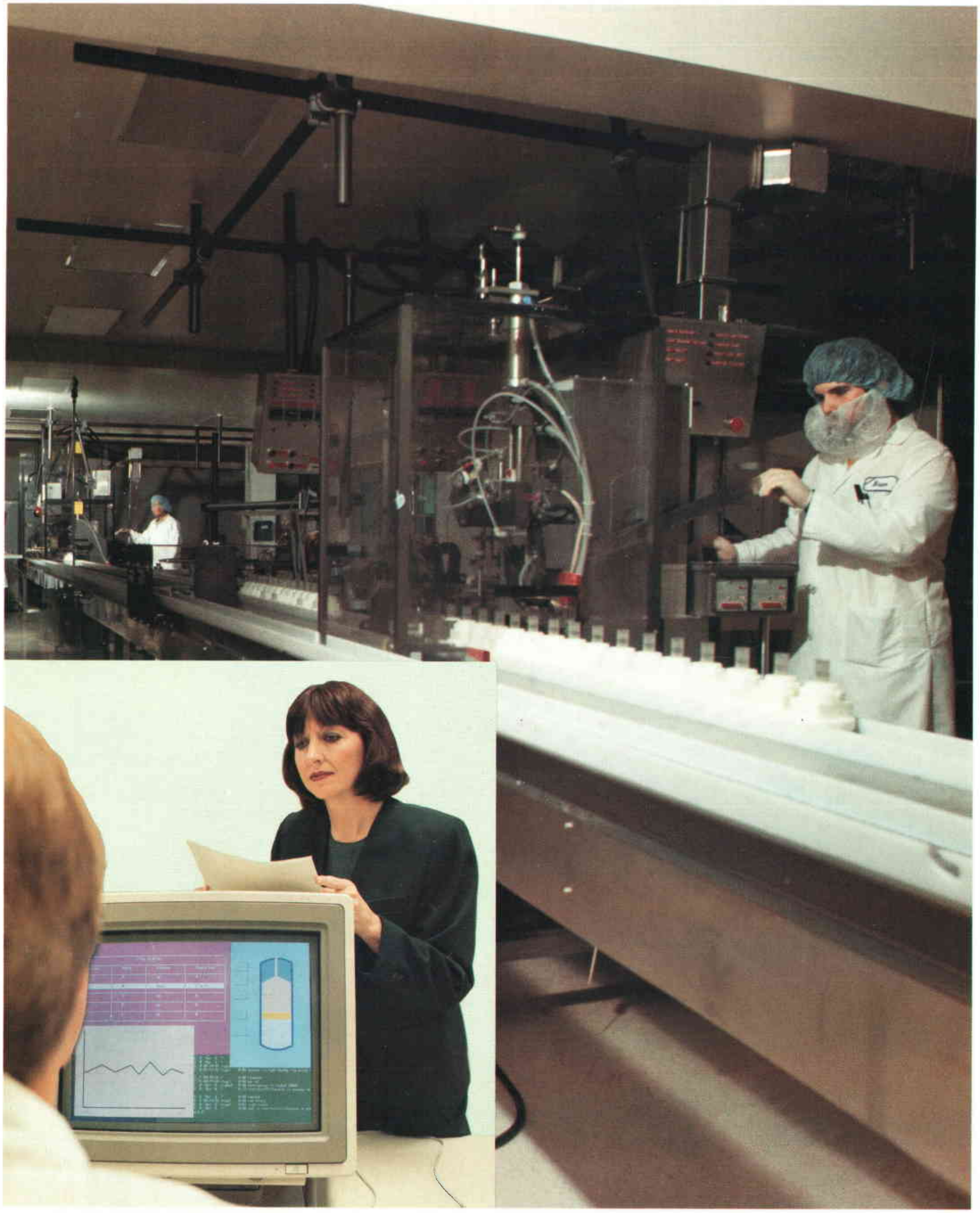
HEWLETT-PACKARD JOURNAL

JUNE 1989



HEWLETT PACKARD

HEWLETT-PACKARD JOURNAL

June 1989 Volume 40 • Number 3

Articles

Editor, Richard P. Dolan • Associate Editor, Charles L. Leath • Assistant Editor, Hans A. Toepfer • Art Director, Photographer, Arvid A. Danielson
Support Supervisor, Susan E. Wright • Administrative Services, Typography, Anne S. LoPresti • European Production Supervisor, Sonja Wirth

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

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

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

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

In this Issue

Computer programs for data base management usually use magnetic disc as their primary data storage medium and enforce rigid protocols to guarantee data consistency and integrity. While these are highly desirable features for most applications, they are not without cost. If many transactions occur in a short time, the system's response to an individual transaction may be slow, or even worse, unpredictable. This isn't acceptable for real-time applications—high-speed production lines, for example. HP Real-Time Data Base, a data base management system for real-time applications running on HP 9000 Series 300 and 800 Computers, is designed for predictable response time and high speed. It's a memory-resident system, using main memory as its primary data storage medium, and it allows the user to disable unnecessary features to increase performance. Tests have shown that using direct access (one of three methods), HP Real-Time Data Base can retrieve data at a rate of 66,666 56-byte records per second. The article on page 6 describes the design of this system and tells how choosing the right design alternatives led to its high performance.

As long as you know how they were measured, MIPS (millions of instructions per second) and MFLOPS (millions of floating-point operations per second) can be useful measures of the relative performance of computer system processing units (SPUs). The new SPU for HP 9000 Model 835 technical computers and HP 3000 Series 935 commercial computers has been tested at 14 MIPS and 2.02 MFLOPS running particular benchmark programs (see the footnote on page 19). This represents more than a 300% increase in floating-point performance and more than a 50% increase in integer performance over this SPU's predecessor, the Model 825/Series 925 SPU. Responsible for these increases are processor design improvements and a new floating-point coprocessor, as explained in the article on page 18. A new 16M-byte memory board was also designed and is manufactured using an advanced double-sided surface mount process, described on page 23.

Half-inch reel-to-reel tape drives are widely used for backing up large disc memories in computer systems. Desirable characteristics are high speed for minimum backup time and large reel capacity so that fewer reels have to be handled and stored. The HP 7890XC Tape Drive uses a sophisticated data compression scheme to increase reel capacity, as explained in the article on page 26. It also uses a complementary technique called super-blocking to deal with certain features of its industry-standard 6250 GCR tape format that tend to limit the capacity improvement possible with data compression alone. Super-blocking is explained in the article on page 32. Using both data compression and super-blocking, the HP 7980XC has achieved capacity improvements of 2.5 to 5 times, depending on the data.

High-speed fiber optic communications systems are made up of four basic types of components. For example, there are amplifiers, which have electrical inputs and electrical outputs, laser diodes, which have electrical inputs and optical (light) outputs, photodiodes, which have optical inputs and electrical outputs, and optical fibers, which have optical inputs and optical outputs. Accurate measurements of the transmission and reflection characteristics of all of these device types, needed by both component designers and system designers, are provided by HP 8702A Lightwave Component Analyzer systems. Each system consists of a lightwave source, a lightwave receiver, the HP 8702A analyzer, and for reflection measurements, a lightwave coupler. In the article on page 35, you'll find a description of these systems and a comprehensive treatment of their applications and performance. The design of the lightwave sources and receivers is presented in the article on page 52. A comparison of the reflection measurement capabilities of the HP 8702A and the HP 8145A Optical Time-Domain Reflectometer (December 1988) appears on page 43.

Videoscope, the subject of the article on page 58, is a combination of hardware and software that automates the testing of application software for HP Vectra Personal Computers. While a test is being run manually, Videoscope records the human tester's keystrokes and mouse movements, and with the human tester's approval, the correct responses of the application being tested. It can then rerun the test automatically. Unlike other similar testers, Videoscope doesn't affect the performance or behavior of the application being tested. The key to this difference is the hardware, a plug-in card that nonintrusively monitors the video signal of the system running the application being tested and, for each screen, develops a single-number representation called a signature. Signature analysis isn't new, having been used for many years for troubleshooting digital hardware, but its adaptation to software testing is an ingenious and elegant solution to the problem of capturing screens. (Videoscope is an in-house HP tool, not a product.)

A lot of research has been based on the conjecture that if we could simulate the human brain's basic elements—neurons—on a computer, we could connect a bunch of them in a network, and we might be able to solve some of the problems that regular computers find difficult but the brain handles with ease. This approach has met with some success, particularly with certain optimization problems. The theory of neural networks is expressed in differential equations, and its application to practical problems is not intuitive. Seeking and not finding a simpler, higher-level method of determining the right neuron interconnections, gains, and component values to solve a given problem, Barry Shackleford of HP Laboratories developed one. In the paper on page 69, he explains his approach and applies it to several classic optimization problems such as the traveling salesman problem and the eight queens problem.

While we usually think of metal as something very stable, engineers and physicists who deal with integrated circuit chips know that a high enough current density in a thin metal film will cause the metal atoms to move. Over long periods of time, the metal piles up in some places and leaves holes in other places, causing chip failures. Although electromigration has been studied extensively, we still don't have a complete mathematical theory for it. The paper on page 79 reports on a new two-dimensional mathematical model that makes it possible to simulate electromigration with good accuracy on a computer using exclusively classical physics, not quantum mechanics. The model was developed jointly by scientists at HP Laboratories and the California State University at San Jose.

R.P. Dolan
Editor

Cover

One of the potential application areas for the HP Real-Time Data Base is in computer integrated manufacturing, where data such as the status of each station on a manufacturing line can be monitored in real time for quality control purposes. The picture shows a veterinary bolus (large pill) assembly line at the ALZA Corporation in Palo Alto, California. ALZA Corporation researches, develops, and manufactures, and markets drug delivery systems. ALZA Director of Quality Assurance Carol L. Hartstein is shown in the inset photo with a simulated monitor screen. Our thanks to ALZA Corporation for helping us illustrate this application.

What's Ahead

In the August issue we'll bring you the designers' view of the HP NewWave environment, HP's state-of-the art user interface for personal computers. The evolution of an existing quarter-inch tape drive into the HP 9145A with twice the speed and twice the cartridge capacity will also be featured.

Data Compression in a Half-Inch Reel-to-Reel Tape Drive

A proprietary data compression algorithm implemented in a custom CMOS VLSI chip improves the throughput and data capacity of the HP 7980XC Tape Drive.

by Mark J. Bianchi, Jeffery J. Kato, and David J. Van Maren

HP 7980 TAPE DRIVES are industry-standard, half-inch, reel-to-reel, streaming tape drives that operate at 125 inches per second, have automatic tape loading, and can be horizontally rack-mounted for better floor space utilization.[1] They are available in a variety of configurations and support three industry-standard tape formats: 800 NRZI, 1600 PE, and 6250 GCR.

The HP 7980XC Tape Drive is a new member of this family. Its special contribution is its use of a sophisticated real-time data compression scheme that provides extended performance to the 6250 GCR format.

The implementation of data compression in the HP 7980XC involves two different but complementary components. The first component is the data compression engine. This engine resides in the HP 7980XC and consists of a proprietary integrated circuit and support circuitry. The second component is a packing process, referred to as *super-blocking*, that is performed on the data packets that have been compressed by the compression engine. Super-blocking is performed in the firmware that resides in the HP 7980XC drive. When these two components are combined, the resulting compression scheme provides high *tape compaction*. Tape compaction is a figure of merit for compression performance. It is the ratio of the amount of tape used by a standard 6250 GCR half-inch tape drive to that used by the HP 7980XC in compression mode. It is a higher ratio than that for compression alone, since super-blocking provides additional tape savings. This article addresses the design and implementation of data compression in the HP 7980XC. For more detailed information on super-blocking, see the article on page 32.

The Data Compression Engine

The performance improvement in the HP 7980XC is provided by the Hewlett-Packard data compression (HP-DC) subsystem. This subsystem can both compress and decompress the data being passed through it. Fig. 1 shows how the HP-DC engine fits architecturally into the data path of the HP 7980XC Tape Drive. The data compression or decompression occurs between the interface hardware and the cache buffering hardware. When data is written to the tape drive, it flows from the interface to the HP-DC subsystem where it is compressed and packed, and then proceeds to the cache buffer, where it is queued to be written to the tape. Conversely, when data is read from the tape drive, it proceeds from the buffer to the HP-DC subsystem, where it is unpacked and decompressed, and then to the interface and the host computer.

Data Compression Development

Development of the Hewlett-Packard data compression algorithm began at HP Laboratories, where the basic data structures for the algorithm were developed. Years of work culminated in an algorithm design that is similar to the widely known public-domain version of the Lempel-Ziv algorithm,[2,3] but offers distinct advantages. It is adaptive, and it is more flexible and offers better performance than the public-domain Lempel-Ziv scheme.

The HP-DC algorithm was presented to the Greeley Storage Division in the form of an algorithm-based Pascal program. To realize this algorithm in silicon, a number of changes were made to the program so that, once implemented in hardware, the algorithm would still provide the high throughput needed by the HP 7980XC's high-speed data path. A state-machine simulator was used to benchmark the performance of the integrated circuit and verify the data integrity of the algorithm. This simulator was then used to architect and design the proprietary IC.

The Data Compression Algorithm

The underlying principle behind data compression is the removal of redundancy from data. The HP-DC scheme performs this by recognizing and encoding patterns of input characters. Each time a unique string of input characters occurs, it is entered into a dictionary and assigned a numeric value. Once a dictionary entry exists, subsequent occurrences of that entry within the data stream can be replaced by the numeric value or codeword. It should be noted that this algorithm is not limited to compressing ASCII text data. Its principles apply equally well to binary files, data bases, imaging data, and so on.

Each dictionary entry consists of two items: (1) a unique string of data bytes that the algorithm has found within the data, and (2) a codeword that represents this combination of bytes. The dictionary can contain up to 4096 entries.

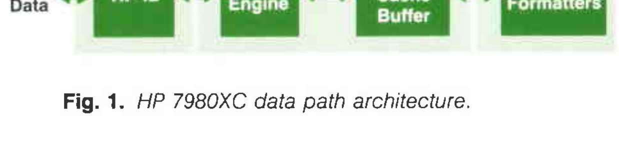


Fig. 1. *HP 7980XC data path architecture.*

The first eight entries are reserved codewords that are used to flag and control specific conditions. The next 256 entries contain the byte values 0 through 255. The remaining locations are linked-list entries that point to other dictionary locations and eventually terminate by pointing at one of the byte values 0 through 255. Using this linked-list data structure, the possible byte combinations can be anywhere from 2 bytes to 128 bytes long without requiring an excessively wide memory array to store them.

In the hardware implementation of the HP-DC scheme, the dictionary is built and stored in a bank of random-access memory (RAM) that is 23 bits wide. Each memory address can contain a byte value in the lower 8 bits, a codeword or pointer representing an entry in the next 12 bits, and three condition flags in the upper 3 bits. The codewords range in length from 9 bits to 12 bits and correspond to dictionary entries that range from 0 to 4095. During the dictionary building phase, the first 512 entries have 9-bit codewords, the next 512 entries have 10-bit codewords, the next 1024 entries have 11-bit codewords, and the final 2048 entries have 12-bit codewords. Once the dictionary is full, no further entries are built, and all subsequent codewords are 12 bits in length. The memory address for a given dictionary entry is determined by a complex operation performed on the entry value. Since the dictionary can contain 4096 entries, it would appear that 4K bytes of RAM is all that is needed to support a full dictionary. However, in practice, more than 4K bytes of RAM is needed because of dictionary "collisions" that occur during the dictionary building phase. When a dictionary collision occurs, the two colliding values are recalculated to two new locations and the original location is flagged as a collision site.

An important property of the algorithm is the coupling between compression and decompression. In the HP-DC IC, these two operations are tied together both in the compression and decompression processes and in the packing and unpacking of codewords into a byte stream. The nature of the compression algorithm requires that the compression process and the decompression process be synchronized. Stated differently, decompression cannot begin at an arbitrary point in the compressed data. It begins at the point where the dictionary is known to be empty or reset. This coupling provides one of the fundamental advantages of the HP algorithm, namely that the dictionary is embedded in the codewords and does not need to be transferred with the compressed data. Similarly, the packing and unpacking process must be synchronized. This implies that compressed data must be presented to the decompression hardware in the proper order.

A Data Compression Example

Fig. 2 is a simplified graphical depiction of the compression algorithm implemented in the HP-DC compression engine. This example shows an input data stream composed of the following characters: R I N T I N T I N. To follow the flow of the compression process, Fig. 2 should be viewed from the top to the bottom, starting at the left and proceeding to the right. It is assumed that the dictionary has been reset and initialized to contain the first 256 entries of 0 to 255. The dictionary must always be initialized in this way to satisfy the requirements of the algorithm's data structure.

The compression algorithm executes the following process with each byte in the data stream:

1. Get the input byte.
2. Search the dictionary with the current input sequence and, if there is a match, get another input byte and add it to the current sequence, remembering the largest sequence that matched.
3. Repeat step 2 until no match is found.
4. Build a new dictionary entry of the current "no match" sequence.
5. Output the codeword for the largest sequence that

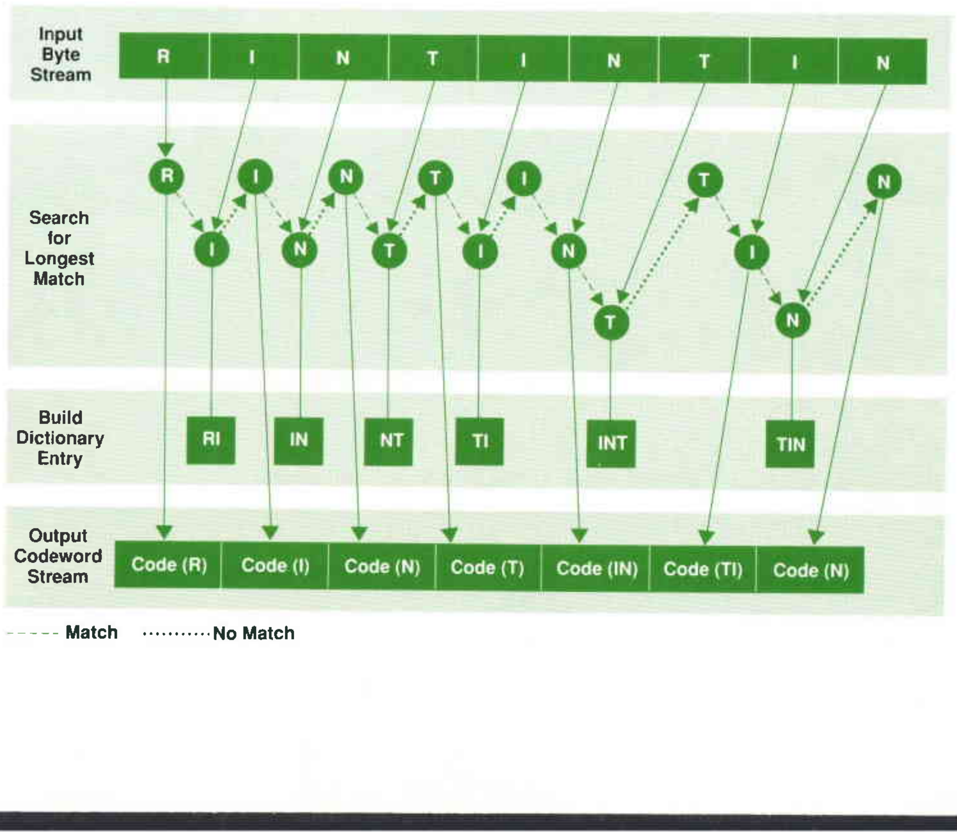


Fig. 2. *Compression algorithm example.*

matched. The following lines of code are an algorithmic representation of these steps:

```
current_byte_sequence := GET_INPUT_BYTE;

REPEAT

    REPEAT
        matched := SEARCH_DICTIONARY(current_byte_sequence,
                                     returned_codeword);
        IF (matched = TRUE) THEN
            BEGIN
            longest_byte_sequence := current_byte_sequence;
            longest_codeword := returned_codeword;
            current_byte_sequence := current_byte_sequence +
                                               GET_INPUT_BYTE;
            END;
    UNTIL (matched = FALSE);
    BUILD_DICTIONARY(current_byte_sequence);
    OUTPUT_CODEWORD(longest_codeword);
    current_byte_sequence := current_ byte_sequence -
                                     longest_byte_sequence;

UNTIL (no more input bytes to compress);
```

In this example, the compression algorithm begins after the first R has been accepted by the compression engine. The input character R matches the character R that was placed in the dictionary during its initialization. Since there was a match, the engine accepts another byte, this one being the character I. The sequence RI is now searched for in the dictionary but no match is found. Consequently, a new dictionary entry RI is built and the codeword for the largest matching sequence (i.e., the codeword for the character R) is output. The engine now searches for I in the dictionary and finds a match just as it did with R. Another character is input (N) and a search begins for the sequence IN. Since IN does not match any entries, a new one is built and the codeword for the largest matching sequence (i.e., the codeword for the character I) is output. This process continues with a search for the letter N. After N is found, the next character is input and the dictionary is searched for NT. Since this is not found, a dictionary entry for NT is built and the codeword for N is output. The same sequence occurs for the characters T and I. A codeword for T is output and a dictionary entry is built for TI.

Up to this point, no compression has occurred, since there have been no multiple character matches. In actuality, the output stream has expanded slightly, since four 8-bit characters have been replaced by four 9-bit codewords. (That represents a 32-bit to 36-bit expansion, or a 1.125:1 expansion ratio.) However, after the next character has been input, compression of the data begins. At this point, the engine is searching for the IN sequence. Since it finds a match, it accepts another character and begins searching for INT. When it doesn't find a match, it builds a dictionary entry for INT and outputs the previously generated codeword for the sequence IN. Two 8-bit characters have now been replaced by one 9-bit codeword for a compression ratio of 16/9 or 1.778:1.

This process continues and again two characters are replaced with a single codeword. The engine begins with a T from the previous sequence and then accepts the next character which is an I. It searches for the TI sequence and finds a match, so another byte is input. Now the chip is searching for the TIN sequence. No match is found, so a TIN entry is built and the codeword for TI is output. This sequence also exhibits the 1.778:1 compression ratio that the IN sequence exhibited. The net compression ratio for this string of 9 bytes is 1.143:1. This is not a particularly large compression ratio because the example consists of a very small number of bytes. With a larger sample of data, more sequences of data are stored and larger sequences of bytes are replaced by a single codeword. It is possible to achieve compression ratios that range from 1:1 up to 110:1. The performance section of this article presents measured compression ratios for various computer systems and data types.

A simplified diagram of the decompression process implemented in the HP-DC IC is shown in Fig. 3. This example uses the output of the previous compression example as input. The decompression process looks very similar to

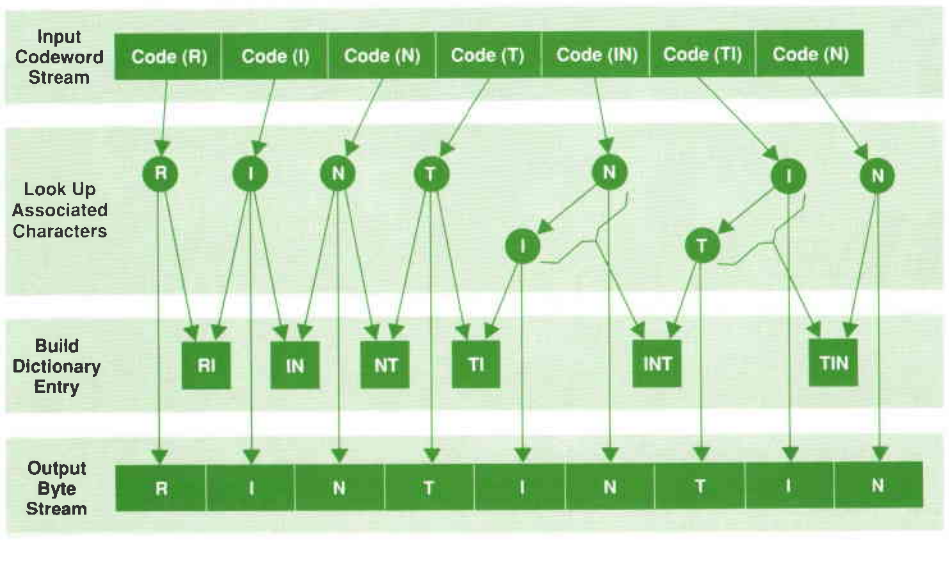


Fig. 3. *Decompression algorithm example.*

the compression process, but the algorithm for decompression is less complicated than that for compression, since it does not have to search for the presence of a given dictionary entry. The coupling of the two processes guarantees the existence of the appropriate dictionary entries during decompression. The algorithm simply uses the input codewords to look up the byte sequence in the dictionary and then builds new entries using the same rules that the compression algorithm uses. This is the only way that the decompression algorithm can recover the compressed data without a special dictionary sent with each data packet. The following lines of code represent the algorithm used by the decompression process:

```
current_codeword := GET_INPUT_CODEWORD;
REPEAT
    codeword := current_codeword;
    REPEAT
        byte := LOOKUP_DICTIONARY(codeword);
        PLACE_BYTE_ON_OUTPUT_STACK(byte);
        FIND_NEXT_ENTRY_IN_LIST(codeword,pointer_to_next_
            entry);
        codeword := pointer_to_next_entry;
    UNTIL (codeword points to tail of list one of bytes 0-255);

    BUILD_DICTIONARY(previous_codeword,byte);

    REPEAT
        output_byte := POP_ BYTE_FROM_OUTPUT_STACK;
        OUTPUT_BYTE(output_byte);
    UNTIL (stack is empty);

    previous_codeword := current_codeword;
    current_codeword := GET_INPUT_ CODEWORD;

UNTIL (no more input codewords to decompress);
```

As in the compression example, it is assumed that the dictionary has been reset and initialized to contain the first 256 entries of 0 to 255. The decompression engine begins by accepting the codeword for R. It uses this codeword to look up the byte value R. This value is placed on the last-in, first-out (LIFO) stack, waiting to be output from the chip. Since the R is one of the root codewords (one of the first 256 entries), the end of the list has been reached for this codeword. The output stack is then dumped from the chip. The engine then inputs the codeword for I and uses this to look up the byte value I. Again, this value is a root codeword, so the output sequence for this codeword is completed and the byte value for I is popped from the output stack. At this point, a new dictionary entry is built using the last byte value that was pushed onto the output stack (I) and the previous codeword (the codeword for R). Each entry is built in this manner and contains a byte value and a pointer to the next byte in the sequence (the previous codeword). A linked list is generated in this manner for each dictionary entry.

The next codeword is input (the codeword for N) and the process is repeated. This time an N is output and a new dictionary entry is built containing the byte value N and the codeword for I. The codeword for T is input, causing a T to be output and another dictionary entry to be built. The next codeword that is input represents the byte sequence IN. The decompression engine uses this codeword to reference the second dictionary entry, which was generated earlier in this example. This entry contains the byte value N, which is placed on the output stack, and the pointer to the codeword for I, which becomes the current codeword. This new codeword is used to find the next byte (I), which is placed on the output stack. Since this is a root codeword, the look up process is complete and the output stack is dumped in reverse order, that is, I is output first, followed by N. The same process is repeated with the next two codewords, resulting in the recovery of the original byte sequence R I N T I N T I N.

Data Compression Hardware

Fig. 4 shows a block diagram of the HP-DC engine subsystem. The heart of the engine is a custom VLSI chip developed using a proprietary HP CMOS process. This chip can perform both compression and decompression on the data presented to it. However, only one of the two processes (compression or decompression) can be performed at any one time. Two first-in, first-out (FIFO) memories are located at the input and the output of the chip to smooth out the rate of data flow through the chip. The data rate through the chip is not constant, since some data patterns will take more clock cycles per byte to process than other patterns. The instantaneous data rate depends upon the current compression ratio and the frequency of dictionary entry collisions, both of which are dependent upon the current data and the entire sequence of data since the last dictionary reset. The third section of the subsystem is a bank of static RAM that is used for local storage of the current dictionary entries. These entries contain characters, codeword pointers, and control flags.

Fig. 5 shows a block diagram of the HP-DC integrated circuit. The HP-DC chip is divided into three blocks: the input/output converter (IOC), the compression and decompression converter (CDC), and the microprocessor interface (MPI). These blocks are partitioned for effective management of the boundary conditions of the algorithm. Each block is well-defined and the coupling between blocks is

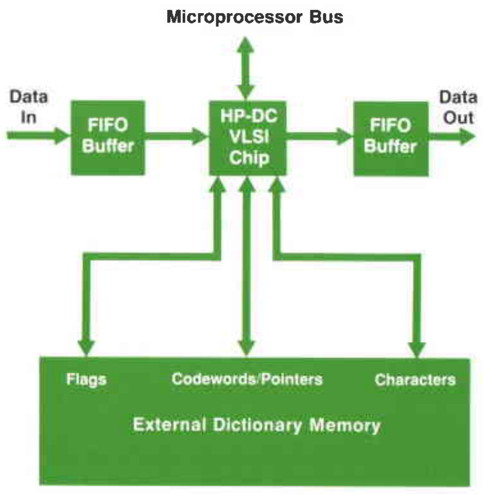


Fig. 4. *HP-DC engine block diagram.*

very low. As a result, each of the blocks runs independently of the other two. This results in maximum chip performance.

The MPI section provides facilities for controlling and observing the chip. It contains six control registers, eight status registers, two 20-bit input and output byte counters, and a programmable automatic dictionary reset circuit. The control and status registers are accessed through a general-purpose 8-bit microprocessor interface bus. The control registers are used to enable and disable various chip features and to place the chip into different operating modes (compression, decompression, passthrough, or monitor). The status registers access the 20-bit counters and various status flags within the chip.

During the development of the HP-DC algorithm, it was found that compression ratios could be improved by resetting the dictionary fairly frequently. This is especially true if the data stream being compressed contains very few similar byte strings. Frequent dictionary resets provide two important advantages. First, resetting the dictionary forces the codeword length to return to 9 bits. Second, new dictionary entries can be made that reflect the present stream of data (a form of adaption). The HP-DC chip's interface section contains circuitry that dynamically monitors the compression ratio and automatically resets the dictionary when appropriate. By writing to an interface control register, this circuitry can be programmed to reset automatically at a wide range of compression ratio thresholds. Another, faster, reset point when the data is expanding guarantees a better worst-case compression ratio, which in turn provides a level of expansion protection. Most data compression algorithms will expand their output if there is little or no redundancy in the data.

The IOC section manages the process of converting between a byte stream and a stream of variable-length codewords (ranging from 9 bits to 12 bits). Two of the eight reserved codewords are used exclusively by the IOC. One of these codewords is used to tell the IOC that the length of the codewords must be incremented by one. With this function controlled by a codeword in the data stream, the process of incrementing codeword size is decoupled from the CDC section. The IOC operates as an independent pipeline process, thus allowing the CDC to perform compression or decompression without being slowed down by the IOC. Another benefit of using a reserved codeword to increment the codeword size is that any future HP-DC engines that have larger codeword sizes will be backward compatible with this HP-DC engine.

The second reserved codeword alerts the IOC that the next codeword is the last one associated with the current packet of data. From this information, the IOC knows to finish its packing routine and end on a byte boundary. This feature allows compression of multiple input packets into one contiguous output packet while maintaining the ability to decompress this packet into its constituent packets. The IOC is also capable of allowing data to pass straight through from input to output without altering it, and of allowing data to pass through while monitoring the potential compression ratio of the data. These features can be used as another level of expansion protection.

The CDC section is the engine that performs the transformation from uncompressed data to compressed data and vice versa. This section is composed of control, data path, and memory elements that are finely tuned for maximum data throughput. The CDC interfaces with the IOC via two 12-bit buses. During compression, the IOC passes the input bytes to the CDC section, where they are transformed into codewords. These codewords are sent to the IOC where they are packed into bytes and sent out of the chip. Conversely, during decompression the IOC converts the input byte stream into a stream of codewords, then passes these codewords to the CDC section, where they are transformed into a stream of bytes and sent to the IOC. The CDC section also interfaces directly to the external RAM that is used to store the dictionary entries.

The CDC makes use of two reserved codewords. The first is used any time a dictionary reset has taken place. The occurrence of this codeword causes two actions: the IOC returns to the state in which it packs or unpacks 9-bit codewords, and the CDC resets the current dictionary and starts to build a new one. Dictionary resets are requested by the MPI section via microprocessor control or the automatic reset circuitry. The second reserved codeword is generated during compression any time the CDC runs out of usable external RAM while trying to build a new dictionary entry. This event very rarely happens, given sufficient external RAM. However, as the amount of memory decreases, it is more likely that the CDC will encounter too many dictionary collisions and will not be able to build new dictionary entries. With the reduction of external memory and the inevitable increase in dictionary collisions, the data throughput and compression performance will be slightly degraded. The HP-DC chip supports three different memory configurations, so a subsystem cost-versus-performance trade-off can be made with regard to individual system requirements. This full-dictionary codeword is also used during decompression by the CDC to ensure that the

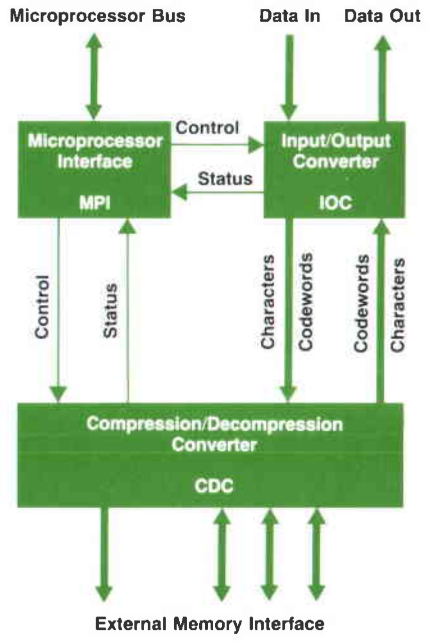


Fig. 5. *HP-DC chip block diagram.*

decompression process stops building dictionary entries at the same point as the compression process.

Compression Performance Results

The two most important performance measures for the HP-DC engine are data throughput and data compression ratio. Throughput performance is measured as the data rate that can be sustained at the uncompressed side of the HP-DC engine (i.e., by the host device). This data rate is primarily dependent upon the compression ratio of the data with some minor dependency upon the data pattern. During compression, the HP-DC engine will have a minimum throughput of 1.0 Mbytes/s and can achieve a maximum of 2.4 Mbytes/s. During decompression, the HP-DC engine will have a minimum throughput of 1.1 Mbytes/s and can achieve a maximum of 2.0 Mbytes/s. The worst-case throughput occurs when the input data is completely random and as a result is expanding. In any case, the compressed data rate is equal to the uncompressed data rate divided by the compression ratio.

The second performance measure and perhaps the most important one is the data compression ratio for various data types. This performance was measured by compressing real user backup data from a variety of computer systems. The table below is a summary of the compression ratios achieved by the HP-DC engine using this data. The test setup included HP 7980A and HP 7980XC half-inch tape drives. All of the test data was copied from various backup tapes to the HP 7980XC in compression mode, then read back and verified while monitoring the compression ratio of the HP-DC engine alone. The article on super-blocking (see page 32) discusses the effects these compression ratios have on the actual tape compaction ratios.

Summary of Data Compression Benchmark Results

Data Description	Volume (Mbytes)	Compression Ratio
MPE/MPE XL on HP 3000s		
Series 68 (HP Desk)	528	3.93
Series 68 (Data Base)	2924	4.31
Series 68 (Misc. Data)	1559	4.30
Series 70 (Manufacturing)	2924	4.31
Series 930 (Code)	311	3.44
HP-UX on HP 9000s		
Series 800 (Commercial HP-UX)	226	2.06
Series 500 (Code)	363	2.38
Series 500 (Data Base)	336	4.07
Series 500 (VLSI)	785	2.52
Series 300 (Archive)	329	2.30
DEC		
DEC VAX (Code)	423	2.31
HP 9000 Running Pascal O.S.		
Series 200 (Misc. Data)	467	2.47
Amdahl		
Amdahl (HP Corporate Data)	5000	3.79

Acknowledgments

The authors wish to thank the rest of the HP 7980XC project team who helped successfully implement the concept of data compression into ½-inch reel-to-reel technology. The team includes manager Mike Tremblay along with Dave Ruska, Virgil Russon, Robert Moss, and Kelly Reasoner. Special thanks goes to Kai Yui and Taher Elgamal at HP Laboratories who presented us with and supported the original algorithm work on data compression. Additional thanks goes to Gordon Thayer, Greg Allen, and Robert Moss along with Chuck McCord and his group at the HP Northwest IC Division for their assistance in implementing data compression in a VLSI chip.

References

1. J.W. Dong, et al, "A Reliable, Autoloading, Streaming Half-Inch Tape Drive," *Hewlett-Packard Journal*, Vol. 39, no. 3, June 1988, pp. 36-42.
2. T.A. Welch, "A Technique for High-Performance Data Compression," *IEEE Computer*, Vol. 17, no. 6, June 1984, pp. 8-19.
3. J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, Vol IT-23, no. 3, May 1977, pp. 337-343.

Maximizing Tape Capacity by Super-Blocking

Interrecord gaps on the tape limit the capacity improvement attainable with data compression in the HP 7980XC Tape Drive. Super-blocking eliminates most of these gaps.

by David J. Van Maren, Mark J. Bianchi, and Jeffery J. Kato

SUPER-BLOCKING is a proprietary Hewlett-Packard method for maximizing half-inch tape data capacity. This capacity improvement is achieved by the removal of some of the interrecord gaps ordinarily placed between host data records. It is performed in real time by the firmware residing in the cache buffer of the HP 7980XC Tape Drive.

To understand how super-blocking works, one must understand the general format of half-inch tapes. When a packet of data is sent from a host to a tape drive, the tape drive must place this packet on the tape in such a way that it can recover the packet and return it to the host exactly as it was received. Normally, physical gaps are placed on the tape between each data record. These gaps, which are areas on the tape containing no flux reversals (and consequently no data), guarantee that the data packets can later be individually recovered. This format of data records with interrecord gaps is required to maintain compatibility with the ANSI standard for 6250 GCR tapes.

A typical host will send data records to the drive in sizes that range from 4K bytes to 32K bytes. Assuming that a tape is written at 6250 bytes per inch, a typical record will be between 0.65 inch and 5.25 inches long. The minimum interrecord gap length is approximately 0.3 inch. From these numbers, one can see that a tape written with 4K bytes of data will contain 69% host data and 31% blank tape. This means that about one third of the tape is wasted by interrecord gaps.

Super-blocking is a formatting technique that removes as many as possible of these capacity-limiting gaps while retaining enough information to separate individual records. This process will pack together as many records as it can without exceeding a maximum super-block length of 60K bytes. Included at the end of each super-block is information that is used in the data retrieval process to separate the super-block into its original records. A graphical illustration of the super-blocking process is shown in Fig. 1.

Fig. 2 demonstrates the effect that decreasing the record size has upon overall tape capacity. As the size of the data records gets smaller, there is a corresponding decrease in the amount of data that can be stored on one tape. The advantage of super-blocking is that it makes the tape capacity independent of record size. The effect of super-blocking is to minimize the portion of tape capacity lost to interrecord gaps. For example, a normal tape written with 16K-byte records will waste 12.5M bytes compared to a super-blocked tape.

What Fig. 2 does not show is the effect on capacity of file marks. A file mark is a special pattern written on the tape that denotes a break in the host data. A file mark uses a very small portion of tape. However, there is an additional gap for each file mark. Because of this extra gap, super-blocking also absorbs all file marks and keeps track of where they were originally located. For simplicity, it is assumed that the ratio of the number of file mark gaps to the number of data record gaps is typically very small. Therefore, the effect on tape capacity of the absorption of file marks will not be considered in this article. One should note that the advantage of super-blocking for increased tape capacity would only improve for each file mark requested by the host.

As explained in the article on page 26, the HP 7980XC Tape Drive is capable of performing data compression on the data that it receives. Referring to Fig. 2, a tape written with 16K-byte records will contain 154M bytes of host data. If this data were compressed by the HP 7980XC and exhibited a compression ratio of 4:1, one would expect the tape capacity to increase by a factor of four to 616M bytes. How-

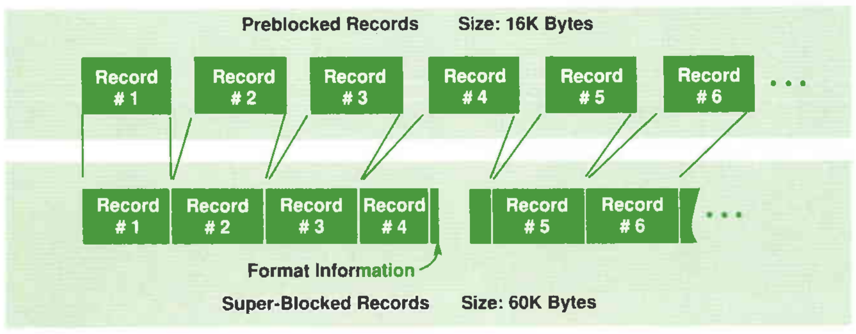


Fig. 1. *The super-blocking process combines normal records into super-blocks as large as 60K bytes.*

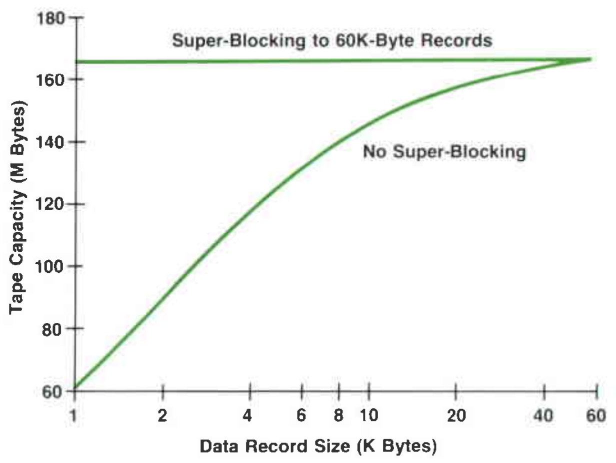


Fig. 2. *Tape capacity versus data record size for a 2400-foot tape written at 6250 bpi with 0.3-inch gaps.*

ever, this is not the case, since only the physical record size is reduced in proportion to the compression ratio. Thus the original 16K-byte records are indeed 4K bytes long after compression, but the expected 616M-byte tape capacity is only 471M bytes, which is 24% less. It is to prevent this effective loss of capacity that super-blocking is needed.

Using the example of 16K-byte records compressed to 4K-byte records, the effect of super-blocking can readily be seen. The compressed 4K-byte records are super-blocked and packed into 60K-byte records instead of being written directly to the tape. This results in a tape capacity of 666M bytes instead of 471M bytes. This is a capacity improvement of approximately 41.5%. By combining data compression with super-blocking, the limitations that the half-inch tape format imposes on data compression are overcome. In addition to obtaining the full benefit of data compression, super-blocking further improves the tape capacity. The table below demonstrates how super-blocking affects this example:

Condition (16K-Byte Input Records)	Tape Capacity (M Bytes)	Tape Compaction
No Data Compression or Super-Blocking	154	1.00:1
Super-Blocking Only	166	1.08:1
4:1 Data Compression Only	471	3.06:1
4:1 Data Compression and Super-Blocking	666	4.32:1

Fig. 3 illustrates the combination of data compression and super-blocking implemented in the HP 7980XC.

Complications

Implementing this concept of super-blocking in the HP 7980XC was made more complex by the constraints imposed by the host interface, the drive mechanism, and the industry standards for the half-inch tape format. The physical format of a super-blocked, data-compressed tape written by the HP 7980XC does not violate the ANSI 6250 GCR specification, but the logical meaning of the data is changed. This means that another 6250 GCR tape drive can read a compressed tape, but only an HP 7980XC will be able to decipher the data that was sent by the original host. This does not preclude the HP 7980XC's being used for data interchange with other GCR drives, since the drive can easily be configured to write the data it receives in an uncompressed format, just as any other 6250 GCR drive would do.

Since the physical specifications of the 6250 GCR format are maintained on a compressed tape, a method for differentiating a compressed tape from a normal GCR tape was needed. The method chosen to accomplish this is to write special noncompressed records at the beginning of a compressed tape. Whenever a tape is loaded into an HP 7980XC, the drive automatically searches for these records. If they are not found, the tape is treated as a normal uncompressed tape. If they are found, the tape is recognized as compressed and the drive separates the super-blocks and decompresses the records before sending them to the host.

Another complication stems from the embedded gaps and file marks within a super-block. To execute the typical

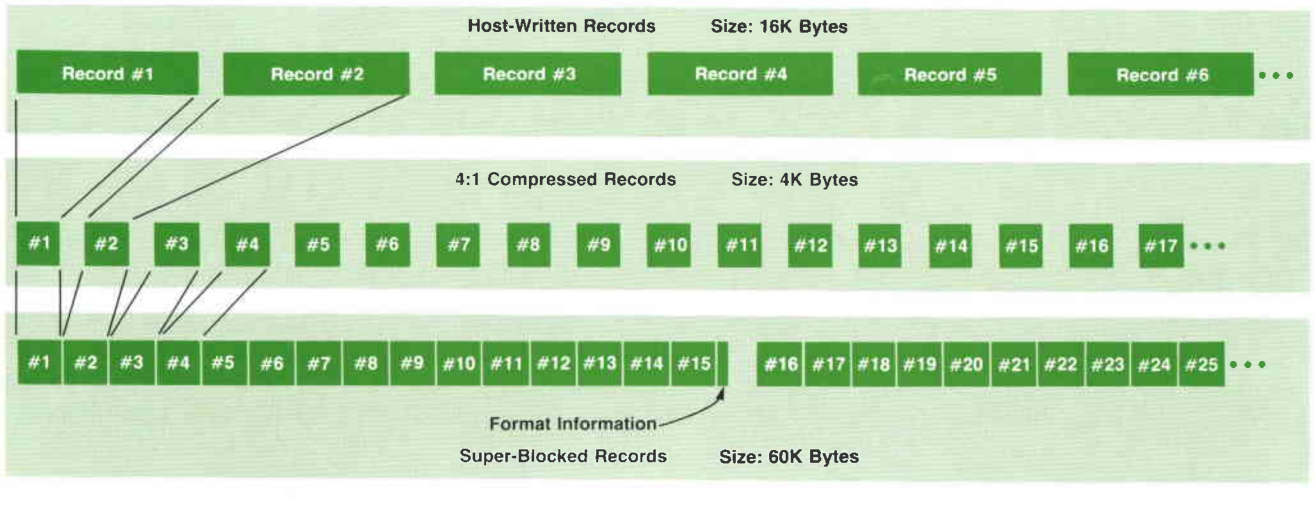


Fig. 3. *Data compression combined with super-blocking.*

host command to space to a record or file, all super-blocks must be read and processed to determine the location of the next record or file. This is not a problem when the tape is moving forward, since no performance penalty is incurred by reading the data instead of spacing over it. However, since the HP 7980 family of drives cannot read data when the tape is moving in reverse, reverse record/file spacing becomes much more complicated. Super-blocks on the tape must first be backed over and then read in the forward direction. Hypothetically, if a backspace file command were issued near the end of the tape and the beginning of the preceding file was very near the beginning of the tape, all of the super-blocks on the tape would have to be backed over and then read, a situation that might be intolerable.

The backspace file problem is solved by recording in each super-block the running count of how many super-blocks have been written since the last file mark was written. This provides the information needed to determine how many records can be safely backed over without missing the file mark. Thus, single backspace file commands can be executed efficiently. The backspace record command does not negatively impact performance because the previous record is typically within the current super-block or the preceding one.

Another issue that had to be addressed was overwriting. This occurs when a host writes and fills the entire tape, rewinds the tape, and then writes a directory at the beginning of the tape, expecting the rest of the previous writes to remain intact. This practice is strongly discouraged for sequential access devices, but does occur. If it is done, it invalidates the backspace file information in some of the super-blocks. This is because extra records and/or file marks are put back onto the tape after the previous backspace file information was written.

To support this activity, a physical tape mark is written to the tape whenever the host switches from writes to any other tape motion command. If a tape mark is encountered during backspacing, it indicates that some data has been previously overwritten. The backspace operation must read the super-block in front of the tape mark because the previous information used in the backspace file command may have been corrupted by an overwrite condition. By reading this super-block, the tape drive gets accurate information regarding the start of the file.

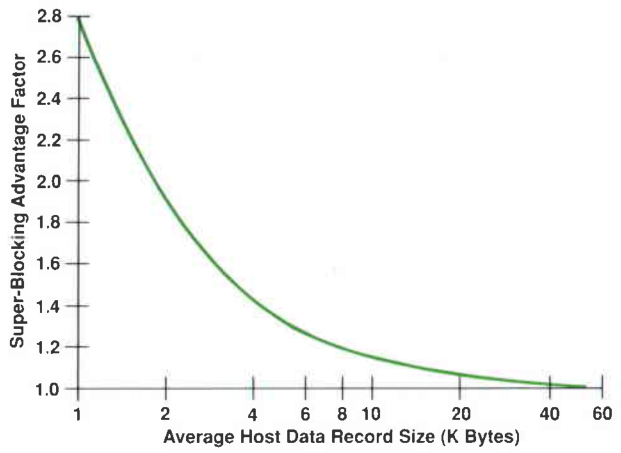


Fig. 4. *Super-blocking advantage factor (SAF) versus data record size for a tape written at 6250 bpi with 0.3-inch gaps.*

Results

The true figure of merit for the HP 7980XC in compression mode is the observed tape compaction ratio. This ratio combines the benefits of the HP data compression algorithm with the advantages of super-blocking. The tape compaction ratio is equal to the compression ratio of the host data times the super-blocking advantage factor (SAF). The SAF is dependent upon the average host data record size. A graph of SAF versus average record size is shown in Fig. 4. The compression ratio is a function of the amount of redundancy exhibited by the host's data.

The following table shows the data compression benchmark results previously outlined on page 31 with the overall tape compaction results obtained with an HP 7980XC Tape Drive.

Summary of HP 7980XC Tape Compaction Results

Data Description	Volume (Mbytes)	Compression Ratio (alone)	Tape Compaction
MPE/MPE XL on HP 3000s			
Series 68 (HP Desk)	528	3.93	4.35
Series 68 (Data Base)	2924	4.31	4.83
Series 68 (Misc. Data)	1559	4.30	5.04
Series 70 (Manufacturing)	2924	4.31	4.83
Series 930 (Code)	311	3.44	3.97
HP-UX on HP 9000s			
Series 800 (Commercial HP-UX)	226	2.06	2.73
Series 500 (Code)	363	2.38	2.57
Series 500 (Data Base)	336	4.07	4.39
Series 500 (VLSI)	785	2.52	3.34
Series 300 (Archive)	329	2.30	3.05
DEC			
DEC VAX (Code)	423	2.31	2.65
HP Series 200 Running Pascal O.S.			
Series 200 (Misc. Data)	467	2.47	2.67
Amdahl			
Amdahl (Corporate Data)	5000	3.79	3.86

Authors

June 1989

Michael J. Wright

Software engineer Mike Wright joined the team developing the HP Real-Time Data Base during the early design stages in 1987. His main responsibility was the interactive query/debug software. He joined the Manufacturing Productivity Division of HP in 1985, offering the experience of some twenty years of programming and systems work in business and manufacturing systems. Mike attended the University of Wisconsin, from which he received a master's degree in 1965. He is married and enjoys riding motorcycles.

Cynthia Givens

Cynthia Givens' responsibilities for the HP RTDB project ranged from the initial investigation, to internal/external design, to testing and documentation. She has since moved on to the development of an application integration tool. Among her past software projects are the MMC/1000 manufacturing application and AGP/DGL graphics packages. Cynthia's BA degree in computer science is from the University of Texas at Austin (1983). Born in Durango, Colorado, she's married and lives in Santa Clara, California. She enjoys hiking, skiing, and camping.

6 Real-Time Data Base

Michael R. Light

Mike Light joined HP in 1980, shortly after receiving his BS degree in computer science from the University of Vermont. He contributed to the development of the HP RTDB product as an R&D engineer, and his past responsibilities include the Image/1000, Image/1000-2, and Image/UX environments. Mike was born in Panama City, Florida, and lives in San Jose, California. "Games in any form" is how he describes his leisure interests.

Le T. Hong

Contributing to all development phases of the HP Real-Time Data Base project, Le Hong analyzed the user requirements, assisted in scheduling and prioritizing, and generally acted as the technical leader for the project. She has since moved to technical marketing in HP's Value-Added Channels program. In earlier assignments, she has contributed to the maintenance and enhancement of the IC-10 integrated-circuit lot tracking system, the EN-10 engineering data collection system, and the PCB/3000 printed-circuit-board lot tracking system. Le's BA degree in computer science is from the University of Washington (1983). She was born in Saigon, Vietnam, and lives in Fremont, California.

Feyzi Fatehi

Working on the HP Real-Time Data Base for over three years, Feyzi Fatehi designed and implemented the indexing mechanisms and contributed to all phases of developing this precision tool. He came to HP in 1986, after working as a plant automation engineer at a Texas power plant. Feyzi's BSME degree (1982) is from the University of Texas at Austin, and his master's degree in computer science (1985) is from Southwest Texas State University. He's currently studying toward an MBA degree at Santa Clara University. He was born in Teheran, Iran, lives in Sunnyvale, California, and serves as a Junior Achievement advisor at the nearby Mountain View High School. His favorite pastimes include tennis, hiking, and skiing.

Ching-Chao Liu

A software development engineer at HP's Industrial Applications Center, Ching-Chao Liu contributed his expertise to all phases of the RTDB project. In previous assignments, he was the technical leader of the HP ALLBASE DBCORE project, the project leader for the HP-UX MULTIPLAN tool, and a designer of other software projects. He came to HP in 1980. Ching-Chao coauthored two papers for data base conferences and is a member of the Association for Computing Machinery and of SIGMOD. His BS degree in nuclear engineering is from the National Tsing Hua University in Taiwan (1972), and his MS degree in computer science is from Oregon State University (1979). He was born in Taiwan, is married, and has two children who sparked his special interest in child education. He lives in Sunnyvale, California. In his leisure time, he likes swimming, playing bridge, and listening to classical music.

John Keller

Design of the floating-point controller was John Keller's main contribution to the HP 9000 Model 835 project. His list of past design projects includes CMOS processes, RAMs, and circuits for the HP 3000 Series 950, 925, and 955, and HP 9000 Models 850, 825, and 855 Computers. He now designs ICs for future computer products. His BSEE degree is from the University of Wisconsin (1981) and his MSEE degree is from the University of California at Berkeley (1985). He has authored and coauthored a number of papers and articles for conferences and publications. John was born in Milwaukee, Wisconsin. He is a volunteer literacy tutor in Cupertino, California, where his lives. In his spare time, he likes studying languages, skiing, and travel.

Floyd E. Moore

Floyd Moore designed the 16M-byte memory circuitry for the HP 9000 Model 835 and worked on the design and testing of the HP 3000 Series 935 system. He is presently working on the design of an SPU for a future HP Precision Architecture system. He came to HP in 1986, working on a project associated with the tape-automated bonding technique. Floyd was born in Richmond, California. His bachelor's degree is from the California Polytechnic State University at San Luis Obispo. He is married and lives in Fort Collins, Colorado. His favorite pastimes are photography and audio engineering.

Russell C. Brockmann

Most of Russ Brockmann's recent design activities have concentrated on the processor circuit for the HP 9000 Model 835 and HP 3000 Series 935 Computers. He also designed the battery backup unit used in the HP 9000 Models 825 and 835 and HP 3000 Series 925 and 935 Computers. He completed design of the Model 825/Series 925 processor circuit. Currently, he is developing components for future SPUs. He joined HP shortly after obtaining his BSEE degree from Oregon State University in 1985. He also attended Western Baptist College in Salem (1977-1979) and Lane Community College in Eugene (1981-1983), both in Oregon. Russ teaches Sunday school and serves in a variety of other church activities in Fort Collins, Colorado, where he lives. He was born in Myrtle Point, Oregon, is married, and has three children. Fishing, camping, playing a 12-string guitar, and bible study are some of his favorite pastimes.

18 Midrange Computers

Thomas O. Meyer

Tom Meyer was the project manager for the HP 9000 Model 835 SPU hardware. Since he joined HP in 1977, his design projects have included a memory board for the HP 250 Computer, a power supply for the HP 9000 Model 520 Computer, the battery backup regulator for the HP 9000 Model 825 Computer, and project management for the HP 9000 Model 825 and HP 3000 Series 925 Computers. Tom joined HP in 1977, soon after obtaining his BSEE degree from the South Dakota School of Mines. He has coauthored two previous articles for the HP Journal. He was born in Rapid City, South Dakota, and lives in Fort Collins, Colorado. His list of outside interests includes sailing and sailboat racing, scuba diving, skiing, hiking, and four-wheel-drive vehicles.

Jeffrey G. Hargis

Designing the processor-dependent hardware and conducting environmental testing of the HP 9000 Model 835 were Jeff Hargis' first major projects after joining HP's Systems Technology Division in 1987. He has since moved on to the design of components for new SPUs. He attended Ohio State University, where he obtained a BSEE degree in 1987. Jeff was born in Athens, Ohio, and is married. He lives in Fort Collins, Colorado. He enjoys playing the piano, basketball, and backpacking.

26 Data Compression

Jeffery J. Kato

During development of the HP 7980XC Tape Drive, Jeff Kato's contributions focused on the architecture and design implementation for the data compression chip and firmware design. He has also designed read electonics for the HP 7978A Tape Drive and the PLL and PLL IC for the HP 7980A Tape Drive. He came to HP in 1982, the same year he received his BSEE degree from Montana State University. He is named as a coinventor in three pending patents describing data compression and blocking techniques. Jeff has coauthored a previous article for the HP Journal. He is an active member of his church and a United Way volunteer. He was born in Havre, Montana, is married, and lives in Greeley, Colorado. Among his spare-time activities, he likes skiing, basketball, softball, and camping.

Mark J. Bianchi

Analog circuit design and control systems are Mark's principal professional interests. He was the R&D engineer for the design, layout, and testing of the data compression chip for the HP 7980XC. On previous projects, he designed the read channel electronics for the HP 9144A and HP 9142A Tape Drives. Mark received his BSEE degree from Pennsylvania State University in 1984, the year he also joined HP's Greeley Division. Born in Vineland, New Jersey, he lives in Fort Collins, Colorado. His list of leisure activities includes weightlifting, softball, volleyball, basketball, boardsailing, skiing, camping, and photography.

David J. Van Maren

Dave Van Maren joined HP's Vancouver Division in 1980, after receiving his BSEE degree from the University of Wyoming. His responsibilities as an R&D engineer on the HP 7980XC Tape Drive included the data compression and tape capacity benchmarks, the tape format definition and firmware, and the data buffer management firmware. In past projects, he worked on formatting VLSI tools for both the HP 7979A and HP 7980A Tape Drives and on the servo firmware for the HP 7978A. He coauthored an article for the HP Journal in 1983 about the latter project. Dave's work on the VLSI FIFO circuit for the tape drives resulted in a patent, and he is named coinventor in four pending patents describing data compression and blocking techniques. He was born in Casper, Wyoming, is married, and has three young sons. He lives in Fort Collins, Colorado. He and his wife spend much of their free time teaching natural family planning.

32 Super-Blocking

Mark J. Bianchi
Author's biography appears elsewhere in this section.

Jeffery J. Kato
Author's biography appears elsewhere in this section.

David J. Van Maren
Author's biography appears elsewhere in this section.

35 Lightwave Component Analysis

Michael G. Hart

As development engineer, Mike Hart was involved in designing firmware for the HP 8702A Lightwave Component Analyzer and, earlier, for the HP 8753A Network Analyzer. He continues to work on similar assignments for new HP products. He attended Utah State University, where he earned his BSEE degree in 1983. His MSEE degree is from Cornell University (1984). He joined HP in 1984. The lightwave component analyzer is the subject of a paper Mike coauthored for an RF and microwave symposium. He is a member of the IEEE. He was born in Sacramento, California, and in his off-hours, he serves as the organist for his church in Santa Rosa, California, where he lives. Other recreational activities include playing the piano, softball, tennis, and travel.

Paul Hernday

Paul Hernday is an R&D project manager in HP's Network Measurements Division in Santa Rosa, California. With HP since 1969, he has been involved with new-product developments in sweepers, scalar and vector network analyzers, and lightwave component analyzers. His most recent project has been the development of a dual-laser heterodyne system for the calibration of lightwave receivers. Paul earned his BSEE degree at the University of Wisconsin in 1968. He is married, has two children, and lives in Santa Rosa, California. Boardsailing, music, and robotics are among his diverse leisure interests.

Geraldine A. Conrad

As a development engineer on the HP 8702A Lightwave Component Analyzer, Gerry Conrad worked on measurement accuracy and system performance analysis. She continues to be involved in similar developments, more specifically in microwave circuit design and optical system evaluation. In earlier years of her career at HP, she worked first as a product marketing engineer and later joined a design team on the HP 8753A Network Analyzer. Gerry originally joined HP as a summer student in 1980, then accepted a permanent position two years later. Her BSEE degree is from the University of Florida (1982). She has authored a paper describing an RF network analyzer verification technique and coauthored a symposium paper about high-frequency measurement of lightwave systems. She is a member of the IEEE. Born in Trincomalee, Sri Lanka, Gerry is married and lives in Santa Rosa, California. Her leisure interests include travel, quilting, camping, hiking, cooking, and reading.

Roger W. Wong

Lightwave and microwave measurement technologies are Roger Wong's special interests, and as the R&D program manager, he carried overall responsibility for the development of the HP 8702A Lightwave Component Analyzer. Past responsibilities included scalar network analyzer detectors, directional bridges and accessories, and the development of microcircuits and associated components for microwave applications. Roger joined the Microwave Division of HP in 1968, after obtaining his MSEE degree from Columbia University. His BSEE degree is from Oregon State University (1966). He is a member of the IEEE and the National Society for Professional Engineers. He has authored or coauthored a number of papers and articles on microwave transistor modeling, microwave amplifier design, and high-speed lightwave measurements. Several patents describing lightwave measurement techniques Roger developed are pending. He was born in Honolulu, Hawaii, is married, and has a five-year-old son. He lives in Santa Rosa, California. His favorite pastimes include swimming, hiking, cooking, and baking bread.

52 Lightwave Sources and Receivers

Kenneth W. Shaughnessy

The HP 8753A and the HP 8754A Vector Network Analyzers and a number of lightwave instruments are among the major projects to which Ken Shaughnessy has contributed design ideas. On the HP 8702A Lightwave Component Analyzer System, he worked as a product designer. He joined the Santa Rosa (California) Division of HP in 1975 as a printed circuit board designer, after previous positions as a mechanical designer at Sperry Marine Systems and Teledyne Avionics. Ken attended the University of Virginia School of Engineering. He was born in Chicago, Illinois, is married, and has five children. He lives in Kenwood, California. Woodworking and automobile and bicycle repair are his favorite spare-time activities.

Kent W. Leyde

In development of the HP 8702A Lightwave Component Analyzer, Kent Leyde's design work concentrated on microcircuits and optics for the lightwave receivers. As a development engineer, he has since started work on the signal acquisition and processing elements of a new product. Kent's BSEE degree (1984) and MSEE degree (1985) are from Washington State University. While attending college, he worked for local companies on such product developments as process controls and digital protective relays for high-voltage ac transmission systems. He joined HP in 1985. He coauthored an article describing an optical measurement system, soon to be published in Optical Engineering. Born in Seattle, Washington, he is married and has a small daughter. He lives in Santa Rosa, California. In his off-hours, he enjoys boardsailing and skiing.

Rollin F. Rawson

The HP 8753A and HP 8754A Network Analyzers and the HP 8756A Scalar Network Analyzer are among the many product developments to which Fred Rawson has contributed. His work on the HP 8702A Lightwave Component Analyzer has focused on source leveling and the thermal loops, the receiver power supplies, the RF attenuator and source, and the RF interface. He has worked for HP since 1960. Fred's BSEE degree is from California State University at San Jose. Before enrolling at San Jose, he served as a staff sergeant in the U.S. Air Force. Born in Laguna Beach, California, he is married and has four children. He lives in Santa Rosa, California. In his leisure time, he enjoys collecting, refurbishing, and driving Studebaker automobiles; he also collects stamps.

Robert D. Albin

Dale Albin was project manager for the lightwave sources and receivers discussed in this issue of the HP Journal. In his twelve-year career at HP, he has been a production engineer at the Microwave Technology Division working on device testing and GaAs FET processing and a development engineer/project leader on millimeter source modules at the Network Measurements Division. His BSEE degree (1977) is from the University of Texas at Arlington, and his MSEE degree is from Stanford University. Two patents relating to finline technology are based on his ideas. Dale has delivered papers at HP symposia and has written a previous HP Journal article about millimeter source modules. He was born in Dallas, Texas, and lives in Santa Rosa, California. His outside interests include running, skiing, bow hunting, reading, and aviation.

58 — Videoscope

Myron R. Tuttle

Before working on the hardware and firmware design for the Videoscope tool, Myron Tuttle's responsibilities included development of the HP 45981A Multimode Video Adapter and the HP 2625 and HP 2628 Terminals. He joined the Advanced Products Division of HP in 1974 and is a member of both the Association for Computing Machinery and the IEEE. Myron's BSEE degree is from the University of California at Berkeley. He served in the U.S. Navy as an electronics technician. Born in San Francisco, California, he is vice president of a homeowners association in Santa Clara, California, where he lives. His hobbies are amateur radio and computer programming.

Danny Low

Danny Low joined the Cupertino Division of HP in 1972, shortly after obtaining a degree in computer science from the University of California at Berkeley. He developed the host software for the Videoscope tool and continues to support it. In the past, his responsibilities included software quality control for the original MPE system. He also developed system software for the HP 300 and for the MPE-V computers. Danny was born in Canton, China, and lives in Mountain View, California. His favorite off-hours activities focus on computers, science fiction, and photography.

69 — Neural Data Structures

J. Barry Shackleford

Barry Shackleford spent almost three years as a development engineer at Yokogawa Hewlett-Packard in Japan, where he worked on a Kanji computer terminal. His more recent work in the Systems Architecture Laboratory of HP Laboratories yielded the background for the neural network programming approach he describes in this issue of the HP Journal. Before joining HP in 1981, he worked for Hughes Aircraft Corporation, designing a telemetry computer for the still-functioning Pioneer Venus spacecraft, and for Amdahl Corporation, developing hardware for Models 470 and 580 mainframe computers. Several pending patents are based on his ideas. Barry's BSEE degree is from Auburn University (1971), and his MSEE degree is from the University of Southern California (1975). He is a member of the IEEE. He was born in Atlanta, Georgia, and lives in Sunnyvale, California. He speaks Japanese and practices Japanese brush writing as a hobby. He has a pilot license and likes large-format photography, woodworking, and hiking.

79 — Electromigration Model

Vladimir Naroditsky

As a professor of mathematics at California State University at San Jose, Vladimir Naroditsky contributed his expertise in the electromigration simulation project described in this issue of the HP Journal. He emigrated from the Soviet Union in 1979, and his bachelor's degree is from Kiev University (1976). His PhD degree is from the University of Denver (1982). Vladimir has authored 24 papers in the field of mathematical physics, his professional specialty. He is a member of the American Mathematical Society, the Mathematics Association of America, and the Society of Industrial and Applied Mathematics. He was born in Kiev, is married, and lives in San Francisco, California. In his leisure time, he enjoys classical music.

Wulf D. Rehder

Wulf Rehder, who describes his place of origin as "a tiny village in Northern Germany," pursued studies at universities in Hamburg, Freiburg, Tokyo, Berkeley, and finally Berlin, where he earned his PhD degree in 1978. Ancient languages, mathematics, and physics are among his subjects of study, and he has held various teaching positions, most recently as professor of mathematics at California State University at San Jose. He was a statistician at HP's System Technology Division until last December, when he became systems performance manager at Metaphor Computer Systems in Mountain View, California. Wulf is a prolific writer and has published some 35 papers on mathematics, statistics, philosophy, and linguistics. He's working on his third book. He is married, has two children, and lives in Santa Clara, California. His hobbies include the study of the middle ages, especially the 11th century. He also specializes in early 19th-century literature.

Paul J. Marcoux

Paul Marcoux is a project manager for studies involving failure analysis and failure physics for integrated circuits. In this issue of the HP Journal, he reports on a new model for simulating electromigration in thin metal films. Aspects of integrated circuit process technology have been focal to most of his past projects at HP. He has written about 20 articles about chemistry and IC processing for professional journals, and he is a member of the American Vacuum Society. A patent has been awarded for an IC process he helped develop. Paul's BS degree is from Villanova University (1970), and his PhD degree in chemistry is from Kansas State University (1975). He did postdoctoral work in chemical kinetics at Pennsylvania State University. Born in Pautucket, Rhode Island, Paul is married and has two daughters. He lives in Mountain View, California, and his favorite pastime is photography.

Paul P. Merchant

As a project leader at HP Laboratories, Paul Merchant has had a variety of R&D projects involving electromigration, silicide process development, and multilevel metallization. He handled modeling, testing, and planning in the electromigration study discussed in this issue. Processing and properties of thin films are his specialty. He has published many papers on solid-state physics and chemistry and on microelectronics and is a member of both the American Physical Society and the American Vacuum Society. Paul's BS degree in physics is from the University of Vermont (1972), and his ScM degree (1974) and his PhD degree in physics (1978) are both from Brown University. Two postdoctoral positions, one in France and one at Brown University, involved laser materials and photoelectrolysis. He is married, has three sons, and lives in Menlo Park, California. In his off-hours, he enjoys playing the piano, astronomy, and bicycling.