HEWLETT-PACKARD
JOURNAL

JUNE 1989

HEWLETT
PACKARD

HEWLETT-PACKARD
JOURNAL

June 1989 Volume 40 • Number 3

Articles

Research Reports

Departments

The **Hewlett-Packard Journal** is published bimonthly by the Hewlett-Packard Company to recognize technical contributions made by Hewlett-Packard (HP) personnel. While the information found in this publication is believed to be accurate, the Hewlett-Packard Company 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.

A Data Base for Real-Time Applications and Environments

HP Real-Time Data Base is a set of subroutines and a query facility that enable real-time application developers to build and access a real-time, high-performance, memory-resident data management system. The software runs in an HP-UX environment on an HP 9000 Series 300 or 800 Computer.

by Feyzi Fatehi, Cynthia Givens, Le T. Hong, Michael R. Light, Ching-Chao Liu, and Michael J. Wright

A REAL-TIME ENVIRONMENT deals with current phenomena rather than past or future events. If information is lost, it is lost forever since there is rarely an opportunity to reclaim it. A typical real-time situation is a factory floor where a computer is monitoring the status of machines and materials and constantly checking to make sure that everything is working properly. Frequently, the data from these checks can be discarded once it is determined that all is indeed satisfactory, although some data might be retained for statistical purposes. If the checking process reveals something amiss, a real-time process might be invoked to correct the situation, such as rejecting a flawed part or shutting down an entire assembly line if a machine is overheating. Data from such incidents is frequently saved for later analysis (e.g., statistical quality control).

A real-time environment needs to respond reliably when an action must be taken quickly within a brief predetermined span of time, such as receiving and storing a satellite data transmission. If the process of receiving and storing the data can always be expected to finish within 80 milliseconds, then the satellite can reasonably transmit every 100 milliseconds without fear of losing any data.

Data capture in a real-time environment may involve sampling large amounts of raw information with data arriving unexpectedly in bursts of thousands of bytes or even megabyte quantities. A real-time data base must be capable of efficiently storing such large amounts of data and still support the expectations of the user for reliable and predictable response.

When a real-time process requests data, it should be given that data immediately, without any unreasonable delay. Whether or not the data is consistent may be less of a concern than that it is the most current data available. Given sufficient urgency, a real-time application may not require guarantees of either consistency or integrity of data. An application designer must be aware of the risks and should only violate normal data integrity rules when absolutely necessary. A real-time data management system must tolerate such violations when they are clearly intentional.

Finally, a real-time data base must be scalable to the needs of different users. This means that users should be able to implement or eliminate functionality according to the needs of the application. The performance impact of unused functionality must be minimal.

Traditional Data Bases

Traditional data bases are generic and flexible, intended to support the widest possible range of applications. Most traditional data bases use magnetic disc as the primary data storage medium because of its large capacity, relatively high-speed access, and data permanence. Disc-based data bases in the gigabyte range are now possible.

However, traditional data bases are too slow for most real-time applications. Disc access speeds are still two to three orders of magnitude slower than dynamic random access memory (DRAM) access. Even when the average speed of a traditional data base is acceptable, its worst-case speed may be totally unacceptable. A critical need of real-time systems is the ability to provide a predictable response time. Traditional data bases support transaction operations, which may require commit protocols, logging and recovery operations, and access to disc. They also define data access methods that rigidly enforce internal rules of data consistency and integrity. Given a large number of simultaneous transactions, it becomes nearly impossible to guarantee pre-

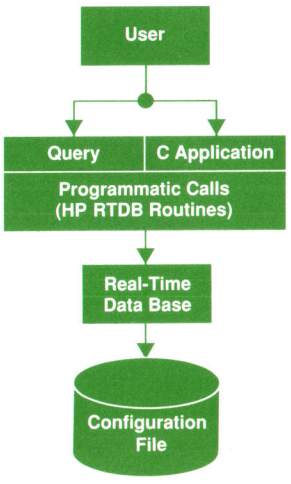


Fig. 1. *An overview of the HP Real-Time Data Base System.*

dictable response time. For example, data that is modified as part of an update transaction may not be available to a reader process until the entire transaction is committed. If the reader in this case were a real-time assembly line control process, it could be disastrously delayed waiting for a process of much less importance to complete.

Real-Time Data Bases

Because of extremely high performance requirements, real-time data bases are often custom-designed to the particular needs of a given application. This limits their usability for other applications and causes portability problems if their performance relies upon any hardware characteristics. They are also usually expensive to program and maintain.

Real-time data bases have taken two common approaches, acting either as a very fast cache for disc-based data bases or as a strictly memory-resident system which may periodically post core images to disc. Real-time data bases acting as a high-speed cache are capable of quickly accessing only a small percentage of the total data kept on disc, and the data capacities of purely memory-resident data bases are severely limited by the amount of available real memory. In either case, real-time data bases must coexist with disc-based data bases to provide archival and historical analysis functions in real-time applications. Eventually, a portion of real-time data is uploaded to a disc-based data base.

Data transfer between real-time and disc-based data bases requires commonly understood data types, and may require reformatting or other treatment to make it digestible to the target data base. Frequently, data is transferred over a network interface as well. The problems of interfacing real-time data bases with disc-based data bases are often further complicated by the customized, nonstandard nature of most real-time data bases.

HP Real-Time Data Base

HP Real-Time Data Base (HP RTDB) is one of the Industrial Precision Tools from HP's Industrial Applications Center. The Industrial Precision Tools are software tools intended to assist computer integrated manufacturing (CIM) application developers by providing standard software solutions for industrial and manufacturing applications problems. The HP Real-Time Data Base is the data base tool. HP RTDB is a set of software routines and interactive query commands for creating and accessing a high-performance real-time data base. It is designed for the specific needs of real-time systems running on HP 9000 Series 300 and 800 Computers.

Column

Tuple

Machine	Operator	Work_Order	Parts_So_Far	Rate_Hr
Machine 1	Jack_Swift	A123	112	4.67
Machine 2	Mike_Grell	A124	110	4.58
Machine 3	Olive_Frap	A125	155	6.46
Machine 4	Judy_Tyron	A126	125	5.20
Machine 5	Sammy_Silas	A127	175	7.29
Machine 6	Kurt_Bralin	A128	000	0.00

Fig. 2. *A table structure in HP RTDB consisting of six tuples and five columns.*

Fig. 1 shows an overview of the HP Real-Time Data Base system. Access to the data base for the user is through the query commands or an application program written in C that uses the HP RTDB routines. The HP RTDB routines provide the application developer with the ability to:

- Define or change the data base schema
- Build the data base in memory
- Read or write data from or to the data base
- Back up the schema and data.

The query commands provide an interactive facility for configuring and debugging the data base, and for processing scripts in batch mode and on-line without writing a program. The configuration file is automatically created when the user defines the data base. It contains the system tables and control structures for the data base.

Besides the two interfaces to the data base, HP RTDB also provides the following features:

- Performance. HP RTDB supports predictable response time, and to ensure speed, HP RTDB is entirely memory-resident. Several design alternatives were chosen to ensure this high performance, such as preallocation of all data base memory to minimize memory management overhead, alignment of data on machine word bound-

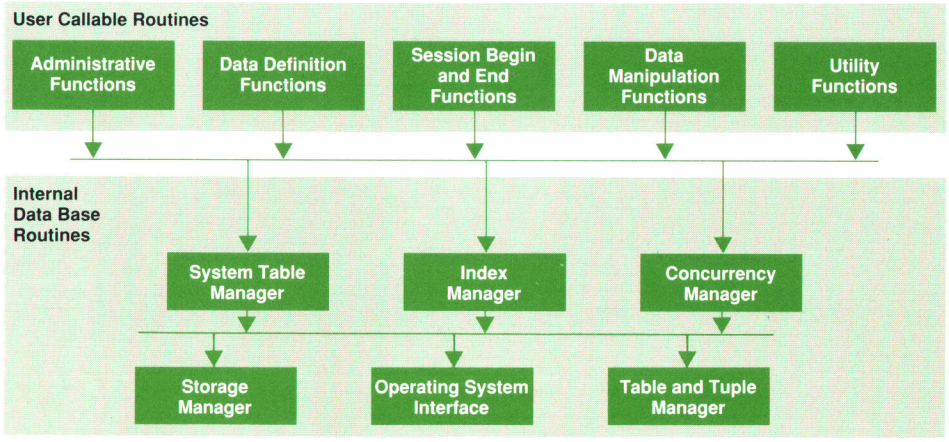


Fig. 3. *HP RTDB module hierarchy.*

aries, simple data structures, and the extensive use of in-line macros to reduce the overhead of function calls. The design alternatives chosen produced performance results that exceed initial goals. For example, performance tests showed that 66,666 56-byte records can be directly retrieved from an HP Real-Time data base in one second.

- Multiple Data Base Access. HP RTDB resides in shared memory so that multiple processes can access the data base concurrently. Also, one process can access multiple data bases.
- Simple Data Structures. Data is stored in HP RTDB in two forms: tables and input areas. A table is an array of columns and rows (tuples) that contain related information (see Fig. 2). Input areas are areas in the data base designed to receive large blocks of unstructured data. Often this data comes from high-speed data acquisition devices.
- Data Access. Retrieval routines are constructed so that any specified data can be accessed directly, sequentially, or by hash key values. Direct access is available to a tuple (row) of a table and to an offset in an input area.
- Dynamic Reconfiguration. Tables and input areas can be added or deleted quickly without having to recreate the data base.
- Security. HP RTDB provides three levels of password protection: data base administrator access, read-write access, and read-only access.
- Backup and Recovery. The schema (data base structure), and optionally the user's entire data base can be saved to a disc file.
- Locking. HP RTDB provides tables and input area locking. This means that an application can exclusively access a table or input area until it decides to release the lock. If the application requires, read-through and/or write-through locks are allowed.
- Scalability. HP RTDB is scalable. If some features are not required they can be eliminated to improve performance.
- Documentation Aids. HP RTDB is supplied with a self-paced tutorial complete with a query/debug script to build the data base that is used in the tutorial examples. There is also an on-line help facility for the interactive query/debug utility.
- Programming Aids. HP RTDB programming aids include:
  - A standard C header file defining the constants and data structures required to use the HP RTDB subroutines
  - Prototyping and debugging capabilities of the query/debug utility
  - On-line access to explanations of HP RTDB error codes
  - User-configurable error messages, formats, and hooks for user-written error routines
  - Native language support which includes 8-bit data, 8-bit filenames, and message catalogs.

HP RTDB Modules

The HP Real Time Data Base modules can be grouped into two main categories: user-callable routines and internal data base routines (see Fig. 3). The user-callable routines include the following functions.

- Administrative Functions
  - Define the data base including its name, passwords, and system limits (MdDefDB)
  - Build or rebuild the data base in memory (MdBuildDb)
  - Remove a data base from memory (MdRmDb)
  - Change data base system limits or passwords (MdChgDb, MdChgPwd).
- Data Definition Functions
  - Define a table or input area (MdDefTbl, MdDefIA)
  - Define or add column(s) to a user table (MdDefCol)
  - Define an index on column(s) in a defined table (MdDefIx)
  - Remove a table or an input area (MdRmTbl, MdRmIA)
  - Remove an index from a table (MdRmIx).
- Session Begin or End Functions
  - Open the data base and initiate a session (MdOpenDb)
  - Close the data base and terminate a session (MdCloseDb).
- Data Manipulation Functions
  - Open a table or input area for access (MdOpenTbl, MdOpenIA)
  - Get a tuple by sequential search (MdGetTplSeq), hash key index (MdGetTplIx), or tuple identifier (MdGetTplDir)
  - Compare a tuple value with a set of expressions (MdCompare)
  - Add or remove a tuple to or from a table (MdPutTpl, MdRmTpl)
  - Update a tuple (MdUpdTpl)
  - Get or put a value from or to an input area (MdGetIA, MdPutIA)
  - Lock or unlock a table or an input area (MdLock, MdUnlock).
- Utility Functions
  - Save the data base schema and optionally the entire data base to disc (MdTakeImage)
  - Release resources held by prematurely terminated processes (MdCleanup)
  - Provide information on the columns of a table (MdColInfo)

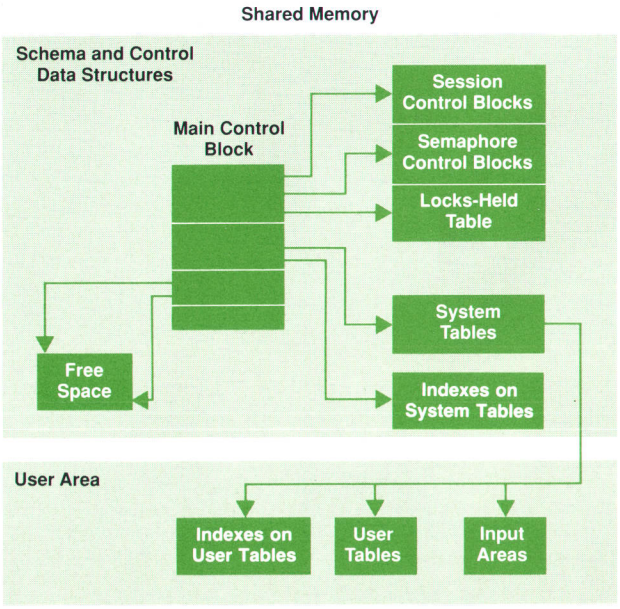


Fig. 4. *HP RTDB data structures in shared memory.*

- Provide information on the minimum data base and schema size in bytes (MdDbSizeInfo)
- Provide information on all or one specific user table, index on a table, or input area (MdTblInfo, MdIxInfo, MdIAInfo)
- Provide information on system tables and session use (MdSchInfo).

The internal data base routines are used by either the user-callable routines or other internal routines. They are implemented as C functions or macros. The macro implementations are used for small pieces of code. The resulting code is slightly larger but faster. The functions performed by the internal data base routines include:

- System Table Manager. These routines handle the tables that define the schema and configuration of the data base.
- Index manager. These routines handle hashing, index manipulation, and formulation of hash index key values.
- Concurrency Manager. These routines handle locking operations and control concurrent processes using the data base.
- Storage Manager. These routines handle memory management, which includes keeping track of allocated and available shared memory.
- Operating System Interface. These routines provide a clean and consistent interface to the HP-UX operating system.
- Table and Tuple Manager. These routines handle functions related to tuples and tables such as copying, adding, or deleting tuple values.

Data Structures

The data structures in HP RTDB are divided into two categories: those that manage and control access to the data base and define the schema, and those that contain the user data. Fig. 4 shows an overview of these structures in shared memory. The data structures in the schema and control section are automatically created when the data base is defined. The data structures in the user area are added later when the data base is built. Only two of these data structures are visible and accessible to the user—user tables and input areas.

- Main Control Block. The main control block contains the data base status, limits, and pointers to other data structures in the schema and control section of the data base. It also contains information used by the storage manager, such as pointers to the beginning and end of free memory storage space, a pointer to the list of free storage blocks, and the total amount of free storage left.
- Session Control Blocks. A session control block is allocated to each process accessing the data base. Each block contains a session identifier, a pointer to the main control block, and other information about the process, such as the user identifier (HP-UX uid) and the process identifier (HP-UX pid). The session identifier is returned to the user when the data base is opened, and is used in subsequent calls to access the data base. The number of session blocks determines the number of users that can have access to the same data base at any one time. This number is determined when the data base is created.
- Semaphore Control Blocks. There is a semaphore control block for each lockable object in the data base (i.e., user tables and input areas). These blocks contain HP-UX semaphore identifiers.
- Locks-Held Table. Each entry in the locks-held table indicates whether a lock is being held by a session on a certain data base object (user table or input area), and if so, what type of lock.
- Index Tables. Index tables contain the data for performing fast access (i.e., hash indexing) to system and user tables.
- System Tables. System tables contain the schema (structure) of the data base and information about the locations and attributes of all data base objects, including themselves.
- User Tables and Input Areas. The application data managed by the user is contained in the user tables and input areas.

Tables. The table, which is a two-dimensional array consisting of rows (tuples) and columns, is the fundamental data structure in HP RTDB. There are three types of tables: system tables, user tables, and index tables. All tables, whether they are system, index, or user tables, have the same structure, called a table block (see Fig. 5). A table block is divided into two sections: control structures and data. Control structures contain the information needed to locate, add, or delete data in a table. The data portion of the table contains the system or user data. The information in the control structures includes:

- Table Block Header. The header contains information needed to access information within the table, such as data offsets and table type (i.e., system, index, or user).
- Slot Array. Each entry in the slot array indicates whether a tuple in a table is filled or vacant. The slot array is accessed when adding or deleting tuples, and when searching sequentially.
- Column Descriptor Array. The entries in the column descriptor array describe the columns in the data portion of the table block. Each column descriptor defines the column type (i.e., character, byte string, integer, float, input area offset, etc.), the column length, and the column offset in bytes from the start of the tuple (see Fig. 6).

The data in each type of table is stored in tuples. The tuple format, which is the number, length, and type of columns, must be the same for all tuples in any one table. However, the tuple format may be different for each table. The number and size of tuples in a table are limited only

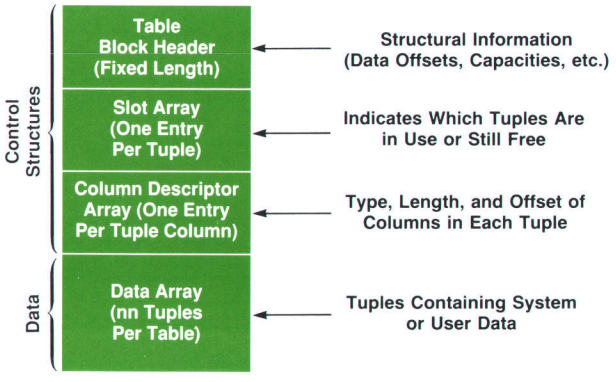


Fig. 5. *HP RTDB table block.*

by the amount of real memory available. Each tuple and all columns within a tuple are word-aligned. Variable-length columns and null columns are not supported. To support only fixed-length data and columns may seem wasteful of real memory, but this scheme more than offsets the increased size and complexity of code needed to support variable-length data, and the resulting performance degradation. Another benefit is that the size of a table has little effect upon the speed of accessing any given tuple. Since all tuples in a table are the same length, a tuple's location is fixed and can be quickly determined with one simple calculation. Once located, copying the tuple's data between the data base and a user buffer can be done by words (four bytes at a time) rather than one byte at a time, since all data is aligned on machine word boundaries.

Data in user tables can be any supported C data type or an offset into an input area. Users can also store and retrieve unsupported data types in table columns defined as a byte string type. Using the byte string type, the user can store pointers to other tuples in the same or any other table. Data compression, alignment of values in tuples, and verification of data types is left to the user's application, where these functions can be done more efficiently. HP RTDB routines store user data exactly as it is received and retrieve user data exactly as it is stored. A positive side effect of this is that the storage and retrieval integrity of 16-bit data (e.g., Katakana or Chinese text) can be guaranteed without special routines.

Because all table types have the same table block structure, the same code can be used to perform operations on system, index, and user tables. However, system table access is so critical to performance that operations on system tables are often performed by special code that takes full advantage of the known, fixed locations and formats of system tables.

Tuple Identifiers. A tuple identifier or tid uniquely identifies each tuple in every table in the data base including system tables, index tables, and user tables. Tuple identifiers are used by the user to access user tables and by

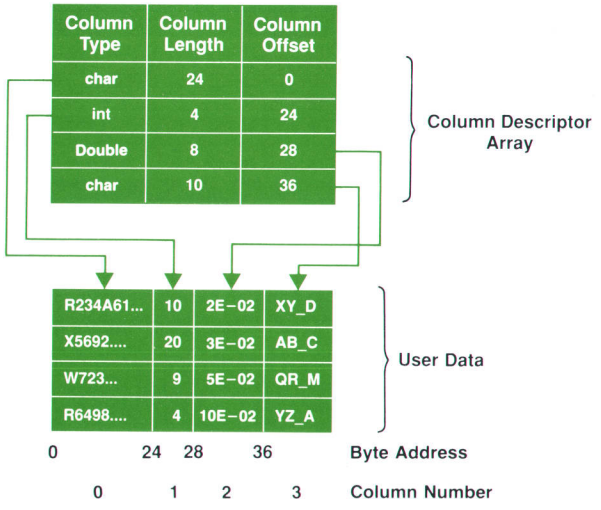


Fig. 6. *HP RTDB table block showing the association between the column descriptor array and the columns in the data portion of the user table.*

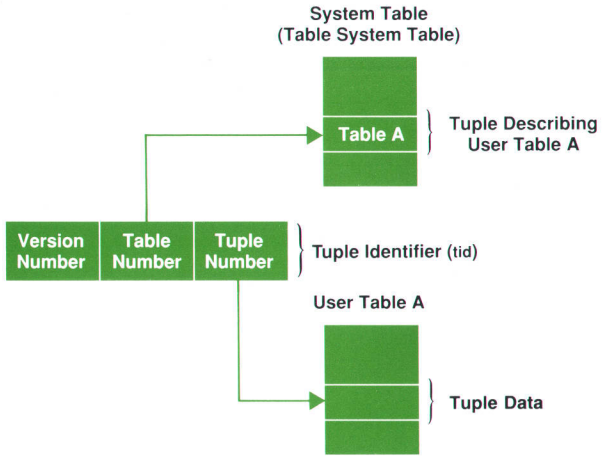


Fig. 7. *Tuple identifier data structure and its relationship to system tables and user tables.*

internal HP RTDB routines to access all the tables in the data base. A tuple identifier is returned to the user when a tuple is added to a table (MdPutTpl) or after a successful table search (MdGetTplSeq) or after a successful indexed access (MdGetTpllx). Once obtained, a tuple identifier can be used in subsequent calls to provide extremely fast, direct access to the same tuple for rereading, deletion, or update. Directed access by tuple identifier is by far the fastest access method in the HP RTDB data base.

The data type for a tuple identifier is called tidtype and contains three elements: a table number, a tuple number, and a version number.

- The table number is the tuple number for a tuple in a system table that describes the table associated with the tuple identifier. Fig. 7 shows the tid for a user table and the use of the table number and tuple number entries. For system and user tables, the system table containing the tuples of table descriptions is called a table system table, and for index tables the system table is called an index system table. System tables are described in detail later in this article.

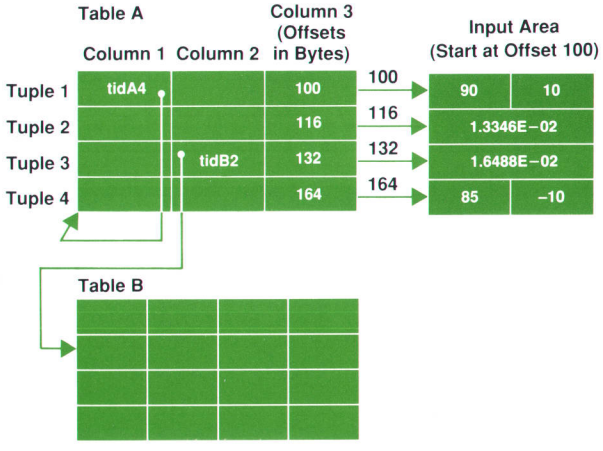


Fig. 8. *Tuple identifiers can be used to link tuples logically in the same table (tidA4) or other tables (tidB2). Also, offsets for input areas are stored in the table.*

- The tuple number indicates the row in a table containing the tuple data.
- The version number is used to ensure that a tuple being accessed directly by a tid is the same tuple that was accessed when the tid was first obtained. For example, suppose user A adds a tuple to table X and saves the returned tid for subsequent rereading. If user B accesses table X and deletes the tuple added by user A and then adds another tuple to table X, it is possible that the tuple added by user B could occupy the same location as the tuple originally added by user A. When user A attempts to use the same tid on table X for reading the tuple that was changed by user B, the version numbers won't match and user A will be prevented from accessing the tuple and notified of the problem.

Tuple identifiers can be used to chain tuples logically. Users can build logical relationships between tuples by inserting the tuple identifier of one tuple as a column value of another tuple. This concept is illustrated in Fig. 8, where tidA4 and tidB2 are tuple identifiers. The tuple identifier is designed so that its value remains constant across data base restarts and dynamic schema changes. Thus relationships of tuples, whether system-defined or user-defined, are not lost when a data base is shut down and restarted.

System and User Tables. The system tables contain the schema for the data base, and the user tables and input areas contain the user's application data. The relationship between these data structures is shown in Fig. 9. There are four types of system tables:

- Table System Table. The table system table contains information on all the user tables and system tables in the data base including itself. One section of the table describes system tables and another section describes user tables. Each tuple in the table system table describes one table, and the columns contain relevant information about the attributes of the table described by the tuple (e.g., table name, tuple length, number of columns, and so on). Fig. 10 shows a portion of a tuple in the table system table for a user table (Usertbl02). The entry CSTtid is the tuple identifier for the starting tuple in the column system table assigned to Usertbl02, and the entry ISTtid is the tuple indentifier for the starting tuple in the index system table assigned to Usertbl02. The entry FstBlkOff is an offset in bytes to the first block of storage for Usertbl02. When the user adds or deletes a table, the table system table is updated accordingly. Likewise, when certain changes (e.g., add indexes) are made to the user table these changes are reflected in the associated tuple in the table system table.
- Column System Table. The column system table contains information on all columns in a user table. Each tuple describes one column in a user table. Some of the information kept in the column system table includes column type, length, and offset for each user table column. This same information is kept in the the column descriptor array of the user table control block described earlier. The reason for having this data in two places is that it eliminates one level of indirection when accessing data in user table columns. A new tuple is added to the column system table when a new column is added to a user table.
- Index System Table. The index system table contains information on the indexes for system and user tables. Each tuple in the index system table describes an index defined on a system or user table. Indexes on system tables are predefined by HP RTDB and indexes on user

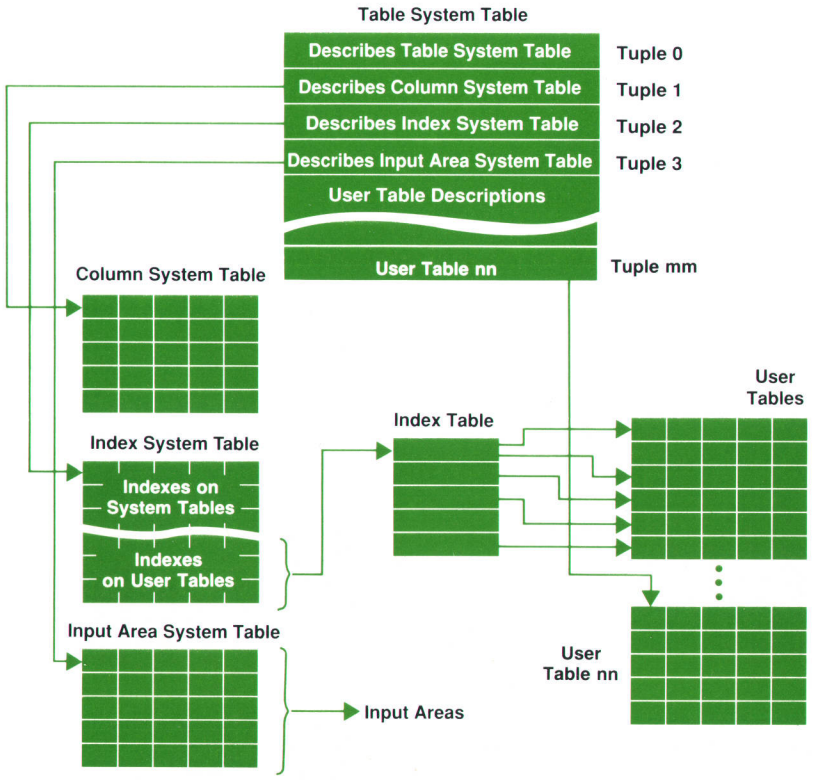


Fig. 9. *Relationship between system tables and user tables.*

tables are defined only by the user. Indexes are described in more detail later in this article.

- Input Area System Table. The input area system table contains information on user-defined input areas. Each tuple contains the input area name, the input area size, and the offset (in bytes) of the beginning storage location allocated to the input area.

Indexes. Indexes are defined on tables to provide faster access to a data tuple. HP RTDB provides hash indexing. Fig. 11 shows the organization of the hash indexing scheme employed in HP RTDB. In this scheme a key value, which is composed of one or more columns in a table, is sent to a hash function that computes a pointer into a table of tuple identifiers. Once the tuple identifier is known, the desired tuple can be accessed.

The columns that are used for the key value are designated in the index system table described ealier. Fig. 12 shows the relationship between the index system table and the columns in a user table designated for key values. These columns are specified when an index is defined for a table. In many hashing schemes the hashing function transforms a key value into a storage address where the user's data is stored. HP RTDB does not use hashing for both storage and retrieval of tuples, but only as a very fast retrieval mechanism.

The process of inserting a new tuple into a table with a hash index takes the following steps:

- The tuple is inserted in the first available slot in the user table without regard to any index defined on the table.
- A location is found in the index table by applying the hash function to the key value of the tuple. This location is called the primary location for the tuple. If the hash function returns a primary location that is already in use, a secondary location is found and linked to the primary location using a synonym chain.
- The tuple identifier of the inserted tuple is stored in the designated primary (or secondary) location in the index table.

The process of retrieving an existing tuple from a table using the tuple's key value takes the following steps:

- The hash function is applied to the key value to obtain the primary location for the corresponding tuple identifier in the index table.
- If the primary location has no synonyms, the tuple addressed by the tuple identifier in the primary location is accessed and returned.
- If synonyms exist, then each one is accessed in turn until one is found with a key value that matches the unhashed key value of the requested tuple. If the hash index is defined with the option to allow duplicate key values, then each tuple with a matching key value will be returned in the order found.

Independence of retrieval from storage provides HP RTDB with some major advantages:

- Multiple hash indexes. Each table can have multiple hash indexes defined for it, allowing the same table to be searched with any number of different keys as shown in Fig. 12.
- Constant tuple identifiers. A hash index can be rehashed without causing any data migration (the data tuple locations do not change). This means that applications can use direct access by tuple identifier and never be concerned that rehashing might cause a tuple identifier to change. This feature also significantly improves the performance of applications that frequently update table columns used for key values.
- Dynamic hash index definition. Unlike direct hashing algorithms, hash indexes can be added to or removed from existing tables.
- Fixed space overhead. The space overhead incurred because of defining a hash index is a direct function of the number of tuples in a table and does not depend on the number of columns, so it does not increase as new columns are added to a table.

However, no matter how carefully a hash function is designed, it cannot guarantee that collisions will not occur

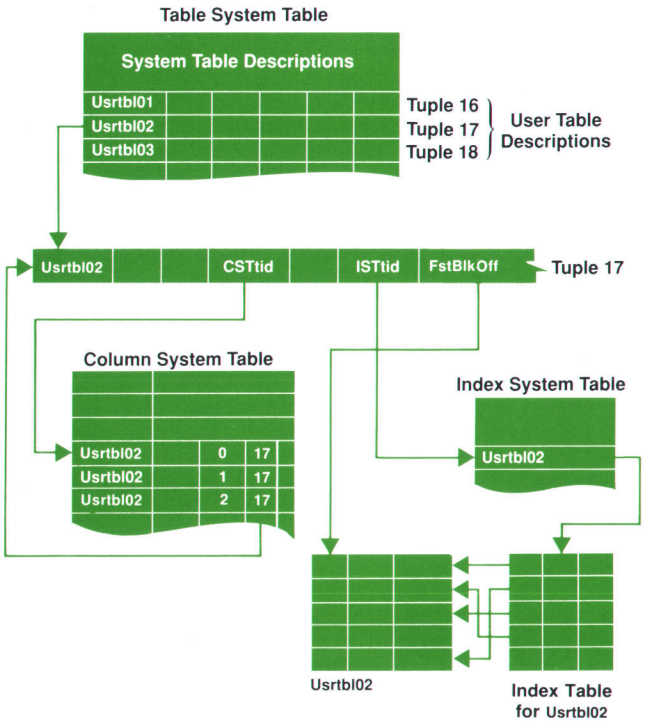


Fig. 10. *A partial view of a tuple in a table system table that describes a user table named* Usrtbl02, *and the connection to other system tables that contain information about* Usrtbl02.

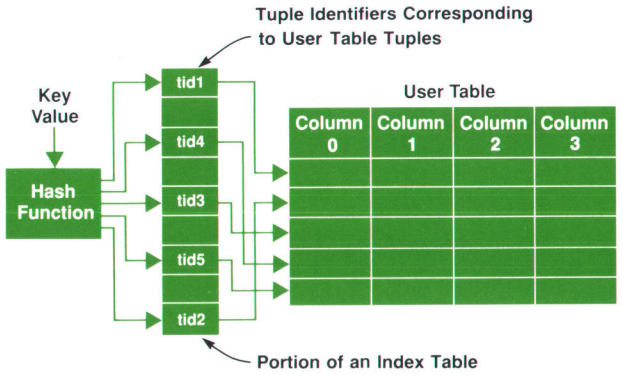


Fig. 11. *Index hashing scheme in HP RTDB.*

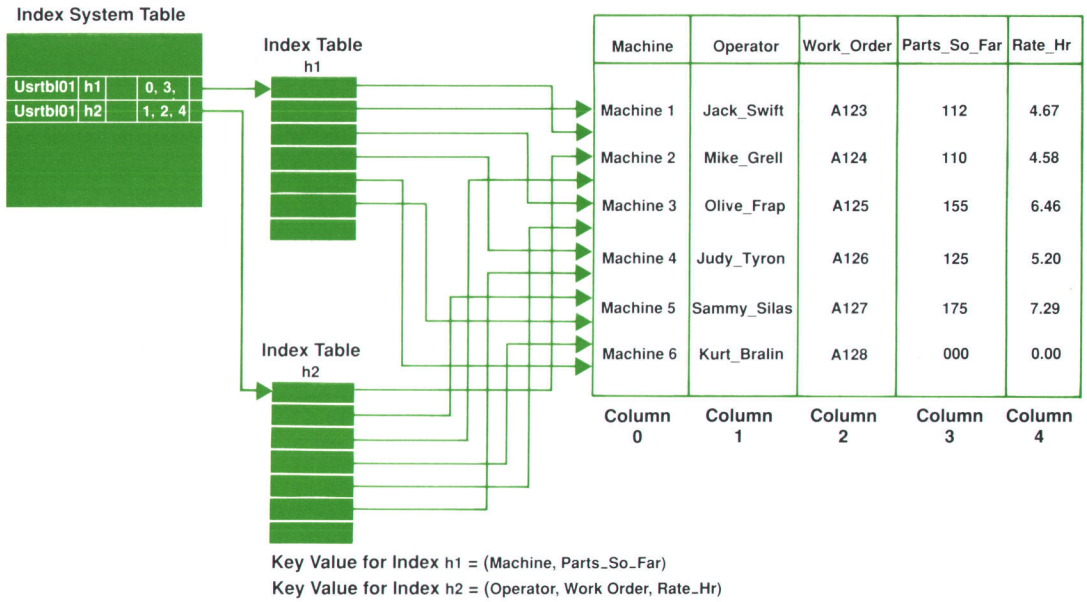


Fig. 12. *A user table with indexes defined on it. The index system table contains the numbers of the columns in the user table that make up the key values for index tables* h1 *and* h2.

when the function is applied to diverse and unpredictable sets of keys. Therefore, when a collision does occur, there must be a means of specifying an alternate location in the index table where the new tuple identifier can be stored. HP RTDB uses a form of separate chaining in which each hash index consists of two segments, a primary segment and a secondary segment (see Fig. 13).

The primary segment contains the tuple identifiers of all keys that hash to an unused primary location. To reduce the probability of collisions, the number of locations in the primary segment can be configured by the user to be more than the maximum number of tuples in the data table. For example, if the number of primary segment locations is 1.25 times the number of tuples in the data table (primary ratio), then the load factor (packing density) of each primary segment cannot exceed 80 percent. What this means is that for a table with eight tuples the number of primary segments is 10 (1.25 × 8), and if there are no collisions, at most eight of the tuples in the primary segment will be occupied. A higher primary ratio will reduce the probability of collisions but will increase the primary segment size. Users can adjust each index's primary ratio to achieve the best performance and minimum memory consumption.

The secondary segment contains the tuple identifiers of all data values that hash to a primary location that is already in use. This segment provides a separate overflow area for secondary entries (synonyms), thus eliminating the problem of migrating secondaries (existing synonyms that must be moved to make room for a new primary entry). The secondary segment is allocated based upon the number of tuples in the data table and is guaranteed to be large enough for even a worst-case index distribution. After a collision occurs at a location in the primary segment, the primary location becomes the head of a linked-list synonym chain for all secondaries that hash to that primary location.

Input Areas. Input data in a real-time environment may be expected or unsolicited, and can arrive in streams, small packets, or large bursts. This data may also involve complex synchronization of processes to handle the data. In all cases, there is a need to respond to the arrival of the new data within a predictable time before it is too old to be of value, or is overwritten by the next arrival.

Input areas provide highly efficient buffering for receiving and storing unstructured data into the data base. Users

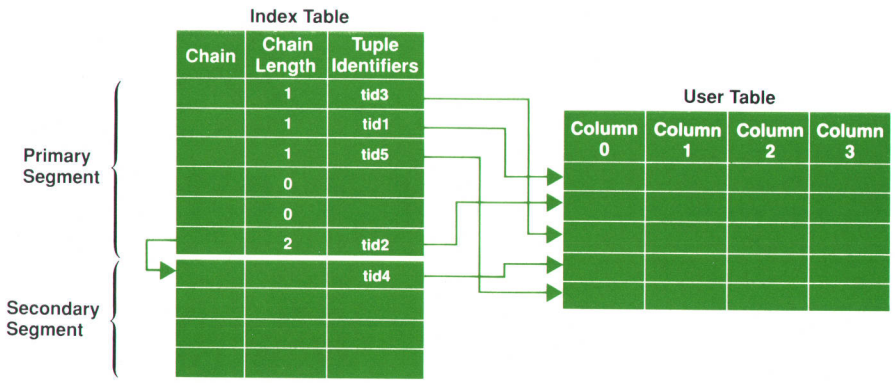


Fig. 13. *Primary and secondary segments in the hashing and collision resolution scheme.* Tid4 *is put into the secondary segment because it hashed into the same location as* tid2.

can configure a data base with any number of input areas of any size up to the limits of available shared memory. Values in input areas can be read or updated either using offsets in a named input area or, for even higher performance, using an input area's actual address as if it were a local array variable. Like tables, input areas can be explicitly locked and unlocked for control of concurrent access.

Data Access

Traditional data base transactions are not supported in HP RTDB because each access to the data base is considered a transaction, and each access is guaranteed to be serialized and atomic. However, a system designer can still define and implement an application transaction as a set of two or more data base accesses, which will complete or fail as a group.

The general data access flow in HP RTDB is shown in Fig. 14. The sequence of events to access the data base to update a tuple would be:

- Obtain the address of the main control block using the session identifier (SessID). The session identifier is returned to the user when the data base is opened and is used in subsequent calls to access the data base.
- Obtain the address of the table system table from the main control block, and using the table identifier (TblTid) obtain the tuple of the user table from the table system table. The user is given a table identifier when the table is opened.
- Obtain the entries in the locks-held and semaphore control blocks and lock the user table. The addresses for these entries are obtained from the main control block.
- Finally, obtain access to the tuple in the user table using the user table address obtained from the table system table and the tuple identifier (tid).

This process is the same for input areas, except that the input area system table is accessed rather than the table system table, and the input area offset is used instead of the tuple identifier.

Performance tests to assess the data access performance characteristics of the HP Real-Time Data Base routines were run on an HP 9000 Model 825. The benchmark for these tests consisted of 56-byte tuples. During the tests, shared memory was locked into physical memory. The results of these performance tests are summarized in Fig. 15.

Table Access. There are three methods of locating tuples in user tables. Tuples can be read using sequential access, hashed key index access, or direct tuple identifier access. However, updates and deletions of tuples can only be done by direct tuple identifier access. This means that to update or delete a tuple, it must first be located by one of the three read access methods. Sequential access starts at the beginning of a table and returns each tuple in the order in which it is encountered. Since tuples can be inserted or deleted at any location, the physical order of tuples in an active table is unpredictable. The user can set up search conditions for sequential searching, such as the number of comparisons, columns to use, comparison operator (e.g., EQ, NE, etc.), and ASCII type (e.g., hexadecimal, octal, etc.). If the search conditions are specified, then only those tuples that meet the conditions are returned. Sequential access is the slowest mode of access, but for very small tables of 10 to 15 tuples, the speed of sequential searching is comparable with indexed searching. Sequential access is most appropriate for serially processing most or all of a table's data, since it does not use additional memory for index tables.

Indexed access, which uses the indirect hashing technique discussed earlier, is much faster than sequential access, but still slower than direct access by tuple identifiers. Index keys can consist of up to five contiguous or noncontiguous table columns of mixed data types and any number of indexes can be defined for a single table. Although there is no hard limit as to how many indexes can be defined for a table, each index requires additional memory for index tables and additional processing time to create or update each index key defined. Indexed access is best for applications that need fast, unordered access to data and that mainly perform reads and updates rather than insertions and deletions.

The HP RTDB tuple update routine allows three alternative courses of action when columns that make up an index key value are about to be updated. One option specifies that the indexes should be modified (that is, rehashed) to reflect the update. A second option is to deny the update and return an error when an index key value is about to be changed. For top performance, there is an option to update a tuple and bypass checking for index modification. This option should only be used if the user's application can ensure that a tuple's index key values are never changed after its initial insertion into a table.

Direct access by tuple identifier is by far the fastest form of access. A tuple's tuple identifier is returned when it is first added to a table and also when the tuple is accessed by a sequential or indexed search. The returned tuple identifier can then be used to update, delete, or reread the same tuple directly any number of times since tuple identifiers do not change, even when a table is rehashed. This feature offers spectacular benefits when a small set of tuples is repeatedly accessed. The tuple identifier can be obtained once, stored internally, and then used to access the same tuples directly in all subsequent operations.

Input Area Access. Data in input areas can only be read or updated. Since input areas are usually updated by being overwritten with new input data, HP RTDB does not provide separate routines for deleting and inserting data elements in input areas. Nor are these functions really needed, since updating an element to a null or non-null value accomplishes the same end.

Since the structure and content of input areas are appli-

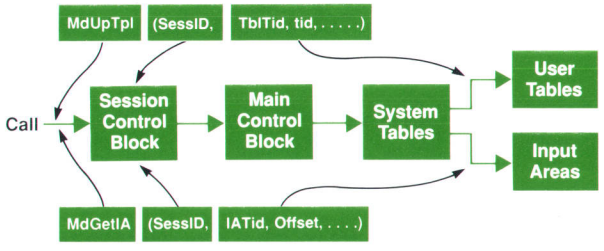


Fig. 14. *Data access in HP RTDB.*

cation dependent and can change at any time, HP RTDB does not try to map input areas as it does tables. Data elements in input areas are accessed by naming the input area and specifying the element's offset and length. HP RTDB then locates the start of the input area, adds the offset, and reads or updates the selected data element. For maximum performance, the input area may optionally be addressed directly as if it were a local array, but this access mode bypasses HP RTDB's address validity checking and concurrency control mechanisms and should only be used if the extra performance is critical. The application must then ensure correct addressing to avoid data base corruption.

Users can also associate a table column with an input area data element by defining the column's data type as a pointer to an input area and then assigning the data element's offset as the column's value. The input area offsets shown in Fig. 8 are input area pointer types.

Configuration and Creation

Much effort was made to keep the process of configuring and creating a data base as simple and flexible as possible. The final definition of data base objects intended for future implementation can be deferred until they are actually needed, and other data base objects can be added and/or removed after the data base is created. Also, all data base configuration and maintenance functions can be done with the interactive HP RTDB query/debug commands as well as by calling HP RTDB subroutines. This allows users to prototype and test data base designs without writing a single line of program code.

Defining the Data Base Schema. The first step in creating an HP RTDB data base is to define the system limits of the data base, that is, the data base name, the configuration file name, the maximum number of tables, input areas, indexes, columns, and sessions that will be accommodated by the data base at any time. The user may choose to defer the actual definition of some data base objects until a later time, but must inform HP RTDB of how many of each object may eventually be defined. This information is used to ensure that the system tables and control structures for the data base are as large as the maximum requested. Preallocation of contiguous storage for the maximum size of the data base objects instead of allocating on an as-needed basis eliminates the overhead of checking for available space when adding tuples to a table. It also eliminates the overhead associated with dynamically allocating and deallocating blocks in a multiuser environment.

When the system limits are defined, a skeletal schema and control structures are generated in shared memory and saved on disc in the configuration file. At this point the data base schema can be filled out with the definition of user tables, columns, input areas, and indexes either through query/debug commands or by calls to the HP RTDB subroutines (MdDefTbl, MdDefCol, MdDefIx, and MdDefIA). As these other data base objects are defined, the information about them is entered into the system tables and the schema grows more complex. However, no storage is allocated for

		No Locking	Locking
Data Retrieval	Direct Get	~15 μs/Transaction (66,666 Transactions/s)	~478 μs (2,092/s)
	Index Get	~71 μs (14,085/s)	~692 μs (1,445/s)
	Input Area Get	~42 μs (23,809/s)	~542 μs (1,845/s)
Data Update	Update Tuple (No Index Operations)	~21 μs (47,620/s)	~530 μs (1,887/s)
	Update Input Area	~50 μs (20,000/s)	~544 μs (1,838/s)
		Explicit Lock Done Once for All Updates	Implicit Lock Done Once for Each Update
Insertion/ Deletion	Add Tuple (No Index Operations)	~50 μs (20,000/s)	~673 μs (1,486/s)
	Delete Tuple (No Index Operations)	~37 μs (27,027/s)	~625 μs (1,600/s)
Data Base Backup	Save Data Base Image Data Base Size: 34,000 Bytes 6 Tables, 2 Input Areas	~2 Backups/s	

Fig. 15. *Results of performance tests for the HP RTDB data base routines. The tests were run on an HP 9000 Model 825. The benchmark for these tests consisted of 56-byte tuples. In HP RTDB each data base access is a transaction.*

newly defined data base objects until the data base is built. The user can at any time save a copy of the current memory-resident schema to the configuration file. When the data base is fully operational and contains data, the data as well as the schema can be saved in the same file.

Building a Data Base in Memory. Once the system limits of the data base are set and the schema defined, the data base can be built. First, the schema must be loaded into memory. The schema will already be in memory if the data base is newly defined. Otherwise, the schema file on disc is opened and loaded into memory (MdOpenDb). Using the memory-resident schema data, the HP RTDB build routine (MdBuildDb) allocates shared memory to each data base object, builds and initializes any data or control structures associated with the objects, and sets up the logical links between the structures. HP RTDB also provides routines to calculate the minimum memory needed to build the data base from the schema. Additional memory may optionally be allocated to allow for future implementation of data base objects that are not yet defined in the schema. After a data base is built, it is ready to be initialized with application data.

Locking and Concurrency Control

There are three components to the synchronization of concurrent access to a data base: session management, lock management, and semaphore management. As each new user process opens the data base, HP RTDB allocates a session control block for the new process and the process becomes attached to the data base. A session identifier is returned to the process for use in subsequent calls to access the data base, and the session control block is filled with information about the process such as the HP-UX process identifier (pid) and user identifier (uid). The session identifier is used to index into the locks-held table. With this and other data the session manager is able to perform its role in controlling concurrent access to the data base.

Locking in HP RTDB is provided only at the table and input area level rather than at the tuple and data item level. This coarse granularity of locking is acceptable because in a memory-resident data base, locks are normally held for very short periods of time. Each object (user table and input area) in the data base is owned by a specific semaphore. Locking of data base objects is accomplished by acquiring the object's semaphore and associating it with the process's session identifier. The lock is released by freeing the semaphore and breaking the link between the semaphore and the session identifier.

HP RTDB controls the locking and unlocking of semaphores, but all queuing and rescheduling of blocked processes is handled by the HP-UX operating system. This gives HP RTDB a simple, efficient, and reliable concurrency control mechanism that is guaranteed to be compatible with other HP-UX features. For example, a user could easily integrate HP RTDB and the HP real-time extension features to implement real-time priorities in an application. HP real-time extensions are additions to HP-UX that allow users to set high priorities for certain processes.

If the application does not explicitly lock a data base object before trying to access it, the HP RTDB routine called to do the access will normally apply an implicit lock of the object by default. There are options to allow users to read and write through locks, but these options may compromise the integrity of the data base and should be used with caution when higher performance is critical. A read-through lock allows a high-priority process to access data in the data base that may be in the process of being updated by another process.

Security

HP RTDB provides three levels of security through the use of passwords. A password is required to access the data base. The password level, which can be data base administrator, read-write, or read-only, determines the user's access authorization. The data base administrator password allows a user process to perform any operation on the data base supported by HP RTDB subroutines or by the query/debug commands. The read-write password allows a user process to read, modify, and delete data in the data base, but not to perform any of the data base definition functions, such as adding or removing a table or index. The read-only password allows a user only to read data in the data base, but not to delete or modify data, or perform any data base definition functions.

In addition to the password security, the data base administrator (or the root user) can also alter the access permissions of the schema file on disc or the data base's shared memory segment to limit access to the data base.

Backup and Recovery

Memory-resident data bases are particularly vulnerable to power failures and operating system failures because both types of failures usually destroy the contents of main memory. Battery backup power systems can provide excellent protection against power failures, but system failures pose a problem that really has no good solution.

The traditional backup methodology of logging all changes to a disc file cannot be used if high performance is desired; yet there is no other way to keep a secure backup with close parallel to the state of memory. HP RTDB provides a "snapshot" backup which allows each application to choose an acceptable trade-off between performance and secure backup.

At any time, the application can call an HP RTDB routine to save an image of the data base schema or the schema and data to a disc file. For a data base of 34,000 bytes consisting of 6 tables and 2 input areas, a single snapshot takes about 0.5 second on an HP 9000 Series 825. Snapshots can be taken as often or as rarely as the user application chooses, and can be triggered periodically or by specific events. Users who can afford the overhead can take more frequent backups while users who require top performance may rarely or never take a backup. In some real-time applications, there is little point in taking a backup since the data would be obsolete long before the system could be restarted. Data base recovery is also very fast but a data base can only be recovered to the point where the last snapshot was taken. Either the schema or the schema and the data can be recovered. Recovery of the schema only would create an empty data base which could then be reinitialized with data by the application.

Query/Debug Utility

The query/debug utility is included as part of the HP Real-Time Data Base software to provide real-time application developers with a tool to:

- Assist with prototyping, testing, and debugging applications
- Create, modify, and maintain HP RTDB data bases in both development and production environments
- Use as a simple and flexible HP-UX filter, which when combined with other HP-UX commands and utilities, can provide useful application functions to HP RTDB users without the need for additional code.

The query/debug utility supports nearly all of the functionality of the HP RTDB subroutines. However, it is highly generalized and is designed to be safe and friendly rather than fast. Therefore, most query/debug functions are significantly slower to execute than equivalent subroutine calls.

The query/debug command syntax is modelled after the Structured Query Language (SQL), an ANSI industry-standard relational interface. This resemblance to SQL is intended only to make it easier for users who are already familiar with SQL. The query/debug utility is not intended to support the SQL standards of inquiry or reporting functionality.

The query/debug utility was designed as an HP-UX filter and conforms to the conventions for filters in its use of HP-UX input/output files stdin, stdout, and stderr. This allows it to be used with input and output redirection, pipes, and so on. Output can optionally be produced without headings to enable clean output data to be piped directly into other filters or user-written programs.

Query/debug commands can be entered interactively for ad hoc work, or can be read from ordinary disc files for repetitive tasks. For example, the commands to define and initialize a data base could be saved in a disc file to ensure that the data base is always recreated the same way. Likewise, simple reports can be generated using query/debug command files or by combining query/debug command files with HP-UX shell scripts and utilities.

The query/debug commands provide the following functionality:

- Define, reconfigure, build, remove, and back up a data base
- Change passwords and shared memory security permissions
- Initialize table and input area values in a new data base
- Display data base configuration and status information
- Generic add delete, update, and select of tuple values based upon indexed or sequential searches
- Display or print all or selected data from all or selected tuples in a table in either tabular or list format
- Generic update, select, and display of input area values
- Load or store data base values from or to disc files
- Debugging aids such as hexadecimal data display, a "peek" function, and an error trace option for tracing all errors that may occur during any query/debug processing of the user's data base
- On-line help facility for all query/debug commands
- Built-in octal, decimal, and hexadecimal integer calculator
- Execution of HP-UX commands without leaving query/debug.

Conclusion

The goal of a high-performance data base drove many of the design decisions and implementation techniques for HP Real-Time Data Base. The performance goals were met and exceeded with simple data structures (tables and input areas), programming techniques such as macros, and options that allow users to eliminate features that affect performance. The result is a set of routines that enables real-time application developers to create custom data bases for capturing and retrieving the diverse data structures found in real-time environments.

Acknowledgments

The authors wish to thank Marie-Anne Neimat, Ming-Chien Shan, and Bob Price for their assistance in the design of RTDB, and Mark Butler for his direction in the creation of RTDB.

New Midrange Members of the Hewlett-Packard Precision Architecture Computer Family

Higher performance comes from faster VLSI parts, bigger cache and TLB subsystems, a new floating-point coprocessor, and other enhancements. A new 16M-byte memory board is made possible by a double-sided surface mount manufacturing process.

by Thomas O. Meyer, Russell C. Brockmann, Jeffrey G. Hargis, John Keller, and Floyd E. Moore

NEW MIDRANGE HP PRECISION ARCHITECTURE computer systems have been added to the HP 9000 and HP 3000 Computer families. The HP 9000 Model 835 technical computer and the HP 3000 Series 935 commercial computer share the same system processing unit (SPU). Designed with significantly improved floating-point and integer performance, the Model 835/Series 935 SPU meets the computational needs of mechanical and electrical computer-aided engineering (CAE) and multiuser technical and commercial applications.

The HP 3000 Series 935 (Fig. 1) is configured for business applications and runs HP's proprietary commercial operating system, MPE XL. HP 9000 Model 835 products include the Models 835S and 835SE general-purpose multiuser computers, the Models 835CHX and 835SRX engineering workstations with 2D and 3D (respectively) interactive graphics, and the powerful Model 835 TurboSRX 3D solid-rendering graphics superworkstation with animation capability (Fig. 2). All Model 835 systems run the HP-UX operating system. As a member of the HP Precision Architecture family, the Model 835/Series 935 SPU supports a wide variety of peripherals, languages, networks, and applications programs.

User Requirements

Like its predecessor, the Model 825/Series 925 SPU,[1] the Model 835/Series 935 SPU's definition was driven by requirements from several different application areas. In addition to the requirements of small size, low power dissipation, low audible noise, flexible I/O configurations, and tolerance of a wide range of environmental conditions normally required for a midrange technical or commercial product, the Model 835/Series 935 SPU design addresses several other needs. For scientific computation and me-

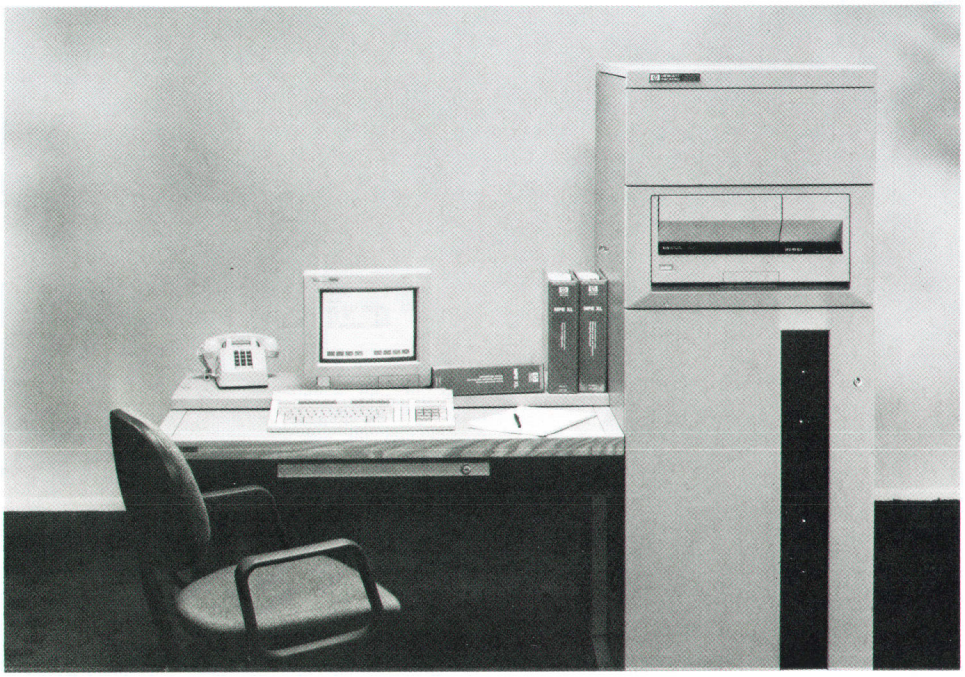


Fig. 1. *The HP 3000 Series 935 is configured for business applications and runs HP's proprietary commercial operating system, MPE XL.*

chanical and electrical CAE applications, high floating-point and integer computational performance is desired. The Model 835/Series 935 SPU provides more than a 300% increase in floating-point performance and more than a 50% increase in integer performance over the Model 825/Series 925. The Model 835/Series 935 has been benchmarked at 14 MIPS and 2.02 MFLOPS.*

Customers who own or plan to purchase a Model 825 or Series 925 want the ability to upgrade to the faster Model 835/Series 935 without having to replace the whole computer. To meet this requirement, the Model 835/Series 935 processor is designed so that an upgrade can be easily done at a customer's site by exchanging two boards. Because of HP Precision Architecture compatibility, user applications migrate and realize enhanced performance without modification or recompilation.

For all application areas, main memory capacity is an important influence on overall system throughput. To meet increased memory requirements, a compact, double-sided surface mount 16M-byte memory board has been made available. Designed to work in any of the Model 825/Series 925 or Model 835/Series 935 products, this board doubles memory capacity to either 96M or 112M bytes depending on the configuration.

Design Overview

The Model 835/Series 935 uses many of the same components as the Model 825/Series 925 SPU. Common components include the mechanical package, the power supply, I/O cards, the I/O expander, the battery backup unit, and the memory cards. This high degree of commonality not only assures easy upgrade potential but also minimized design time.

*MIPS (million instructions per second) performance is relative to a Digital Equipment Corporation VAX 11/780 computer in a single-user, multitasking environment, as calculated from the geometric mean of a suite of 15 integer benchmarks. MFLOPS (million floating-point operations per second) performance is measured using the Linpack benchmark, double-precision, with coded Basic Linear Algebra Subprograms (BLAS).

A block diagram of the Model 835/Series 935 SPU is shown in Fig. 3. The two boards unique to this SPU are the processor and processor dependent hardware (PDH) boards highlighted in the block diagram and shown in Fig. 4.

The following sections will explain the approaches taken to meet the performance requirements mentioned earlier. In addition, the design considerations for a compact 16M-byte memory board using a new double-sided surface mount manufacturing process will be discussed.

Processor Board

The Model 835/Series 935 processor board reuses much of the technology developed for the Model 825/Series 925, a practice frequently called "leverage" within HP. Eight VLSI integrated circuits make up the core of the processor board: the CPU (central processing unit), the SIU (system interface unit), two CCUs (cache controller units), the TCU (TLB controller unit), the FPC (floating-point controller), and two commercially available floating-point chips. Of these, the CPU, SIU, TCU, and two CCUs are functionally identical to those used in the Model 825/Series 925 processor but run 20% faster. These parts were designed in HP's NMOS-III VLSI process.[2,3] The FPC and the floating-point chips, new for the Model 835/Series 935 processor, will be discussed later.

In addition to faster VLSI, a number of performance enhancements over the Model 825/Series 925 processor board are found on the Model 835/Series 935 processor board. These include:

- An eight-times-larger cache (128K bytes by 2 sets, unified instructions and data).
- A two-times-larger translation lookaside buffer or TLB (2K instruction entries and 2K data entries). Since HP Precision Architecture defines page sizes to be 2K bytes, this allows 8M bytes of main memory to be mapped directly into the TLB.

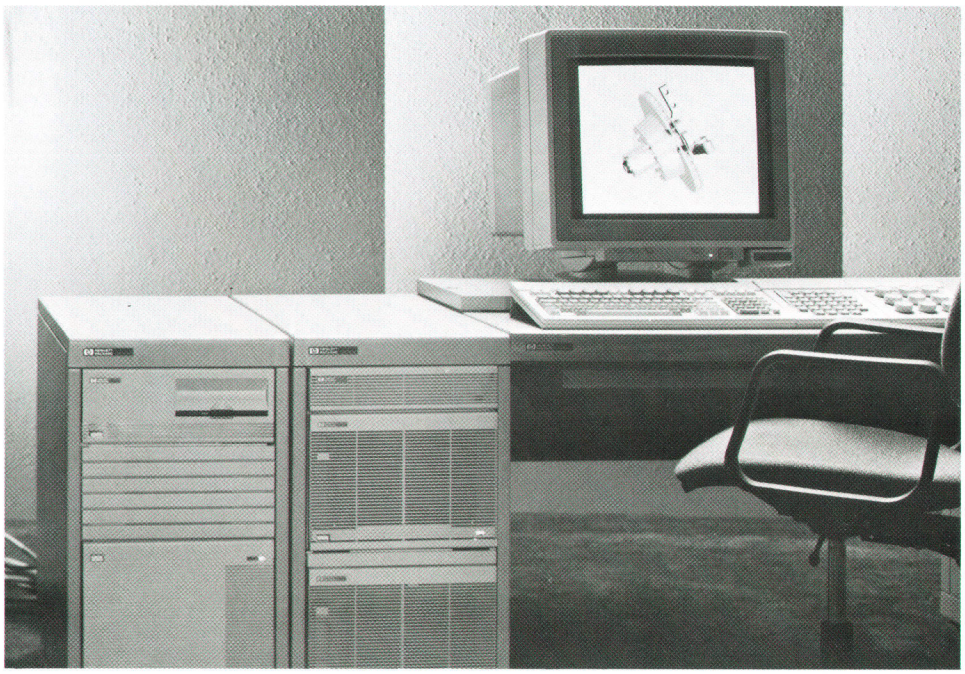


Fig. 2. *The HP 9000 Model 835 TurboSRX is a 3D solid-rendering graphics superworkstation with animation capability. Like other Model 835 multiuser technical computers, it runs the HP-UX operating system.*

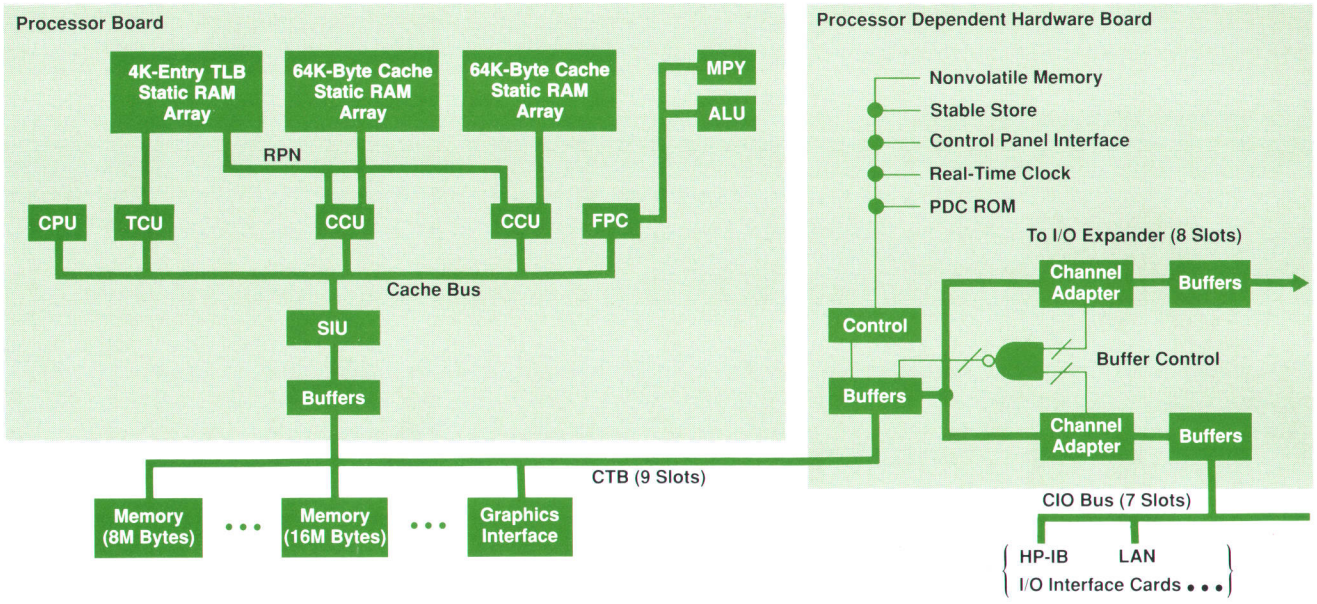


Fig. 3. *Block diagram of the Model 835/Series 935 system processing unit.*

- A new single-chip clock buffer that replaces over 40 discrete parts and, along with faster VLSI, allows a 20% increase in clock frequency.
- A new floating-point coprocessor based on a new NMOS-III VLSI floating-point controller (FPC) and two floating-point chips.

The increased cache size and faster cycle time account for the increased integer performance and partially account for the improved floating-point performance. However, the new floating-point system is mainly responsible for the greater than 300% improvement in overall floating-point performance.

The performance of a central processing unit can be described by three items:

- The amount of work each instruction does
- The average number of cycles per instruction executed (CPI)
- The cycle time of the CPU.

In general, reduced instruction set computers (RISC) trade off the first item with the last two. By performing extensive simulations to determine the most important instructions and making appropriate trade-offs in instruction power versus cycle time, HP has been able to minimize the impact of this trade-off. As a result, the HP Precision Architecture instruction set is very powerful despite being classified as RISC. Consider, for instance, the 14-MIPS rat-

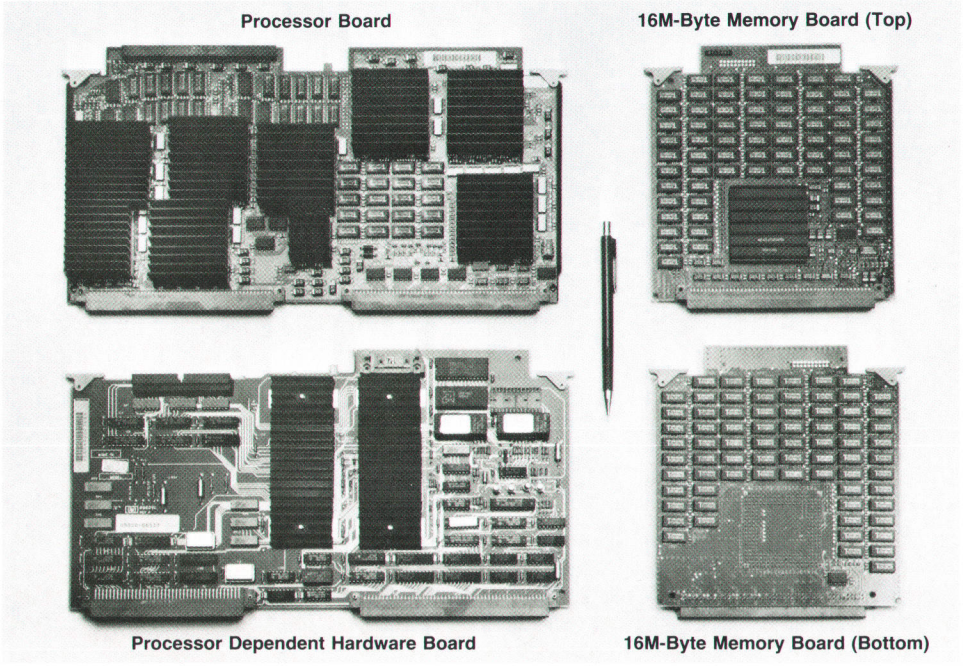


Fig. 4. *Printed circuit boards designed or modified for the Model 835/Series 935 SPU.*

ing and the 15-MHz peak instruction rate of the Model 835/Series 935 SPU. The sustainable instruction rate is approximately 10.8 MHz,[4] not considering cache and TLB misses. The actual rate will be lower. The reason that the SPU is rated at 14 MIPS relative to a Digital Equipment Corporation VAX 11/780 when the actual instruction rate is less than 10.8 MHz is that for this suite of benchmarks an average mix of HP Precision Architecture instructions performs more work than an average mix of instructions for the VAX 11/780, a complex instruction set computer (CISC).

Since the Model 835/Series 935 uses the same CPU chip as the Model 825/Series 925, the improvements in CPI have been made external to the CPU. When the CPU executes directly from the on-board cache memory, it proceeds at nearly the full instruction clock rate. However, when an instruction or data is needed that is not in the cache, the CPU temporarily halts execution while the needed data is fetched from main memory. This has the effect of increasing the average number of cycles to execute an instruction. By increasing the cache size by a factor of eight and increasing the TLB size by a factor of two, the cache and TLB hit rates are significantly improved. The result is that the CPU can execute directly from the cache a much greater percentage of the time.

The third item, the cycle time of the CPU, has also been improved in the Model 835/Series 935. The Model 825/Series 925 processor board clock is designed to provide a 25-MHz clock rate. Operation beyond 25 MHz would push the design beyond original specifications. To overcome this limitation, an NMOS-III clock chip designed for an earlier HP product has been adapted to meet Model 835/Series 935 design requirements. As an added benefit, part count is reduced by more than 40, freeing up critical space for other functions.

By using the faster VLSI chips and the NMOS-III clock buffer, the Model 835/Series 935 processor board runs at a frequency of 30 MHz. Operation at 30 MHz provides the additional benefit that the CTB (central bus, formerly Mid-Bus), which is designed to operate at any frequency from 6.67 MHz to 10 MHz, can be run at its peak frequency of 10 MHz. This is because the CTB frequency is derived from the system clock by a simple divide-by-three circuit in the SIU. All CTB boards designed to run at the full 10-MHz CTB frequency operate in either the Model 825/Series 925 or the Model 835/Series 935.

All of these enhancements, of course, don't come free. The larger cache and TLB would require almost twice the area they occupied on the Model 825/Series 925 processor board using standard through-hole printed circuit technology. In addition, to achieve the floating-point performance goals, the floating-point chips had to be located on the processor board to allow faster communication between the FPC and the floating-point chips. (In the Model 825/Series 925 SPU, the floating-point coprocessor is split between two boards.) These changes, along with additional bypassing and other changes, add three large VLSI chips (two are 169 pins each and one is 72 pins) and a total parts count increase of 79 parts.

Examination of the Model 825/Series 925 processor board reveals that there is very little room for more parts. To fit all the extra components onto the processor board, surface mount technology (SMT) is used. SMT is rapidly gaining favor as the board technology of choice within HP, largely because of its increased density and potential for lower manufacturing cost. In addition to fitting the extra 79 parts onto an already crowded board, SMT allows more than 96% of the board to be machine-loaded.

Design Time

The Model 835/Series 935 project schedule called for introduction less than one year after the Model 825/Series 925.The tight schedule depended upon reusing as much technology as possible. Significant work had already gone into developing tools based on HP's Engineering Graphics System (HP EGS). Most of the VLSI chips are unchanged, so a significant part of the design is taken directly from the Model 825/Series 925 processor board design. Additional custom macros were added to HP EGS to speed layout. The flexibility of HP EGS allowed easy addition of SMT capability to the editor. Software tools developed by the lab to perform design rule checks directly on photoplot output and to compare results to the original schematic were enhanced to understand the additional requirements of SMT. The extra effort in tool development paid off well. The very first printed circuit boards booted the HP-UX and MPE XL operating systems without any workarounds.

Floating-Point Coprocessor

The Model 825/Series 935 floating-point coprocessor provides hardware assistance for floating-point math operations and is implemented by a floating-point controller (FPC) and two floating-point integrated circuits. One of the floating-point ICs, the ALU, performs the addition, subtraction, compare, and convert operations, and the other, the MPY, performs the multiply, divide, and optional square root operations. All floating-point operations can be either single-precision or double-precision and fully support the IEEE 754 floating-point standard.

The FPC, as the name implies, is the central control circuit for the floating-point coprocessor. It interprets the floating-point instructions and manages the flow of operands and results to and from the floating-point chips. The FPC contains twelve 64-bit floating-point registers, a status register, seven operation-exception registers, and a configuration register.

The FPC gets its floating-point instructions and operands over the cache bus. Instructions come from the CPU, but operands are read into the 12 floating-point registers directly from the cache. Double-precision operands require three cache bus cycles to transfer the data. The first cycle transfers the floating-point load instruction and the next two transfer the operand. Single-precision operands require only two cache bus cycles. When a floating-point operation is begun by the CPU, the operands are loaded into the operation-exception registers from the floating-point registers to be forwarded to the floating-point chips over the 64-bit math bus. Although the FPC has seven operation-exception registers, it only uses the first two. (The remaining five are for architectural compliance.) These registers act as a queue for the operations and also

indicate exceptions in the event of a trap. The first register contains the currently executing operation, while the second may contain a queued operation waiting to begin.

Feature Set

Besides supporting the HP Precision Architecture floating-point operations, the FPC has performance enhancements that decrease the number of states during which the CPU suspends the cache bus while waiting for an FPC instruction to be accepted. Three major performance enhancements designed into the FPC are:

- Queuing of two floating-point operations
- Register bypassing on interlocking stores
- Interlock result bypassing on the math bus.

The FPC executes floating-point operations in the order issued by the CPU. Queuing up at most two floating-point operations allows the CPU to issue two floating-point operations back-to-back without having to wait for the first operation to complete before the second operation is accepted. Since performance analysis shows that the FPC is likely to complete a floating-point operation before a third operation is issued by the CPU, the FPC is designed to accept only two successive floating-point operations. If a third operation is issued to the FPC while there are still two operations in the queue, the FPC will suspend the CPU until the first floating-point operation is completed.

When a floating-point store instruction is encountered in a program, the data in one of the floating-point registers will not be immediately available to send to the cache if the store specifies a register that is the target of a pending floating-point operation. In this case the FPC will suspend the CPU until the operation is done before providing the data. The penalty for this interlocked store is reduced two states by driving the result onto the cache bus at the same time it is being stored into the target floating-point register.

The third FPC enhancement allows a result on the math bus to be reloaded into the math chips as an operand for the next operation. This means the FPC does not have to reissue the operand over the math bus for the next operation. This saves three states in the execution of the following operation for double precision and two states for single.

Circuit Design

The FPC is physically divided into ten principal blocks (Fig. 5): a cache bus interface, a math bus interface, a register stack, interlock logic, two cache bus control programmable logic arrays (PLAs), two math bus control PLAs, a test state PLA, and a clock generation block. The math bus side of the chip includes 64 driver/receiver pairs that are routed to the register stack by a 64-bit data bus, and 21 pairs that provide various control functions. The cache bus interface includes 32 driver/receiver pairs for cache bus data and 27 pairs for control and identification of transactions and coprocessor operations.

An important feature of the FPC is the hardware interlock block, which performs two main functions. The first is to detect register bypassing opportunities by comparing the registers referenced by coprocessor operations and manage the bypass data traffic. The second is to determine the interlocks for loads and stores, allowing them to be received by the FPC and be handled in parallel with arithmetic operations if the referenced register is not involved in operations queued in the FPC.

A problem frequently encountered in the design of integrated circuits with wide bus interfaces such as the FPC is the generation of noise on the circuit's power supplies when many of the chip's pads are driven at the same time. Since VLSI packages in a system are decoupled with bypass capacitors to provide stable supply levels at the package leads, this noise is caused primarily by the parasitic inductance of the pin-grid array (PGA) package. The magnitude of this noise is given by the relationship

$$v = L\frac{di}{dt},$$

where v is the noise voltage generated, L is the parasitic inductance of the package, and di/dt is the derivative of the current through the package leads. The expression suggests the two ways used to reduce package noise in the FPC.

First, the number of leads devoted to power supplies is increased, effectively decreasing the parasitic inductance between the internal circuitry and the stable voltage levels provided by the printed circuit board decoupling. A high-pin-count package, a 272-pin PGA, was chosen to minimize this effect for the FPC. Second, current switching (di/dt in the inductance voltage expression) can be decreased. This is the most important effect for the FPC. To minimize this effect, attention is focused on the pad drivers, normally the most heavily loaded circuits on an IC. The critical factor is not the current capability of these drivers once they are fully turned on, but how rapidly they are turned on. Fig. 6 illustrates the solution to achieving low noise in high-current drivers. The final driver, inverter 4 in the chain driving the PC board load, is a large driver capable of sinking or sourcing the off-chip load. Inverter 3 is much smaller, with less than one tenth the drive capability of inverter 4, and turns on inverter 4 somewhat slower. This slow turn-on of inverter 4 doesn't greatly compromise how quickly the off-chip load is driven. It is the final current in inverter 4, once it is fully turned on, that is the first-order factor in how rapidly the off-chip load is driven.

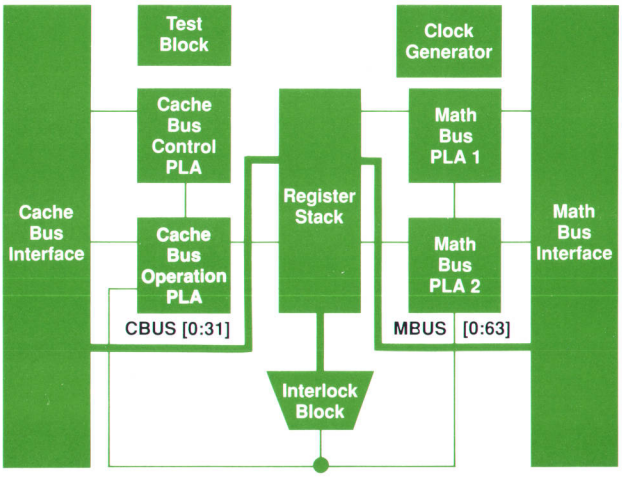


Fig. 5. *Block diagram of the floating-point controller chip.*

Double-Sided Surface Mount Process

Requests for increased component packing densities by product development labs have prompted the development of a full double-sided surface mount process (SMT-2) with the ability to place reasonably large ICs on the bottom side of the printed circuit board. A standard process was developed for use by all Hewlett-Packard surface mount manufacturing sites by the Surface Mount Development Center of the HP Computer Manufacturing Division. The process was implemented by the Colorado Surface Mount Center to build the 16M-byte memory board described in the accompanying article. Other HP manufacturing sites are currently installing this process.

Design Requirements

Requirements for the SMT-2 process were received from product R&D labs and all the surface mount manufacturing sites. First, component packing density improvements were needed. Up to two times single-sided surface mount densities were required to add memory capability, reduce printed circuit board sizes, and minimize the number of printed circuit assemblies.

Several requests were made for ICs in SOJ (small-outline J lead), SOIC (small-outline IC), and PLCC (plastic leaded chip carrier) packages on both sides of the printed circuit board. To be useful in most applications, 28-pin and sometimes 44-pin ICs need to be placed on the bottom side of the board. Larger ICs could be restricted to the top of the board for most applications.

Through-hole components would be required for most assemblies. In general, these would be types of components that are still difficult to obtain in SMT packages, such as connectors and PGAs (pin-grid arrays).

From a processing standpoint, the SMT-2 process had to be compatible with SMT-1, HP's single-sided surface mount process, on the same manufacturing line. Minimum additions or changes could be made to major pieces of equipment for the new process.

Problems and Solutions

To develop a process to meet these requirements, several questions had to be answered. The first was how to adhere the components on the bottom side while reflowing components on the top side. Second, how would the through-hole parts be soldered? Finally, would the joints formed be as reliable as joints on a single-sided surface mount assembly?

The alternatives considered for adhering the bottom-side components during reflow of the top side included:

- Glue one side and reflow both sides at once.
- Make two passes through reflow using different solder pastes that reflow at different temperatures. The components on the bottom side would be reflowed first using a higher-temperature solder paste.
- Reflow the components on the bottom side first. During top-side reflow, surface tension would be relied on to keep the components on the bottom side from falling off.

The feasibility of relying on the surface tension to hold parts has been demonstrated.[1] The mass of a plastic leaded chip carrier that can be supported by the surface tension of molten solder can be simply calculated from a knowledge of the perimeter of each lead wetted by the solder, the surface tension of molten solder, and the number of leads per package. This model shows that PLCC packages up to 68 pins should not fall off. Experiments have shown, however, that problems start to occur above 44 pins. This is probably because the model does not take into account all the factors affecting the parts during reflow, such as belt vibration and incline. Since surface tension will support the required components, there is no need for gluing or using special solder pastes which would add considerable complexity to the process.

(continued on next page)

Risk Reduction and Flexibility

One of the FPC design features is that it controls floating-point chips of varying speeds from different vendors. This allows the use of the FPC in the HP 9000 Models 850S and 855S as well as the Model 835/Series 935 SPU.

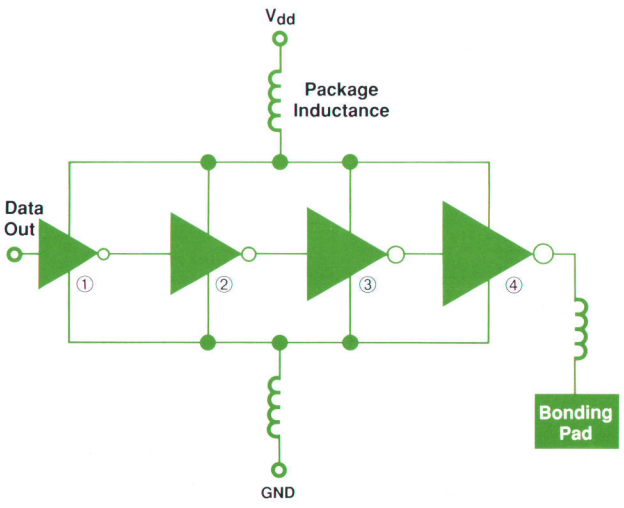


Fig. 6. *Low-noise, high-current pad driver.*

Several design options have been implemented to allow the FPC to be as flexible as possible in accommodating the various vendor parts requirements. A configurable math count register is included in the FPC to accommodate various time requirements for ALU operations, multiplication, division, and square root. This register regulates the number of cycles the FPC waits before unloading a result from the math chips. It is initialized to a default value during power-up and can be reset by system software using a special implementation dependent instruction. This feature had the added benefit of allowing the FPC to be developed concurrently with the floating-point chips without knowing the final number of cycles needed to complete an operation.

High-Productivity Design

A number of factors contributed to high design productivity for the FPC. One is the extensive reuse of circuits first designed for other ICs. The building blocks that went into the state machines, clock generation circuitry, cache bus interface, and parts of the math bus interface were originally designed for chips used in other HP Precision Architecture computers.

Finding a solution for soldering the through-hole parts turned out to be quite difficult. The normal solution of wave soldering could not be used because of concern about damaging the ICs on the bottom of the board. Using a solder pot would give selective soldering, but was undesirable because each component would have to be soldered separately with a special fixture. A single-point applicator could be developed to apply paste one point at a time, but would take considerable hardware and software development and would be difficult to use with parts like PGAs, unless the paste was applied from the bottom side of the board or before the parts were inserted. Hand soldering suffers from extensive labor and poor quality. Solder preforms give fairly good soldering results, but many vendors did not have the capability to provide them and the incremental cost was quite high.

The alternative that gave the best results and caused the least change in the current process was a stencil-reflow process first suggested by AMP.[2] In this process, solder paste is stenciled into the plated-through holes at the same time that paste is stenciled onto the surface mount pads for the top side of the boards. The through-hole components are loaded after the surface mount components and reflowed with either vapor phase or infrared. Critical to forming a good solder fillet is getting enough paste to provide 110% to 140% fill of the hole. This is done by enlarging the holes in the stencil to deposit extra paste on the board surface around the plated-through holes and reducing the plated-through hole sizes to about 0.007 to 0.010 inch larger than the component leads.

The final process is:

- Bottom side: stencil, place surface mount components, and reflow.
- Top side: stencil (for surface mount and through-hole components), place surface mount components, place through-hole components, and reflow.

Reliability testing included standard product environmental and strife tests for the Model 835/Series 935 memory board plus accelerated shock and vibration testing of single-sided and double-sided test boards for comparison. Since most of this testing was done using vapor phase reflow, additional testing was done to compare vapor phase and infrared reflow for solder joint reliability. This was done by subjecting the boards to a predetermined level of destructive random vibration.[3] In these tests, there was no difference between single-sided and double-sided boards, there were no failures in product testing of the 16M-byte memory board on production prototype and pilot run boards, and infrared was statistically better than vapor phase reflow.

Summary

As a result of the dual-reflow process, component packing density has been increased to about twice the single-sided surface mount density, with ICs as large as 44-pin PLCCs being reliably placed on the bottom side. Process and product strife testing has shown no difference in reliability from single-sided surface mount technology. Since the surface mount process has been changed very little from the single-sided process, the processing cost of a double-sided board is less than that of two single-sided assemblies because some of the processes are only done once (e.g., clean, depanel, test, etc.).

Acknowledgments

I would like to extend appreciation and recognize the following individuals for their contributions to the design and implementation of the SMT-2 process: Jim Baker, Mai Chow, Conrad Taysom, and Keith Walden.

Andy Vogen
Project Manager
Surface Mount Development Center

References

1. D.W. Rice, *Double-Sided Surface Attachment: A Guide to What Component Mass Molten Solder Will Support*, Internal HP Paper, March 9, 1987.
2. M. Rupert, "Design Characteristics of a Surface Mount Compatible Through-Hole Connector," *SMTA Expo*, Las Vegas, Nevada, October 16-29, 1987.
3. C. Taysom, *Report on the Results of the IR Reflow Acceptance Testing Using F-16 Double-Sided Boards*, Internal HP Paper, October 14, 1988.

Once the ten principal blocks of the FPC were designed, the 781 signals interconnecting these blocks were routed automatically with almost 100% success. This was accomplished by careful planning of these blocks with constraints on their design to facilitate automatic routing. Part of the routing productivity results from addition of a third level of metallization to the existing NMOS-III process. This coarse interconnect level is reserved for power and clock distribution, saving the finer interconnect levels for signals.

Testing

The FPC is the first HP Precision Architecture coprocessor designed to handle more than one operation at a time. This approach, while increasing performance, also leads to a more complex design and a more difficult testing problem. Unlike the pipeline inside the CPU, which controls when the next operation will arrive, the coprocessor can receive a new floating-point instruction from the CPU at any time. The arrival of this instruction can occur just as the FPC is shifting its internal queue, discovering a trapping condition, or performing some other complex transaction. The number of combinations of conditions is extremely large and verifying correct operation under all conditions is difficult.

To combat this problem in the testing of the FPC, a test program was developed that runs sequences of three floating-point instructions each, with a variable number of non-floating-point instructions inserted between each sequence to create all possible timing conditions. Each test sequence is run once on the hardware, then emulated in software and the results compared. Any differences indicate a failure in the hardware. This test program in its final version runs more than one billion test sequences of three floating-point instructions each.

Processor Dependent Hardware

The Model 835/Series 935 processor dependent hardware (PDH) board is similar to its counterpart in the Model 825/Series 925. It provides the processor dependent hardware interface to the processor board and contains either one or two VLSI channel adapters depending on options the customer has selected. The primary difference is that the board is designed to run at 10 MHz on the CTB, 20% faster than the Model 825/Series 925 board.

The HP CIO channel adapters are the same VLSI design used in other HP Precision Architecture computers except for their higher operating frequency capability. The stan-

dard adapter provides the interface between the CTB and the CIO bus in the SPU. The optional second CIO channel adapter provides an interface to an external CIO expander without consuming a CTB slot in the SPU.

The addition of a second CIO channel adapter chip to the PDH board requires that the two channels share buffers on the CTB side (see Fig. 3). This is possible because the 32 address and data signals from each channel can be directly tied together: only one channel's signals are active at a time. However, the signals that control the direction of the CTB buffers cannot be tied directly together. These signals are NANDed together on the PDH board, electrically separating the control signals from each channel and eliminating the possibility of driver conflict. Pull-up resistors are added to several of the second channel's signals, thus guaranteeing the state of these signals even in the absence of the second channel. This allows the PDH board to be built without the second CIO channel adapter, providing a version of the Model 835/Series 935 with only an internal channel for customers who do not require a CIO expander.

An 8K × 8 static RAM chip is included on the PDH board as nonvolatile memory and is used to store operating system panic and loading information. It is kept alive by the same circuit that provides backup power to the real-time clock. The circuit will provide power from either the powerfail backup system or the SPU clock battery when main power is not available.

16M-Byte Memory

The design challenge for the 16M-byte memory board shown in Fig. 4 was to double the capacity of the already dense memory subsystem in a fixed volume with a minimal increase in power consumption. That challenge has been met by packaging 144 1M-bit DRAM chips, one 272-pin PGA VLSI memory controller, one 100-pin connector, various control and buffering logic chips, and the necessary bypass capacitors on a 6.75-by-7.25-inch printed circuit board (roughly half the size of this page).

To allow interchangeability, the 16M-byte memory board is the same size as the current 8M-byte memory board.[1] Increasing the memory to 16M bytes requires an extra 72 DRAMs and their bypass capacitors on the bottom side of the board. The bottom-side mounting of the DRAMs, which are packaged in 0.300-inch SMT packages, required the development of a new double-sided surface mount manufacturing process (SMT-2) and a new approach to printed circuit board design and verification. For details, see "Double-Sided Surface Mount Process" on page 23.

The circuitry of the 8M-byte memory board released with the Model 825/Series 925 was designed to allow future expansion to 16M bytes with only minor modifications. Slight modifications were made to allow either 8M or 16M bytes to be loaded at the factory using the same board. The 8M-byte version simply omits the bottom-side DRAMs and bypass capacitors. The NMOS-III VLSI memory controller was designed to support 2M, 4M, 8M, or 16M bytes, including single-bit error correction, double-bit error detection, and support for battery backup of the memory contents in case of main power failure.

Although very little new electrical design was required in the 16M-byte memory board, time was spent in verifying operational parameters and increasing the manufacturability of the board. It was designed with all the test points on the bottom side to be electrically tested in an HP 3065 single-sided probe fixture in manufacturing. This design allows a less expensive and more reliable test fixture and the use of the same fixture for both the 8M-byte and 16M-byte versions.

Since the 16M-byte board was the first design to use HP's new SMT-2 process, the printed circuit board development software was modified to evaluate and verify the design of the board before sending data out for board fabrication. The flexibility of HP EGS allowed easy addition of the necessary features to route circuitry to bottom-side components and verify SMT-2 design rule compliance.

Acknowledgments

The Model 835/Series 935 project benefited by having a highly motivated team of individuals involved in all aspects of design, manufacturing, test, and marketing. Our thanks go to everyone who was involved. Particular thanks go to Denny Georg, R&D lab manager, and Russ Sparks, R&D section manager, for their complete support throughout the project. Thanks also go to Dan Osecky, Glenda McCall, Lisa Quatman, Jerry Schell, and Bob Headrick for their management help.

The authors also wish to recognize the efforts of Ann Kloster, Charlie Shilling, Lynne Christofanelli, Alisa Scherer, and Jim Murphy for hardware design, Paul French, Willy McAllister, Pirooz Emad, Dave Goldberg, Gene Mar, Kevin Morishige, Greg Averill, Annika Bohl, Balbir Pahal, Ram Ramjadi, and Valerie Wilson for FPC design, Mark Hodapp, Rose Maple, Jeff Kehoe, and Ed Ogle for firmware and diagnostics, Mark Stolz, Rick Butler, Jeff Rearick, and Asad Aziz for chip and board test development, and Charlie Kohlhardt and Kevin Burkhart for providing 30-MHz VLSI. Thanks also go to Andy Vogen, Conrad Taysom, Keith Walden, and Madi Bowen for their help with SMT design and to Spencer Ure, George Winski, Don Soltis, Arlen Roesner, Gary Stringham, Steve Hanzlik, Marcia Franz, and Karyn Jarvis for handling other manufacturing issues.

References

1. C.S. Robinson, et al, "A Midrange VLSI Hewlett-Packard Precision Architecture Computer," *Hewlett-Packard Journal*, Vol. 38, no. 9, September 1987. pp. 26-34.
2. S.T. Mangelsdorf, et al, "A VLSI Processor for HP Precision Architecture," *ibid*, pp. 4-11.
3. J.D. Yetter, et al, "HP Precision Architecture NMOS-III Single-Chip CPU," *ibid*, pp. 12-18.
4. J.D. Yetter, et al, "A 15-MIPS 32-Bit Microprocessor," *Digest of Technical Papers, International Solid-State Circuits Conference*, February 1987.

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.

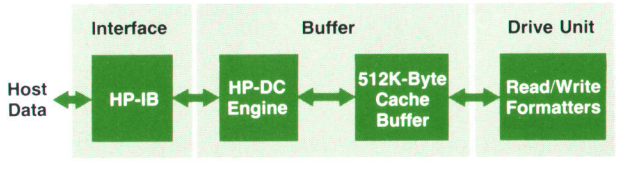


Fig. 1. *HP 7980XC data path architecture.*

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

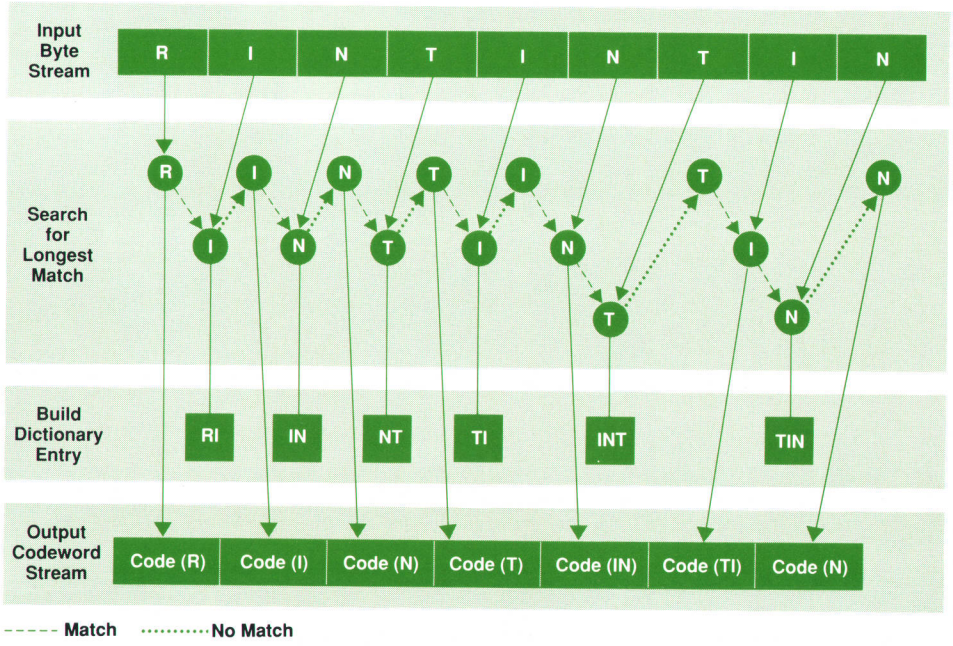


Fig. 2. *Compression algorithm example.*

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

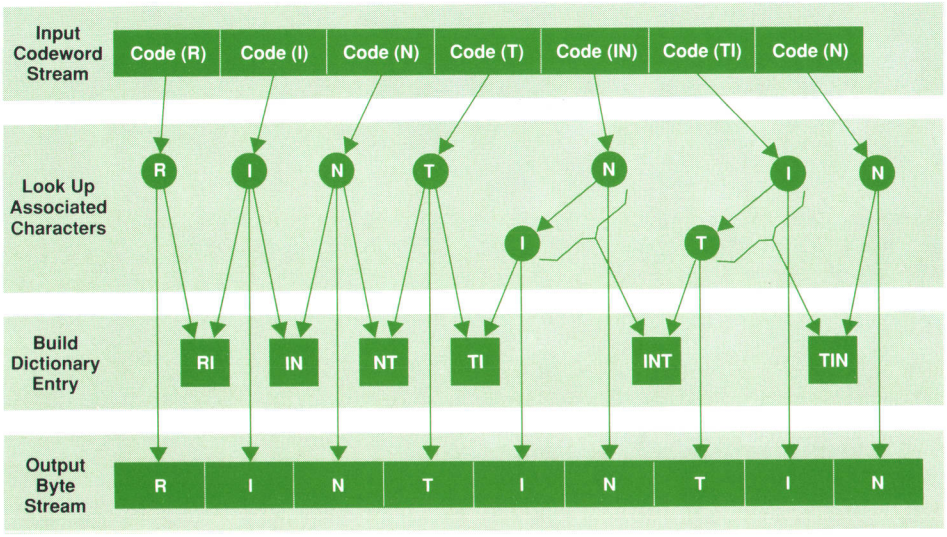


Fig. 3. *Decompression algorithm example.*

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

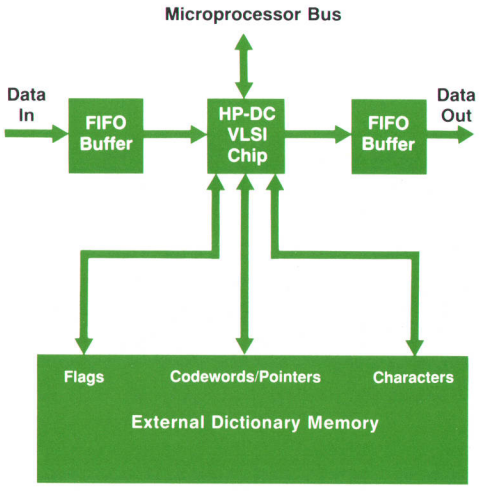


Fig. 4. *HP-DC engine block diagram.*

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

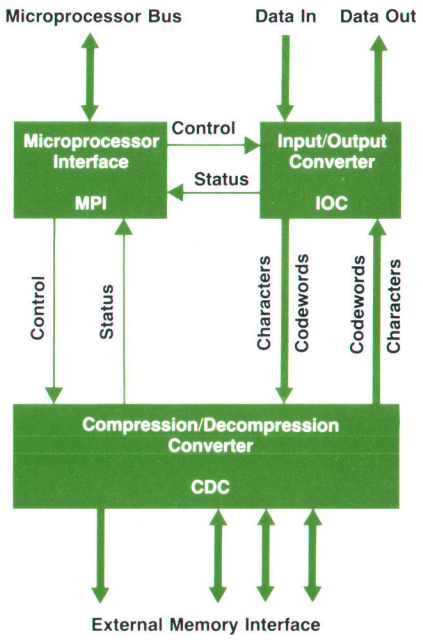


Fig. 5. *HP-DC chip block diagram.*

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-

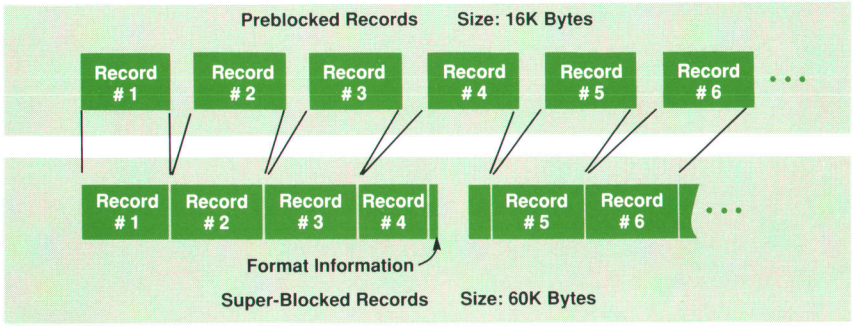


Fig. 1. *The super-blocking process combines normal records into super-blocks as large as 60K bytes.*

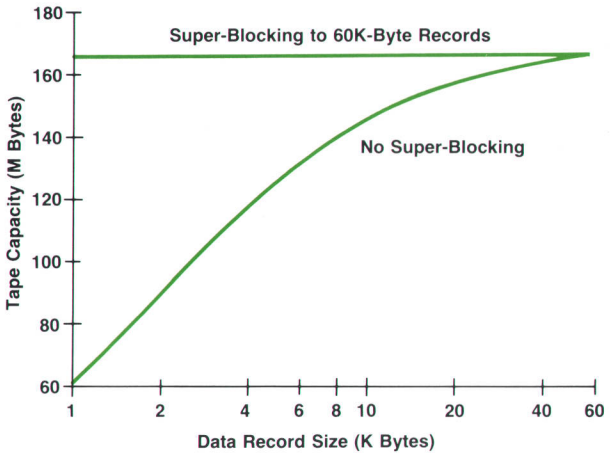


Fig. 2. *Tape capacity versus data record size for a 2400-foot tape written at 6250 bpi with 0.3-inch gaps.*

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

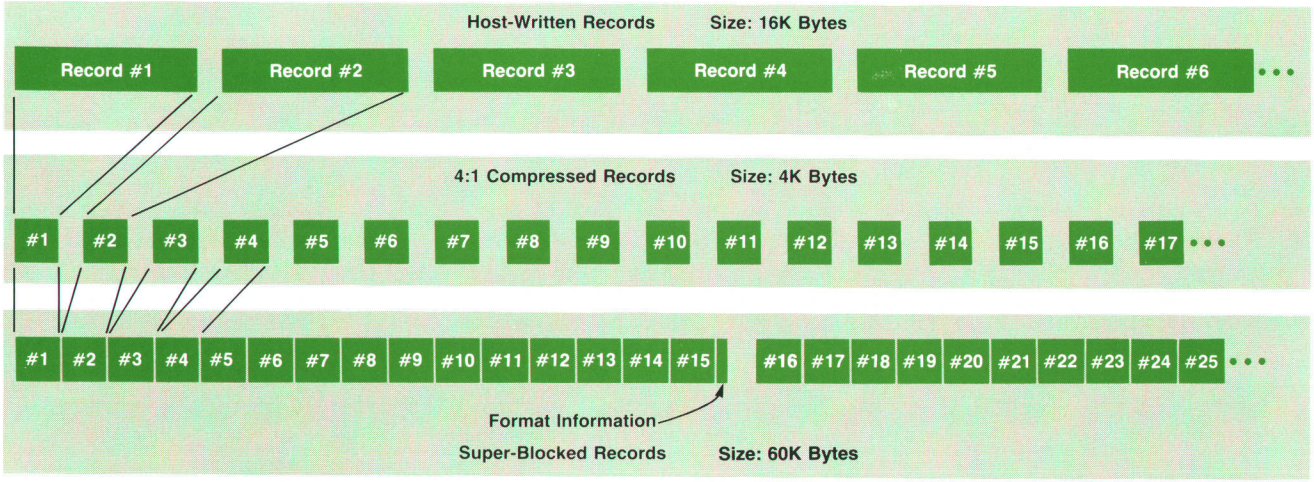


Fig. 3. *Data compression combined with super-blocking.*

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.

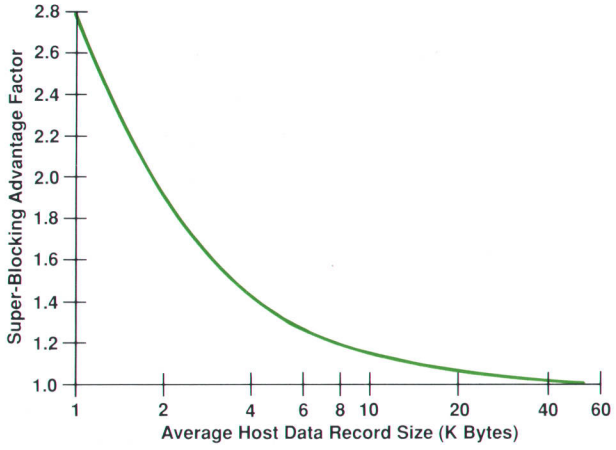


Fig. 4. *Super-blocking advantage factor (SAF) versus data record size for a tape written at 6250 bpi with 0.3-inch gaps.*

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

High-Speed Lightwave Component Analysis

A new analyzer system performs stimulus-response testing of electrical-to-optical, optical-to-electrical, optical-to-optical, and electrical-to-electrical components of high-speed fiber optic communications systems.

by Roger W. Wong, Paul Hernday, Michael G. Hart, and Geraldine A. Conrad

HIGH-SPEED FIBER OPTIC COMMUNICATIONS systems have emerged over the last half decade to compete with other forms of communications systems as a cost-effective means of moving information. A decade ago, the possibility of a commercially installed 500-Mbit/s fiber optic system seemed remote. Today, not only are many fiber optic systems operating at hundreds of megabits per second, but pilot systems are being installed that operate at 1.7 to 2.4 gigabits per second. The trend toward higher system bit rates places more demand upon the lightwave component designer to optimize the performance of each device within the high-speed lightwave system.

Lightwave System Challenges

Fig. 1 shows the typical functional blocks in a fiber optic communication system. The high-speed portions of the lightwave system are the preamplifier, the directly modulated laser, the optical fiber, the photodiode receiver, and the postamplifier. As systems transmit higher bit rates, each of the components needs to be designed to meet the higher speed requirements. However, with the higher speeds, optimization of signal transmission through the various devices becomes more challenging and the interactions of various components become more evident and difficult to minimize.

Fig. 2 shows some of the typical challenges the high-speed component designer encounters as systems move to gigabit-per-second transmission rates. As in lower-bit-rate systems, optical power budgets are affected by the insertion loss of the optical fiber, connectors, and splices. In the higher-bit-rate systems (>500 Mbits/s), interactions between the high-speed devices are significant. Often more extensive analysis and device characterization are required to optimize the electrical and optical interfaces between these high-speed components in a systematic way.

For example, electrical mismatches between the laser and its preamplifier or between the photodiode and its postamplifier can affect the modulation transfer function and the cumulative modulation bandwidth. Also, light reflected back into the laser source affects its modulation transfer characteristics and the system signal-to-noise ratio.

Lightwave Component Analyzer Systems

Fig. 3 shows the key instruments that form the HP 8702A Lightwave Component Analyzer measurement systems. Three basic systems are offered:

- Modulation capability to 6 GHz at 1300 nm
- Modulation capability to 3 GHz at 1300 nm (high dynamic range)
- Modulation capability to 3 GHz at 1550 nm (high dynamic range).

Each HP 8702A system consists of a lightwave source, a lightwave receiver, the lightwave component analyzer, and a lightwave coupler. Fig. 4 shows the HP 83400 family of lightwave sources and receivers, which are important elements of the measurement systems. More information on these sources and receivers can be found in the article on page 52.

The system measures the modulation transfer function of a device under test and provides the modulation amplitude and phase response of that device. The input or stimulus signal can either be a radio frequency (RF) signal or a modulated optical signal, and the output or response signal can either be an RF signal or a modulated optical signal. Thus, the device under test (DUT) can be an electrical-to-electrical, electrical-to-optical, optical-to-electrical, or opti-

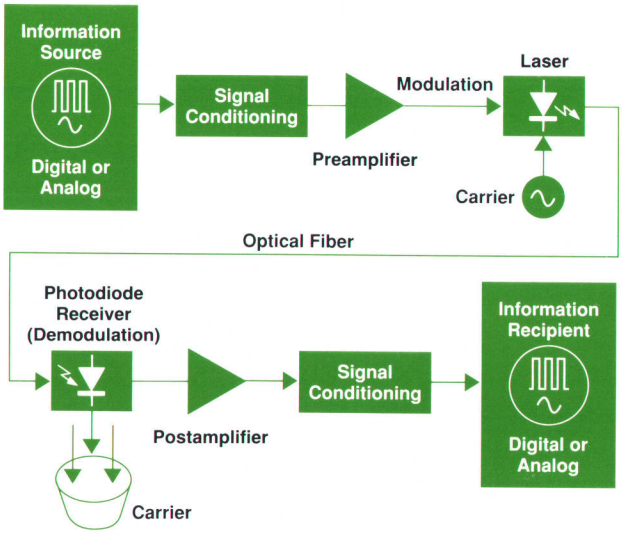


Fig. 1. *Functional blocks in a typical fiber optic communications system.*

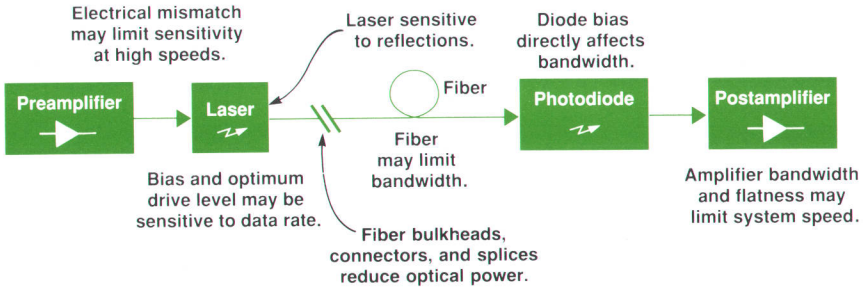


Fig. 2. *Effects of component characteristics on system performance.*

cal-to-optical device, depending upon the measurement block diagram employed and the calibration procedure used. Table I shows typical examples of each device type. By adding an optical signal separation device, such as a lightwave directional coupler, the system can be configured to measure optical reflections in a wide variety of optical devices, such as optical fiber components, connectors, antireflection coatings, bulk optic devices, and so on. Moreover, if there are multiple reflections in a device, each reflection can be located very accurately. Multiple reflections can be resolved when they are only centimeters apart.

In this article, lightwave component analysis means the capability to characterize a given device in terms of its modulation transfer function, electrical driving-point impedance, optical return loss, and length, as appropriate to the device type.

Table I
Types of Lightwave Devices

	Electrical (RF) Output	Optical (Modulated) Output
Electrical (RF) Input	**Electrical-to-Electrical Devices** ■ Amplifiers ■ Coaxial Cables and Passive Components ■ Repeater Links	**Electrical-to-Optical Devices** ■ Laser Diodes and LEDs ■ Optical Sources ■ Optical Modulators
Optical (Modulated) Input	**Optical-to-Electrical Devices** ■ PIN Photodiodes ■ Avalanche Photodiodes ■ Optical Receivers	**Optical-to-Optical Devices** ■ Optical Fibers, Passive Components, Attenuators ■ Optical Modulators ■ Regenerators

Lightwave Component Analyzer Operation

The hardware design of the HP 8702A Lightwave Component Analyzer is virtually identical to that of the HP 8753B RF Network Analyzer. However, the HP 8702A has operating features that make it more appropriate for lightwave measurements.

The HP 8702A consists of three main subsystems that tie together the lightwave measurement system: an RF source, RF receivers, and processing/display (see Fig. 5). The lightwave measurement system is analogous to a lightwave communication system. The HP 8702A performs the functions of an information source and an information receiver. The data processing subsystem uses this information to measure the modulation transfer characteristics of lightwave components.

Signals used to modulate a lightwave source are produced by a synthesized RF source in the HP 8702A. The RF source provides linear, logarithmic, and list frequency

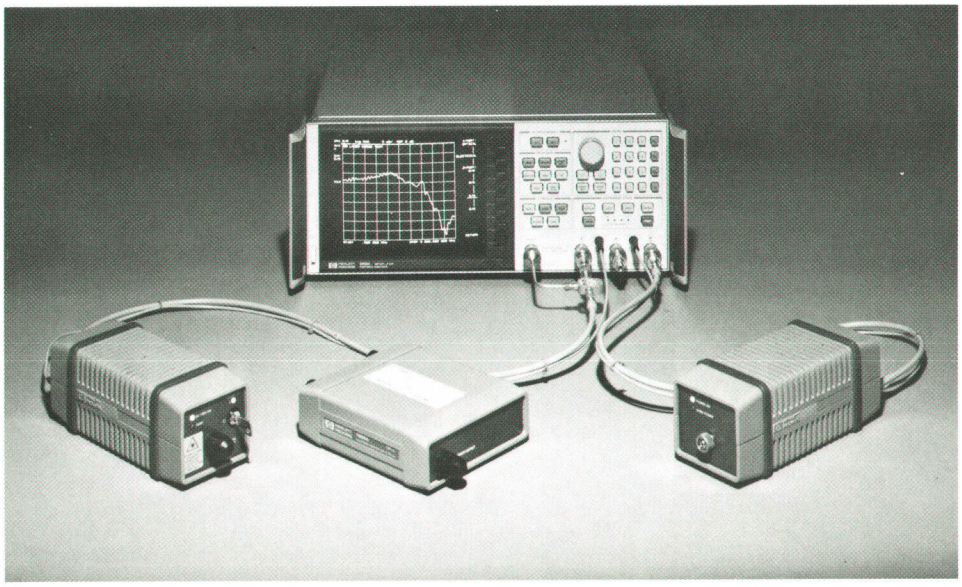


Fig. 3. *HP 8702A Lightwave Component Analyzer systems consist of a lightwave source, a lightwave receiver, the analyzer, and a lightwave coupler. Three basic systems have modulation capability to 6 GHz at 1300 nm or to 3 GHz at 1300 or 1550 nm.*

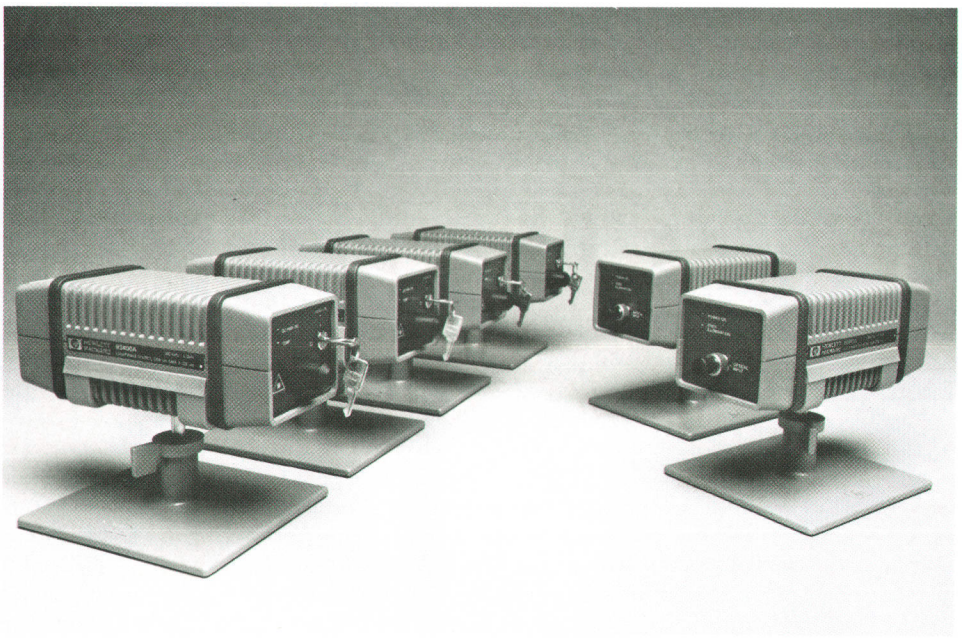


Fig. 4. *The HP 83400 family of lightwave sources and receivers.*

sweeps from 300 kHz to 3 GHz with 1-Hz resolution. Power and CW sweeps may also be generated. The source is phase locked to the R receiver channel, which is described below. The HP 8702A provides the power supply for lightwave sources and receivers.

Demodulated signals from a lightwave receiver are measured by three 300-kHz-to-3-GHz, tuned RF receivers in the HP 8702A. The receivers' bandwidths are extended to 6 GHz with Option 006. Meaurements of electrical devices have a dynamic range of over 100 dB. A portion of the R receiver signal is used to phase lock the source to the reference channel. Input signals are sampled and down-converted to a 4-kHz IF. The 4-kHz IF signals for the A, B, and R inputs are converted into digital words by the analog-to-digital converter (ADC).

The data processing flow from the ADC to the display is shown in Fig. 6. The digital filter performs a discrete Fourier transform (DFT) on the digital words. The samples are converted into complex number pairs. The DFT filter shape can be altered by changing the IF bandwidth. Decreasing the IF bandwidth is an effective technique for noise reduction. A reduction of the IF bandwidth by a factor of ten lowers the measurement noise floor by appoximately 10 dB.

Ratio calculations are performed next, if the selected measurement is the ratio of two inputs, that is, A/R, B/R, or A/B. The ratio is formed by a simple division operation.

The sampler/IF correction operation is applied next. This process digitally corrects for frequency response error, primarily sampler roll-off, in the analog down-conversion

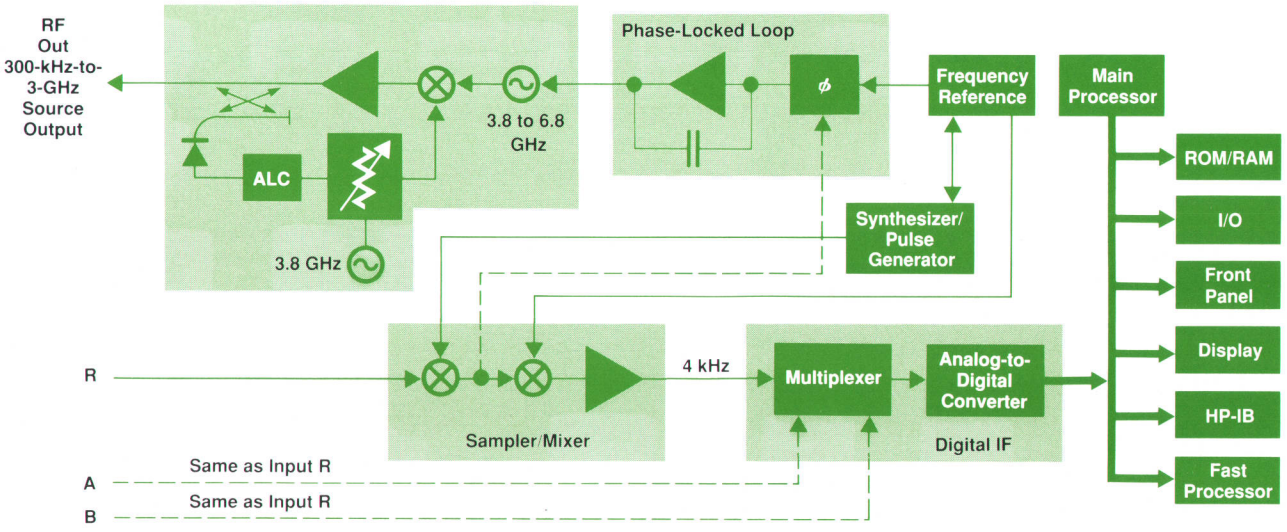


Fig. 5. *HP 8702A Lightwave Component Analyzer block diagram. The hardware is essentially the same as the HP 8753B RF Network Analyzer.*

path.

Sweep-to-sweep averaging is another noise reduction technique. This involves taking the complex exponential average of several consecutive sweeps weighted by a user-specified averaging factor. Each new sweep is averaged with the previous result until the number of sweeps equals the averaging factor. Doubling the averaging factor reduces the noise by 3 dB. This technique can only be used with ratio measurements.

The raw data arrays store the results of all of the preceding data processing operations. All processing up to this point is performed in real time by the fast digital signal processor shown in Fig. 5. The remaining operations are performed asynchronously by the main processor. These arrays can be stored to an external disc drive and can be accessed directly via the HP-IB (IEEE 488, IEC 625).

Vector error correction is performed next, if a measurment calibration has been performed and correction is turned on. Error correction removes repeatable systematic errors (stored in the error coefficient arrays) from the raw arrays. This can vary from simple vector normalization to full two-port (12-term) error correction. Correction for the various types of lightwave measurements is described in more detail below.

The results of error correction are stored in the data arrays as complex number pairs. The data arrays can be stored to disc and accessed via the HP-IB. If the data-to-memory operation is performed, the data arrays are copied into the memory arrays. The memory array is also externally accessible.

The trace math operation selects either the data array, the memory array, or both to continue flowing through the data processing path. In addition, the complex ratio of the two (data/memory) or the difference (data − memory) can also be selected. If memory is displayed, the data from the memory arrays goes through the same data processing flow path as the data from the data arrays.

Gating is a digital filtering operation associated with time-domain transform (Option 010). Its purpose is to remove unwanted responses isolated in time. In the time domain, this can be viewed as a time-selective bandpass or band-stop filter.

The delay block involves adding or subtracting phase in proportion to frequency. This is equivalent to extending or shortening the electrical length in the measurement path or artificially moving the reference plane.

Conversion, if selected, transforms the measured s-parameter data to the equivalent complex impedance or admittance values, or to inverse s-parameters.

The transform operation converts frequency-domain information into the time domain when time-domain transform is enabled (Option 010 only). The results resemble time-domain reflectometry (TDR) or impulse-response measurements. The transform employs the chirp-Z inverse Fourier transform algorithm. Windowing is a digital filtering operation that prepares the frequency domain data for transform to the time domain. The windowing operation is performed on the frequency-domain data just before the transform.

Formatting converts the complex number pairs into a scalar representation for display, according to the selected format. Formats include log magnitude in dB, phase, and group delay. Polar and Smith chart formats retain complex data for display on real and imaginary axes.

Smoothing is another noise reduction technique. When smoothing is on, each data point in a sweep is replaced by the moving average value of several adjacent points. The number of points included depends on the smoothing aperture, which is selected by the user. The result is similar to video filtering.

The results at this point in the data processing chain are stored in the format arrays. Marker values, marker func-

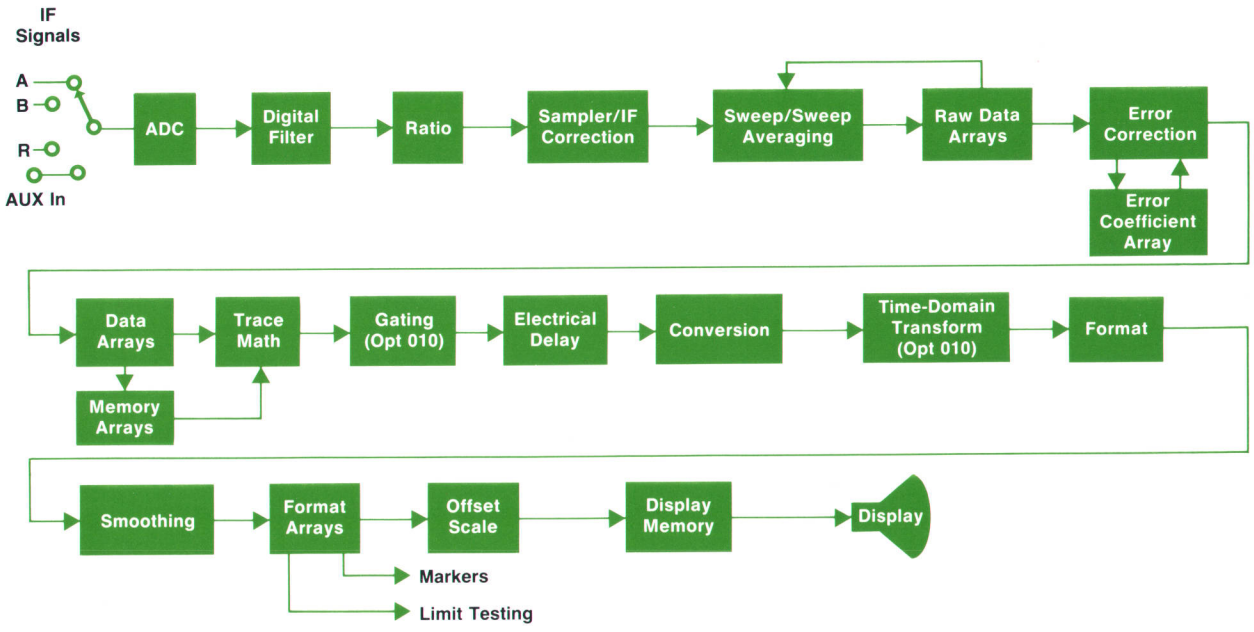


Fig. 6. *HP 8702A Lightwave Component Analyzer data processing flow diagram.*

tions, and limit testing are all derived from the format arrays. The format arrays can be stored to an external disc drive and can be accessed via the HP-IB.

The offset and scale operations prepare the formatted data for display on the CRT. This is where the reference line position, reference line value, and scale calculations are performed as appropriate to the format and graticule type.

The display memory stores the display image for presentation on the CRT. The information here includes graticules, annotation, and softkey labels in a form similar to plotter commands. When hard-copy records are made, the information sent to the plotter or printer is taken from display memory.

The HP 8702A can be connected with an s-parameter test set (HP 85046A) to make electrical reflection measurements, such as s-parameters s_{11} and s_{22} (return loss and impedance). An HP 85047A S-Parameter Test Set can be used to measure modulation transfer function to 6 GHz.

Firmware Features

The principal contributions of the HP 8702A are its firmware enhancements. The firmware was developed using the HP 8753A RF Network Analyzer as a platform. The HP 8702A firmware contains the features of the HP 8753A as well as new features specific to lightwave measurements. The most significant enhancement is the ability to perform measurement calibration of lightwave components.

The measurement calibration process consists of measuring a characterized standard and using it to measure an unknown device. The firmware contains a mathematical model of the calibration standard and the model's parameters. Data from measurement of the standard is used with the calibration models to remove systematic errors from measurements of the test device. Lightwave measurements are also scaled to proper units for the particular component type.

Calibration of optical devices is performed using through connections and known reflections as standards. Calibration is done for transmission measurements by connecting the lightwave source to the lightwave receiver with the test device removed. Reflection measurements require a known reflection as a calibration standard. For example, the Fresnel reflection occurring at the test port connector of a lightwave coupler is a repeatable and convenient reflection standard (3.5%).

Calibrated lightwave receivers and calibrated lightwave sources are used as standards for electrooptical and optoelectrical test device measurements. The calibration process is the same for both types of devices. Calibration information is provided in two forms. The first form is a digitized modulation frequency response of the standard. This information is read by the analyzer from a disc provided with each calibrated source and receiver. The second is a curve fit of the frequency response. Coefficients are entered by the user into the analyzer using values printed on each lightwave source and receiver.

Calibration of electrical devices is the same as in most HP network analyzers. Calibration kits containing standard devices are available for several different connector types. Typical standards include shorts, opens, and loads.

Time-domain transform, an optional feature of the HP 8702A, is an extremely powerful tool in lightwave measurements. Data measured in the frequency domain is converted to the time domain using a chirp Fourier transformation technique. The resulting time scale is extremely accurate and stable because of the synthesized frequency sweep. Measurements of distance can be derived from transmission or reflection measurements using the index of refraction or velocity factor of the test device. The HP 8702A has an enhancement to assist in setting transform parameters. The distance range and resolution of the transformed data depend on the width of the frequency-domain sweep and the number of data points. The transform parameters feature assists the user by displaying range and resolution values as sweep parameters are set (Fig. 7).

Measurement Concept

The lightwave component analyzer measurement concept is shown in Fig. 8. The information source provides a sine wave whose amplitude and phase characteristics are known. This signal serves as the modulation signal to the lightwave source (transmitter). The output signal of the transmitter is an intensity modulated optical carrier at a fixed wavelength. The intensity modulation (i.e., amplitude modulation) envelope of the lightwave signal is proportional to the radio frequency sine wave information signal. Because the laser lightwave source is dc-biased in the linear region of its optical-power-versus-input-current characteristic, the average optical power from the lightwave source is the same whether or not a modulation signal is present.

The intensity modulated signal from the lightwave source is transmitted through the optical medium, most commonly optical fiber, although it could be an open-beam environment. The lightwave receiver demodulates the intensity modulated lightwave signal and recovers the sinusoidal RF envelope, which is proportional to the sine wave from the information source. The demodulated signal is compared in magnitude and phase to the original signal by the HP 8702A analyzer.

The optical signal incident upon an optical device under test is of the form:

TRANSFORM PARAMETER	Channel 1
RANGE	13.6905 m
RESPONSE RESOLUTION	133.79 mm
TRANSFORM SPAN	40 ns
RANGE RESOLUTION	41.067 mm
TRANSFORM MODE	BANDPASS
START FREQUENCY	300 kHz
STOP FREQUENCY	3 GHz
FREQUENCY SPAN	2.9997 GHz
NUMBER of POINTS	201
INDEX of REFRACTION	1.46
PULSE WIDTH	651.55 ps
SOURCE POWER	0 dBm
SWEEP TIME	800 ms
IF BANDWIDTH	3000 Hz

Fig. 7. *The transform parameter display helps the user set up the instrument for the optional time-domain transform.*

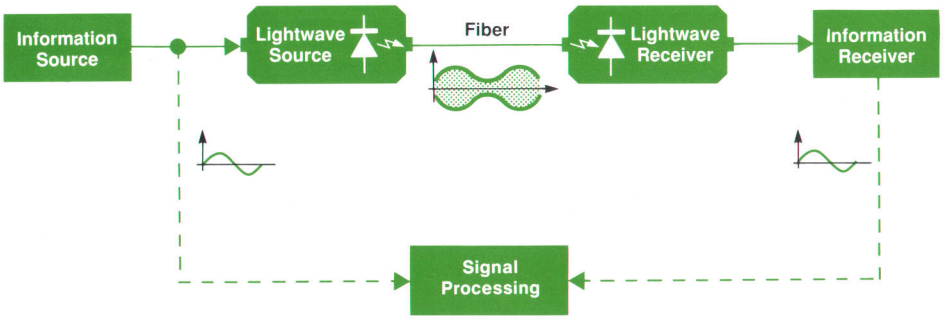


Fig. 8. *Measurement concept. The system compares transmitted and received sine wave modulation superimposed on the 1300-nm or 1550-nm optical carrier.*

$$f(t) = a(t)\cos(\omega t)$$

where a(t) is the RF modulation signal and cos(ωt) represents the lightwave carrier signal at a given wavelength.

The device under test operates on the amplitude of both the modulation envelope and the carrier signal identically and delays both signals by identical amounts, yielding the following relationship for the DUT output:

$$g(t) = |H|a(t + \Delta t)\cos(\omega(t + \Delta t)),$$

where |H| is the magnitude of the transfer function of the DUT, $\Delta t = \phi/\omega$, and ϕ is the phase of H.

The impact of the DUT on the carrier can be determined by measuring the modulation envelope. Basically, the measurement process consists of two steps: (1) calibration of the system, and (2) measurement of the DUT. This measurement process is essentially a substitution method. The system is calibrated by measuring a known quantity and then the DUT is substituted for the known device and measured.

Electrooptical Calibration Theory

Two important electrooptical devices are lasers and photodiodes. Measurements of their modulation transfer characteristics and modulation bandwidths are of primary interest for the design of high-speed lightwave communications systems. The development of electrooptical calibration routines for measuring electrooptical devices such as lasers and photodiodes was a significant challenge.

Fig. 9 shows the relationship between optical power and RF current for typical electrooptical devices, such as lasers, optical modulators, and photodiodes. The slopes of the curves at points (I_a,P_a) and (P_b,I_b) define the slope responsivities of the electrical-to-optical and optical-to-electrical devices, r_s and r_r, respectively, as shown.

For electrical-to-optical devices:

$$\Delta P_o = r_s \Delta I_1 \quad (1)$$

where ΔP_o is the peak-to-peak optical power swing, r_s is the slope responsivity of the electrical-to-optical device in W/A, and ΔI_1 is the peak-to-peak RF current swing.

For optical-to-electrical devices:

$$\Delta I_2 = r_r \Delta P_o \quad (2)$$

where ΔI_2 is the output peak-to-peak RF current swing, r_r is the slope responsivity of the optical-to-electrical device in A/W, and P_o is the peak-to-peak optical power swing.

The relationship between the device slope responsivities and RF current gain can be derived from equations 1 and 2:

$$\Delta I_2/\Delta I_1 = r_s r_r. \quad (3)$$

Equation 3 forms the basis for the electrooptical calibrations and allows the measurement of an electrical-to-optical device separately from an optical-to-electrical device, which is one of the contributions of the measurement system.

For each HP lightwave source or receiver, a calibration data disc is provided, which contains the device's slope responsivity, modulation amplitude frequency response,

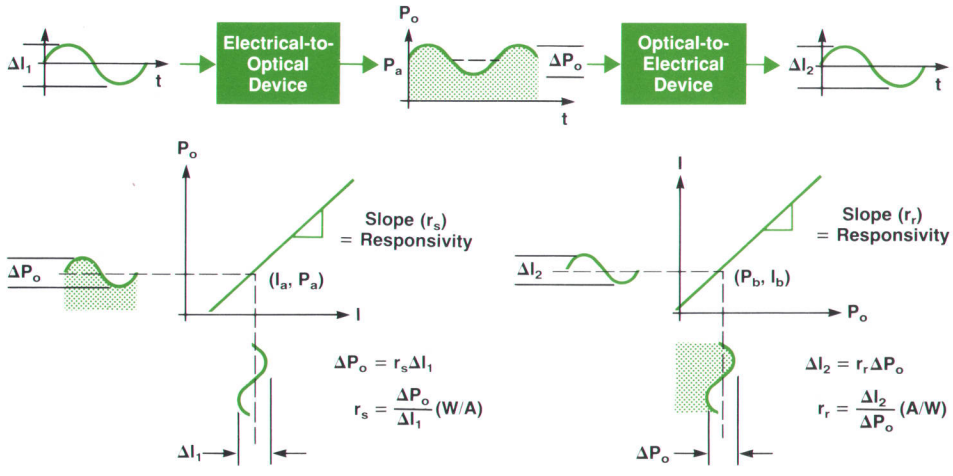


Fig. 9. *Relationship between optical power and RF current for typical electrooptical devices.*

and modulation phase frequency response. This data disc can be downloaded into the analyzer as part of the electrooptical calibration procedure.[1] The calibration data is traceable to an internal HP optical heterodyne system called the Optical Heterodyne Calibration System.

Laser Bandwidth and Power Compression Measurements

Fig. 10 shows the measurement block diagram for lasers and other electrical-to-optical devices. In this configuration, laser diode and/or laser transmitter characteristics such as responsivity, modulation bandwidth, modulation phase or deviation from linear phase, and modulation power compression can be measured.

A commercially available 1-Gbit/s lightwave transmitter is used as an example. The laser was dc-biased in the linear range of its optical-power-versus-input-current curve (about 15 mA above its threshold) and modulated with an incident RF power of approximately +10 dBm. The measured laser responsivity (0.34W/A or −9.35 dB) and modulation bandwidth (about 600 MHz) are shown in the top trace in Fig. 11.

The built-in inverse Fourier transform feature of the HP 8702A allows modulation frequency response data to be converted to the equivalent step or impulse response. For the above example, where the laser was operating in the linear region, an equivalent step response with rise time information can be calculated and displayed, as shown in the bottom trace in Fig. 11. Notice that the transmitter's frequency response is peaked by about 2.5 dB at 428 MHz. This accounts for the underdamped time-domain step response.

To illustrate the modulation power compression measurement, a commerically available electrical-to-optical converter with an internal laser preamplifier was selected. The same block diagram as shown in Fig. 10 was used. The analyzer has the ability to change the RF signal power to

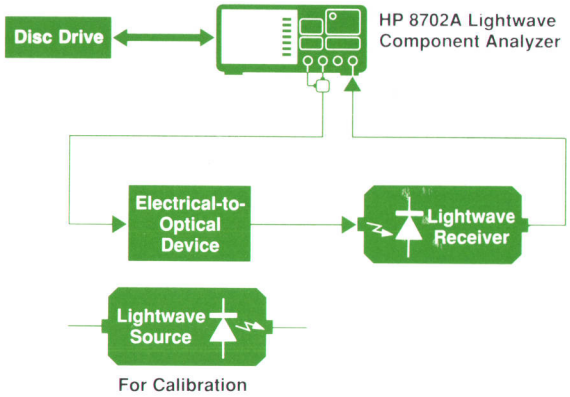


Fig. 10. *Measurement block diagram for lasers and other electrical-to-optical devices.*

the DUT over a 25-dB range at a fixed modulation frequency. New calibration routines were developed that allow the input RF power and modulated optical power measurement planes to be referenced at the connectors of the device under test. Fig. 12 shows two measurements. The top trace shows transmitter modulated optical power out as a function of RF power into the transmitter. The bottom trace shows transmitter responsivity as a function of RF power incident to the transmitter with the modulation frequency fixed at 400 MHz. The top curve shows that the transmitter has a compressed modulated optical power of −1.44 dBm (or 0.72 mW peak to peak) with an incident RF power of 1.8 dBm. The bottom curve shows the transmitter responsivity linearity and compression characteristics. The small-signal responsivity is 0.38W/A (or −8.3 dB) and compresses by 1 dB at −4.4 dBm incident RF power.

Laser Reflection Sensitivity Measurements

Most high-speed lasers are sensitive to back-reflected

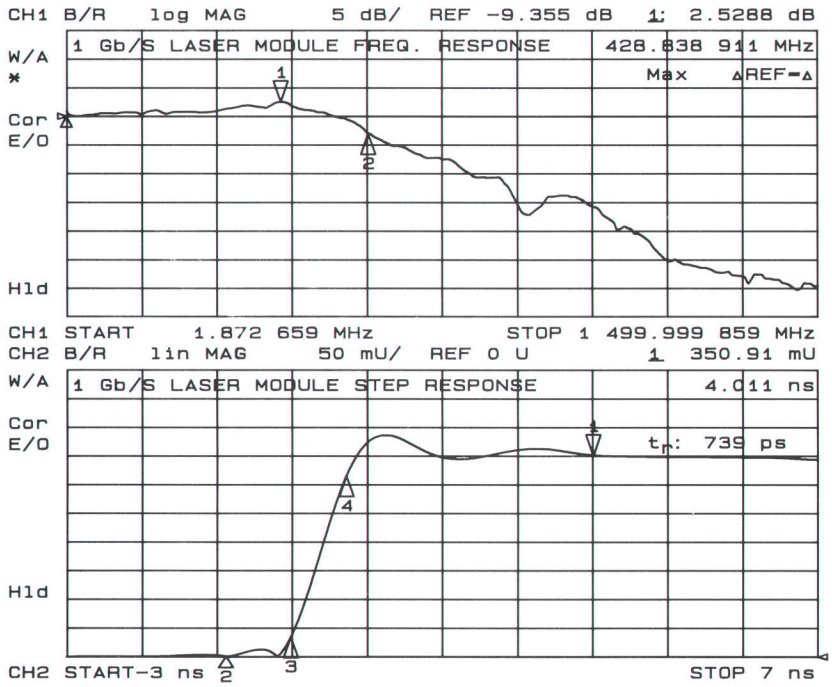


Fig. 11. *(top) Responsivity versus modulation frequency for a 1-Gbit/s laser module. (bottom) Calculated equivalent step response of the laser module.*

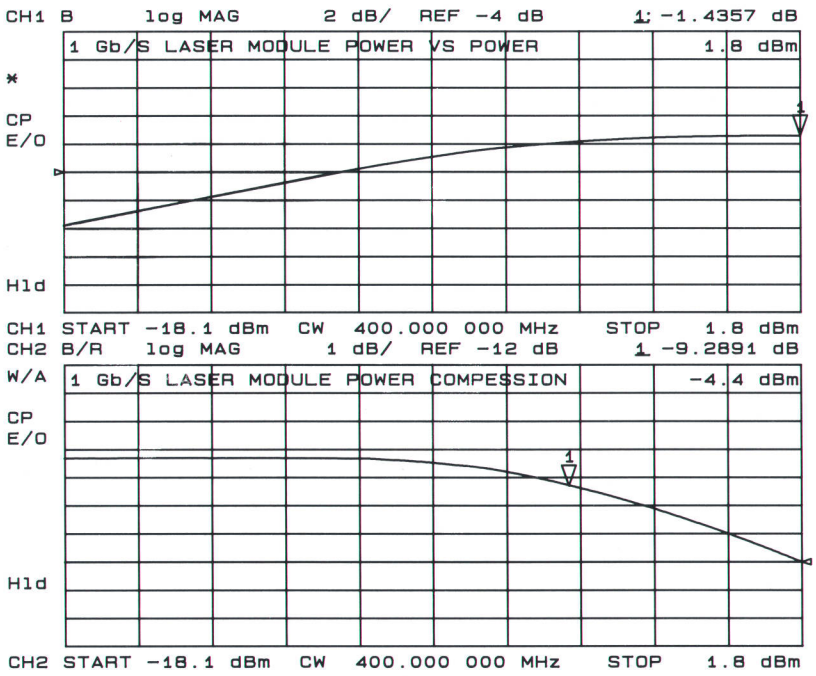


Fig. 12. *Measurements on an electrical-to-optical converter with an internal laser preamplifier. (top) Change in optical power output as a function of RF input power. (bottom) Converter responsivity versus RF input power.*

light. The reflected light can couple back into the laser's cavity, be reamplified, and change the laser's modulation transfer characteristics. The fine-grain ripple that often results is called reflection noise. Fig. 13 shows the measurement setup to characterize the change in the laser's modulation transfer function and bandwidth when different levels of light are intentionally reflected back into the laser.

The directional coupler test port (point A) is where the reflected light condition is developed. The modulation frequency response is referenced to a condition in which no incident light at point A is reflected back toward the laser under test, that is, an optical load is created at point A. This reference condition is normalized to 0 dB. When the reflection condition is changed, the resulting measurement shows the deviation or change of the laser's modulation response for that reflection condition (a laser reflection sensitivity measurement). An example of such a measurement of a commercially available laser transmitter is shown in Fig. 14. The worst response represents the condition when approximately 95% of the light was reflected back to the laser under test. The improved response was achieved when a polarization controller was inserted between the test port and the 95% optical reflector and the polarization of the reflected light was adjusted to minimize the response roll-off.

Photodiode Measurements

Measurements characterizing optical-to-electrical devices, such as photodiodes and lightwave receivers, are similiar to laser measurements. The measurement block diagram is shown in Fig. 15.

Two-Port Optical Device Measurements

The loss, gain, and modulation bandwidth of any two-port optical device can be measured using the measurement block diagram shown in Fig. 16. Examples of such devices are optical connectors, attenuators, other passive optical devices, modulators, and optical regenerators. In this measurement, the input stimulus and output response signals are intensity modulated light signals. The device under test can be a single component or an optical subsystem such as an interferometer or sensor.

If an optical attenuator is selected as the device under test, not only can the attenuator loss be measured, but also the effective measurement system dynamic range can be determined for optical transmission measurements. Fig. 17 shows such a measurement. This particular system displays more than 50 dB of optical dynamic range.

Optical Reflection Measurements

The measurement of optical reflections and the identification of their locations are becoming more important in gigabit-rate lightwave systems, subsystems, optical sensors,

(continued on page 44)

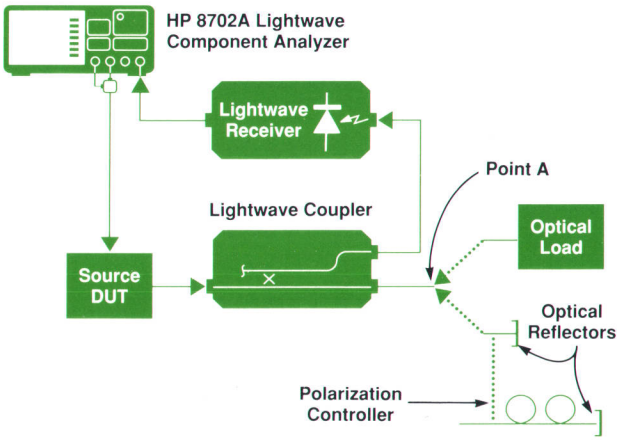


Fig. 13. *Measurement block diagram for laser reflection sensitivity measurements.*

OTDR versus OFDR

The HP 8702A Lightwave Component Analyzer system can display a test device's transmission and reflection characteristics in the modulation frequency domain or the time (distance) domain. Because it shows reflections in the time domain, a comparison of its capabilities to those of an optical time-domain reflectometer is often requested.

An optical time-domain reflectometer (OTDR) like the HP 8145A[1] measures reflections and losses in optical fibers and other devices by sending a probe pulse of optical energy into the device and measuring the reflected and backscattered energy. The HP 8702A Lightwave Component Analyzer described in the accompanying article makes optical reflection measurements differently, that is, by transforming reflected frequency-domain data mathematically into the time domain. Hence it can be thought of as an optical frequency-domain reflectometer (OFDR). Both measurement systems measure optical reflections and lengths and have some overlapping and complementing capabilities, but in general, they are designed for different application areas and therefore have significant differences. The OTDR is primarily a fiber installation or maintenance tool used for installing fiber, checking for faults, and measuring splice loss. The HP 8702A OFDR technique is a lab bench tool used for component and device characterization or location of reflections.

The table below summarizes the principal differences between the OTDR (HP 8145A Optical Time-Domain Reflectometer) and the OFDR (as implemented in the HP 8702A Lightwave Component Analyzer):

	OFDR Mode (HP 8702A)	OTDR (HP 8145A)
Reflection measurement (one-port measurement)	Yes	Yes
Measures loss versus distance (dB/km) (backscatter)	No	Yes
Measures splice loss and breaks in fiber	No	Yes
Measures magnitudes and positions of optical reflections	Yes	Yes
Measures optical return loss of reflections	Read directly	Derivable
Distance range (4% Fresnel reflection)	About 40 km	Greater than 100 km
Dead zone	None	Tens of meters (1)
Single-event resolution (in optical fiber)	<2 mm (2)	Meters
Two-event resolution (in optical fiber)	3.4 cm (3) 1.7 cm (4)	Tens of meters
Gates out unwanted resonses	Yes	No

(1) Dead zone depends upon pulse width used in the measurement.
(2) Assumes that the index of refraction is known accurately and does not limit the measurement accuracy.
(3) Theoretical limit. Assumes a 3-GHz frequency span. 6 cm observed empirically in a nonoptimized experiment.
(4) Theoretical limit. Assumes a 6-GHz frequency span. 2.5 cm observed empirically in a nonoptimized experiment.

Since the OTDR measures backscatter, it can locate and measure discontinuities that do not produce a reflected signal. A sudden drop in backscatter level clearly shows a fiber break. The HP 8702A OFDR technique is not suited to these applications, which are generally required for fiber installation.

Conversely, in designing and manufacturing small components, connectors, or fibers, the excellent resolution and stability of the HP 8702A OFDR technique make it the best method for determining the exact locations of closely spaced reflections. In addition, the gating function can be used to eliminate parasitic responses and isolate the effect of a particular reflection.

Both measurement systems perform a one-port reflection measurement on the optical device under test by injecting an optical stimulus signal and detecting the reflected optical signal. In the case of the OTDR, the injected stimulus signal is an optical pulse or pulse train and the reflected signal consists of the reflected (Fresnel) and backscattered (Rayleigh) power.[1]

In the case of the HP 8702A OFDR mode, the injected stimulus signal is an amplitude modulated optical signal swept over a range of modulation frequencies, and the response is an interference pattern which is the summation of individual reflected (Fresnel) signals caused by differences in the index of refraction at each interface. The optical return loss versus distance information is generated by performing an inverse Fourier transform on the modulation frequency response data. OFDR as implemented by the HP 8702A does not detect the backscattered (Rayleigh) light, and therefore cannot measure loss versus distance in an optical device, such as an optical fiber. (See also "Optical Reflection Measurements," page 42.) However, the HP 8702A OFDR mode can measure optical reflections at each interface where the index of refraction changes and can locate each of these individual reflections very accurately. The system also allows the direct measurement and display of a reflection's magnitude in terms of optical return loss (dB) or reflection factor.

Since the HP 8702A system derives the time/distance information from the frequency-domain data and the system is calibrated to a known reflection (which calibrates the reflection level and location of the known reflection), there is no dead zone. In other words, reflections can be located from the calibration reference plane to many kilometers, depending upon the instrument calibration states.

The single-event resolution refers to the accuracy with which the location of any given reflection can be located. In the case of the HP 8702A system, any given reflection can be located within 2 mm, assuming that the index of refraction of the medium is known to an accuracy that does not limit the measurement system accuracy.

The two-event resolution is the minimum separation at which two adjacent reflections can be detected with at least 3 dB between their respective peaks and the valley between the peaks. The two-event resolution theoretical limit of the HP 8702A system is 3.4 cm and 1.7 cm for frequency spans of 3 and 6 GHz, respectively. Experiments have been conducted to verify the two-event resolution of the HP 8702A system on optical fiber samples cut to lengths of 2.5 and 6 cm. Each fiber end face was cleaved so that it was perpendicular to the fiber's longitudinal axis, yielding an end-face reflection (Fresnel) to air of approximately 3.5% of the incident power.

Reference

1. M. Fleischer-Reumann and F. Sischka, "A High-Speed Optical Time-Domain Reflectometer with Improved Dynamic Range," *Hewlett-Packard Journal*, Vol. 39, no. 6, December 1988, pp. 6-13.

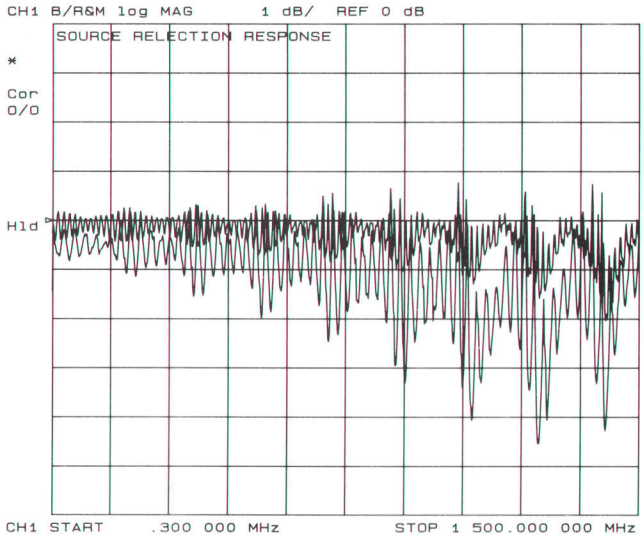


Fig. 14. *Laser transmitter modulation response characteristics. The lower trace is for 95% of the transmitted light reflected back to the laser. The upper, flatter response was obtained with a polarization controller between the transmitter and the 95% reflector.*

(continued from page 42)

and optical components. The HP 8702A system is well-suited to perform optical reflection and length measurements on a wide variety of components and subsystems.

Fig. 18 shows the block diagram for measuring optical reflections and optical return loss of any optical device under test. If a device has more than a single reflection, for example reflections of varying magnitudes spaced out at different distances in the device, the HP 8702A test system can measure the total reflection or each constituent reflection and its respective location. This system can be used to measure reflections of fiber optic components or bulk optic components when the proper collimating optics are added to the lightwave coupler test port in the test system.

If there are two or more reflections in the device under test, the individual reflections will have different phase relationships with respect to the measurement reference plane at a fixed modulation frequency that will sum to a given modulation amplitude and phase. As the modulation

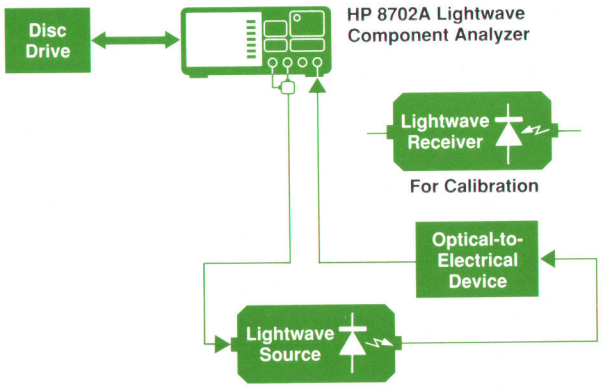


Fig. 15. *Measurement block diagram for optical-to-electrical devices such as photodiodes.*

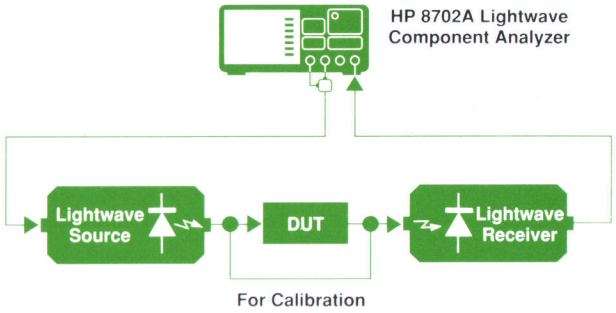


Fig. 16. *Measurement block diagram for two-port optical devices.*

frequency is changed, the phase relationship of each individual reflection will change, depending on its delay time, resulting in a different overall modulation amplitude and phase ripple pattern. The ripple pattern contains the reflection magnitude and location information. By performing an inverse Fourier transform on the ripple pattern, a signature of the individual reflections can be displayed as a function of time (and hence, distance).

The examples presented here show the reflection measurement capabilities of HP 8702A Lightwave Component Analyzer systems on small components.

The first example shows that with a lensed optical fiber connector added to the block diagram of Fig. 18, reflections of optical devices can be measured in an open-beam environment. The devices under test are a glass slide and a flat surface gold-plated to form a 95% reflective surface at 1300 nm. In Fig. 19, the top trace shows the ripple pattern generated from the reflections and rereflections from the glass slide and the gold wafer. The bottom trace shows the individual reflections and rereflections and their respective locations in time (distance).

The second example shows the reflections in a length of fiber that has three internal mirrors fabricated to produce

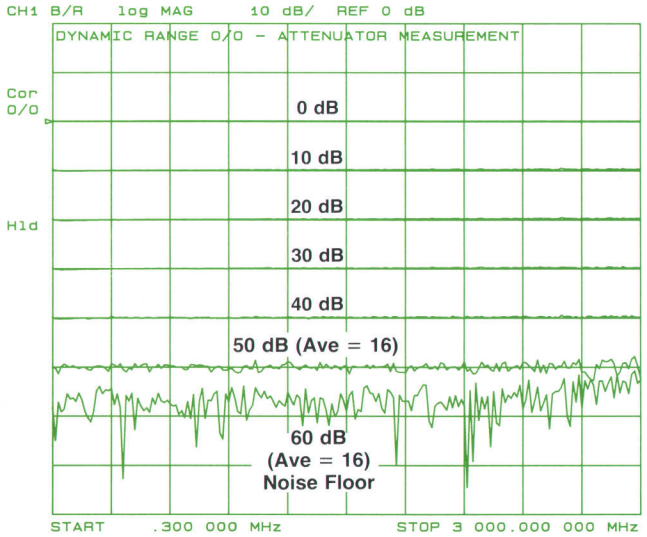


Fig. 17. *This attenuator loss measurement shows not only the insertion loss of the device, but also a dynamic range of at least 50 dB.*

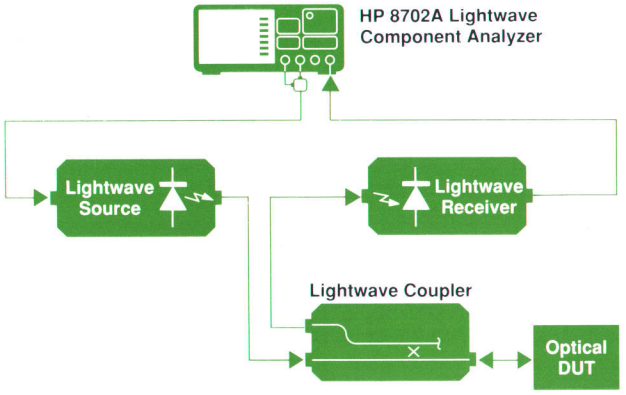


Fig. 18. *Measurement block diagram for measuring optical return loss and reflections.*

approximately 2% reflections at three different locations in the fiber. This component is typical of devices found in various fiber sensor applications. Fig. 20 shows the device dimensions and the measurement of the individual optical reflections and their respective locations. The absolute location of any of the individual reflections can be measured to within a few millimeters, given the correct test conditions.

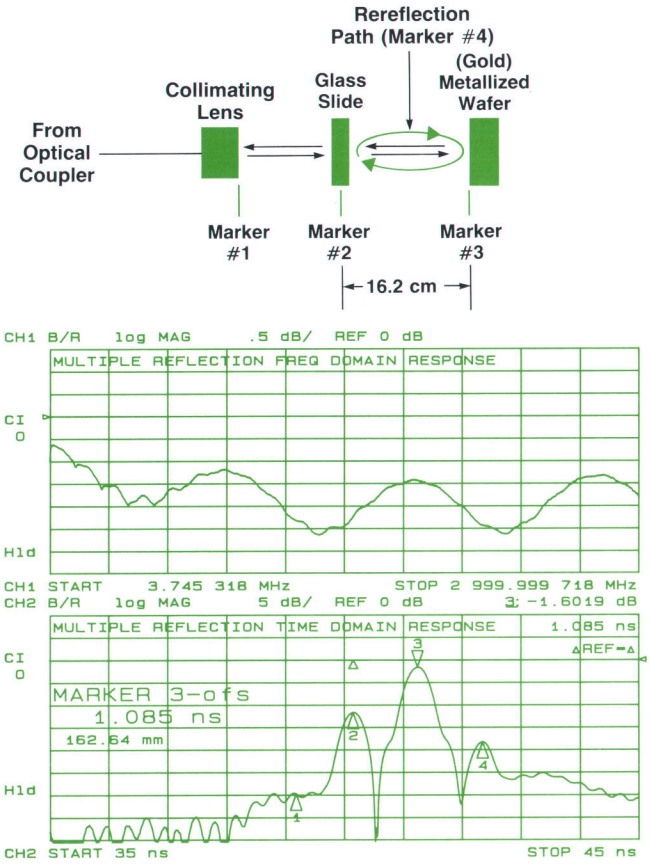


Fig. 19. *(top) A combination of a glass slide and a 95% reflector in an open-beam environment. (middle) Ripple pattern generated by reflections and rereflections in the setup at top. (bottom) Locations of individual reflections.*

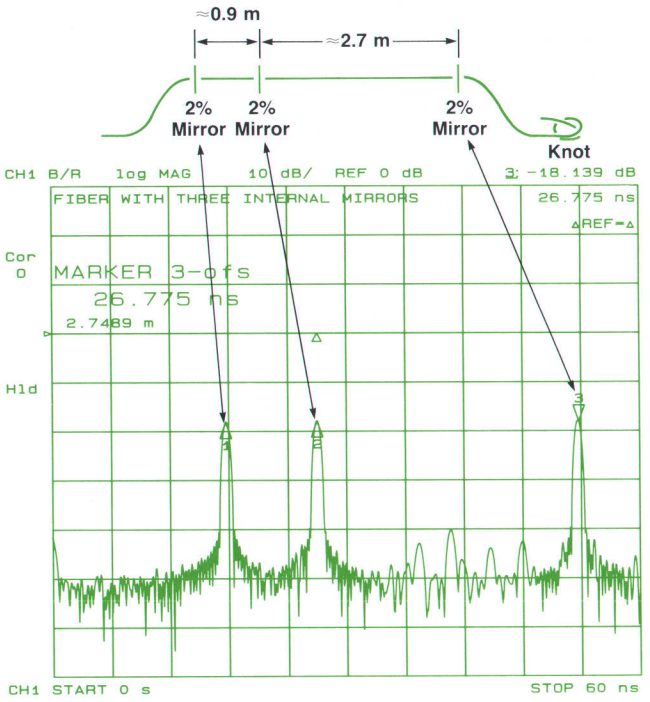


Fig. 20. *(top) A glass fiber with three internal mirrors producing 2% reflections. (bottom) A measurement of the three reflections and their locations. (Fiber internal mirrors courtesy of Texas A&M University EE Department.)*

Fig. 21 shows the optical return loss of the optical launch from a laser chip into the fiber identified by marker 3 (11.46 dB return loss), and the optical return loss of the laser module's optical fiber connector at marker 2 (about 37 dB return loss). Optical return loss of other optical devices and launches such as photodiodes, lenses, and antireflection coatings can also be measured easily.

The widest modulation frequency span limits the minimum separation at which two adjacent reflections can be resolved, that is, the best two-point resolution. For a 6-GHz modulation frequency span, the system's theoretical two-point resolution is about 1.71 cm in fiber. Fig. 22 shows a measurement of two reflections, about 4% each, spaced approximately 2 cm apart. The modulation frequency span was 6 GHz. Individual reflections can be located within less than 2 mm.

Optical Heterodyne Calibration System

The transfer function of each lightwave source and receiver is measured at the factory and stored on a disc, which is shipped with the product. This calibration data is loaded into the HP 8702A Lightwave Component Analyzer during the measurement calibration process.

System accuracy is adjusted at the factory using the simple but powerful heterodyne or beat frequency technique shown in Fig. 23. Light from two highly stable single-line lasers is combined to form a single optical test beam. The receiver under test effectively filters out the optical frequency terms and develops a beat frequency response only. A frequency sweep is obtained by changing the temperature of one of the lasers.[2]

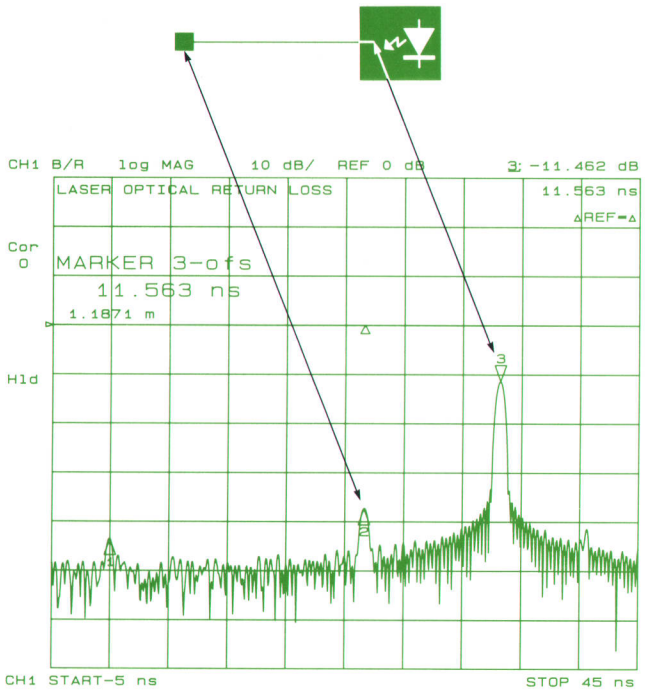


Fig. 21. *Optical return loss of the optical launch from a laser chip (marker 3) and the optical fiber connector (marker 2).*

A spectrum analyzer monitors the swept RF output of the lightwave receiver under test. Narrowband filtering, feasible because of the less than 10-kHz linewidth of the lasers, provides an exceptionally repeatable measurement.

Since the amplitude of the beat signal is a function of the polarization of the two laser beams, the system is implemented in polarization-maintaining fiber. Laser output polarization does not change over the modest temperature tuning range. A second error source, variation of the laser output powers with time and temperature, is eliminated by sampling their outputs throughout the measurement process and compensating for it. The receiver under test is calibrated as an optical average power meter and its bias current is monitored to measure variations in the average optical powers of the two lasers.

The resulting system is capable of generating beat frequencies from dc to over 40 GHz with over 50-dB dynamic range. A special reference receiver is calibrated with this system and used to calibrate sources and receivers.

Exceptional laser performance is obtained from Nd:YAG ring lasers, CW-pumped by shorter-wavelength diode lasers. Frequency tuning is accomplished by changing the temperature of the ring, which is fully contained in a specially faceted crystal.

Measurement Accuracy Considerations

The HP 8702A system performance depends not only on the performance of the individual instruments, but also on the measurement system configuration and on user-selected operating conditions. The HP 8702A system provides a set of measurement calibrations for both transmission and reflection measurements.

The type of calibration depends on the type of device and the measurement. For example, if the measurement is of optical insertion loss, a frequency response calibration would be performed. This calibration removes from the measurement the frequency response of the system. This is done by connecting a cable between the lightwave source and receiver. Once this measurement calibration is stored, the DUT can be connected in place of the cable and a corrected measurement (i.e., the DUT's optical insertion loss) will be displayed.

In any measurement, sources of uncertainty influence the system's measurement accuracy. The major ones are optical and RF connector repeatability, reflection sensitivity (or noise) of the laser source, directivity of a coupler in reflection measurements, and accuracy and repeatability of standards and models used in the measurement calibrations.

Connector repeatability is a measure of random variations encountered in connecting a pair of optical or RF connectors. The uncertainty is affected by torque limits, axial alignment, cleaning procedures, and connector wear. Optical connector repeatability problems can be minimized by using precision connectors such as the Diamond® HMS-10/HP connector,[3] or by using splices instead of connectors.

Reflection sensitivity (or noise) refers to the change in the behavior of a laser (i.e., its transfer characteristics) when reflected light reenters the laser cavity. The effect of the reflected light depends on many factors, including the magnitude, delay, and polarization of the reflected light. Reflection sensitivity can be minimized by buffering the laser with an optical attenuator or an optical isolator.

The term directivity refers to how well a directional coupler (optical or electrical) directs a signal, or how well it separates the incident from the reflected signal. Directiv-

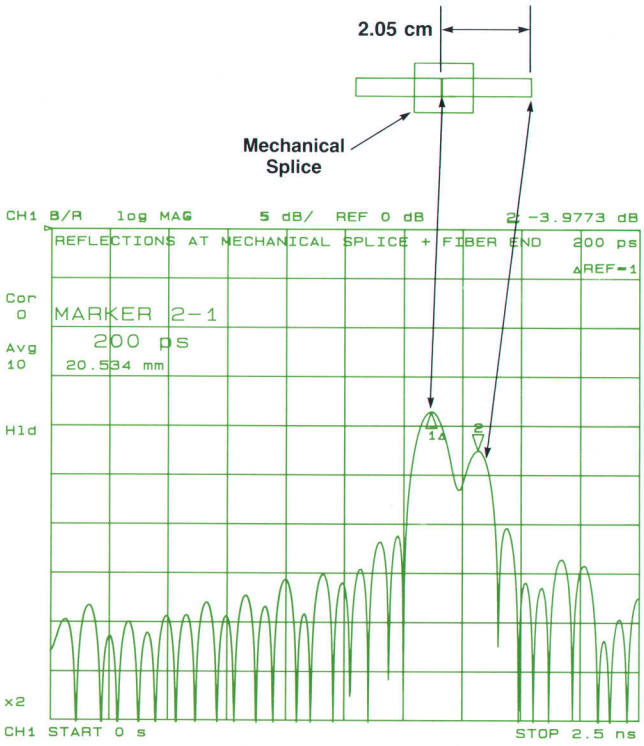


Fig. 22. *A measurement of two 4% reflections 2 cm apart.*

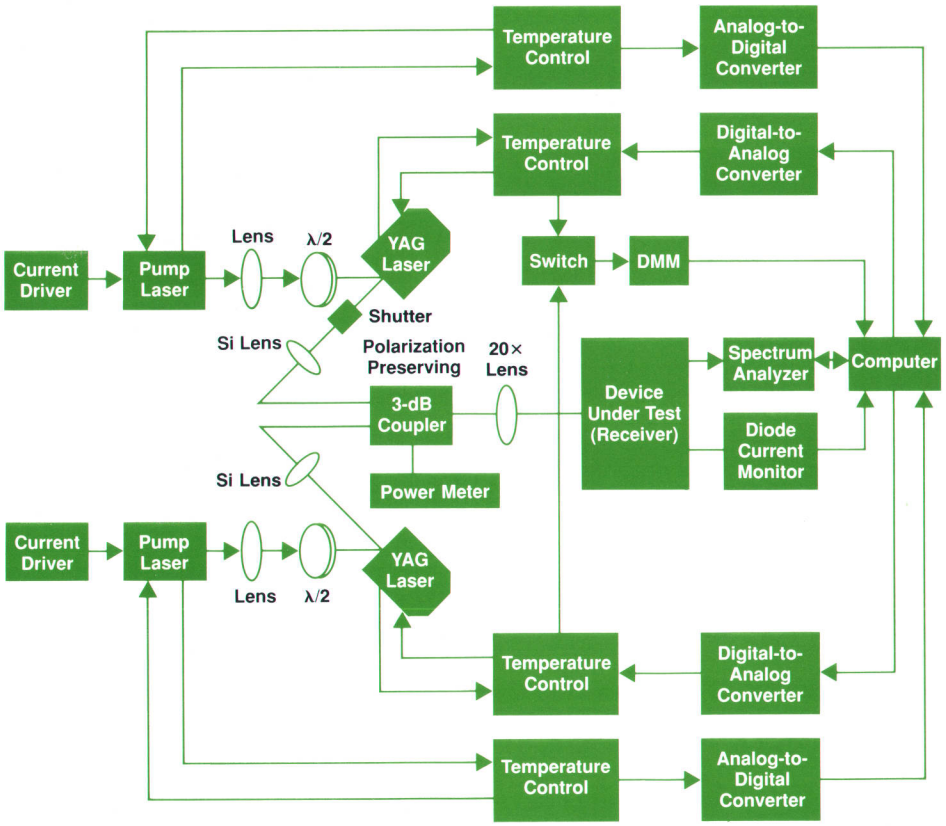


Fig. 23. *Heterodyne system used to calibrate lightwave receivers in production. Calibration data is stored on a disc that is shipped with the product.*

ity is calculated as the difference between the reverse isolation of the coupler and the forward coupling factor (e.g., if reverse isolation is −50 dB and coupling factor is −3 dB, then the directivity is −50 − (−3) = −47 dB). However, while the coupler itself may have > −50 dB directivity, the connectors and internal splices may cause reflections that may reduce the effective directivity of the packaged coupler.

Every measurement calibration uses standards that have default models within the HP 8702A instrument firmware or have data that is provided externally. Each of these standards and models has accuracy and repeatability characteristics that affect the overall system uncertainty. For example, when calibrating for an electrooptical measurement, the user can enter the transfer characteristic data of the lightwave source or receiver into the HP 8702A in two ways: by using the factory-supplied 3.5-inch disc or by entering the calibration factors printed on a label for the source or receiver. The lightwave source or receiver data has some accuracy relative to the factory system on which each instrument is measured. In addition, use of the 3.5-inch disc data offers better model repeatability than the calibration factors printed on the label, since the calibration factors represent a polynomial fit to the data stored on the disc.

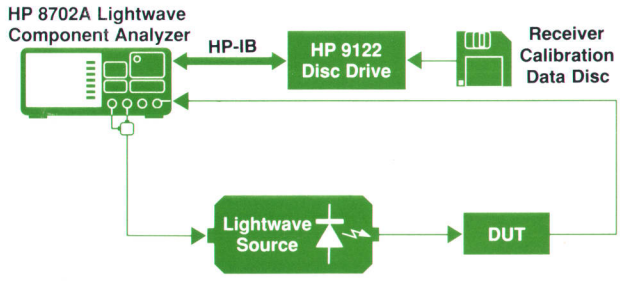


Fig. 24. *Measurement block diagram for a photodiode receiver transmission measurement.*

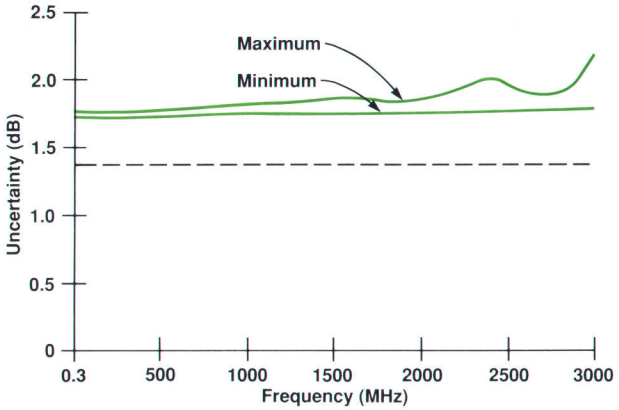


Fig. 25. *Typical ±3σ uncertainty of the receiver responsivity measurement as a function of modulation frequency. The solid lines show the maximum and minimum values for the setup of Fig. 24. The dashed line is the value for the same setup with a low-reflection 10-dB optical attenuator between the source and the receiver.*

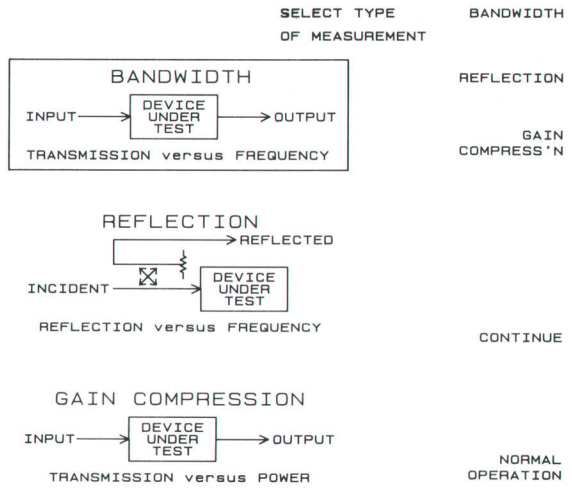


Fig. 26. *HP 8702A guided setup screen for selecting the type of measurement.*

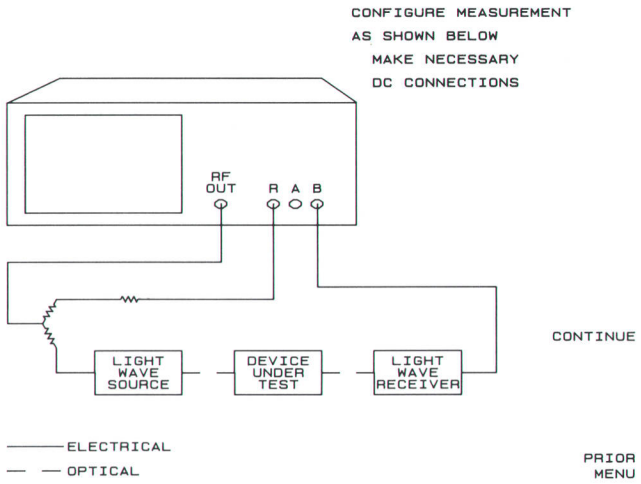


Fig. 28. *Screen for configuring the measurement hardware.*

Example: Receiver Responsivity Measurement

A photodiode receiver with 0.32 A/W or −10 dB responsivity, −14 dB of optical input mismatch, and −14 dB of electrical mismatch was measured by the system shown in Fig. 24. The responsivity of the receiver can be read from the CRT in dB for any given modulation frequency.

The uncertainties considered while computing the accuracy of the measurement are as follows: optical match interaction of the lightwave source and receiver, optical match interaction of the lightwave source and DUT, electrical match interaction of the lightwave receiver and the HP 8702A analyzer input, the same uncertainty for the DUT and the HP 8702A analyzer input, reflection sensitivity of the lightwave laser, dynamic accuracy, lightwave receiver accuracy, lightwave receiver model uncertainty, and wavelength related uncertainty.

Fig. 25 shows the uncertainty (dB) of the receiver responsivity measurement (described above) over an RF modulation frequency range of 300 kHz to 3 GHz. The solid lines represent the maximum and minimum values for the configuration shown in Fig. 24. The dashed line represents the value for the same configuration with a low-reflection 10-dB optical attenuator between the lightwave source and the DUT to reduce the reflection sensitivity of the laser.

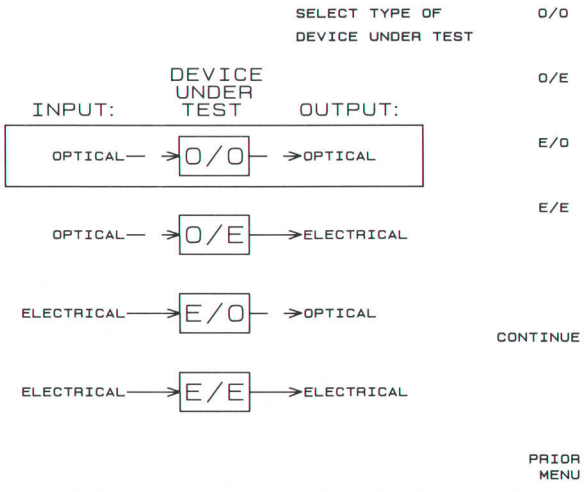


Fig. 27. *Screen for selecting the type of device.*

User Interface

A significant feature of the HP 8702A Lightwave Component Analyzer is the guided setup user interface. It consists of a series of softkey menus, instructions, and graphical displays to assist the user in configuring measurement hardware and in setting basic instrument parameters. Guided setup is one part of the user interface. The user is assisted in making fundamental measurements without facing the entire set of advanced instrument features.

The HP 8702A uses RF and microwave network analysis techniques for making various lightwave measurements. At the beginning of the project it was felt that many of the potential HP 8702A users would be unfamiliar with traditional HP network analyzers. A major goal of the project was to develop a user interface that would be easy to use, particularly for those with no network analyzer experience.

Guided setup provides a subset of the HP 8702A feature

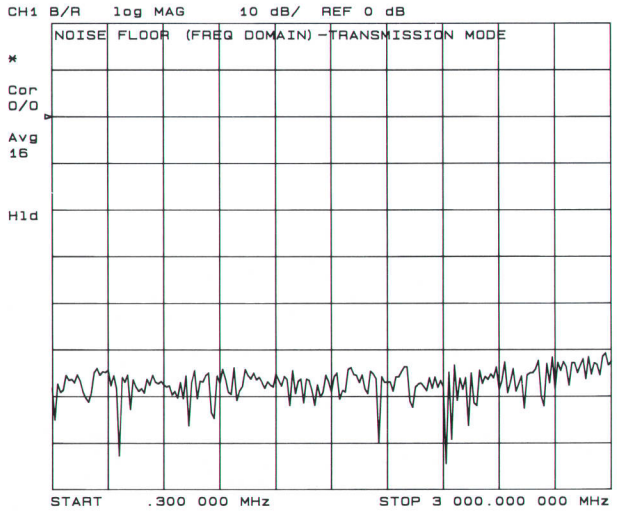


Fig. 29. *Noise floor trace for a 3-GHz system for optical transmission measurements.*

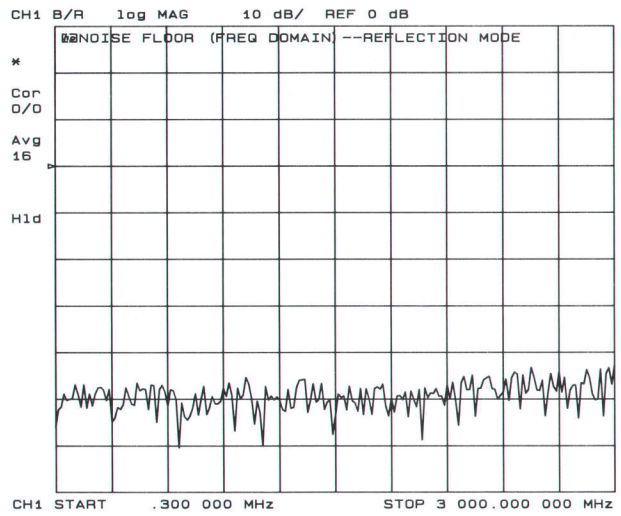


Fig. 30. *Noise floor trace for a 3-GHz system for optical reflection measurements in the frequency domain.*

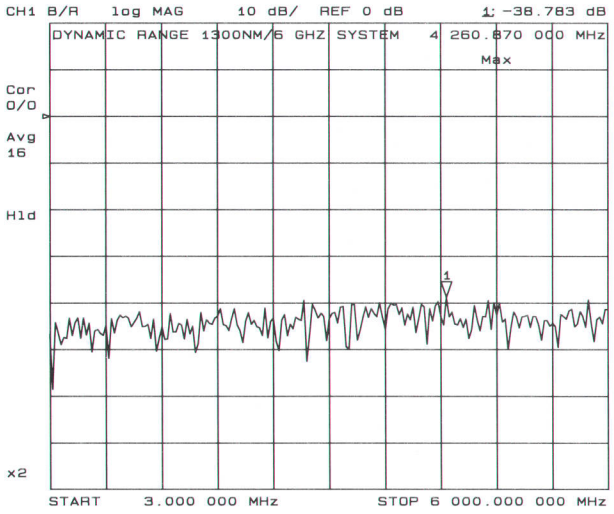


Fig. 32. *Typical 6-GHz system frequency-domain noise performance for optical transmission measurements.*

set. The user is given only the choices needed to set up a basic measurement. The commands are accompanied by textual and graphical instructions in a logical sequence.

When the analyzer is first turned on, the user is given instructions on choosing either guided setup or normal unguided instrument operation. At any time after selecting normal operation, the user can start guided setup through one of the regular softkey menus. Conversely, the user can exit guided setup and go to normal instrument operation at any time.

Guided setup consists of a series of screens that assist the user in configuring a measurement and setting basic instrument parameters. Each screen consists of a softkey menu, instructions, and a graphical display. The screens are ordered to teach the general measurement sequence recommended in the *User's Guide*. Each screen contains an operation to be performed or parameters to be set. The user progresses through guided setup by pressing the CONTINUE softkey. If existing values and/or instrument states are satisfactory, the user can proceed without making changes by pressing CONTINUE. To return to a previous screen, the user presses the PRIOR MENU softkey.

Guided setup has the general sequence: select type of measurement (Fig. 26), select type of device (Fig. 27), configure measurement hardware (Fig. 28), set instrument parameters, calibrate measurement, set measurement format and scale, print or plot measurement, and save instrument state in an internal register.

Guided setup is structured so that action is focused on the display and softkey menus. The user is not required to use the labeled keys on the front panel with the exception of the entry keys. Instrument parameter values are entered using the numbered keys, the up/down arrow keys, or the knob. Values are entered in normal operation with the same method.

System Performance

Typical measurement system performance is dependent

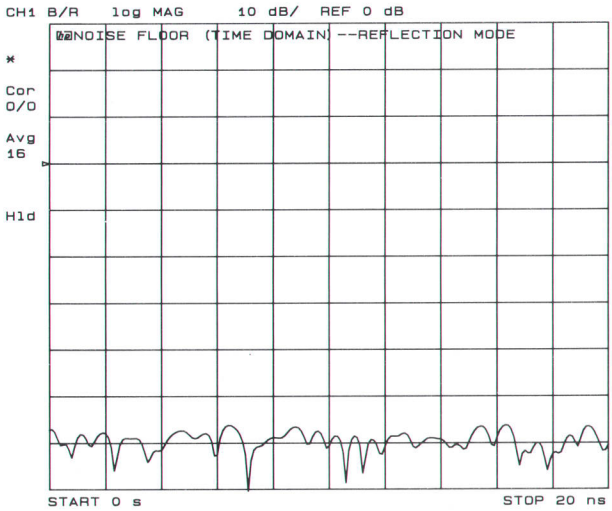


Fig. 31. *Effective 3-GHz system time-domain noise performance for optical reflection measurements.*

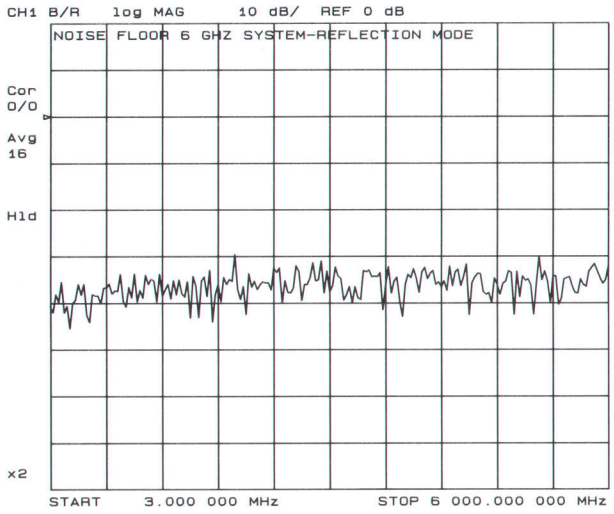


Fig. 33. *Typical 6-GHz system frequency-domain noise performance for optical reflection measurements.*

upon the lightwave source and receiver used with the HP 8702A Lightwave Component Analyzer. In addition, the system dynamic range and noise floor performance are dependent on the calibration routine selected (e.g., response or response/isolation calibration) and the signal processing features used (e.g., IF bandwidth, signal averaging, signal smoothing).

The system dynamic range is defined as the difference between the largest signal measured, usually given by a reference level of 0 dB, and a signal 3 dB above the system noise floor, as measured in the frequency domain. Besides the HP 8702A Lightwave Component Analyzer, the 3-GHz system includes an HP 83400A (1300 nm, 3 GHz, single-mode 9/125-μm fiber), HP 83401A (1300 nm, 3 GHz, multimode 50/125-μm fiber), or HP 83403A (1550 nm, 3 GHz, single-mode 9/125-μm fiber) Lightwave Source, an HP 83410B Lightwave Receiver, and an HP 11889A RF Interface Kit. The 6-GHz system includes an HP 83402A (1300 nm, 6 GHz, single-mode 9/125-μm fiber) Lightwave Source, an HP 83411A Lightwave Receiver, an HP 85047A 6-GHz S-Parameter Test Set, and an HP 8702A Lightwave Component Analyzer Option 006 (6-GHz capability). For reflection measurements, the addition of a lightwave directional coupler is required in the measurement block diagram as shown in Fig. 18. Depending upon the optical fiber size, either an HP 11890A (single-mode 9/125-μm fiber) or an HP 11891A (multimode 50/125-μm fiber) Lightwave Coupler should be used.

To determine the system dynamic range, the system noise floor must be determined for the measurement. For the 3-GHz system, typical noise floor performance is shown in Figs. 29, 30, and 31. Fig. 29 shows an averaged noise floor trace (ave = 16) for optical transmission measurements; it varies from −55 dB at low frequencies to −50 dB at 3 GHz, which yields a 47-dB dynamic range. Fig. 30 shows an averaged noise floor trace (ave = 16) for optical reflection measurement in the frequency domain; it varies from −47 dB to −43 dB. This noise floor yields a 40-dB dynamic range in the frequency domain. Fig. 31 shows the effective system noise floor for an optical reflection measurement viewed in the time domain. It is derived by performing an inverse Fourier transform on the optical reflection noise floor data in the frequency domain shown in Fig. 30. The effect of the inverse Fourier transform on the frequency-domain data is to increase the measurement dynamic range in the time domain. Fig. 31 shows a 12-dB improvement in dynamic range or a noise floor of −55 dB in the time/distance domain.

For the 6-GHz system, typical frequency-domain noise performance for optical transmission and reflection measurements is shown in Figs. 32 and 33, respectively. Typical time-domain or distance-domain noise performance for optical reflection measurements derived from frequency-domain data (Fig. 33) is shown in Fig. 34. In Fig. 32, the noise trace was averaged sixteen times and shows a −38-dB worst-case point, which corresponds to a dynamic range of 35 dB. Fig. 33 shows an averaged (ave = 16) noise floor performance of −30 dB for optical reflection measurements obtained in the frequency domain; this corresponds to a usable dynamic range of 27 dB, typically. For optical reflection measurements in the time or distance domain, the averaged noise floor is reduced to −41 dB, which corresponds to a dynamic range of 38 dB, typically.

Table II summarizes the typical system dynamic range for each combination of lightwave source and receiver in the HP 83400 family when used with the HP 8702A Lightwave Component Analyzer.

Table II
Typical System Dynamic Range

	3-GHz System	6-GHz System
Electrical [1] or Electro-optical [2] (Frequency Domain)	100 dB	80 dB
Optical [3]		
Transmission (Frequency Domain)	47 dB	37 dB
Reflection (Frequency Domain)	40 dB	27 dB
Reflection (Time Domain)	52 dB	38 dB

1. Electrical-to-electrical device:

$$dB = 10\log(P_2/P_1) = 20\log(V_2/V_1)$$

where: P_1 = RF power available at port 1,
P_2 = RF power available at port 2,
V_1 = RF voltage at port 1,
V_2 = RF voltage at port 2
(50 Ω impedance system).

2. Electrical-to-optical device:

$$dB = 20\log((r_s)/(1W/A))$$

where: r_s = slope responsivity of the electrical-to-optical device.

Optical-to-electrical device:

$$dB = 20\log((r_r)/(1A/W))$$

where: r_r = slope responsivity of the optical-to-electrical device.

3. Optical device:

$$dB = 10\log(P_1/P_2)$$

where: P_1 = optical power at port 1,
P_2 = optical power at port 2.

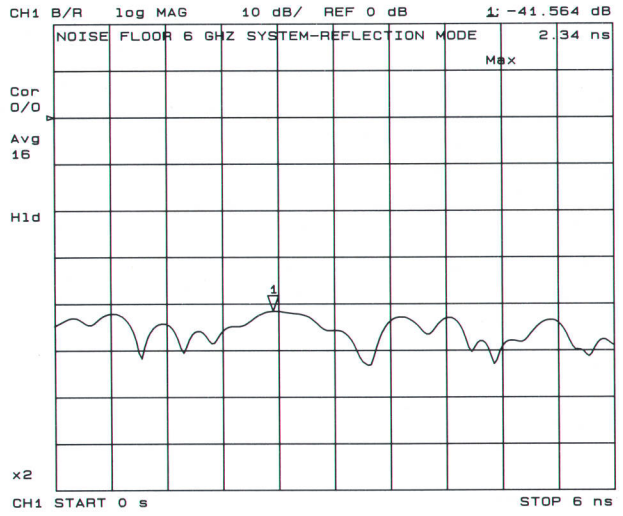


Fig. 34. *Typical 6-GHz time-domain noise performance for optical reflection measurements.*

Acknowledgments

Contributions to the development of the HP 8702A measurement systems came from many people located in HP Laboratories and other divisions of Hewlett-Packard. We appreciate their contributions and thank them for their support. We would like to acknowledge and thank Tom Hornak and his team in HP Laboratories for their guidance and support during the early phases of the program. Finally, we would especially like to thank Hugo Vifian for his strong support and encouragement throughout the development program, and other members of the Network Measurements Division lightwave support teams for their contributions to the program.

References

1. R. Wong, M. Hart, G. Conrad, and D. Olney, *The Lightwave Component Analyzer: High-Frequency Measurements of Lightwave Systems and Components*, Hewlett-Packard Publication No. 5956-4356.
2. T.S. Tan, R.L. Jungerman, and S.S. Elliott, "Calibration of Optical Receivers and Modulators Using an Optical Heterodyne Technique," *IEEE Transactions on Microwave Theory and Techniques*, Vol. MTT-2, May 25, 1988, pp. 1067-1070.
3. W. Radermacher, "A High-Precision Optical Connector for Optical Test and Instrumentation," *Hewlett-Packard Journal*, Vol. 38, no. 2, February 1987, pp. 28-30.

Design and Operation of High-Frequency Lightwave Sources and Receivers

These compact, rugged modules are essential components of HP 8702A Lightwave Component Analyzer Systems.

by Robert D. Albin, Kent W. Leyde, Rollin F. Rawson, and Kenneth W. Shaughnessy

FOR HIGH-FREQUENCY FIBER OPTIC MEASUREMENTS, calibrated transitions are needed from electrical signals to optical signals and back again. In HP 8702A Lightwave Component Analyzer systems, these transitions are provided by the HP 83400 family of lightwave sources and receivers, which are designed for easy integration into HP 8702A measurement systems. Power supply connections, RF connections, signal levels, and calibration data are all designed for direct compatibility with the HP 8702A, which is the signal processing unit in the system.

To date, four lightwave sources and two lightwave receivers have been released. They are:

- HP 83400A Lightwave Source—1300 nm, 3-GHz modulation, single-mode 9/125-μm fiber
- HP 83401A Lightwave Source—1300 nm, 3-GHz modulation, multimode 50/125-μm fiber
- HP 83402A Lightwave Source—1300 nm, 6-GHz modulation, single-mode 9/125-μm fiber
- HP 83403A Lightwave Source—1550 nm, 3-GHz modulation, single-mode 9/125-μm fiber
- HP 83410B Lightwave Receiver—1300 or 1550 nm, 3-GHz modulation, multimode 62.5/125-μm fiber
- HP 83411A Lightwave Receiver—1300 or 1550 nm, 6-GHz modulation, single-mode 9/125-μm fiber.

Source Design and Operation

The signal path through each source starts at the rear-panel RF connector and proceeds through a matching circuit, an RF attenuator. The attenuator output is transformed into a modulated light signal by a laser diode. The optical laser output signal is transmitted through a short piece of optical fiber to the front-panel connector (see Fig. 1).

Power for the source is supplied from the probe power jacks on the front panel of the HP 8702A. Bias current requirements of the internal components exceed the 400 mA available from this 15V supply, so each source includes a dc-to-dc converter, which changes the supply to 3V, 1A for the thermoelectric heat pump

It was decided to use a laser diode rather than an LED as the source element to take advantage of laser diodes' high modulation-rate capability, high power, and narrow optical spectrum. Lasers are, however, fairly sensitive to temperature variations. Output power and wavelength vary as a function of temperature. The lifetime of the laser is affected by the temperature of the environment as well. The degradation of operating life as the diode junction temperature is increased is shown in Fig. 2.

To help minimize these temperature effects, a thermal control loop is used to regulate the temperature of the laser to a constant 20°C. 20°C was chosen to give optimum laser lifetime and temperature regulation range. The thermoelectric heat pump has a range of cooling of approximately 40°C. The thermal loop maintains the temperature of 20°C within 0.1°C over the specified environmental temperature range of 0°C to 55°C.

The laser diode and a temperature sensor are both mounted on the surface of the thermoelectric heat pump. A voltage proportional to the temperature of this surface is generated by the sensor and external circuitry and then

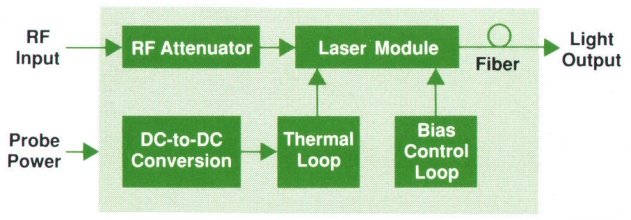


Fig. 1. *Lightwave source block diagram.*

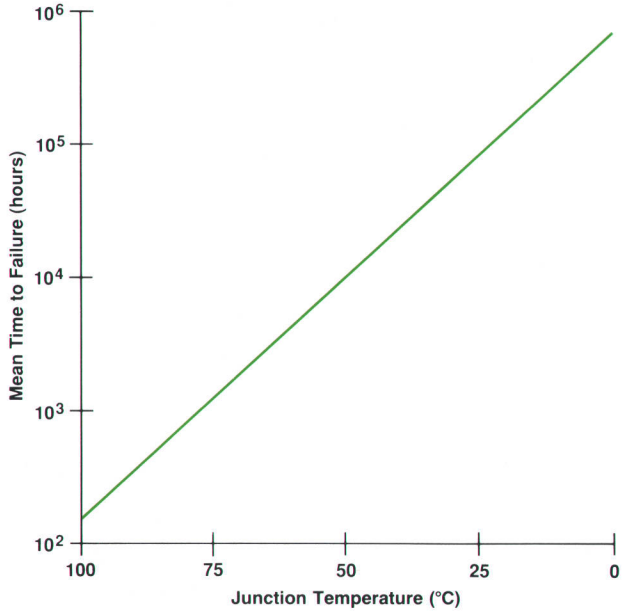


Fig. 2. *Laser diode failure rate as a function of junction temperature.*

applied to an integrator as an error signal. The integrator output serves as a control signal for the 70-kHz pulse width modulated current control circuit. The output of this circuit goes to an H-bridge (see Fig. 3) which directs current through the thermoelectric heat pump in the proper sense to either heat or cool the laser.

The laser operating point is set by another control loop (see Fig. 4) consisting of a photodiode, an amplifier, and the laser bias current source.

The laser diode chip has a front facet and a back facet from which light is emitted. The front-facet light is coupled into the fiber and goes to the front-panel connector. The back-facet light is coupled into a photodiode to generate a current proportional to the emitted light. The bias control circuit receives this current and generates an error voltage, which controls the laser bias current source. The control loop's bandwidth is limited to well below the applied RF frequencies.

It is not desirable for the modulating signal to drive the laser current to its threshold value, since this would cause clipping of the optical signal. $I_{threshold}$ is the current at which the laser diode begins the lasing operation, that is, when the laser bias current is large enough to produce a gain that exceeds the losses in the laser cavity.

The dc transfer function of the laser diode is shown in Fig. 5. At very high diode current, a droop in laser output may occur. This phenomenon is known as a kink. If the laser current is allowed to swing into this region, distortion of the modulation will occur. Therefore, the laser diode operation point is bounded by $I_{threshold}$ on the low end and the kink region on the high end.

RF modulation is applied to the laser through a dc blocking capacitor and an RF attenuator. An impedance matching network is included to match the 50Ω input to the laser impedance (Fig. 6). Some adjustment of the laser transfer function is accomplished by varying the RF attenuator to match the RF swing to the individual laser.

The impedance matching network matches the low-impedance laser diode to 50Ω. A variable capacitor is included in the matching network to flatten the modulation frequency response of the laser. Adjustment of this capacitor results in a typical frequency response flatness of ±1 dB to 3 GHz.

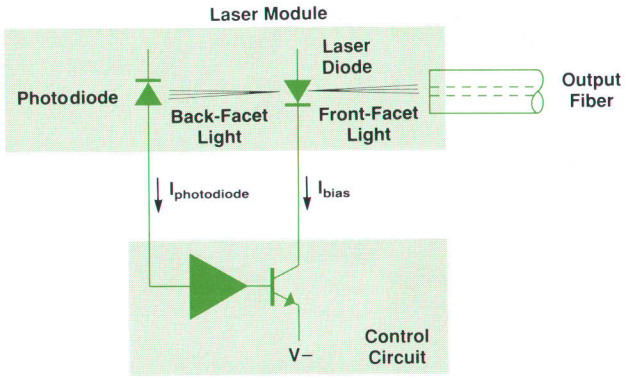


Fig. 4. *Laser diode bias control loop.*

The source microcircuit package is a straightforward deep-well design. The laser package is retained in the microcircuit package by two wedging clamps which force it against x-axis and y-axis datum surfaces while keeping its bottom surface pressed against the housing floor. This approach was chosen to locate the laser precisely relative to the sapphire microcircuit while ensuring adequate heat sinking for the laser's internal thermoelectric heat pump. The thin-film microstrip circuit provides the RF interface between an SMA connector and the RF terminals of the laser package. An epoxy-glass printed circuit board interconnects the dc terminals of the laser with the filtered feedthroughs of the microcircuit package (Fig. 7).

Receiver Design and Operation

The signal path through the receiver starts at the front-panel optical connector (see Fig. 8). Once the modulated optical signal is inside the receiver module, it travels through a short input fiber to the optical launch, where it is coupled to a pin photodiode chip. The output of the photodiode is an alternating current at the same frequency as the modulation. This signal is amplified by a transimpedance amplifier. The output of the amplifier is routed to the back panel of the receiver by a short length of coaxial cable.

While simple in concept, the optical launch is difficult

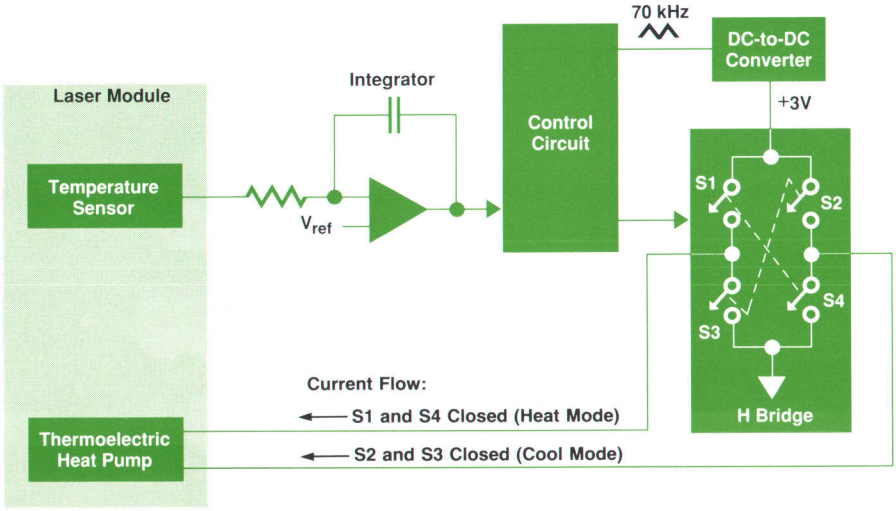


Fig. 3. *The thermal control loop keeps the laser diode within 0.1°C of 20°C.*

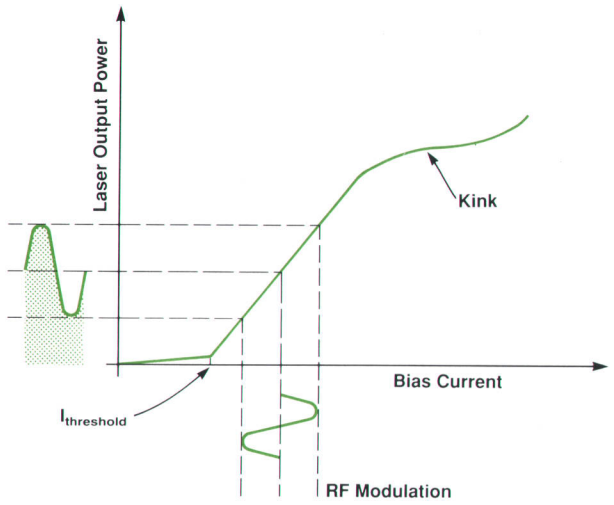


Fig. 5. *The allowable laser operating region is between the threshold current and the kink region.*

to realize because of the dimensions and parameters of the components involved. The most obvious approach, launching the light directly from the fiber end, was tried first. This approach was abandoned because of poor and inconsistent performance, fragility, and difficult assembly. The final design, which offers numerous advantanges, is shown in Fig. 9.

A graded-index (GRIN) lens is the primary optical element in the launch. Whereas normal lenses (e.g., planoconvex) use material with a constant index of refraction material and curved surfaces to refract light, graded-index lenses have flat end faces and refract light by virtue of their quadratically varying internal index of refraction.

The path of a light ray in a graded-index lens is sinusoidal. The length of the path is used to describe the fundamental parameter of the lens, known as pitch. If a light ray traces a sinusoid of 180 degrees, the lens is said to be half-pitch. If the ray traces a sinusoid of 90 degrees, the lens is said to be quarter-pitch, and so on.

The lens used in the lightwave receiver optical launch is just slightly less than half-pitch. Light enters the lens from the input fiber. It then diverges along sinusoidal paths. About halfway through the lens the light beam is collimated and is at its maximum width. Past this point the beam starts to converge. Just before the beam converges, it exits

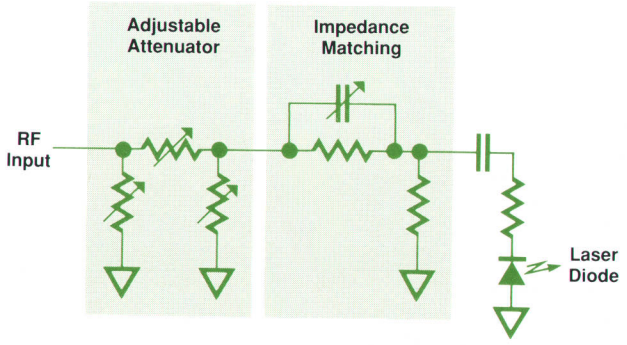


Fig. 6. *Input circuit of the lightwave source.*

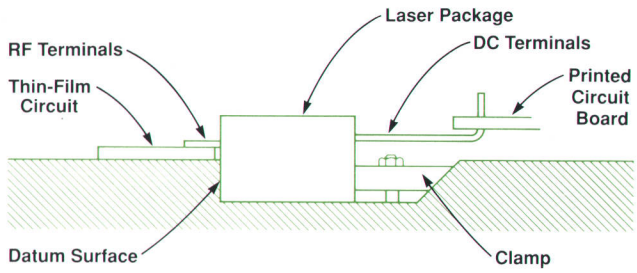


Fig. 7. *Mounting of the laser in the source microcircuit.*

the lens. After traveling a short distance through air, the beam converges and forms an inverted image of the input fiber on the face of the photodiode.

The GRIN lens is mounted in a machined ceramic cap. The ceramic cap was chosen to minimize the effect on the electrical performance of the microstrip thin-film circuit. The ceramic material used provides a thermal match to the sapphire circuit. This is important since the cap is solidly attached to the circuit.

Alignment of the optical launch and photodiode is critical if all of the incident light is to impinge on the small active area of the photodiode detector. Misalignments on the order of a few micrometers can result in substantial signal loss. Achieving this alignment solely through mechanical precision would have been difficult and expensive. Instead, alignment accuracy is achieved by using an interactive technique, as shown in Fig. 10.

The interactive alignment technique works as follows. First, the receiver microcircuit is placed in a test fixture and power is applied. An optical launch assembly consisting of a GRIN lens mounted in a ceramic cap is then coarsely aligned with the photodiode. Modulated optical power is applied to the GRIN lens by the test system lightwave source. Next, micropositioners are used to adjust the position of the optical launch while the output of the receiver microcircuit is monitored by the HP 8702A. When the position of the launch assembly has been optimized for power output, the assembly is fastened to the sapphire thin-film circuit.

The conversion from optical signal to electrical signal takes place at the pin photodiode. The photodiode is of a proprietary design, optimized for this application. Some of its requirements are that it respond strongly to light in the wavelengths of interest, have a flat frequency response that is uniform across its entire active area, have an active

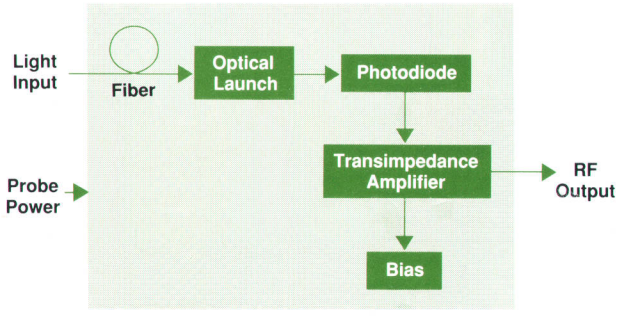


Fig. 8. *Lightwave receiver block diagram.*

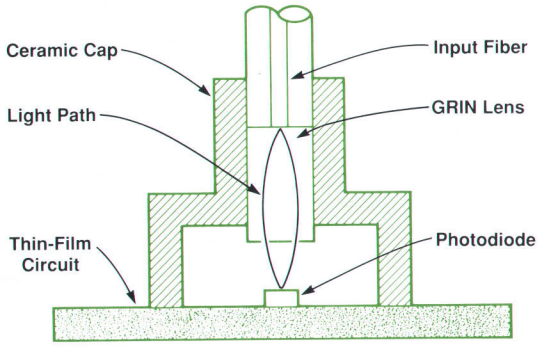


Fig. 9. *Optical launch in the lightwave receiver.*

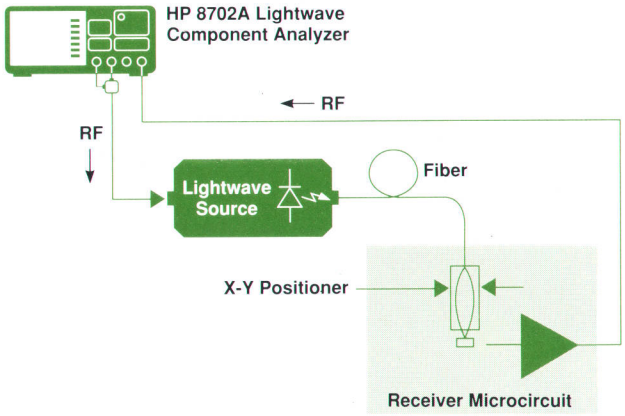


Fig. 10. *Alignment of the optical launch.*

area large enough so that all of the incident optical signal can be focused onto it, be linear over a wide range of input power, and possess good antireflection properties to keep reflected optical signals to a minimum.

The pin photodiode gets its name from its structure. The top layer is p-type semiconductor, the middle layer is i-type or intrinsic semiconductor, and the bottom layer is n-type material. Photons enter the photodiode through the top layer. The bandgap of the material is such that it appears transparent to the photons and they pass right through. An electrical signal is generated when photons are absorbed in the i layer of a reverse-biased photodiode, creating an electron-hole pair. A strong electric field then sweeps out the carriers, creating a current that is amplified and detected in the HP 8702A Lightwave Component Analyzer.

Once the signal has been generated by the pin photodiode, it must be transferred into the measurement system. As is typical in high-frequency applications, the system uses 50-ohm terminations and coaxial cables.

Output impedance is one of the receiver parameters optimized to facilitate system integration. In a system where termination impedances are not well-controlled, standing waves may result. Careful control of the receiver output impedance and the well-controlled input impedance of the HP 8702A minimize these standing waves and the measurement errors they can cause.

The HP 83410B Lightwave Receiver also includes a transimpedance amplifier to increase signal strength. Specifications for the amplifier are derived from HP 8702A system requirements. The fundamental specification of the amplifier is gain. A value for gain is arrived at by considering the output noise of the amplifier and the sensitivity of the HP 8702A's receiver. To realize the best system performance with the least expense and complexity, the

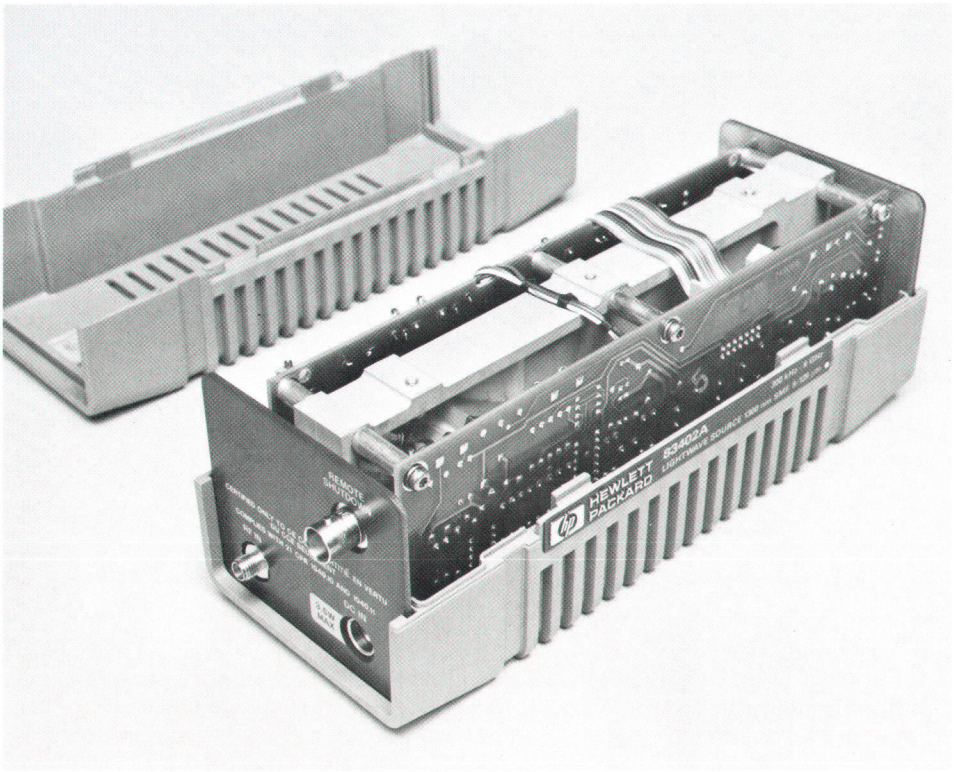


Fig. 11. *Typical lightwave module without enclosure. All components are mounted to the spine.*

High-Speed PIN Infrared Photodetectors for HP Lightwave Receivers

The HP 83400 family of lightwave receivers uses customized InP/InGaAs/InP pin photodetectors. The pin detector works by converting incoming optical energy into an electrical current. Light of wavelengths 1.2 to 1.6 μm passes through the transparent InP p layer. The photons are absorbed in the InGaAs i region creating an electron/hole pair. The device is fabricated on an n-type conductive InP substrate. The device is operated in reverse bias and the electric field sweeps out carriers, creating a current.

A cross section of the device is seen in Fig. 1. The detector epitaxial layers are grown using organometallic vapor phase epitaxy (OMVPE). The mesa structure provides a low-capacitance device for high-frequency applications.

Receiver performance is determined by device dark current, responsivity, frequency response, capacitance, and optical reflections. The photodetector needs a low dark current, which is a measure of the leakage current of the device under reverse bias and no illumination. High dark currents may translate into noise in the lightwave reciever. Dark currents for these devices are <30 nA at −5V. The high-frequency operation is determined by a combination of the RC time constant of the photodetector and the transit time for carriers through the i layer (InGaAs region). Capacitance should be low and transit times short. These two parameters are interconnected. If the i layer is thin for short transit times, the capacitance increases. The design must be optimized with both in mind. Two such designs are used. In the HP 83410B Lightwave Receiver, operation to 3 GHz is achieved, and in the HP 83411A Lightwave Receiver, 6 GHz is obtained. The frequency response must also be flat across the device's active region.

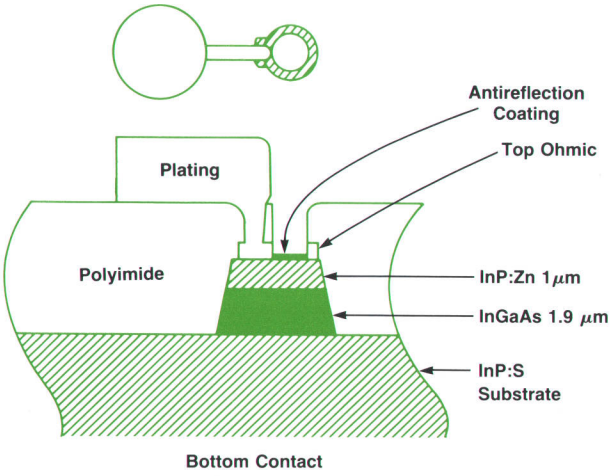


Fig. 1. *Photodetector diode top view and cross section.*

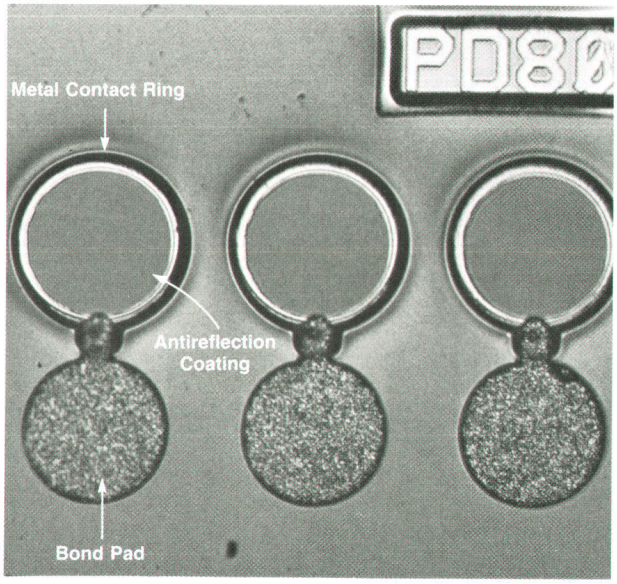


Fig. 2. *Photograph of photodetector chip.*

Responsivity is a measure of diode sensitivity. It is the ratio of photocurrent (I_p) output to absorbed optical power input:

$$r = I_p/P_{optical}.$$

It is important to have high responsivity to absorb as many incoming photons as possible and convert them into photocurrent. Typical responsivity values are 0.9A/W at −5V for incoming light wavelengths of both 1330 nm and 1550 nm.

Low optical reflections are important in a lightwave system to avoid feedback to the laser light source. To achieve the highest quantum efficiency, carriers need to pass through the top InP layer and not be reflected at the top diode surface. An antireflection coating is used to acheive <2% reflection from the diode surface for both 1300 nm and 1550 nm wavelengths of incoming light.

The devices have been tested for long-term reliability by examining the mean time to failure under high stress conditions of 175°C and −5V. The high-temperature operating life tests show lifetimes greater than 3×10^5 hours at 55°C instrument operating temperature.

Fig. 2 shows a photograph of a photodetector chip containing three devices. It shows the metal contact ring, active area with antireflection coating, and device bond pad.

Susan Sloan
Development Engineer
Microwave Technology Division

amplifier has just enough gain so that its output noise is approximately equal to the input sensitivity of the HP 8702A receiver. Any more amplification and the sensitivity of the HP 8702A receiver would be wasted; any less and the system sensitivity would drop.

The amplifier is realized using thin-film circuit construction for optimum wideband frequency response. Silicon bipolar transistors are used instead of GaAs FETs to minimize 1/f noise.

Mechanical Considerations

It was felt that small, rugged modules would offer significant advantages to the user in ease of positioning relative to the DUT and in environments such as light tables, where space is at a premium and operation remote from the analyzer is required. The die-cast housings offer the right

combination of size and shape. When assembled to the modules' aluminum center body, or spine, (the modules' single structural component), these survived shock and vibration levels during strife testing that were ten times the qualification test levels. All components—microcircuits, printed circuit boards, fiber optic and electrical I/O, and cabling—are mounted to the spine, which allows full assembly and testing of the modules before they are installed in their respective enclosures (Fig. 11).

The fiber optic connector adapter system developed by HP's Böblingen Instrument Division is used on both source and receiver modules. Based on the precision Diamond® HMS-10/HP fiber optic connector,[1] the design of these adapters allows easy access to the ferrule for cleaning, and allows the internal HMS-10/HP connector to be mated to any of five different connector systems: HMS-10/HP, FC/PC, ST, biconic, and DIN. A hinged safety shutter is provided on the source modules to comply with safety regulations in certain localities.

Assemblability evaluation method* techniques were used throughout the development of the source and receiver modules. This method was an extremely useful tool in exposing hot spots in the mechanical design, areas where the number and/or complexity of the steps in an assembly operation make it particularly difficult. Perhaps more important, it provided a simple structured way of comparing designs and making rough estimates of their cost of assembly.

Reference

1. W. Radermacher, "A High-Precision Optical Connector for Optical Test and Instrumentation," *Hewlett-Packard Journal*, Vol. 38, no. 2, February 1987, pp. 28-30.

*Assemblability evaluation method is a system developed by Hitachi, Ltd. and refined by General Electric Company. It sets forth objective criteria for evaluating mechanical designs in terms of the number of parts and the relative difficulty of the operations necessary to assemble them.

Videoscope: A Nonintrusive Test Tool for Personal Computers

The Videoscope system uses signature analysis techniques developed for digital troubleshooting to provide a tool that allows a tester to create an automated test suite for doing performance, compatibility, and regression testing of applications running on HP Vectra Personal Computers.

by Myron R. Tuttle and Danny Low

INTERACTIVE TESTING OF APPLICATION SOFTWARE requires the tester to sit at the test system and enter test data using a keyboard and/or some other input device such as a mouse, and observe the results on the screen to determine if the software being tested produces the correct results for each set of test data. This process is time-consuming and error-prone if it is done manually each time the tester wants to repeat the same set of tests. This process must be automated to ensure adequate test coverage and improve the productivity of testing.

Videoscope is a test tool developed and used by HP's Personal Computer Group (PCG) for automated performance, compatibility, and regression testing of interactive applications running on HP Vectra Personal Computers. It is independent of the operating system and nonintrusive. Nonintrusive means that it does not interfere with or affect the performance and behavior of the application being tested or the operating system. Videoscope is for internal use and is not available as a product.

An overview of the operation of Videoscope is illustrated in Fig. 1. During test creation the tester manually enters test data to the application being tested and waits for the correct result to show on the display. As the tester enters the test data, Videoscope records the data into a file called a test script. At the command of the tester Videoscope also records the screen result in the test script. The tester continues this process for each set of test data and at the end of testing the test script contains a sequence of test data interspersed with correct screen results. For retesting the same application, Videoscope automates the process by replacing the tester and playing back the test script to the application (Fig. 1b). The test data is sent to the application as it was entered during test recording. Whenever a screen result is encountered in the test script, Videoscope waits for it to occur on the display and then automatically does a comparison between the current screen and the correct screen results in the test script to determine if the test passes or fails.

The concepts and motivation for developing Videoscope evolved from experiences with trying provide the best test coverage of applications running on the HP Vectra PC. When the HP Vectra PC was developed, a major goal of the product was that it be compatible with the industry standards established by the IBM PC/AT. Compatibility was determined by running various applications written for the IBM PC/AT and evaluating how well they ran on the Vectra. The first iteration of testing was done by hand using most of the engineers in the lab. This was clearly an inefficient and expensive way to run these tests. The tests were then automated using two utility programs, Superkey and Sidekick from Borland International Incorporated. Superkey was used to capture and play back keystrokes, and Sidekick was used to capture screen displays and save them to disc files where they were compared with files containing known-correct screen displays. These tools and certain standards for creating tests were called the regression test system (RTS).

While RTS initially proved adequate, long-term use revealed weaknesses in the system. First, mouse movements

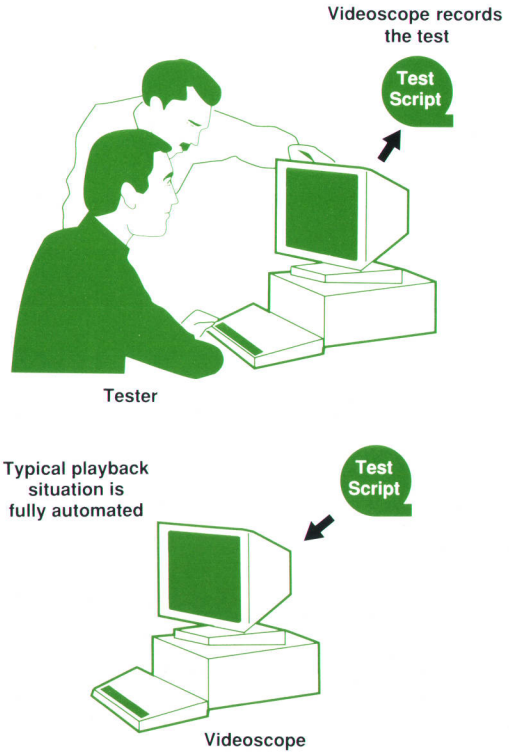


Fig. 1. *An overview of the operation of the Videoscope system. (a) Test recording. (b) Test playback.*

could not be captured or played back, so any program that used a mouse had to be tested by hand. Second, the system was intrusive. This meant that certain programs did not act the same when RTS was loaded. The deviations ranged from running differently to not running at all. For example, Microsoft® Windows could not run at all with RTS because of conflicts over control of the interrupt vectors. Other applications could not run because RTS used up so much memory that there was not enough left for the application. Finally, RTS could not be used to do performance testing since it used system resources and affected the performance of the system.

Videoscope was developed to replace RTS and to compensate for its weaknesses. This resulted in the following design objectives for the Videoscope system:

- It had to have the same capabilities as RTS.
- It had to be nonintrusive.
- It had to be able to do controlled-time and real-time performance testing. Controlled-time means that fixed time delays are inserted in the test scripts to control when certain events take place on the system under test. Real-time performance testing means the ability to determine the actual response time of events taking place on the system under test.
- It had to be able to handle a mouse and any other pointing device that HP sells for the Vectra.
- It had to support HP extensions to the PC standard.
- Test scripts had to be portable. The intent of this objective is to be able to port test scripts to other PC operating systems such as Xenix, OS/2, or even HP-UX. It was also considered necessary to be able to use a multitasking computer system such as the HP 3000 Computer System as a host to test multiple systems on playback.
- It had to be able to handle a list of programs (e.g., Microsoft Windows and HP AdvanceWrite) that we needed to test but were unable to test with RTS.

Videoscope System

The Videoscope system consists of two major parts: a program called vscope that resides in a system known as the host system, and a board called the Videoscope board that occupies one slot in the PC running the application being tested (see Fig. 2). This system is called the system under test (SUT). The vscope program is used by the tester to create and perform the actual tests. The Videoscope board provides the links to the keyboard and pointing device (i.e., mouse, tablet, etc.) ports on the SUT. These connections enable the host keyboard and pointing device to be used in place of the SUT keyboard and pointing device during test recording. The Videoscope board is also connected to the video adapter of the SUT, which enables it to capture the video signal of the screen contents of the SUT. The video signal is used to compute a digital representation of the screen. This representation is called a signature, and it is the signature that is stored in the test script.

Although two complete PCs are required for test development, the SUT does not need to have a keyboard or pointing device. For playback, a monitor is optional in the SUT since normally no human will need to look at it. Also for playback, it is possible to use any computer system with appropriate software as the host—not just a PC. This satisfies the portability objective.

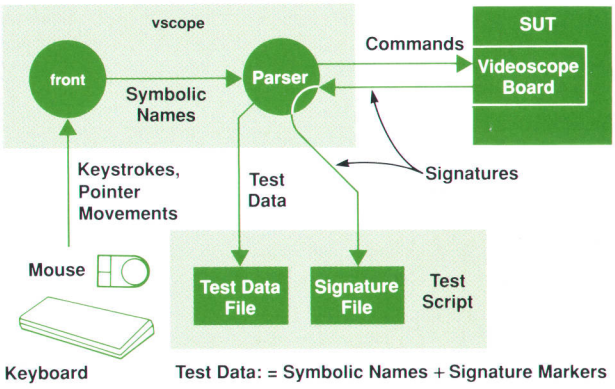


Fig. 3. *Data flow during test script recording.*

Videoscope Software

The vscope program provides the interface between the tester and the recording and playback features of the Videoscope system. For recording the tester uses the keyboard

Microsoft is a registered trademark of Microsoft Corporation.

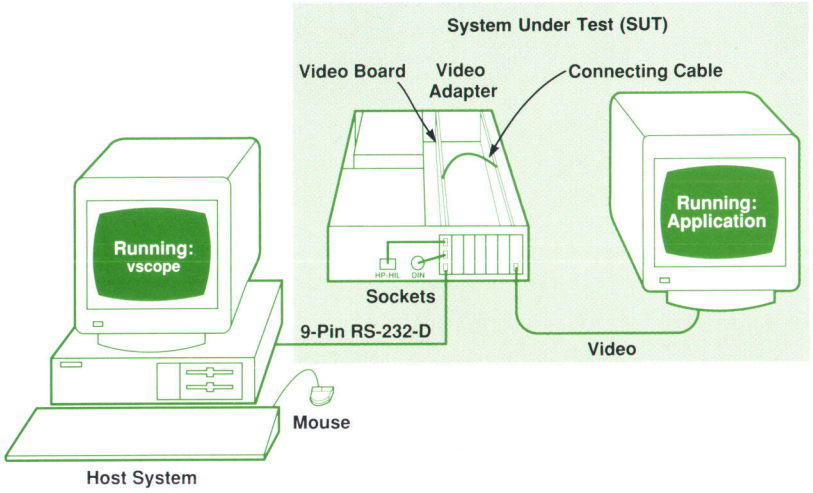


Fig. 2. *Typical setup for using the Videoscope system with an HP Vectra Personal Computer.*

and pointing device on the host and runs the application being tested on the SUT (see Fig. 3). The front routine captures the test data, which is composed of keystrokes from the keyboard and pointing device movements from the HP-HIL[1] (HP Human Interface Link), converts it to symbolic names, and passes it on to the parser. The parser performs two functions with symbolic names: it saves them in the test script, and it translates them into commands for the videoscope board. These commands are transmitted over the RS-232-D line to the videoscope board on the SUT. Periodically, the tester must tell vscope to capture a screen display and save it for comparison during playback. Each time a screen is captured a signature is computed by the code on the videoscope board and passed back to vscope to be saved in the test script. This whole process results in a test script composed of two files. One file (test data) contains keystrokes, pointer device movements and picks, and markers for signatures, and the other file (signature file) contains the screen signatures. Because of blinking fields and cursors on the display, vscope takes several signatures for one screen result. A histogram of the signatures is built and only the most frequent ones are included in a signature list, which goes into the file.

The syntax of the symbolic names and the algorithm to interpret them is based on those commonly used by PC keyboard macro programs.[2,3] The reason for this design decision was to maintain the look and feel of the RTS, and to reuse existing data stuctures and algorithms.

Not all keys are recognized by the front routine. Keys that do not send data to the keyboard buffer (e.g., **CTRL**, **Alt**, **Shift**) are not recognized. These keys, known as "hot keys," are commonly used by terminate and stay resident (TSR) programs such as Superkey or Sidekick to activate themselves. TSRs are so commonly used in PCs that it was decided that vscope had to accommodate their presence on the host and the SUT. This created the problem of how to send such nondata keys to the SUT. Fortunately, the solution to this problem was a natural consequence of the method used to encode and decode the symbolic names created by vscope. For example, the keystrokes **Enter cls Enter dir /w** {cmd}getcrc{cmd} results in a clear screen, a listing of filenames in wide format on the SUT, and the capture of the resulting screen in the signature file. These keystrokes result in the following stream of symbolic names being generated by the front routine:

<ENTER>cls<ENTER>dir /w<ENTER>{cmd}getcrc{cmd}

This stream is interpreted as follows:

- <ENTER> - send press **Enter** key command
- cls - send press **C**, **L**, and **S** key commands
- <ENTER> - send press **Enter** key command
- dir /w - send press **D**, **I**, **R**, Space, /, and **W** key commands
- <ENTER> - send press **Enter** key command
- {cmd}getcrc{cmd} - Execute getcrc command

Pressing a key on the host means sending a press key command to the videoscope board over the RS-232-D line. Under this scheme, an **Enter** key can be inserted in the test script in two ways. The first way is to press the **Enter** key on the host keyboard. The second way is to press the < key, **E** key, **N** key, **T** key, **E** key, **R** key and > key in that order. The pattern <ENTER> will be interpreted as a press **Enter** key entry. Under this scheme, keys that do not generate data for the keyboard buffer can be entered by typing the symbolic name of the key. For example, the hot key combination **CTRL Alt**, which is used to invoke the TSR program Sidekick, can be sent to the SUT by typing on the host keyboard <ctrlalt>. Just pressing **CTRL** and **Alt** simultaneously would invoke Sidekick on the host. With this scheme any key combination can be sent to the SUT and not cause disruption on the host.

The pattern {cmd} is used to mark the beginning and end of a vscope command. Some vscope commands translate directly into commands used by the Videoscope board firmware, and other commands are used only by vscope. For example, the vscope set command translates directly into a Videoscope firmware command to set switches on the board. On the other hand, the vscope log command, which writes information to a file on the host system, has no association with the Videoscope board. Other commands translate into a complex series of operations. The getcrc command is such an example. During test script recording this command retrieves the screen signature and stores it in the signature file. During test script playback, it reads the signatures from the test script file, compares them with the current screen signature and reports the results.

Keystrokes and mouse movements are sent as quickly as possible to the application. A method is provided for slowing them down to a fixed maximum (time command). Associated with each HP-HIL transaction is a delay which can be set from 0 to 32,767 milliseconds. The effect of this delay is to limit the maximum speed at which mouse movements are sent to the application. Many mouse-oriented applications can lose mouse movements if they come too fast. Normally this is of no concern when the user is part of a closed feedback loop and can reposition the mouse. Videoscope is not tolerant of any differences on the display that would be caused by missing a mouse movement. By experimentally changing the delay value, the test can be run at the maximum speed that gives consistent results. Keystrokes can be programmed with separate press and release delays. Each of these delays can be specified in increments of 32 milliseconds over the range of 32 to 8,160 milliseconds.This gives a maximum typing speed of about 180 words per minute. Allowing these fixed and varying wait times to be inserted between keystrokes provides a method for modeling user think times for performance measurements.

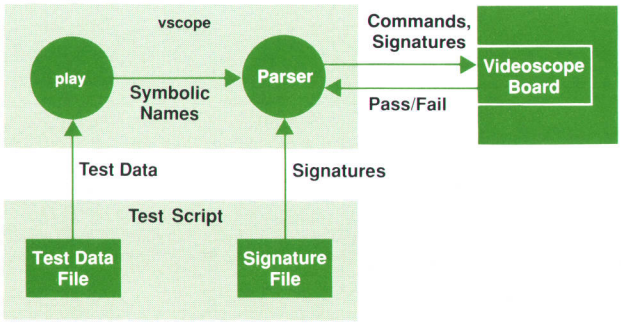


Fig. 4. *Data flow during test script playback.*

For playback mode the tester runs the vscope program, selects the playback option, and specifies the test script files to use. In vscope the play routine shown in Fig. 4 reads the symbolic names from the test script and sends them to the parser. This time the parser does not create another test data file but just translates the data stream and sends it to the SUT. Whenever a signature marker is encountered in the test data file, the associated signature list is retrieved from the signature file and passed to the Videoscope board. The Videoscope board will compare the current screen signatures with signatures passed from vscope and send back a pass or fail indication depending on the outcome of the comparison. If a test fails, vscope will either log the result and continue testing or halt further testing. This decision is based on the options specified by the tester when vscope is set up for playback.

In addition to test recording and playback, vscope provides another operating mode called the replay or regeneration mode. Screen signatures are highly dependent on the video system in use. Even though the display may look exactly the same, signatures from an HP multimode card and a monochrome card are different. If a test developed using a multimode card needs to be played back on a monochrome card (e.g., to test whether the software properly supports the monochrome card), a new set of signatures for the monochrome card needs to be captured. The replay mode automates this process by playing back the test data file and replacing the old signatures with new signatures instead of comparing them as it would in a normal playback. A single test data file can access various signature files, allowing it to be used with several video hardware configurations.

Videoscope Board

The Videoscope board is partitioned into two major sections: the Videoscope processor and the system under test interface (see Fig. 5). The two sections operate independently and are connected by an 8-bit bidirectional port. The processor contains the video signature analyzer (VSA) and the keyboard/HP-HIL emulator. The Videoscope board is a full-length PC/AT-style accessory board (it can be used in a PC/XT-size machine if the cover is left off). During normal operation the board derives power from the +5Vdc, −12Vdc, and +12Vdc lines of the SUT backplane. The PC/AT extended backplane connector is used only to access the additional interrupt lines.

Videoscope Processor. The Videoscope processor is based on an Intel 80188 microprocessor. This microprocessor was chosen because of its low cost and high level of integration, and the fact that it uses the same language development tools as the Intel 80286. The 80188 contains built-in timers, DMA controllers, an interrupt controller, and peripheral select logic. It eliminates the need for at least two 40-pin packages and several smaller-scale chips.

The processor system is equipped with 32K bytes of ROM and 8K bytes of RAM. Connected as peripheral devices are a UART for datacom, two HP-HIL slave link controllers (SLC) for implementing two HP-HIL interfaces, the video signature analyzer, several switches, an LED indicator register, a port to emulate the DIN keyboard, and the SUT interface. The slave link controllers are HP proprietary chips for implementing the HP-HIL protocol.

The Videoscope processor firmware is written entirely in Intel 80188 assembly language. It is modular and all main routines are reached through a jump table in RAM. A loader function is provided so that a user can write a custom module and download it into the processor RAM. The jump table can be overwritten so that the downloaded module is executed instead of the ROM-resident code. There is a command (normally a null operation) that can be used as the entry to a user module. The firmware is structured as a real-time interrupt-driven system. The code normally sits in an idle loop until a command needs to be processed or an interrupt serviced. Some of the command processing routines themselves introduce new interrupt service routines for their operation.

Communication with the host system is through the RS-232-D interface shown in Fig 5. The firmware supports up to 9600 baud using a straightforward command-response protocol with a simple DC1 handshake. The ACK/ENQ protocol used on the HP 3000 is also supported. All data transfers between the host and Videoscope are in ASCII hexadec-

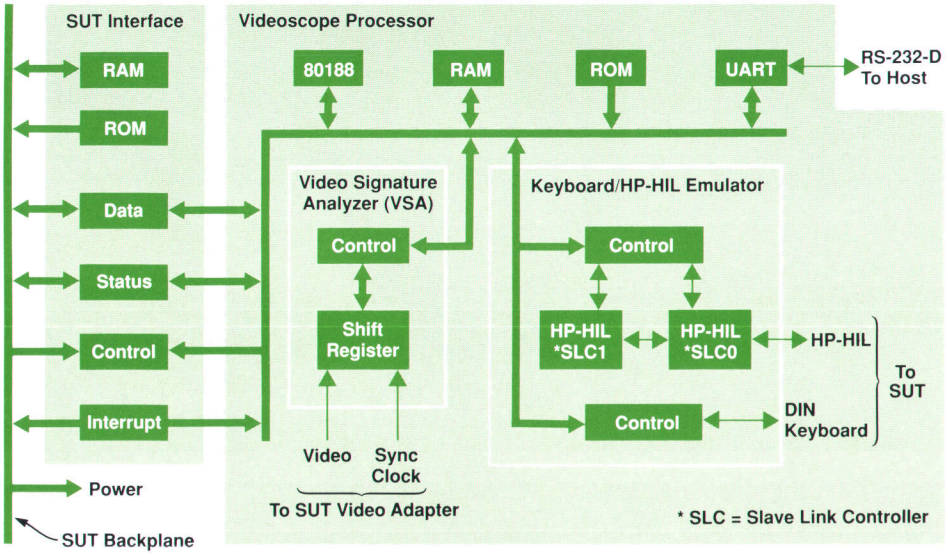


Fig. 5. *Block diagram of the Videoscope board.*

Video Signature Analyzer Operation

Videoscope uses a method of generating signatures based on the HP 5004A Digital Signature Analyzer used for troubleshooting digital systems. The HP 5004A uses a 16-bit linear feedback register to generate a pseudorandom number. A signal under consideration in a properly operating system is combined with this number over a fixed period of time (a specific number of clock cycles) to modify it into a unique representation for that signal. The unique signature is recorded on the schematic diagram or in a troubleshooting guide. This is done for every node in the system. If a malfunction develops, signatures taken from the system can be compared to those recorded when it was operating properly and the fault can be isolated.

The Videoscope signature generator operates in the same way, but it is implemented differently. The heart of the signature generator is a linear feedback register built from three 8-bit shift registers (Fig. 1). To get the best resolution with the minimum number of parts, a length of 22 bits was chosen. This allows the register to operate with $2^{22}-1$ states. The other two bits in the register are not in the feedback path and cause the total number of states to be multiplied by four.

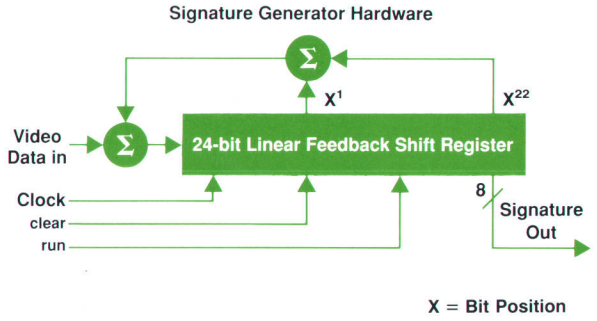


Fig. 1. *The heart of the signature generator is a linear feedback register built from three 8-bit shift registers.*

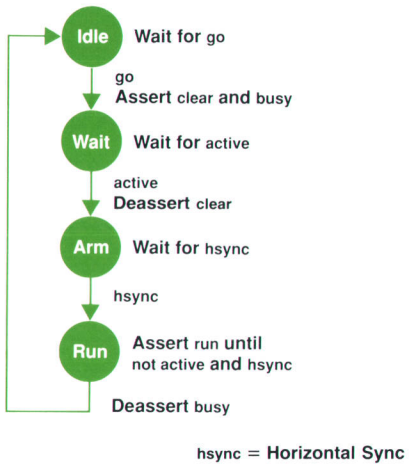


Fig. 2. *Hardware state machine.*

The shift register is controlled by a simple hardware state machine (Fig. 2) which has a resolution of one dot clock. The hardware state machine is controlled by another state machine implemented in firmware, which has a resolution of one scan line. The firmware state machine (Fig. 3) is a set of interrupt service routines (ISRs). The transition from state to state is by an interrupt from either the vertical sync signal (vsync) or a counter reading of zero.

The main code (Fig. 4) starts the signaturing process by resetting the hardware state machine, setting the initial vertical sync interrupt service routine state 0, sending a go pulse to the hardware state machine, and entering a wait loop. While in this loop, other interrupts, such as HP-HIL, can be serviced.

The sequence of states depends on whether the starting line on the display is greater than 0% and the stopping line is less

imal characters. This was chosen to allow complete independence from the host datacom. It also enables test scripts be stored in readable and editable form. No compilation or decompilation of the scripts is necessary. All commands and data from the host parser routine to the Videoscope processor are of the form:

*⟨command⟩⟨length⟩⟨data specific to command⟩⟨checksum⟩CR DC1

with all commands beginning with the marker * and all data characters, including the ⟨command⟩ and ⟨length⟩ fields included in the checksum. The commands recognized by the firmware include the following:

- Set Attribute (*A). Allows the host software to change default settings.
- Load Siglist (*C). Implements downloading of signature lists for screen matching.
- HP-HIL (*H). Sends an HP-HIL device X, Y, and button data frame over the HP-HIL interface.
- Include (*I). Sets the start and stop limits of the area of the screen to include in a signature. Used to avoid time variant areas of the screen.
- Keystroke (*J *K). Sends a keystroke keycode and shift modifier. The *J form uses default timing while the *K form allows explicit press and release times.
- Load (*L). Implements downloading of code routines.
- Resend (*R). Resends the last data record in case of datacom error.
- Signature (*S). Takes a signature of the screen. Used for building the signature file for later playback.
- Test and Report (*T). Provides dumps of various sets of variables.
- Wait for Match (*W). Compares screen signatures until either the downloaded list is matched or a time-out occurs.

Responses to the host are of the form:

⟨+ −⟩⟨optional data⟩CR LF

where the + indicates successful completion of the command and − indicates failure. The optional data varies by command. For successful completion the field may contain actual data. If it does, it is in a format similar to a command, including a length and a checksum. In the case of a failure, an error code, followed by an optional verbose error message (enabled by a switch), is reported.

Video Signature Analyzer. The VSA is the key component

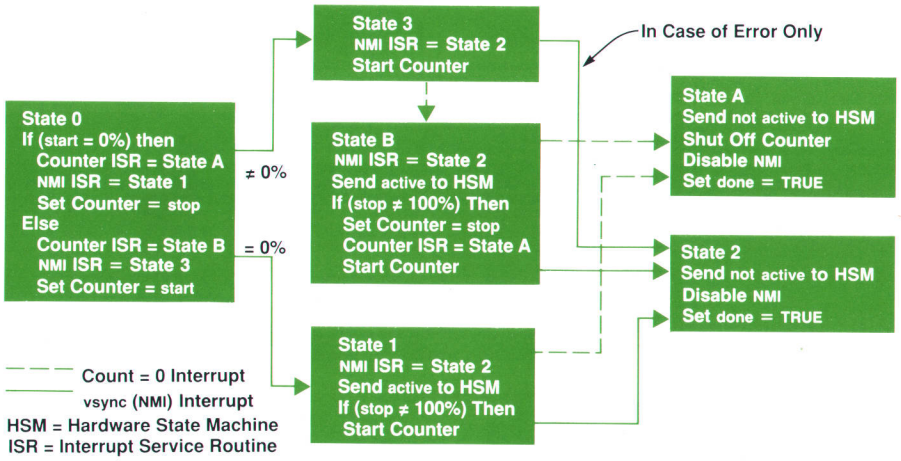


Fig. 3. *Firmware state machine.*

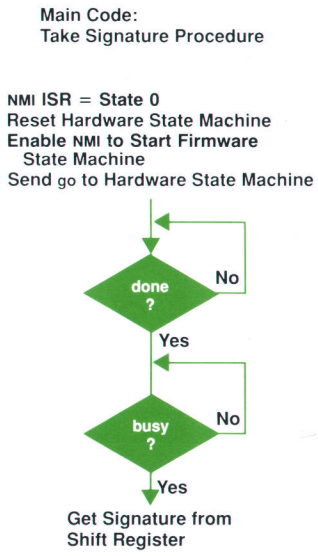


Fig. 4. *Program flow for the main code in the video signature analyzer.*

of the Videoscope concept. By using the technique pioneered by the HP 5004A Digital Signature Analyzer,[4] it is possible to monitor the video signals generated by the SUT's display adapter in real time in a totally nonintrusive manner. The main component of the signature analyzer is a 24-bit linear feedback shift register. The linear feedback shift register is used to accumulate a signature (similar to a cyclic redundancy check) of the video data stream. The signature is a 6-digit hexadecimal number that describes the state of the screen. The linear feedback shift register is a pseudorandom number generator driven by the video signal. This means that even a one-pixel difference will change the signature. A state machine using the display's horizontal and vertical sync signals controls when the signature is taken. Since some applications put time variant data on the screen such as dates, a clock, or file names and paths, a method is provided to allow the signature to be started and stopped by a count of scan lines after the start of the display. In this way only the nonvariant portion of the screen will be included in the signature.

To accommodate the various display adapters used in a PC, the video signature analyzer has an eight-input multiplexer which can select from eight separate video streams. This allows exhaustive testing of all image planes in a multiplane adapter (e.g., EGA or VGA) and minimizes testing when the different planes contain redundant information. The tester can use the vscope set command to control how many planes are signatured. A separate signature is computed for each plane, and when doing a multiplane match to a signature list, all enabled planes must match or a failure is reported.

To reduce the part count while maintaining reasonable precision and speed, the linear feedback shift register is wired as 22-bit maximum length with an additional two

than 100%. If the signature is to include the entire screen, the firmware state machine is started and stopped by the vertical sync interrupt and the state sequence is 0-1-2. If the start line is not at 0% then state 3 is entered and a counter is loaded with the proper number of lines to skip. When the counter reaches 0, state B is entered and the signature started. If the stop line is not 100%, either state 1 or state B will set up the counter to interrupt at the end of the desired area. The final states, 2 and A, shut off the hardware, disable any further vsync or counter interrupts, and signal the main routine via the done flag. The main routine then sits in a loop (busy) waiting for the hardware state machine to finish and then reads the signature in three 8-bit pieces.

As a fail-safe mechanism, another timer runs while the signature is being computed. If this timer expires before the signature is reported as done, an error is assumed, the entire process shuts down, and an error message is issued. Several escape paths are included in the firmware state machine to ensure that it won't go into a lockup state.

bits following. This provides a maximum pattern length of $4\times(2^{22}-1)$ states. Although a typical display has many more states than this (a 640×350-pixel display has over 2^{224000} distinct states), the probability of having an incorrect display with a valid signature is extremely low. In instances where a match of an invalid screen (an alias signature) does occur, the next screen signatured will almost certainly fail. Very few alias signatures have been detected in actual use.

The VSA is controlled from the Videsoscope processor by a high-speed state machine implemented partially in hardware and partially in firmware. The firmware uses a state machine consisting of an idle loop and several interrupt service routines. The present hardware portion of the state machine will operate with video dot rates in excess of 50 MHz and scan rates in excess of 32 kHz. See the box on page 62 for more details about the VSA state machine architecture.

At the completion of the signaturing process the firmware can either pass the signature back to the host or compare it against a list of valid signatures downloaded from the host. The first method is generally used during script creation when the vscope program is building the signature list file, and the second method is used during test script playback. Datacom traffic is kept to a minimum by downloading lists of signatures and doing the comparisons locally on the VSA board.

Keyboard/HP-HIL Emulator. Videoscope has the capability of emulating the keyboard used on the earlier Vectra models as well as the industry standard DIN keyboard used in both the PC/XT and PC/AT protocols. The new Vectra models use the DIN keyboard and are compatible with the PC/XT and PC/AT protocols. For the HP-HIL interface, there are two slave controller chips on the board, which are capable of emulating any two of the following devices: a mouse, a tablet, a touchscreen, or a keyboard. The controllers are directly driven by routines in the Videoscope processor firmware and no additional processors are needed. Additional devices can be emulated by changing the firmware.

SUT Interface. The SUT interface port provides a communication path between the Videoscope processor and the SUT processor. This is an 8-bit bidirectional port with provision for generating an interrupt when written to. In the SUT this can either be a nonmaskable interrupt (NMI) or any one of the interrupt (INTRn) lines on the backplane. The desired line is selected by a switch. The SUT must have an interrupt handler installed for this feature to be used. Also included in the SUT interface are 8K bytes of ROM and 8K bytes of RAM. This 16K-byte address space can be configured within the C0000-DFFFF address range of the SUT. The ROM is seen by the SUT power-up routines as an option ROM and normally includes the necessary interrupt handler. The I/O ports of the SUT interface can be located at any of several addresses normally reserved for the IBM prototype card. Both the memory and the I/O ports can be relocated to avoid conflicts with hardware installed in the SUT. The current implementation of the Videoscope system does not make use of the SUT interface. However, the hooks are available for users to create routines for special-purpose testing requirements.

Conclusion

Videoscope has met or exceeded the original objectives established for the system. One minor disappointment is that the the signature generated from the video signal is not unique. However, the probability of two screens having the same signature is very small, so this is a very minor problem and simple workarounds have been found.

The design and implementation of Videoscope was highly leveraged. The video signature analysis hardware and firmware are based on the HP 5004A Signature Analyzer. The data structures and algorithms for interpreting data in the script file are based on those commonly used by keyboard macro programs, and the data communication software used by vscope to communicate with the Videoscope processor firmware is a package called Greenleaf Data Comm Library from Greenleaf Software, Inc.

Videoscope provides a major productivity tool to improve the quality of software. The PC intrinsically is an interactive computer system. This means that batch-style tests cannot adequately test the capability of any software written for a PC. Videoscope provides an automatic alternative to slow, error-prone, and expensive manual testing. Currently there are over 100 Videoscope systems in use at 18 HP divisions.

Acknowledgments

We would like to acknowledge the help of our manager, Jean-Claude Roy, who made a major contribution to the design of Videoscope, and the entire Videoscope design review team, in particular John Schimandle who provided much appreciated user feedback. The development of Videoscope was a corporatewide effort involving many people, and so we thank all of those who contributed to the successful development of Videoscope.

References

1. R.S. Starr, "The Hewlett-Packard Human Interface Link," *Hewlett-Packard Journal*, Vol 38, no. 6, June 1987, pp. 8-12.
2. *Superkey Owner's Handbook* Second Edition, Alpha Software Corporation, 1985.
3. *Keyworks Users Guide*, Alpha Software Corporation.
4. H.J. Nadig, "Signature Analysis—Concepts, Examples, and Guidelines," *Hewlett-Packard Journal*, Vol. 28, no. 9, May 1977, pp 15-21.

Authors

June 1989

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.

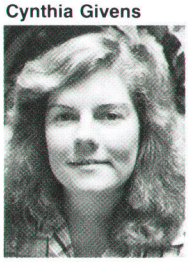
Michael J. Wright

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.

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.

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.

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.

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.

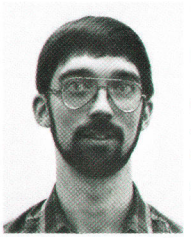
John Keller

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.

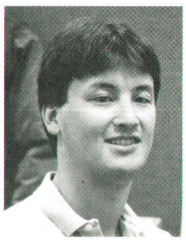
Russell C. Brockmann



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.

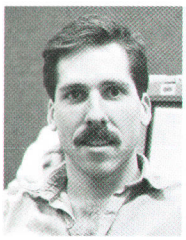
26 Data Compression

Jeffery J. Kato



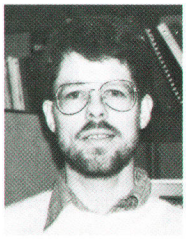
During development of the HP 7980XC Tape Drive, Jeff Kato's contributions focused on the architecture and design implementation for the data compression chip and firmware design. He has also designed read electonics for the HP 7978A Tape Drive and the PLL and PLL IC for the HP 7980A Tape Drive. He came 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

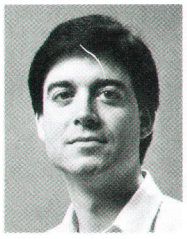
Author's biography appears elsewhere in this section.

David J. Van Maren

Author's biography appears elsewhere in this section.

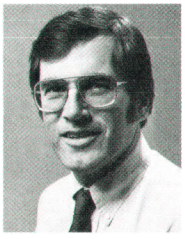
35 Lightwave Component Analysis

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

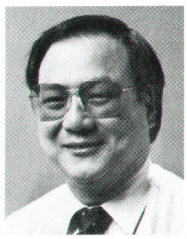


Paul Hernday is an R&D project manager in HP's Network Measurements Division in Santa Rosa, California. With HP since 1969, he has been involved with new-product developments in sweepers, scalar and vector network analyzers, and lightwave component analyzers. His most recent project has been the development 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.

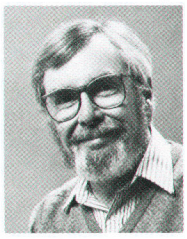
Roger W. Wong



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.

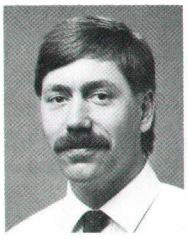
52 Lightwave Sources and Receivers

Kenneth W. Shaughnessy



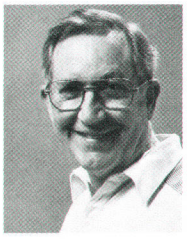
The HP 8753A and the HP 8754A Vector Network Analyzers and a number of lightwave instruments are among the major projects to which Ken Shaughnessy has contributed design ideas. On the HP 8702A Lightwave Component Analyzer System, he worked as a product designer. He joined the Santa Rosa (California) 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



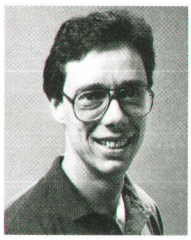
In development of the HP 8702A Lightwave Component Analyzer, Kent Leyde's design work concentrated on microcircuits and optics for the lightwave receivers. As a development engineer, he has since started work on the signal acquisition and processing elements of a new product. Kent's BSEE degree (1984) 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

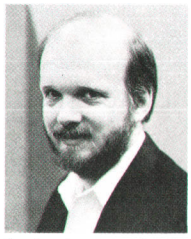
Before working on the hardware and firmware design for the Videoscope tool, Myron Tuttle's responsibilities included development of the HP 45981A Multimode Video Adapter and the HP 2625 and HP 2628 Terminals. He joined the Advanced Products Division of HP in 1974 and is a member of both the Association 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.

Neural Data Structures: Programming with Neurons

Networks of neurons can quickly find good solutions to many optimization problems. Looking at such problems in terms of certain neural data structures makes programming neural networks natural and intuitive.

by J. Barry Shackleford

A FEW YEARS AGO at HP Laboratories we were privileged to have John J. Hopfield, professor with the Divisions of Chemistry and Biology at the California Institute of Technology, give us a talk on computing with neurons. The lecture was fascinating. I was particularly excited by the fact that networks of neurons could quickly find good solutions to optimization problems like the traveling salesman problem. I had once written a computer program to route the interconnections on a computer circuit prototyping board (Fig. 1). Thus, I was painfully aware that simply proceeding to the closest unconnected point can lead to disasters when the objective is to minimize the total length of wire connecting a set of terminals.

At the end of the talk I still could not see how to program these networks. Being an engineer, I wanted to ask, "How do you determine the resistor values and interconnections for the traveling salesman problem?" However, I was reluctant to ask such a prosaic question in light of the arcane questions I was hearing concerning differential equations and energy surfaces.

I assumed that I could get the answer from my colleagues, so the question went unasked. The answer was not forthcoming, however. No one that I asked had come away with the insight of how to construct such a network.

After a week of intensive study of some of Hopfield's published work with HP Laboratories colleague Thomas Malzbender, progress was made. Tom's mathematical insight and my desire for some type of higher-level representation of what was happening at the differential equation level produced the approach presented here. I knew I was on the right track when a week later, I solved the eight queens problem using the neural data structure models that had emerged the week before.

Introduction

By constructing networks of extremely simple nonlinear summation devices—now termed "neurons"—complex optimization problems can be solved in what amounts to be a few neural time constants. This is a collective, consensus-based style of computing where a different but probably equally good answer may result the second time the problem is solved. This is a style of computing for areas where good (i.e., within a few percent of optimum) solutions must suffice—where there is seldom time to wait for the "best" answer.

In many cases, the data itself serves to determine the architecture of the neural network required for a given problem. Other cases require the problem to be mapped onto structures that may not be obvious at first sight. By looking at problems in terms of certain neural data structures, we may find neural programming to be quite natural and intuitive.

To develop an intuition for programming with neurons, a conceptual model is needed. This model has three layers. The innermost layer is the Hopfield neuron. Changing the properties of the neuron has a global effect on the problem. The second layer is composed of elemental data structures suited to the properties of neurons. The third layer is the method by which the gap between the data structure and the problem statement is bridged. It can be explained and observed but, like programming in conventional computer languages, it is best practiced.

Hopfield Neurons

Forming the core of our conceptual model is the Hopfield neuron (Figs. 2 and 3). Simply stated, it is a nonlinear

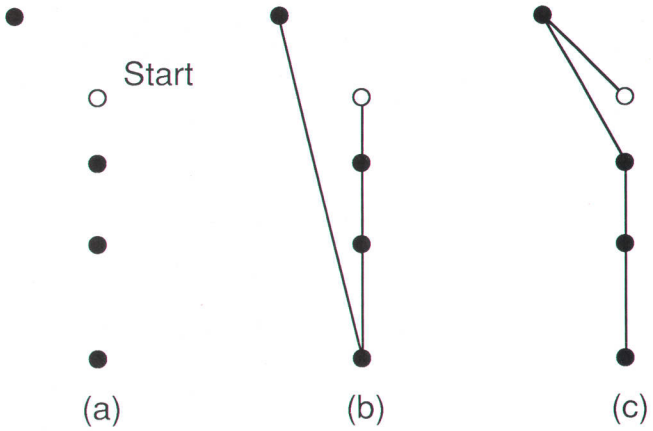


Fig. 1. *(a) A problem similar to the traveling salesman problem—five terminals to be connected together in the shortest possible distance with a single strand of wire. (b) The simple heuristic approach of proceeding to the closest unconnected node can often yield poor results. (c) The optimum result needs to consider all the data at once.*

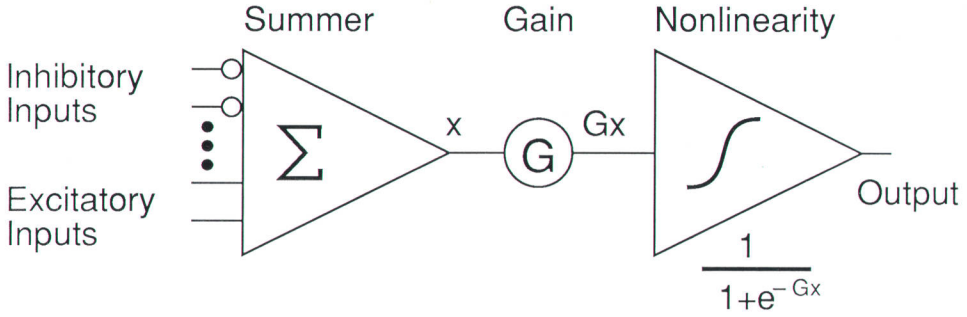


Fig. 2. *Conceptual circuit model for a Hopfield neuron. The number of either excitatory or inhibitory inputs is unconstrained, as is the positive or negative output swing of the summer. The gain element multiplies the summer's output by a constant and then feeds it to a nonlinear element where it is compressed between 0 and 1.*

summing device. We can view the Hopfield-type neuron as being divided into three sections.

The first section does an algebraic summation of the input signals. These can be both excitatory and inhibitory. The excitatory inputs are summed directly and the inhibitory inputs are first inverted (i.e., multiplied by -1) before summation. Let us call the output of the summation element x.

The output of the summation element is then sent to the second section, a gain element, where it is multiplied by a constant G. High values of G make the neurons very sensitive to small variations of x around zero at the cost of reducing the compliancy of the entire system. A very high gain causes the neuron to behave as an analog/digital circuit known as a comparator. A comparator has an output of 1 if the sum of its inputs is the least bit positive. Otherwise its output is 0. In a network composed of comparators there is no in-between, no compliancy; the network loses its ability to compromise. On the other hand, if the gain is too low, all of the neurons will be floating somewhere near the same value. Like a large party with all the guests talking at the same level, no one conversation can be distinguished from the others.

The third section is the output stage, which performs a nonlinear transformation on the signal Gx. The relation:

$$\frac{1}{(1 + e^{-Gx})}$$

provides a symmetric sigmoid (i.e., S-shaped) transfer function (Fig. 4). This type of curve is used in photographic films and signal compression systems where a balance must be struck between fidelity and wide dynamic range. The curve ranges from 0 for large negative values of Gx to 1 for large positive values of Gx. When Gx is zero, the output is one half. The nonlinear characteristic of the neuron effectively gives a network of neurons a wider dynamic range and a higher degree of compliancy.

N-Flops

The n-flop (Fig. 5) represents a natural first stage of organization for neurons. We can say "natural" because it is easy to construct, and once constructed, it is robust. Being robust, it can serve as a base for other organizational structures. There is precedence in nature for n-flops in the form of neural cells exhibiting *lateral inhibition* (defined later).

An n-flop is a local aggregate of n neurons programmed by their interconnections to solve the constraint that only one of the n will be active when the system is in equilibrium. The term n-flop is derived from flip-flop, a computer circuit that has only two stable states. An n-flop has n stable states.

Two independent 6-flops would behave much the same as a pair of dice used in games of chance—the final state of each should be a random number from 1 to 6. However, dice can be "loaded" by moving the center of gravity away from the geometric center. The same can be done to a 6-flop

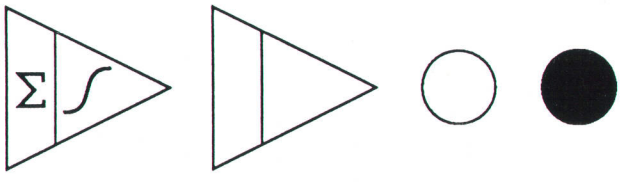


Fig. 3. *Various neuron symbols—the circle being the most abstract. Often a circle will be shaded to indicate that a neuron is active. The diameter of the circle can also be varied to show the relative activity of the neuron.*

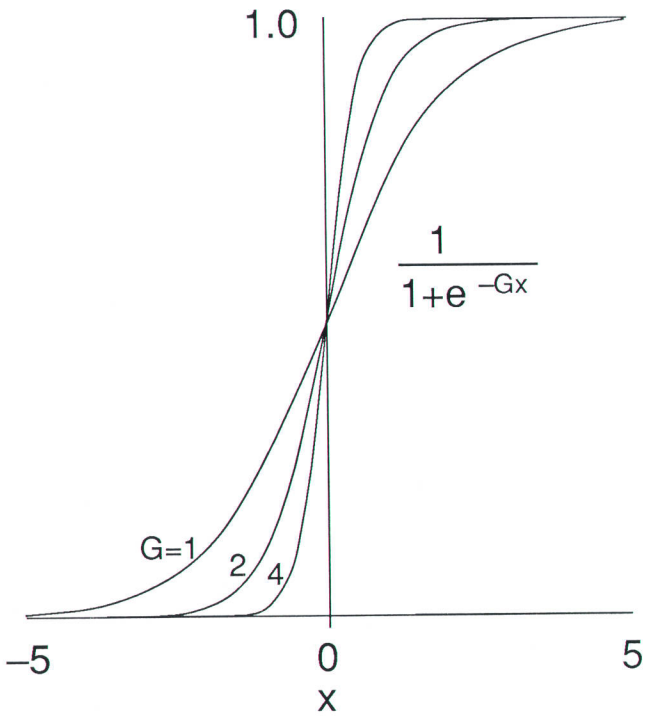


Fig. 4. *Higher gain values sharpen the sigmoid curve and thus reduce the compliancy of the system. At very high gains the shape of the curve approaches that of a step function.*

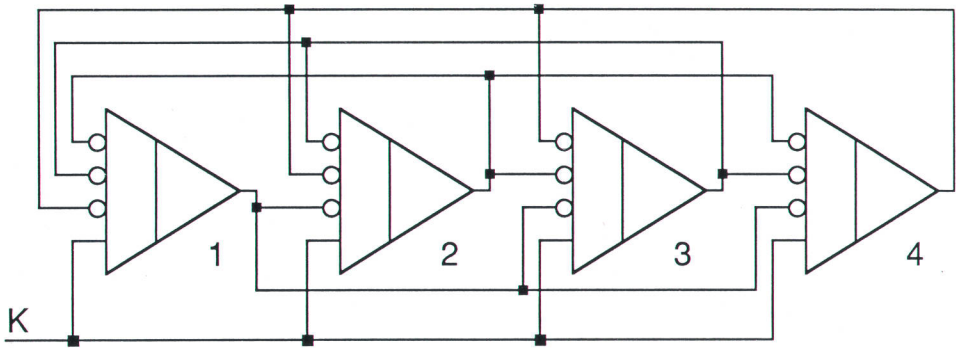


Fig. 5. *Each neuron in the n-flop strives to suppress the others. Eventually, one wins out. K supplies the energy that the winning neuron will use to suppress the others.*

by applying a small excitation to one of the neurons. There is an even greater potential here; the two 6-flops can be interlinked so that they always produce the same sum, say 7. Let the number 1 neuron of the first 6-flop serve as an excitatory input for the number 6 neuron of the second 6-flop. Number 2 goes to number 5 and so on. Then the second 6-flop can be wired back to the first in a similar manner. The stronger the bias weights (i.e., excitations), the higher the probability that the total outcome will be 7.

The interconnections for an n-flop are arranged such that the output of each neuron is connected to an inhibitory input of each of the other $n-1$ neurons (Figs. 5 and 6). We will borrow a term from the biologists and refer to this kind of connection as *lateral inhibition*. Additionally, there is a fixed excitatory input, K, to each of the neurons. Because K is excitatory, it will tend to drive each neuron's output towards 1.

Like a pencil balanced on its point, an n-flop also has an unstable equilibrium state. In this state all of the neurons are balanced somewhere between 0 and 1. Starting from this unstable equilibrium state, an n-flop composed of physical neurons would eventually become unbalanced because of random thermal noise. During this process one neuron would begin to predominate (Fig. 7).

This action would force the other neurons towards zero. This in turn would lessen their inhibitory effect on the predominant neuron, allowing the excitatory energy of the K input to drive its output more towards 1.

Simulations with n-flops have shown that the lower bound on G for a 4-flop is about 4. For a 20-flop it is about 5 or 6, depending upon K.

Acceptable values of K for a 4-flop range from about 0.75 to 1.25. For a 20-flop, the range is roughly 1.25 to 1.75.

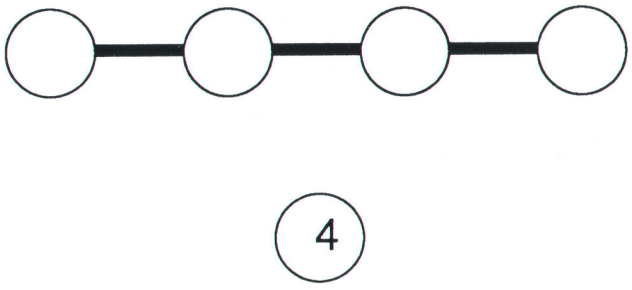


Fig. 6. *The heavy line connecting the four neurons represents their mutual inhibitory connections. A circle containing the number of states presents a more compact, abstract view.*

Too much energy from the K input will allow two or more neurons to predominate above the others, producing an m-of-n-flop.

The above values are for n-flops that rely solely on lateral inhibition to produce the 1-of-n final state. There is an additional constraint that could be applied to ensure that the summation of all the outputs will be close to some value; 1 would be a good choice for the n-flop. Hopfield called this constraint *global inhibition*. The global inhibition constraint is added to the n-flop by first summing all of the outputs of the n neurons. From this sum the desired total value is subtracted. The final value is then sent to an inhibitory input of each neuron in the n-flop. For example, if all of the neurons in an n-flop were initialized to 0, then a value of -1 would be applied to an inhibitory input of each neuron. Applying -1 to an inhibitory input is the same as applying $+1$ to an excitatory input. The effect is to drive the outputs up towards 1 so that the lateral inhibition factor can then take over to ensure a 1-of-n state.

The global inhibition mechanism seems to be somewhat analogous to an automatic gain control. When in place, it allows the n-flop to operate over a wider variation of parameters (Fig. 8), thereby avoiding the m-of-n-flop problem.

It's easy to connect neurons into a 1-of-n structure and then make the aggregate structure work reliably. We might even say that this represents a natural form for neurons. Neurons connected in this manner represent a more ordered condition. Their stable equilibrium state represents

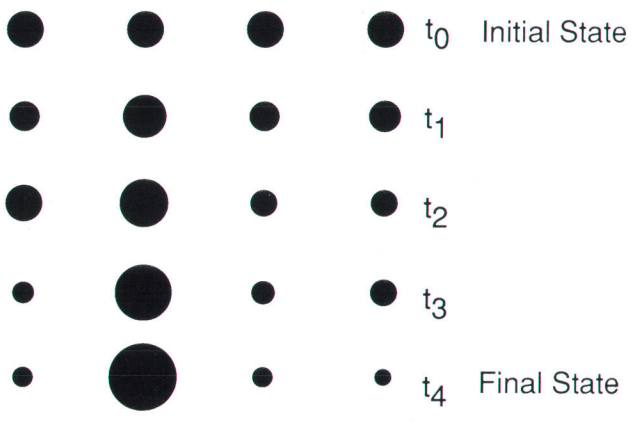


Fig. 7. *Snapshots in time of a 4-flop proceeding from an initial unstable equilibrium state to a final stable equilibrium state. An n-flop with no external biasing inputs will achieve a random final state.*

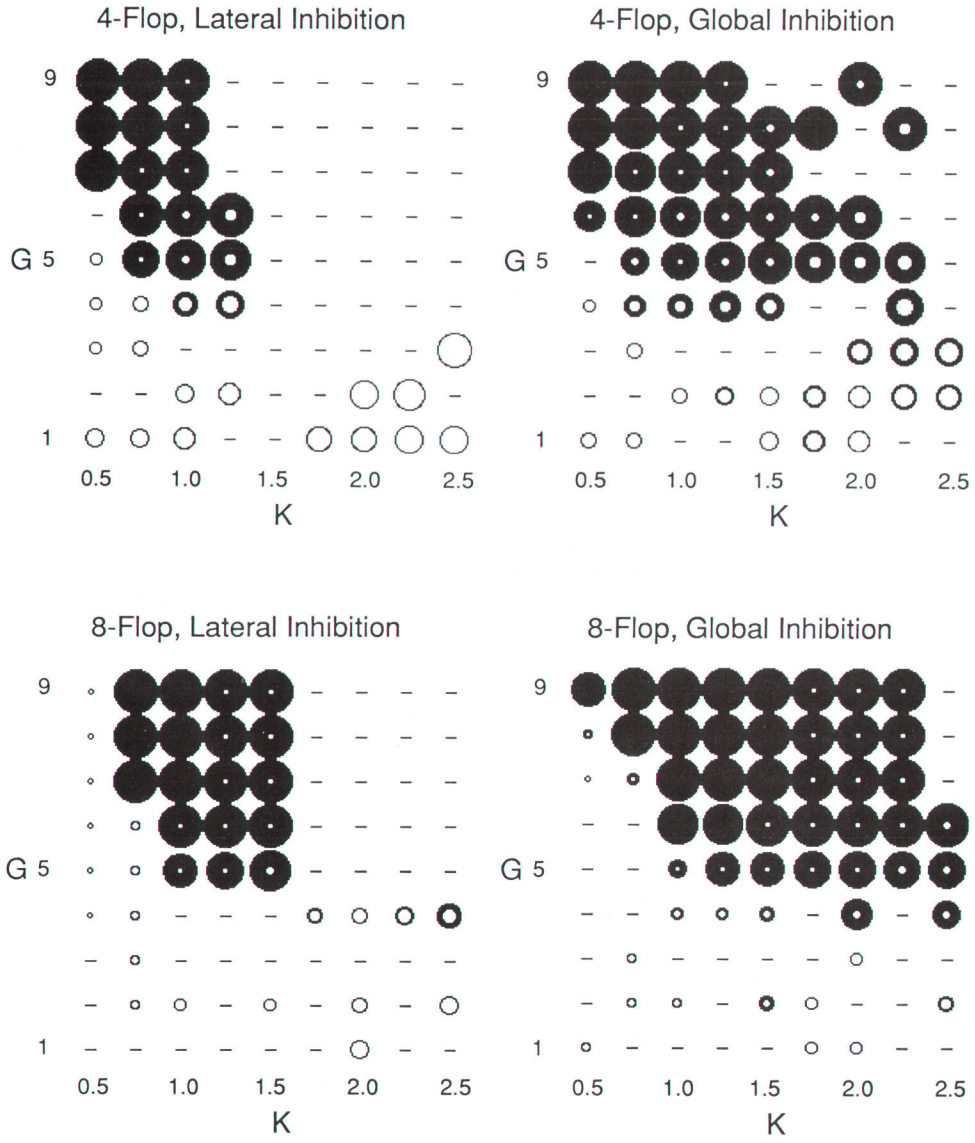


Fig. 8. *Parameter sensitivity studies for a 4-flop and an 8-flop. The two charts on the left show the operational region with lateral inhibition only. On the right, both lateral and global inhibition are used to ensure the n-flop's 1-of-n characteristic. The outer dark circle represents the relative strength of the predominant neuron. The inner white circle represents the average strength of the nonpredominant neurons. Dashes represent combinations of G and K that did not result in a valid 1-of-n state.*

the resolution of the constraint: "Pick only one of these n." This 1-of-n condition can be thought of as a data structure—a means to represent the answer to an optimization problem. For example, in the four-color map problem, a syntactically correct answer is that each country is colored only one of four colors. A qualitatively correct answer requires additionally that each country be a different color from each of its neighbors.

N×N-Flops

Beyond n-flops there is an even higher degree of order that can be achieved. By taking n^2 neurons and arranging them into a matrix of n rows and n columns, a new, more highly constrained data structure can be formed. We could call this an n-by-n-flop (Fig. 9).

By applying the n-flop connection scheme to both the rows and the columns of neurons, we obtain a structure that will allow only one neuron to be active in each row and only one in each column. This type of data structure has n! states (Fig. 10).

The number of connections increases as n^3. Each neuron in the matrix is connected to the inhibitory inputs of the other $n-1$ neurons in its column and similarly to the $n-1$ other neurons of its row. Thus, for this type of network there are $2(n-1)n^2$ or $2(n^3-n^2)$ connections. For a 32×32-flop there are 63,488 connections.

Programming with Neurons

Problems best suited for networks of Hopfield neurons are those of optimization and constraint resolution. Additionally, the solution to the problem should be easily represented by a data structure that is compatible with neural structures.

The first step is to construct an array that represents a syntactically correct solution to the problem. Often the same underlying structure will be found to be common to a number of problems. For example, the n-by-n-flop data structure is used to represent a syntactically correct answer to the traveling salesman problem.

The next step is to provide biases to the individual neurons that represent the characteristics of the problem. The biases can be either constants or functions of other neurons and serve either as inhibitory or excitatory stimuli.

The solution realization can be thought of as a dynamic

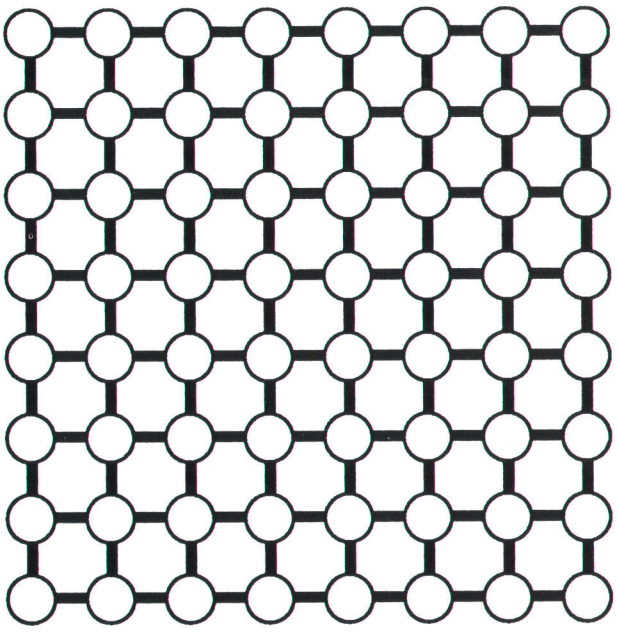


Fig. 9. *The lines between neurons represent n-flop inhibitory wiring. Only one neuron can be active in each row and one in each column.*

system achieving its lowest energy state. For example, a 2-flop might be visualized as a ball balanced on a hill between two valleys. When released from its unstable equilibrium state and subjected to small random perturbations it will settle in either valley with a 50 percent probability. Applying a bias will alter the probability of achieving a given final state.

The Eight Queens Problem

The eight queens problem is one of simply satisfying constraints. The queen is the most powerful chess piece, being able to move the entire length of any row, column, or diagonal that it might occupy. The challenge is to place eight queens on a standard eight-by-eight chessboard such that no queen is attacking any other queen, that is, no two queens occupy the same row, column, or diagonal (Fig. 11).

To solve this problem with artificial neurons, we might consider the neural data structures defined earlier. The n-by-n-flop has the row/column exclusivity that is needed by the problem. An 8-by-8-flop by itself has solved two of the three constraints of the problem. All that remains is to provide diagonal inhibitory interconnections to each

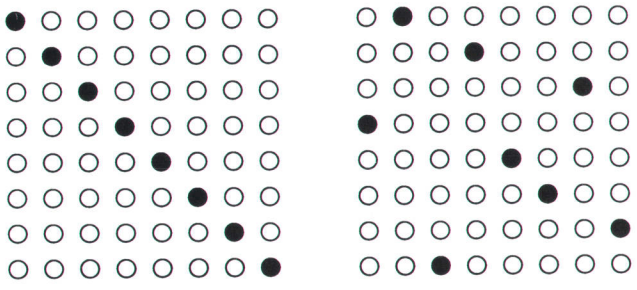


Fig. 10. *Two of the 8! possible states for an 8-by-8-flop.*

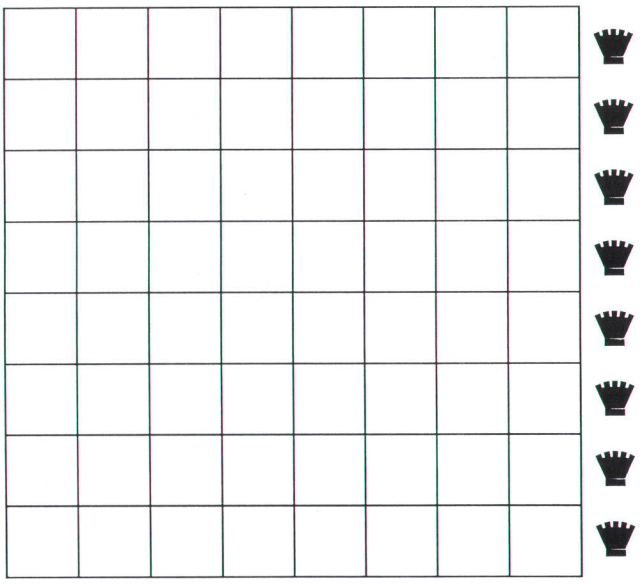


Fig. 11. *The eight queens problem—place the eight queens on the board so that none is under threat of attack from any other.*

neuron in the 8-by-8 array much the same as the row/column inhibitory connections. So now, each of the 64 neurons (each representing a position on the chessboard) has the constraints specified in the problem applied to its inputs (Fig. 12).

A solution is obtained by initializing all neurons to some near-equilibrium (but unstable, "high-energy") state and then releasing the network to seek a stable, "low-energy" state (Fig. 13). In practice, the network will sometimes become mired in a local minimum and only be able to place seven of the eight queens, much the same as its biological counterparts. This susceptibility to local minima may stem from the fact that all of the neurons are no longer identically programmed. The neurons in the center of the board are fed with 27 inhibitory inputs where those on the edges are fed with only 21. Perhaps lessening the gain of the center neurons in proportion to their additional inputs would tend to "flatten out" the local minima on the solution surface. Another strategy to avoid local minima is to

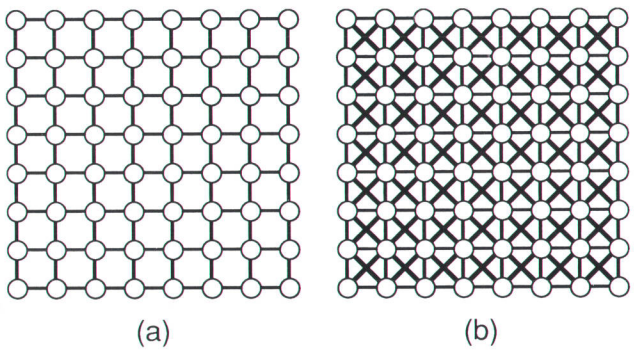


Fig. 12. *The row and column constraints are managed by the underlying n-by-n-flop data structure (a). Adding diagonal inhibition completes the specification of constraints for the eight queens problem (b).*

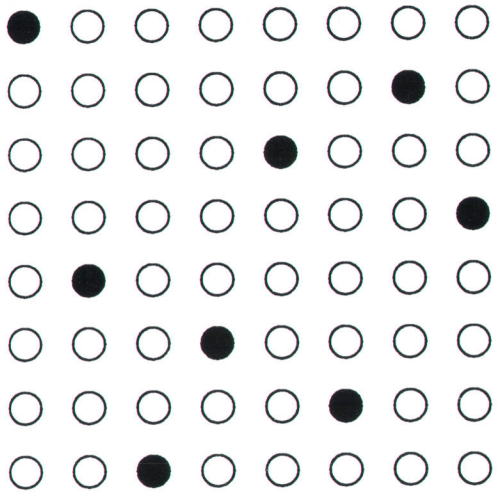


Fig. 13. *One of 92 possible solutions to the eight queens problem. The neurons that have outputs close to 1 are shown as shaded circles. These represent queen placements.*

increase the gain of all of the neurons slowly from a low value to a point that is adequate to allow a solution to emerge. This technique has been used by Hopfield to achieve superior solutions to the traveling salesman problem. Starting at a low gain seems to allow a complex network to initialize itself close to its unstable equilibrium point.

Egon E. Loebner of Hewlett-Packard Laboratories has pointed out that there is often a "knight's jump" relationship between queen positions in solutions to the eight queens problem. Egon has suggested that each neuron in the 8-by-8 array be allowed to apply an excitatory stimulus to its knight's-jump neighborhood. So, instead of being totally specified by constraints, there would be a mixture of both inhibitory constraints and excitatory stimuli.

The Four-Color Map Problem

In the eight queens problem it was possible to construct an analog of a chessboard with neurons—one neuron per board position. With a problem such as the four-color map problem, a similar (but perhaps not so obvious) analog will be used.

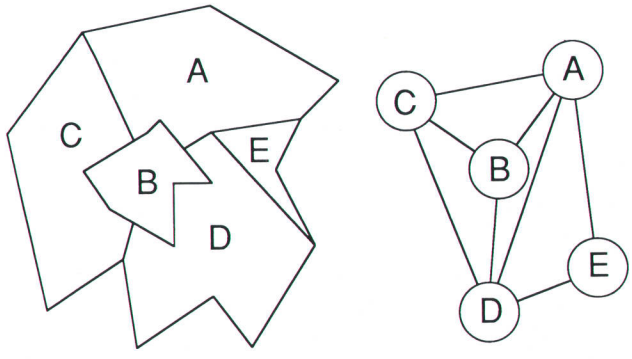


Fig. 14. *On the left is a generic tourist map. On the right is the equivalent graph. The circles (vertices) represent the countries and the lines connecting them (edges) represent the adjacencies.*

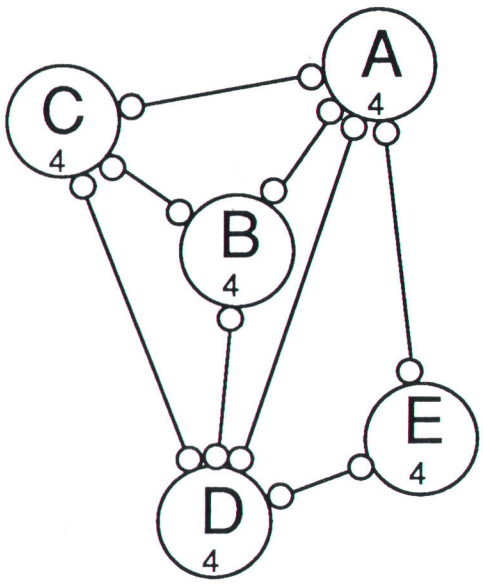


Fig. 15. *The four-color map problem as a network of 4-flops. The tiny circles represent inhibitory inputs. The lines between nodes represent eight connections (four going each way).*

The four-color map theorem states that at most four colors are required to color any map that can be drawn on a flat surface or the surface of a sphere.[1] There are some special cases. A map drawn on a flat surface with straight lines that begin and end at edges will require only two colors. A map that is drawn on a Möbius strip may require up to six colors. One colored on a torus may require up to seven.

The first step is to transform the problem from the form so familiar to all of us—an outline map—to a form more familiar to mathematicians—a graph (Fig. 14). In this case, the graph does not take the form of the stock market's daily highs and lows but resembles more a subway map. In our graph, the nodes will represent the countries and the lines connecting the nodes will represent common borders between countries. Mathematicians refer to the nodes as vertices and the connecting lines as edges.

The next step is to place at each node (representing each country) a 4-flop that will indicate which one of four colors the country is to be (Fig. 15). The 4-flop satisfies the first constraint of the problem: use only one of four colors for each node/country.

The second constraint states that adjoining countries be

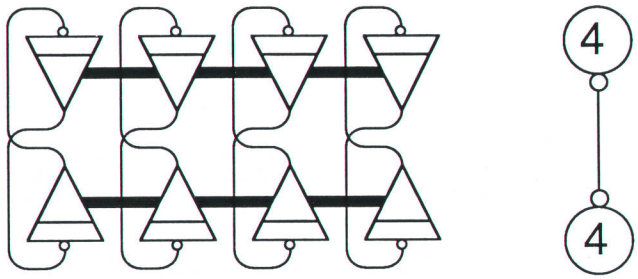


Fig. 16. *Mutual inhibitory connections between two 4-flops produce a network in which each 4-flop will have a different state. At the right is the high-level notation.*

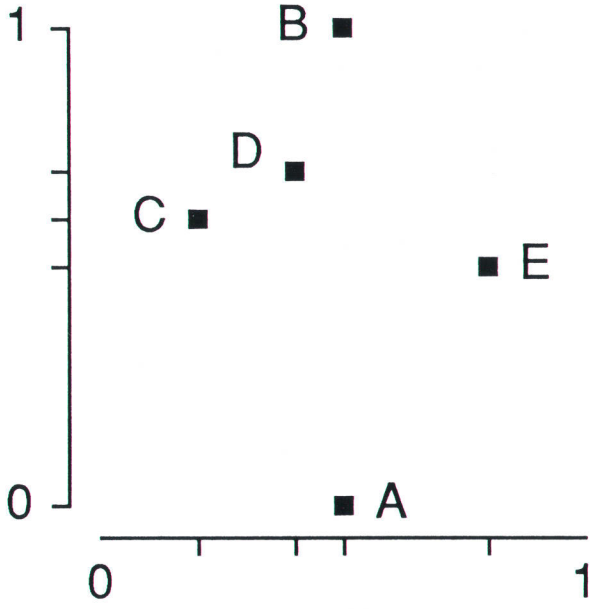


Fig. 17. *The traveling salesman problem—link all of the cities in the shortest possible distance.*

of different colors—i.e. connected nodes must be in different states. In the case of two 4-flops connected together, this mutual exclusivity can be obtained by connecting the output of each neuron to an inhibitory input on its counterpart in the other 4-flop. Thus each line on the graph actually represents eight inhibitory connections, four going each way (Fig. 16).

The solution is obtained as in the eight queens problem. Place all of the neurons at some intermediate level and then let them go.

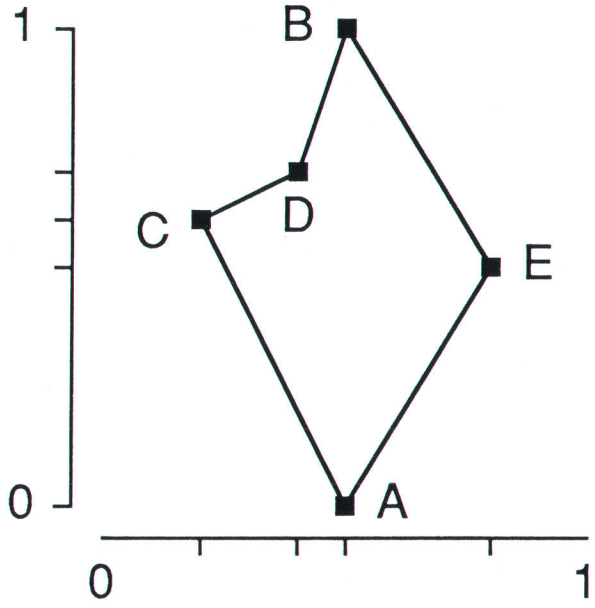


Fig. 18. *A given path is denoted by a list that indicates the order of visits. The list CAEBD is interpreted as: starting at C, first visit A, then E, B, and D, and then return to C. Every path belongs to a set of 2n equivalent paths, where n is the number of cities on the path.*

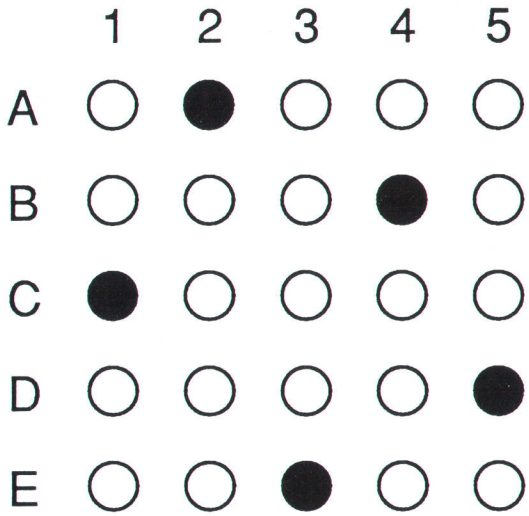


Fig. 19. *The state of the n-by-n-flop (shown by the shaded circles) indicates the path CAEBD. The cities are represented by the rows of neurons and the order of visits is indicated by the columns. The n-by-n-flop serves as the base data structure for many optimization problems. Its characteristic resting state allows only n neurons to be on—one per row and one per column.*

The Traveling Salesman Problem

The traveling salesman problem is perhaps the most famous of all optimization problems. The problem provides us with n cities, each at some distance from the others (Fig. 17). The objective is to visit each of the n cities once (and only once) and then return to the starting city while at the same time minimizing the total distance traveled.

The traveling salesman problem is classified by mathematicians as being *np-complete*, which means that the time required to find the optimal path on a conventional computer will grow exponentially as the number of cities increases. The number of solutions is n! (similar to the n-by-n-flop). The number of distinct solutions is only somewhat less. Each distinct path (Fig. 18) has n versions based upon which city is the origin. Another factor of two stems from the freedom to start out in either direction from the city of origin. Thus there are n!/2n distinct closed paths for the traveling salesman problem.

For a dozen cities there are 12!/24 = 19,958,400 distinct paths. A modern personal computer could probably search them all during a coffee break. However, a problem that is not even three times larger (32 cities) contains 32!/64 = 4.11×10^{33} paths. A multiprocessing supercomputer searching a billion paths a second would require more than 10^{17} years for a complete search.

Again, the first step is to find a representation for the problem solution that is compatible with a network of neurons, probably a graph or a matrix. As it turns out, the traveling salesman problem can be neatly represented with an n-by-n matrix of neurons (Fig. 19). The rows represent the n cities and the columns (labeled 1 through n) represent the stops along the path. The characteristics of the n-by-n-flop (only one active neuron per row and one per column)

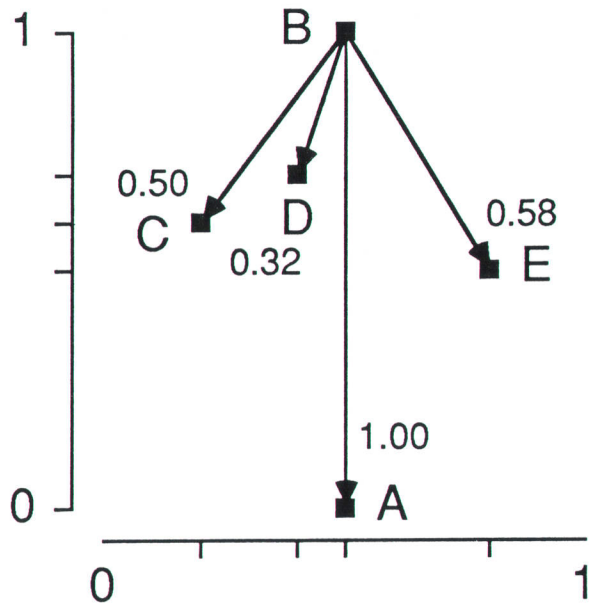


Fig. 20. *Distances between city B and every other city normalized to city pair AB, the most widely separated city pair on the tour.*

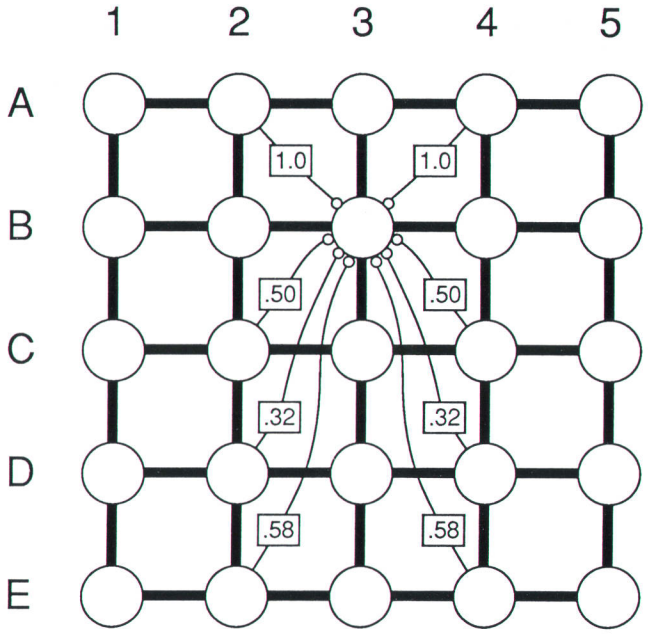


Fig. 21. *The n-by-n-flop provides a syntactically correct solution. Adding inputs to each neuron in the form of graded mutual inhibits between cities serves to improve the quality of the answer. Shown are additional inputs—in effect, the programming inputs—for a single neuron, B3.*

ensure a syntactically correct solution—all cities visited (one per column) and each city visited only once (one per row).

Before proceeding, a path-length minimization heuristic must be selected. A fairly intuitive choice is when at any given city, proceed to the closest nonvisited city. Applying this heuristic sequentially can lead to results far from optimum. However, application to all neurons simultaneously tends to give results close to optimum.

To program a particular version of an n-city traveling salesman problem, we need to know the distance between every city and every other city. These distances will be used to form an additional set of inhibitory constraints. Let the longest distance between any two cities be considered 1.0, then normalize all of the other distances with respect to it (Fig. 20). Thus, the most widely separated city pair has a mutual inhibition factor of 1.0, and other city pairs have weaker mutual inhibitions proportional to their closeness. Each neuron is then connected to the $n-1$ neurons in the column to its left and to the $n-1$ neurons in the column to its right. The strength of the inhibit signal is modulated by the mutual inhibition factor for the particular city pair (Fig. 21). The extreme left and right columns are considered to be adjacent.

Let us take the viewpoint of a single neuron (first recalling that the columns represent the order of the path and that the rows represent the cities). We are sitting in one of the columns and we see a column of neurons to the left, each representing a city that's a possible stop before coming to us. A similar column lies to the right. We can think of one side as the "coming-from" column and the other side as the "going-to" column, although the direction that we perceive as going or coming is not really significant. Now, let's assume that we are the strongest neuron in both our row and our column. In addition to sending out row and column inhibits (to ensure a syntactically valid answer), we will send out inhibits of varying strengths to both the coming-from and going-to columns adjacent to us. The strongest inhibits go to those cities farthest from us, and the weakest inhibits go to the closest cities. The strength of the inhibit is simply the normalized distance fraction times the output of our neuron.

Again, as in the eight queens problem, a solution is obtained by forcing all of the neurons to some arbitrary initial

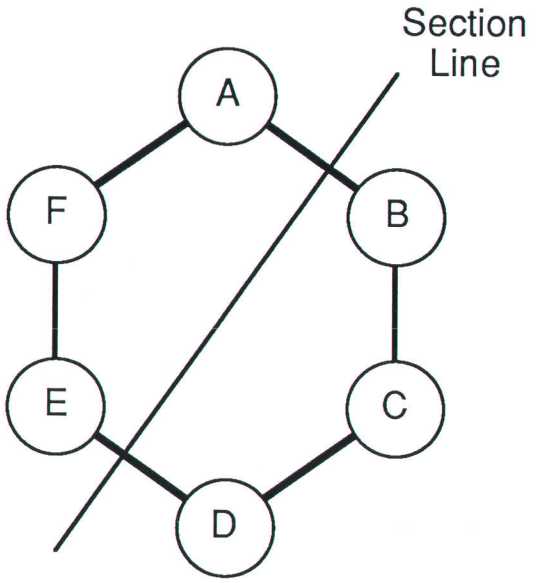


Fig. 22. *An example of the graph sectioning problem using a simple ring network. For $n = 2$, the problem is to bisect the network such that each partition has an equal number of nodes and a minimum number of line crossings.*

value. This represents an unstable condition—one that the preprogrammed constraints inherent in the n-by-n-flop will attempt to rectify. As the neurons collectively "fall away" from their initial values towards values that will satisfy the constraints of the n-by-n-flop data structure, the biasing network (representing the distances between all of the cities) will attempt to "pull" the network towards a final state that has the lowest overall residual error. The solution will generally not be a perfect one but it will probably be a good one.

The Graph Sectioning Problem

At first, the graph sectioning problem sounds somewhat conjectured—take an arbitrary graph and partition the nodes into n sections. The constraints are that each section should have an equal (or near equal) number of nodes. Additionally, the nodes should be assigned to the sections in such a way as to minimize the node connection lines crossing between sections (Fig. 22).

Consider the logic circuitry of a digital computer. The individual logic circuits and their interconnections can be abstracted as a huge graph. The logic circuits form the nodes (or more properly, the vertices) and the input and output connections form the graph edges. A typical computer logic circuit might have three or four inputs and its output might be connected to half a dozen or more other logic circuits.

Currently, research is being performed on automatically synthesizing the logic circuitry of an entire computer. The problem then becomes how to divide 100,000 logic circuits into chips, each of which has a finite area for logic circuitry and a limited number of input/output pins to accommodate connections to other chips.

Like the four-color map problem, the graph sectioning problem will require an n-flop at each node of the graph. The number of sections is represented by n. It is interesting to note how well-prepared the problem is to solve. Simply replace each logic circuit by an n-flop and use the existing interconnections to connect the n-flops. This connection will be an excitatory connection. Once a node is assigned a section, assigning all connected nodes to the same section will tend to minimize the connections crossing between sections.

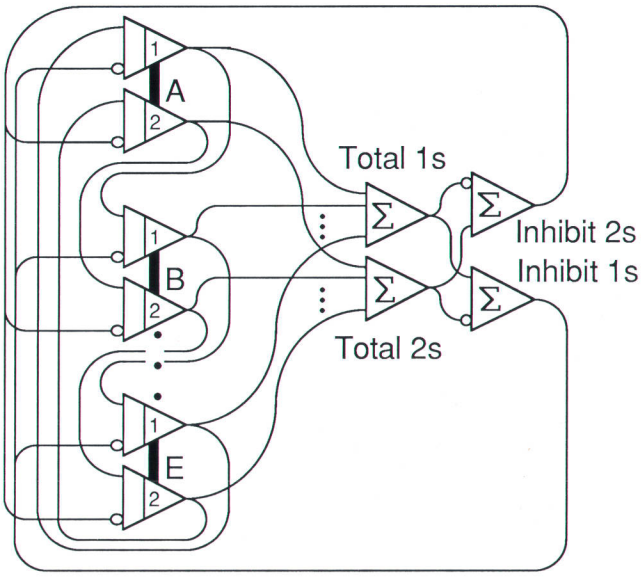


Fig. 23. *Excitatory connections try to keep adjacent nodes in the same section. Global balance constraints keep all nodes from being put into the same section.*

To ensure even distribution among the sections, a global constraint term is required. As the solution emerges, the sum of nodes assigned to each section is monitored. If an imbalance occurs, an excitatory bias is applied to encourage assignment to the deficient sections. At the same time an inhibitory bias is applied to discourage assignment to a section that has a surplus (Fig. 23).

The solution is obtained much the same as in the other problems. First the neurons are placed at an arbitrary initial value and then the network is released from its initial constrained state. Immediately, the underlying data structure constraints will seek a global compromise with the biasing (i.e., programming) constraints. The result will generally be a good solution but probably not a perfect one.

Why can't perfect solutions be readily obtained? Consider the following rationalization. A structure such as an n-flop is a very stable entity when it is in its final equilibrium state; the energy required to change its state is considerable. On the other hand, just after the network is released from its initial state, it is very susceptible to the effects of the biasing network. This is because the effect of the data structure constraints does not become evident until an imbalance begins to occur.

At the point of initial release the output of the biasing network is at its strongest. Its effect will be to "steer" the network towards a state that will produce a lesser error. As the network nears a state that tends to satisfy the biasing constraints, the total energy available from the biasing network will diminish. At some point, the effect of the underlying data structure will begin to predominate and subsequently pull the network to its final equilibrium state. So, at the final equilibrium state there will probably still be some small correctional signal from the biasing network (representing a less than perfect solution) but its strength will be small compared to that required to shift the network from its resting state.

Summary

Simple networks of nonlinear summing devices have demonstrated the collective property of being able to resolve elementary constraints. By viewing these networks as neural data structures, more complex networks can be easily conceptualized. These highly interconnected networks are able to find near-optimal solutions to difficult np-complete optimization problems such as the traveling salesman problem.

The method of matching a solution network to a problem is twofold. First, a network must be realized that yields a syntactically correct answer. Then additional constraints or programming biases relating to the problem are added to the network. These additional inputs serve to select qualitatively good answers from the set of syntactically correct ones.

Reference

1. K. Appel and W. Haken, "The Solution of the Four-Color-Map Problem," *Scientific American*, Vol. 237, October 1977, pp. 108-121.

Bibliography

1. J.J. Hopfield and D.W. Tank, "'Neural' Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 52, 1985, pp. 141-152.
2. J.J. Hopfield and D.W. Tank, "Computing with Neural Circuits: A Model," *Science*, Vol. 233, no. 4764, August 8, 1986, pp. 625-633.
3. D.W. Tank and J.J. Hopfield, "Collective Computation in Neuronlike Circuits," *Scientific American*, Vol. 257, no. 6, December 1987, pp. 104-114.
4. C. Peterson and J.R. Anderson, *Neural Networks and NP-Complete Optimization Problems: A Performance Study on the Graph Bisection Problem*, MCC Technical Report No. EI-287-87, December 14, 1987.

CORRECTION

In the April 1989 issue, Fig. 1 on page 66 should have an edge from node d to node e. The correct figure is shown here.

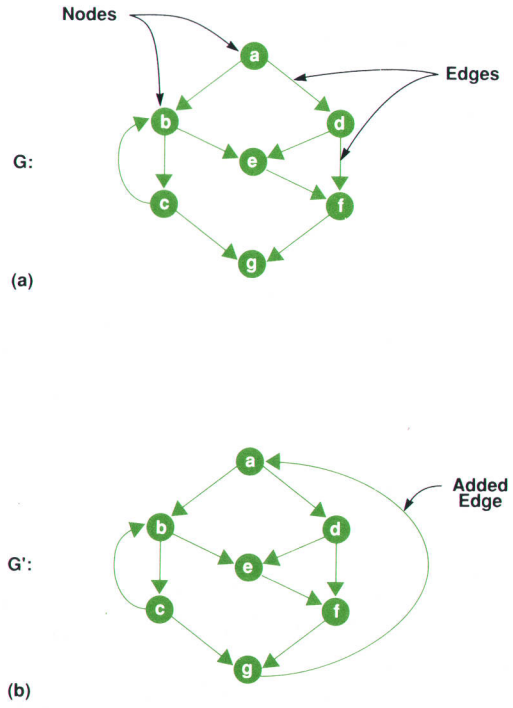


Fig. 1. *a) Program flow graph for a program with seven nodes (blocks of code) and ten edges (branches). b) Same control graph with added edge to satisfy the requirement that the graph must be strongly connected.*

A New 2D Simulation Model of Electromigration

Electromigration in miniature IC interconnect lines is simulated in HP's sophisticated two-dimensional model, giving new quantitative and graphical insights into one of the most important metallization failure sources for VLSI chips.

by Paul J. Marcoux, Paul P. Merchant, Vladimir Naroditsky, and Wulf D. Rehder

WHEN THIN METAL FILMS, such as the interconnect lines of integrated circuits, are stressed by high current densities, a slow migration of atoms is induced, which for aluminum and its alloys proceeds in the direction from cathode to anode. It can be inferred from transmission electron microscopy (TEM), scanning electron micrographs (SEM), and diffusion studies that these atoms travel predominantly along grain boundaries. If structural inhomogeneities develop in the conductor, for example at material interfaces or at triple points or at grain size divergences, then the current-induced atom flow is nonuniform. As a consequence, there exists a nonzero divergence between the incoming and the outgoing flux at these locations, so that material piles up and vacancies form. While such accumulation of material (sometimes called hillocks or, in special cases, whiskers) may short adjacent conductors, the vacancies, on the other hand, will deteriorate the line until voids have coalesced sufficiently to form cracks that eventually lead to electrical opens.

Over the past 20 years, this complex phenomenon, known as electromigration, has become a subject of increasing concern for the entire chip industry because of its deleterious effect on IC reliability. It is especially troublesome now, in light of the continuing shrinkage of IC dimensions below the one-micrometer level.

Hundreds of papers have been written about electromigration and special task forces have been established involving the main chip makers worldwide, but a detailed theoretical understanding of this phenomenon is still in its early stages. Two main approaches have evolved. The first and earlier method saw researchers test their partial theories by deriving analytical formulas from plausible physical assumptions, with computational results that can be tested against the substantial body of empirical data. Two of the most prominent analytic expressions are Huntington's formula[1] for the atomic flux (see equation 7 below), and Black's semiempirical equation (see equation 8 below) for the median time to failure.[2]

The second approach starts from basic physical principles and established general laws such as the continuity equation and diffusion laws, and uses these to drive simulation models built to mimic the dynamic sequence of events in an interconnect line through a time-dependent Monte Carlo method. Early simulation models are those of Attardo[3] and Nikawa.[4]

This paper gives an outline of a new 2D simulation model for electromigration, which is the result of a four-year collaboration of HP's Integrated Circuits Laboratory and the Center for Applied Mathematics and Computer Science at California State University at San Jose.

Classical Methods Used

Some quantum mechanical approaches purport to be able to treat field and current effects and the so-called direct and wind forces in a self-consistent manner. However, the practical gain of these more intricate models over classical models appears limited, for two reasons. First, the quantum theoretical formulas often coincide, at least for the special and manageable cases of interest, with the classical expressions (see, for example equation 25 in reference 5). Second, there are still fundamental conceptual difficulties as to the

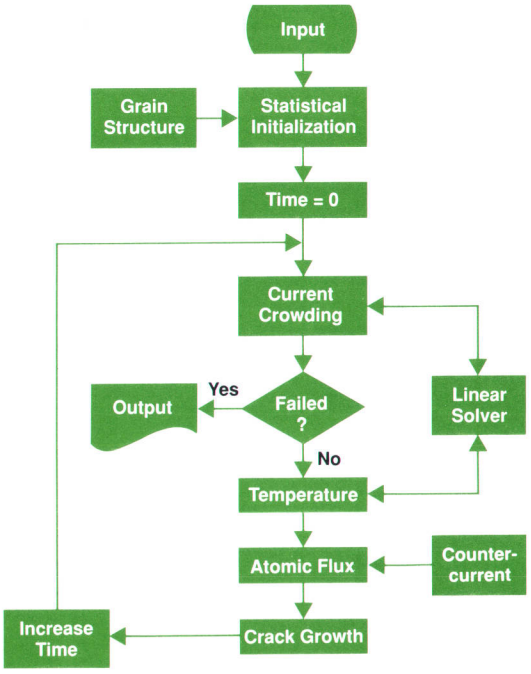


Fig. 1. *Flowchart of the basic algorithm for the simulation program.*

nature of these underlying forces at the sites of impurities (see the controversy reported in reference 6 and the recent paper by Verbruggen[7]). Faced with these theoretical difficulties, but also encouraged by the considerable data validation of Huntington's classical formula and restrained by considerations of simplicity and computer adaptability (codeability), our team took a pragmatic approach and kept all model assumptions, all mathematical techniques, and all underlying physical principles entirely classical, neglecting quantum effects. This limitation led us to the adoption of the following model hypotheses about the two forces that are ultimately responsible for electromigration in metals such as aluminum.

The electrostatic direct force caused by the applied external field acts on the metal ions. The wind force associated with the electron current that accompanies the electric field, which is sometimes called electron drag or electron friction, is a consequence of the momentum transfer from the electrons to the ions during collisions. In aluminum alloys, this latter scattering term dominates the electrostatic force. Hence the resulting migration is towards the anode.

A major feature of the new HP model is that only one generic type of equation, the potential equation, describes both the heat development (Helmholtz equation) and the current flow (Laplace equation). The potential equation is a two-dimensional partial differential equation, allowing continuous or discontinuous step functions as coefficients. A finite element method in two dimensions applied to this potential equation renders a large system of linear equations with a sparse coefficient matrix for both cases. A Fortran subroutine was written to solve this system for temperature and current.[8]

By means of another crucial model building block, the continuity equation, we keep track of the material movement resulting from the inhomogeneities mentioned above. Material flow in polycrystalline films at temperatures below about two thirds of the melting temperature of the metal occurs predominantly through grain boundaries. Thus, we consider material flow only through the grain boundary network.

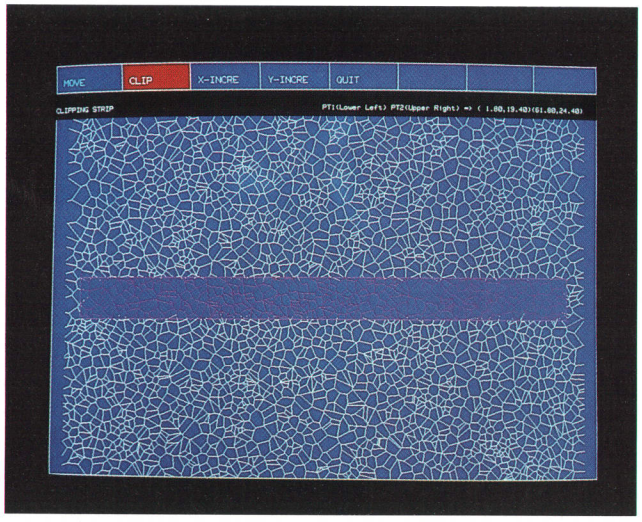


Fig. 2. *Screen showing the interactive clipping process, in which individual lines that will be subjected to simulated stress are cut from a large area. This operation mimics the lithography and etching steps in the formation of real test structures.*

Model Overview

The flowchart of the algorithm is shown in Fig. 1.

Grain Structure Generation. First, using a Monte Carlo technique, we generate a two-dimensional geometrical pattern that simulates the grain structure of a thin metal film by a plane dissection method called Voronoi tesselation. A Voronoi tesselation in two dimensions is a partition of the plane into cells bounded by polygons. Cells are defined by seeds, which are randomly distributed using a Poisson distribution, and the bounding polygon is the set of points that has equal distance from the cell's seed as well as from the seeds of neighboring cells. These cells then represent a carpet of metal grains from which we then clip our metal interconnect line (Fig. 2).

Our package calculates the following statistical characteristics of the grain structure thus simulated:

- The area distribution of the grain sizes

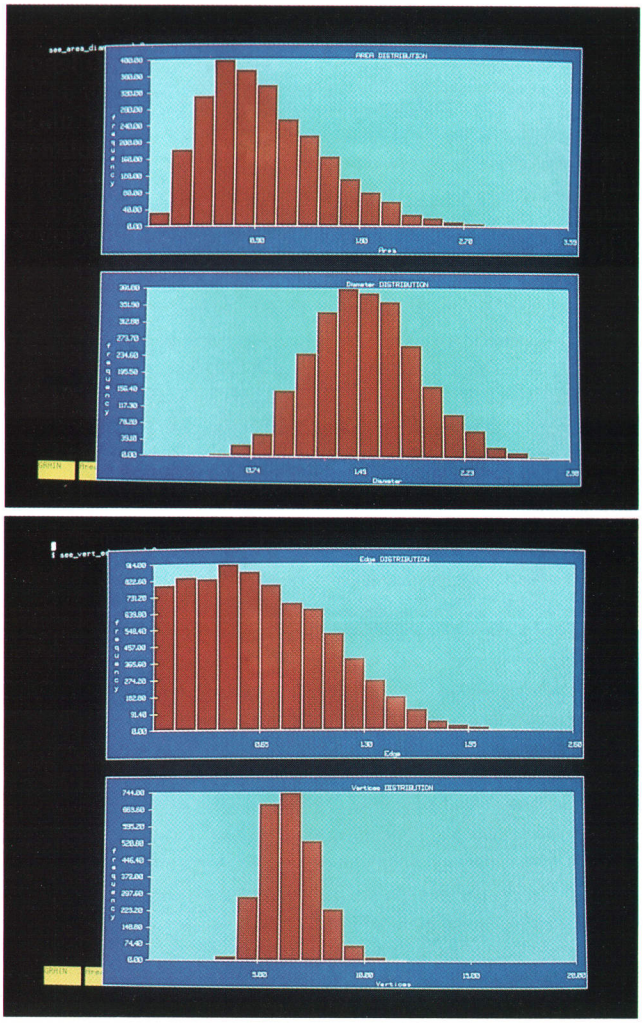


Fig. 3. *(a) Distributions of grain areas (top) and average diameters (bottom). (b) Distributions of lengths of grain boundary segments (top) and the number of vertices per grain (bottom).*

- The diameter distribution of the grains
- The distribution of segment length
- The number of triple points
- The number of vertices.

Figures 3a and 3b show typical histograms for areas, diameters, segment length, and number of vertices for the grain structure shown in Fig. 2. These distributions are characteristic of real deposited films. Thus, our model is useful in studying the correlation between failure distributions and deposited film grain structures.

Current Flow. The main advantage of the HP model over others like Nikawa's[4] is that both the current flow and the heat equation are truly two-dimensional (as is the generation of the grain structure just described). The steady-state behavior of the current flow is described by the Laplace equation

$$(ku_x)_x + (ku_y)_y = 0, \tag{1}$$

where simple discontinuities in the electrical conductivity $k = k(x,y)$ are allowed. The value $k = 0$ is assigned to those cells in the finite element grid where no current is being conducted because of cracks. In electrically active cells, on the other hand, the function k assigns a constant value $K > 0$.

Once the Fortran solver subroutine has solved this Laplace equation for the potential $u = u(x,y)$, we obtain a discrete approximation for the current density j as follows. In the defining equation for j,

$$j = -\,k\,\mathrm{grad}(u) \tag{2}$$

substitute for grad(u) the finite differences of the first order,

$$u(i+1,j) - u(i,j)$$

and (3)

$$u(i,j+1) - u(i,j).$$

Temperature Distribution. Current flow in the interconnects leads to Joule heating, which is dissipated to the surrounding environment. In our model we consider the heat flow through a two-dimensional rectangle in the finite element grid and derive the following partial differential equation for the temperature function $T = T(x,y)$ in the metal line:

$$(\tau T_x)_x + (\tau T_y)_y + j^2\rho_0(1 + \alpha T) = 0. \tag{4}$$

Here $\tau = \tau(x,y)$ is the thermal conductivity coefficient, ρ_0 is the resistivity of the metal at a reference temperature $T = T_0$, α denotes the temperature coefficient of resistance, and j is the absolute value of the current density. Because the conducting line on an electromigration test chip ends on either side in relatively large bonding pads or at contacts to the substrate, which are at an ambient temperature $T = T_a$, the boundary conditions are also well-defined.

It is clear that a numeric procedure to solve equation 4 will also solve the more specialized Laplace equation 1, since formally putting $\rho_0 = 0$ transforms equation 4 into equation 1, where the temperature function T is replaced by the potential u, and the electrical conductivity k assumes the role of the thermal conductivity τ. This observation makes it possible for one Fortran subroutine to solve both equations.

Atomic Flux. In addition to temperature gradients and current density changes, there are other parameters influencing the flux of atoms through the grain boundary network

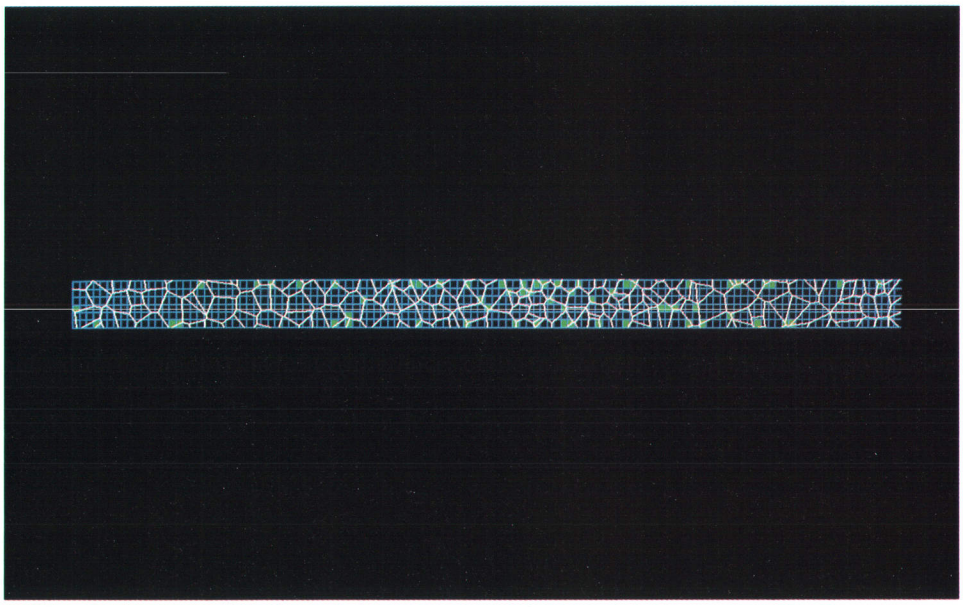


Fig. 4. *Screen showing void formation in a test structure after current stressing. Cells containing a voided grain boundary are highlighted in blue-green. The grid size is larger than that normally used in simulations to make the voiding process visible. Electron flow is from left to right.*

of the metal line. The two most important geometrical parameters are θ and ψ, where the angle θ denotes the misorientation of adjacent grains, and the angle ψ denotes the angle between the grain boundary and the current flow vector. Their values define, together with the mobility constant A, the preexponential term D_{ob}:

$$D_{ob} = A\sin(\theta/2)\cos(\psi) \quad (5)$$

for the grain boundary diffusion term

$$D = D_{ob}\exp(E_a/kT) \quad (6)$$

where E_a is the activation energy and k is the Boltzmann constant. In our model, θ and ψ are assigned to each grain boundary segment by a Monte Carlo process. The diffusion term D then enters the Huntington formula for the atomic flux J_a:

$$J_a = DN_bZ_b^*e/kT\rho(j - j_c). \quad (7)$$

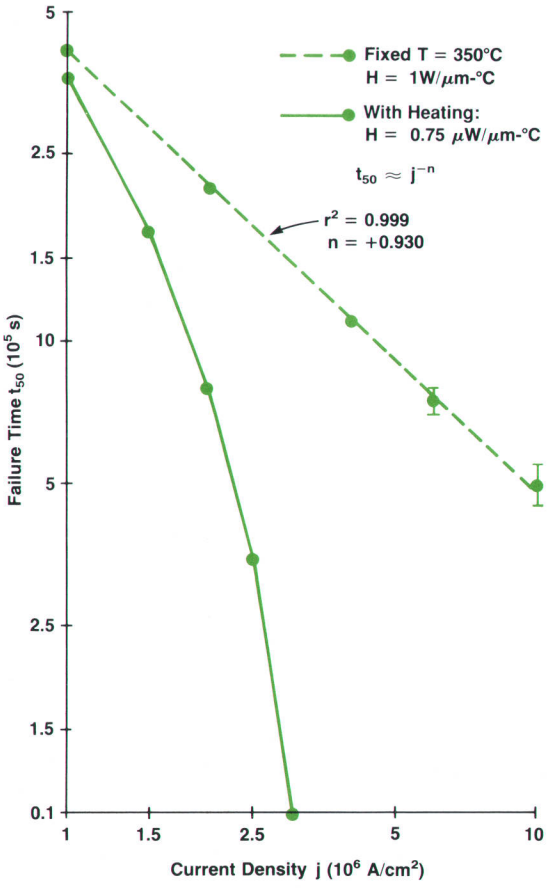


Fig. 5. *Plot of the failure time of a single line structure that has been subjected to simulated stresses at various current densities under conditions of normal Joule heating (solid) and fixed temperature (broken). H is the film-to-substrate heat transfer coefficient. A large value of H ensures a constant temperature. The value of 0.75 μW/μm-°C is typical of real systems. The quantity r^2 is the regression coefficient and n is the slope from the least squares fit.*

Here $\rho = \rho_0(1 + \alpha(T - T_0))$ is the resistivity at temperature T, Z_b^*e is the term for the effective charge, the material constant N_b denotes the atom density (e.g., for aluminum) in grain boundaries, and j_c is a stress-induced countercurrent density term which in our model (see also reference 4) depends on temperature.

Crack Growth and Line Failure. The program tracks the change in the number of atoms coming into and flowing out of each triple point. Cracks start growing at these triple points as soon as the atom count in the transporting grain boundaries shows a density of incoming atoms below a certain threshold (Fig. 4). Hence, if the material concentration drops below this level in a particular area, then this location ceases to be electrically conducting. In a similar vein, mass accumulation above a certain value creates hillocks. It happens occasionally in this model (as it does in reality!) that small cracks recover and fill in again, and some hillocks decrease or vanish completely.

Normally, the program is run until the simulated line fails. Failure is determined by calculating the gradient of the potential along vertical grid lines that are superimposed over the metal film (and used for the discretization procedure necessary for the finite element method). If this gradient is zero for two adjacent grid lines we know that no current is flowing (which signifies an open circuit) and the metal conductor has failed.

Creating several independent metal stripes (by clipping them out of the simulated grain structure) and subjecting them sequentially to the same accelerated test conditions provides a sample of failure times whose statistical distribution can be plotted and studied.

In the remainder of this paper we are concerned with determining how the new HP simulation package provides an accurate representation of the real-world situation of accelerated electromigration testing. There is a wealth of published experimental data available, and we can address here only a few of the more critical experiments.

Time to Failure versus Current Density

One of the early models, proposed on empirical grounds

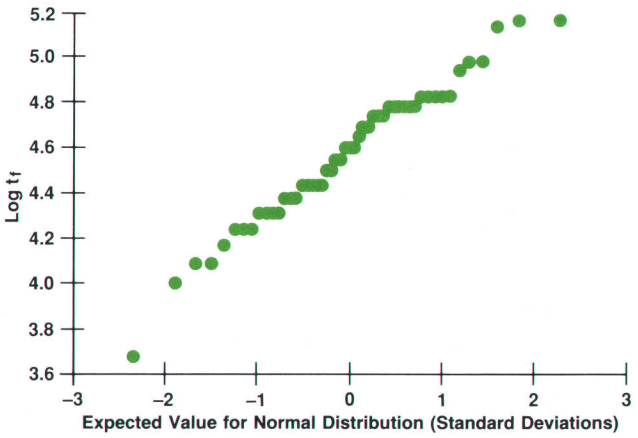


Fig. 6. *Cumulative distribution of the logarithms of simulated failure times. Units on the abscissa are standard deviations from the median (0). A straight line represents a lognormal failure distribution.*

by Black[2] and later analytically derived (under certain conditions) by Chhabra and Ainslie,[9] states a direct relationship between the median time to failure (MTTF) t_{50}, the current density j, and the absolute temperature T. A simplified version of this relationship is given by the following Arrhenius-type equation:

$$t_{50} = Bj^{-n}\exp(E_a/kT), \quad (8)$$

where B is a proportionality constant and k denotes the Boltzman constant.

Much attention has been given in the literature to the exponent $-n$, which on a logarithmic plot of $\ln(t_{50})$ versus j can be interpreted as the slope of a straight line. With this model we can fix the thermal boundary conditions to hold the metal line temperature constant. A comparison of failure times of the same metal line structure at several current densities under conditions of fixed temperature and Joule heating is shown in Fig. 5. This result shows that Joule heating can account for the curvature of the solid line plot. Under the assumptions of our model there are no other sources of curvature.

Notice that because of the high value of the current density j in VLSI devices, even small changes in the exponent n have a major impact on the MTTF t_{50} if extrapolations are made from high (test) to lower (operating) current densities.[10,11,12,13] Thus it is important to have an understanding of the origin of the behavior.

Time to Failure versus Linewidth

An interesting feature of the linewidth dependence of the failure time is the so-called *bamboo effect*, which may be described as follows: if the width of an interconnect line decreases from several (median) grain diameters down to about one grain diameter, the failure time decreases linearly. However, a further decrease in linewidth results in far longer MTTFs, that is, most lines survive longer.

We have observed that the standard deviation of the failure time follows the same trend, implying that very narrow aluminum lines, while living longer, may have the serious drawback of unpredictably large variations as measured by the standard deviation. As a consequence, the quality of a sample of such thin stripes would vary widely.

In the framework of the structural model presented here, the bamboo effect can be explained rather easily. If the width drops below the size of a single typical metal grain, hardly any triple points remain, and the thin line looks like a sequence of grains stacked together similar to the segments of a bamboo stick. That some residual electromigration still occurs is usually attributed to surface diffusion, but data for this phenomenon in aluminum films is presently lacking.

Results from simulations further suggest that larger variations in the grain size distribution trigger the bamboo effect somewhat earlier than for a more homogeneous size distribution. An exact relationship is not yet known.

Distribution of the Failure Time

When talking about failure times of thin metal lines because of electromigration, we must be aware that the times to failure are measured under accelerated conditions, notably for current densities and ambient (oven) temperatures much higher than encountered under real-life operating conditions. Hence, even if we succeed in determining the distribution of the time to failure, there still remains the problem of extrapolating this accelerated distribution to an actual operating environment. This recalculation can be solved with the help of Goldthwaite plots,[14] or by direct computer calculations, once the failure time density f(t) and the cumulative fail-time distribution function F(t) are estimated. From these the failure rate function $\lambda(t)$ can be calculated:

$$\lambda(t) = f(t)/(1 - F(t)). \quad (9)$$

Under accelerated conditions almost all samples of simulated failure times showed a good eye-fit to straight lines on lognormal paper, suggesting a lognormal distribution of failure times. The cumulative plot of the logarithm of simulated failure times of 50 lines is shown in Fig. 6.

However, other two-parameter distributions like the Weibull or the gamma family can be fitted fairly well, at least over certain ranges of failure rates. Attardo[3] notes that "for example, life test data on aluminum conductors fit the lognormal and Weibull distribution equally well at relatively large percentages of failure (0.1 to 1.0%), but at lower percentages of failure—that is, within the region of interest—the projected failure rates differ by several orders of magnitude."

Empirical evidence seems to indicate that electromigration failure times can be described better by the the more optimistic lognormal than by the Weibull distribution.[15] Relatively small sample sizes, large margins of measurement errors and the normal variability because of the randomness of the grain structure, together with the lack of a deeper understanding of the underlying electromigration forces, preclude at this point a definite decision about the true distribution of the time to failure. However, simulations of larger numbers of samples are more convenient to perform than costly life tests.

Summary

To date, we have used this simulation tool to verify the origins of the bamboo effect for electromigration and the curvature of plots of lifetime versus current density. The model can also reproduce the quantitative effects seen in Arrhenius plots of lifetime versus temperature and exhibits failure distributions representative of actual life tests of metal lines under current and temperature stress. Future efforts will be directed at better understanding the correlation between grain structures and metal film properties and the resultant failure distributions for various stress conditions.

Acknowledgments

Over the four years of the electromigration simulation project many colleagues at Hewlett-Packard Laboratories and students at San Jose State University have contributed their time and expertise. The support and encouragement of Drs. Dragon Ilic, Ed Middlesworth, and Yoshio Nishi are greatly appreciated. We are indebted to the enthusiastic help and continuous advice from Professor Jane Day, Director

of the Center for Applied Mathematics and Computer Science at San Jose State University. Professor Igor Malyshev's contributions were important during the final stages of this project. Crucial for a good start was the excellent work on the heat package and the Fortran solver by Dr. Les Foster. Nothing would have been achieved, however, without the hard work and inventiveness of our outstanding students Tung Nguyen, Jackson Shyu, Joe Loos, Takashi Tamasu, Kit Chatsinchai, and Kent Okasaki.

References

1. H.B. Huntington and A.R. Grone, "Current-Induced Marker Motion in Gold Wires," *Journal of Physical Chemistry Solids*, Vol. 20, nos. 1-2, 1961, pp. 76-87.
2. J.R. Black, "Electromigration: A Brief Survey and Some Recent Results," *IEEE Transactions on Electron Devices*, Vol. ED-16, 1969, pp. 338-347.
3. M.J. Attardo, R. Rutledge, and R.C. Jack, "Statistical Metallurgical Model for Electromigration Failure in Aluminum Thin-Film Conductors," *Journal of Applied Physics*, Vol. 42, 1971, pp. 4343-4349.
4. K. Nikawa, "Monte Carlo Calculation Based on the Generalized Electromigration Failure Model," *Proceedings of the 19th IEEE International Reliability Physics Symposium*, 1981, pp. 175-181.
5. P. Kumar and R.S. Sorbello, "Linear Response Theory of the Driving Forces for Electromigration," *Thin Solid Films*, Vol. 25, 1975, pp. 25-35.
6. R.S. Sorbello, "Theory of the Direct Force in Electromigration," *Physical Review B*, Vol. 31, no. 2, 1985, pp. 798-804.
7. A.H. Verbruggen, "Fundamental Questions in the Theory of Electromigration." *IBM Journal of Research and Development*, Vol. 32, no. 1, 1988, pp. 93-98.
8. L. Foster, *Heat Flow in Semiconductor Structures*, Final Report, San Jose State University, 1983.
9. D.S. Chhabra and N.G. Ainslie, *Open-Circuit Failure in Thin-Film Conductors*, IBM Technical Report 22.419, 1967.
10. L. Braun, "Electromigration Testing: A Current Problem," *Microelectronics and Reliability*, Vol. 13, 1974, pp. 215-228.
11. J.M. Towner and P. van de Ven, "Aluminum Electromigration under Pulsed DC Conditions," *Proceedings of the 21st IEEE International Reliability Physics Symposium*, 1983, pp. 36-39.
12. F.M. d'Heurle and P.S. Ho, "Electromigration in Thin Films," in J. Poate, K. Tu, and J. Mayer, eds., *Interdiffusion and Reactions*, Wiley, 1978.
13. P. Merchant, "Electromigration: An Overview," *Hewlett-Packard Journal*, Vol. 33, no. 8, August 1982, pp. 28-30.
14. P.B. Ghate, "Electromigration-Induced Failures in VLSI Interconnects," *Proceedings of the 20th IEEE International Reliability Physics Symposium*, 1982, pp. 292-299.
15. D.J. La Combe and E.L. Parks, "The Distribution of Electromigration Failures," *Proceedings of the 24th IEEE International Reliability Physics Symposium*, 1986, pp. 1-6.

Hewlett-Packard Company, 3200 Hillview Avenue, Palo Alto, California 94304

ADDRESS CORRECTION REQUESTED

Bulk Rate
U.S. Postage
Paid
Hewlett-Packard
Company

HEWLETT-PACKARD JOURNAL

June 1989 Volume 40 • Number 3

Technical Information from the Laboratories of Hewlett-Packard Company
Hewlett-Packard Company, 3200 Hillview Avenue
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
P.O. Box 529
1180 AM Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

CHANGE OF ADDRESS: To subscribe, change your address, or delete your name from our mailing list, send your request to Hewlett-Packard Journal, 3200 Hillview Avenue, Palo Alto, CA 94304 U.S.A. Include your old address label, if any. Allow 60 days.

5953-8575