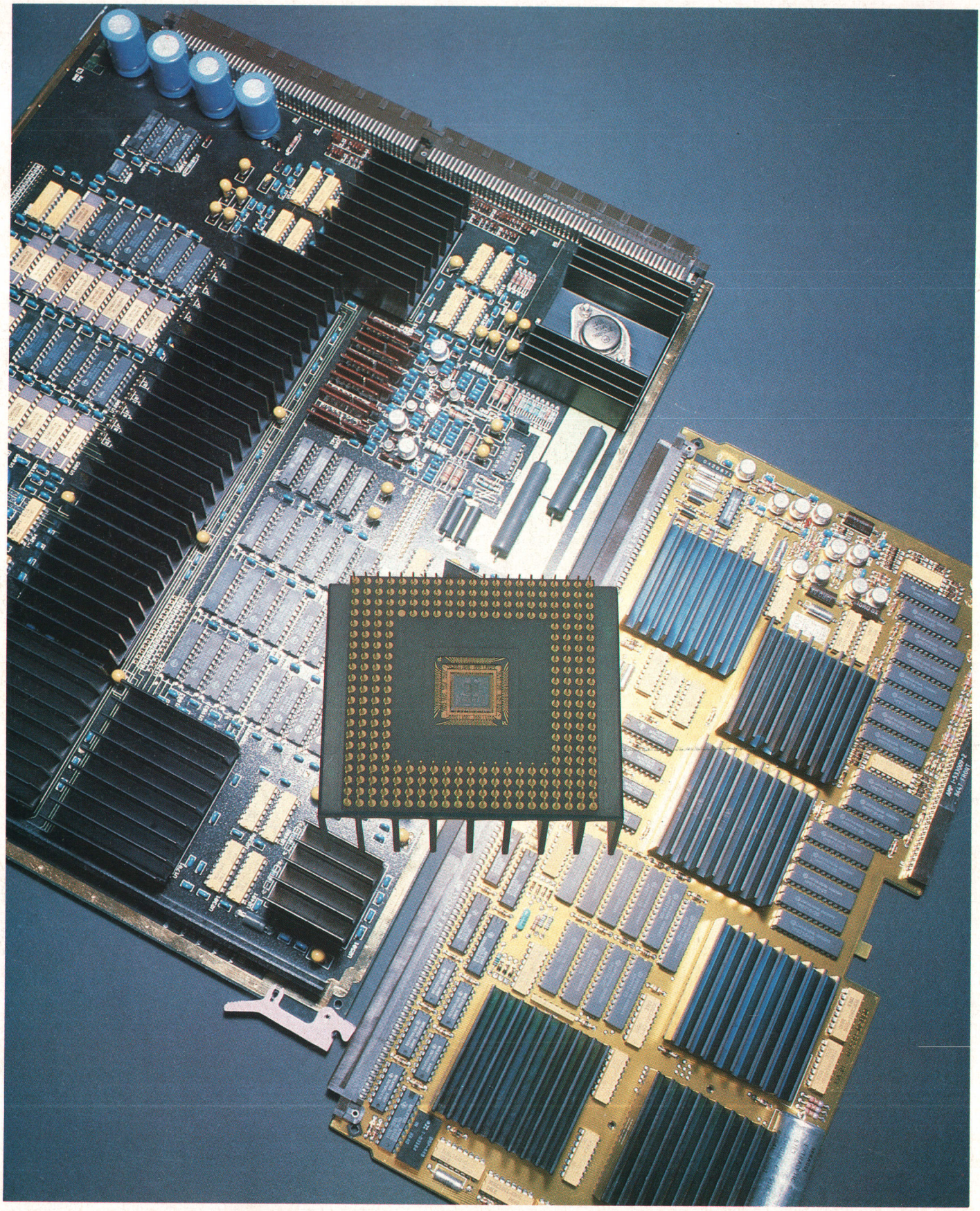
HEWLETT-PACKARD
JOURNAL

SEPTEMBER 1987



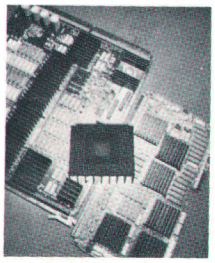
HEWLETT-PACKARD
JOURNAL

September 1987 Volume 38 • Number 9

Articles

Departments

In this Issue



This issue continues our coverage of the HP Precision Architecture hardware contributions. In the March 1987 issue we described the HP 3000 Series 930 and HP 9000 Model 840 Computers, which were HP's first realizations of HP Precision Architecture in off-the-shelf TTL technology. In this issue we present two more HP Precision Architecture computer developments that leverage a common VLSI effort using HP's proprietary NMOS-III integrated circuit technology. One of these system processing units has been introduced as the HP 3000 Series 950 running HP's MPE-XL commercial operating system and as the HP 9000 Model 850S running HP-UX, HP's version of AT&T's industry-standard UNIX® operating system environment. These are the new top-of-the-line HP 3000 and HP 9000 computers. The other system processing unit, a midrange computer, has been introduced as the HP 9000 Model 825 running HP-UX both in a single-user workstation environment and as a multiuser HP-UX system.

The effort described in this issue dates back to 1982 and has been quite large. The VLSI development involved 12 devices with up to 150 thousand transistors per chip and was the largest multichip custom VLSI project HP has undertaken to date. The two computers described share eight common VLSI devices but have quite different design centers. The top-of-the-line SPU maximizes performance, memory, and input/output capacity in a fairly large data-center style enclosure while the midrange SPU is designed for maximum achievable performance in the smallest possible package. Yet both are capable of running exactly the same operating environment and are compatible members of the HP Precision Architecture computer family.

Because the VLSI development was so significant towards the realization of the two computers we have placed the papers describing it in the front of the issue followed by the SPU papers.

-Peter Rosenbladt
Manager, High-Performance Systems Operation

Cover

Processor boards from the HP 9000 Model 825 Computer (smaller board) and the HP 9000 Model 850S/HP 3000 Series 950 (larger board), shown with an unmounted pin-grid array package housing an NMOS-III VLSI chip.

What's Ahead

The October issue will complete the design story (begun in August) of the HP-18C and HP-28C Calculators, with articles on the calculators' thermal printer, infrared data link, and manufacturing techniques. Also featured will be the design of the HP 4948A In-Service Transmission Impairment Measuring Set. Another paper will relate Harvard Medical School's experience with computer-aided training in their New Pathway curriculum, and we'll have a research report on formal methods for software development.

A VLSI Processor for HP Precision Architecture

by Steven T. Mangelsdorf, Darrell M. Burns, Paul K. French, Charles R. Headrick, and Darius F. Tanksalvala

THIS PAPER DESCRIBES the VLSI chip set used in the processors of three HP Precision Architecture computers: the HP 3000 Series 950 and the HP 9000 Models 850S and 825. The Series 950 and Model 850S processors are identical. All of the chips are designed in HP's NMOS-III process.[1] NMOS-III is a high-performance NMOS process with 1.7-μm drawn channel lengths (0.95-μm effective channel lengths), 2.5-μm minimum contacted pitch, and two levels of tungsten metallization. The chips have been designed for a worst-case operating frequency of 30 MHz, although with the static RAMs available for caches at present, the Model 850S/Series 950 processor operates at 27.5 MHz and the Model 825 processor operates at 25 MHz. A 272-pin ceramic pin-grid array package was developed to support the electrical and mechanical requirements for the chip set (see "Pin-Grid Array VLSI Packaging," page 10).

Overview

Each processor consists of a CPU, two cache controller chips (CCUs), a translation lookaside buffer controller chip (TCU), a system bus interface chip (SIU), a floating-point interface chip (MIU), and three floating-point math chips. All chips except the SIU are common to both computer systems. There are two versions of the SIU. The SIU in the Model 850S/Series 950 interfaces to a high-bandwidth 64-bit system bus (SMB), and the SIU in the Model 825 interfaces to a medium-performance 32-bit system bus. The floating-point math chips are the same NMOS-III chips that are used in the HP 3000 Series 930 and HP 9000 Model 840 Computers.[2]

Fig. 1 shows the block diagram of the processor. The diagram is applicable to both computers. The only differences are in the SIU chips, the system buses, and the sizes of the caches and TLBs (translation lookaside buffers). All the chips communicate via the cache bus which consists of 32 address lines, 32 data lines, and 63 control lines. The cache bus protocol is transaction-based; only the CPU or the SIU can be masters.

The CPU has most of the hardware for fetching and executing instructions. It is described in the paper on page 12.

Each CCU responds to cache bus transactions for doing various cache operations and controls an array of commercial static RAMs. The Model 850S/Series 950 has a two-way set-associative 128K-byte cache, and the Model 825 has a two-way set-associative 16K-byte cache. The CCUs also perform instruction prefetching and correction of single-bit errors in the cache RAM.

The TCU responds to virtual address cache bus transactions which require a virtual-to-real address translation and permission checking. It controls an array of commercial static RAMs. It generates appropriate traps for illegal accesses and TLB misses.

The SIU interfaces the processor to the system bus on which main memory is located, fetches cache blocks from memory during cache misses, and performs other system interface functions.

The MIU responds to floating-point coprocessor transactions on the cache bus and controls the floating-point math chips. It also reports exceptions and trap conditions.

All chips on the cache bus also respond to various diagnostic instructions for system dependent functions such as self-test.

Cache and TLB

Cache Function

The cache speeds up CPU memory accesses by keeping the most recently used portions of memory in a high-speed RAM array. The array has a large number of rows called sets and two columns called groups. Each array entry contains a 32-byte block of memory and a tag identifying the block's address.

Cache operation is shown in Fig. 2. When the CPU accesses memory, the low-order address bits determine which of the many sets the data may be in. The tag of each of the

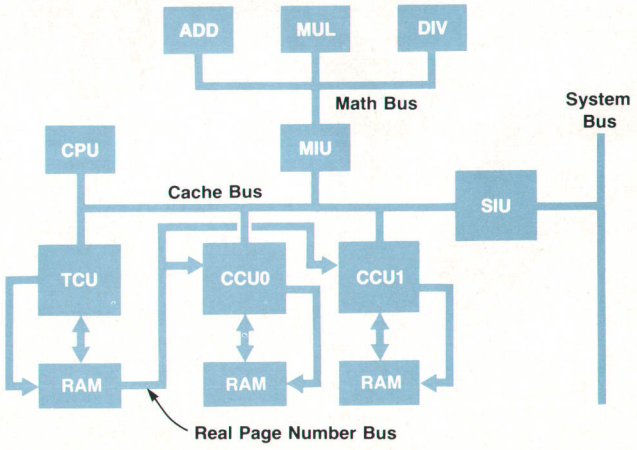


Fig. 1. *HP Precision Architecture VLSI processor block diagram. The VLSI chips are the central processing unit (CPU), the cache control units (CCU), the TLB control unit (TCU), the system bus interface unit (SIU), the floating-point math interface unit (MIU), and three floating-point math chips (ADD, MUL, DIV).*

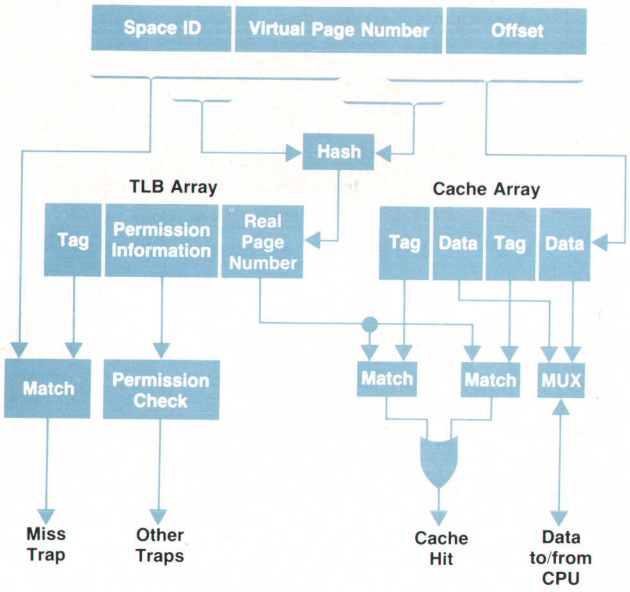


Fig. 2. *Cache and TLB operation.*

two entries in this set is compared against the address. If a hit occurs, the data can be quickly transferred to or from the CPU. If a miss occurs, one of the two blocks in the set is randomly selected to be removed to make room for the new block. This block is written back to main memory if its tag indicates it is dirty (i.e., it has been modified since being moved into the cache). The new block is read from main memory and placed into the vacated entry. The data is then transferred to or from the CPU. Cache misses are managed by the SIU and are not visible to software.

Hits are much more common than misses because programs tend to access a cached block many times before it is removed. For example, the expected miss rate of the Model 850S/Series 950 is 1.5%. The effective memory access time is therefore very close to that of the cache itself. This can be made quite short since the cache is small enough to be implemented with high-speed RAMs and is physically close to the CPU. The cache access time of the Model 850S/Series 950, for example, is about 65 ns measured at the pins of the CPU.

TLB Function

The main function of the translation lookaside buffer (TLB) is to translate virtual addresses to real addresses (Fig. 2). The TLB determines the real page number mapped to by the space ID and the virtual page number. The real page number concatenated with the invariant offset field gives the 32-bit real address. The page size is 2K bytes. HP Precision Architecture allows 64-bit virtual addresses, but these computers implement only 48 bits to economize on hardware. This is more than adequate for current software needs.

The TLB also verifies that the current process has permission to access the page. Two major types of checking are defined by HP Precision Architecture. Access rights checking verifies that the process has permission to access the page in the requested manner (read, write, or execute) and that its privilege level is sufficient. Protection ID checking verifies that the page's 15-bit protection ID matches one of the four protection IDs of the process. A trap is issued if a violation is found.

The TLB cannot possibly store the real page number and permission information for every virtual page since there are 2^{37} of them. Instead, it stores the information for the most recently used pages and operates much like the cache. The TLB has only a single group instead of two. Each entry contains the real page number and permission information plus a tag indicating the page's space ID and virtual page number. Half of the entries are dedicated for instruction fetches and half for data accesses.

When the CPU performs a virtual memory access, the address bits determine which of the many entries the page may be in. This entry's tag is compared against the address. If a hit occurs, the real page number is sent to the cache and the permission checks are performed. If a miss occurs, the CPU is interrupted with a trap. The trap handler inserts the required entry into the TLB, replacing the previous entry. The trap handler returns to the instruction that caused the miss; this time, a hit will occur. While misses are serviced by software, they are not visible to the programmer.

Organization

The cache and TLB system organization is shown in Fig. 1. Each of the two cache controller units (CCUs) controls one of the two groups in the cache array. Similarly, the TLB controller unit (TCU) controls the TLB array. The cache and TLB arrays are implemented with commercial CMOS static RAMs. These provide excellent speed and density while eliminating the design effort and risk associated with an on-chip RAM array.

Operation of the cache system is best understood by considering a typical virtual-mode load cache bus transaction as shown in Fig. 3. The transaction begins with the trans-

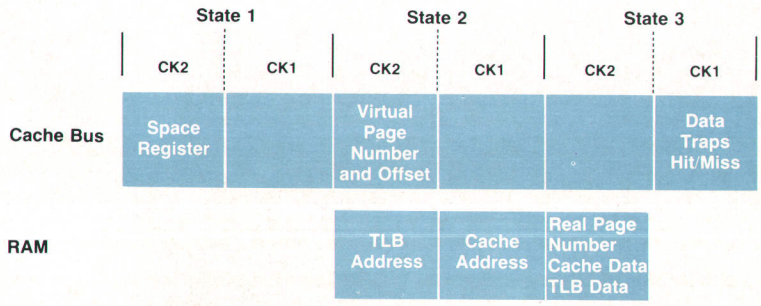


Fig. 3. *Typical virtual-mode load cache bus transaction.*

mission of the space register number from the CPU to the TCU during clock 2 of state 1. The TCU dumps the space ID out of the selected space register, and the virtual page number arrives during clock 2 of state 2. The TCU forms the TLB RAM address from the space ID and virtual page number and drives it out to the RAM array during the same clock 2. One bit of the address is set by whether the transaction is for an instruction fetch or data access; this effectively partitions the TLB array into two halves as discussed above. At the end of clock 2 of state 3, the RAM data is valid. The real page number field goes to the two CCUs for their tag compare. The other fields go back to the TCU for its tag compare and for permission checking. A miss or a permission violation causes the TCU to issue a trap during clock 1 of state 3.

The CCUs process the transaction in parallel. They receive the address during clock 2 of state 2 and drive it out to their RAM during the next clock 1. At the end of clock 2 of state 3, the RAM data is valid along with the real page number from the TLB RAM. A tag compare is then performed. If a hit occurs, the CCU drives the data out onto the cache bus during clock 1 of state 3. If a miss occurs, all the CCUs allow the bus to float high, which is interpreted as a miss by the other chips.

It is important to note that the CCUs can address their RAM with the virtual address before the real page number has arrived from the TLB, greatly reducing overall access time. This is permitted because HP Precision Architecture specifies a one-to-one mapping between virtual and real addresses, and requires software to flush pages before their mapping changes. This is a very important contribution of HP Precision Architecture over other architectures, which allow only the offset portion of the address to be used for efficient cache addressing.

For real-mode transactions, the TCU does not read its RAM. Instead, the TCU receives the real page number from the cache bus and drives it to the CCUs through TTL buffers. This simplifies the CCU by making real and virtual accesses look alike. Note that the TCU does not connect to the real page number RAMs. The CCUs and the TCU cooperate during transactions such as TLB inserts where the real page number is sent over the cache bus.

Stores are similar to loads, except that the CCU writes the new tag and data into the RAM during clock 1 of state 3. Byte merge is performed by combining the old and new data words. If a miss occurs, the write is not inhibited, but the old tag and data are written instead of the new. When a trap occurs such as a TLB miss, the indication comes too late on the cache bus for this. Instead, states are added to the transaction to write the old tag and data back.

Instruction Fetch Timing

The load transaction described above is effectively two states long since its first state can overlap with the last state of the previous transaction, and it causes the CPU to interlock for one state. This interlock would also occur if instruction fetch transactions required two states and would cause an excessive performance loss. Therefore, a prefetching algorithm is implemented by the CCU to shorten fetch transactions to one state and eliminate the interlock in almost all cases.

There are three types of fetch transactions: sequential, branch, and random. They all have one thing in common: at the end of the transaction, the CCU prefetches the next sequential instruction and saves it in an internal register.

A sequential fetch transaction fetches the instruction that follows the previous one. The CCUs can simply dump out the prefetched instruction from their registers. The transaction is only one state long and does not cause an interlock.

A branch fetch transaction fetches the target of a taken branch instruction. The branch target address is sent from the CPU to the CCUs in a prefetch target transaction issued on the previous state. This gives the CCUs a one-state head start in fetching the branch target, so the fetch transaction is only one state long and does not cause an interlock. The CPU must send the prefetch target before it knows whether the branch is taken. Untaken branches result in prefetch targets followed by sequential fetches.

A random fetch transaction is issued for the first instruction after a reset, the first instruction of any interruption handler, or the first or second instructions after a return from interruption instruction. The fetch transaction is two states long and causes a one-state interlock, but these cases are very rare.

The TCU uses a similar prefetching algorithm. However, it is not required to prefetch the next sequential instruction on every fetch transaction. Since all instructions from the same page have the same translation and permission information, the real page number and permission check results are simply saved and used for subsequent sequential fetches. Whenever a sequential fetch crosses a page boundary, a one-state penalty is incurred while the TLB is reaccessed to determine the real page number and check permission for the new page. It is also necessary to reaccess the TLB when an instruction is executed that might change the real page number or permission check results. As a result of this simplification, the TCU RAM is cycled at most every other state (the CCU RAM is cycled every state).

Design Trade-Offs

HP Precision Architecture does not require the cache or TLB to be flushed after each context switch like many other architectures. Therefore, it can never hurt performance to make the arrays larger, provided that the clock rate is not affected. Each CCU can control 8K, 16K, 32K, or 64K bytes of data, and each TCU can support 2K or 4K total entries. Supporting larger arrays would not have been difficult except that pins for additional address lines were not available. The Model 850S/Series 950 has 64K bytes of data per CCU (128K bytes total) using 16K × 4 RAMs, and a 4K-entry TLB using 4K × 4 RAMs. The Model 825 has 8K bytes per CCU (16K bytes total) using 2K × 8 RAMs, and a 2K-entry TLB also using 2K × 8 RAMs.

The number of groups (columns) in the cache array is called the associativity. Increasing the associativity tends to decrease the miss rate. This is because wider sets (rows) decrease the chance that several heavily used blocks will compete for a set that cannot hold them all. The improvement is substantial going from an associativity of 1 to 2, but further increases bring little improvement. Unfortunately, increasing associativity generally tends to increase cache access time because data from the groups must be

multiplexed together following the tag compare. In these processors, access time is limited by parity decoding, which is performed in parallel with the multiplexing. Therefore, we decided to support from one to four groups, each implemented by a CCU and its associated RAM. The Model 850S/Series 950 and Model 825 both have a cache associativity of 2. However, the TLB associativity is fixed at 1 because the improvement in miss rate did not justify the increase in access time and cost.

When the cache associativity is greater than 1, it becomes necessary to decide which of the entries in the set is to be replaced when a cache miss occurs. The performance advantage of a least recently used strategy over a random strategy was too small to justify the extra implementation cost. With the random strategy, each group is selected on every Nth miss where N is the associativity.

The cache block size is fixed at 32 bytes. Larger blocks tend to decrease miss rate up to a limit, but cache misses require more time to service. The block size was chosen to minimize the performance penalty for cache misses, which is the product of the miss rate and the miss service time.

The cache uses a write-back store policy. This means that system memory is not updated on a store until the block is eventually replaced from the cache or explicitly flushed by software. Because a block can be stored to many times before being replaced, a write-back policy minimizes system bus traffic, which is critical in multiprocessor systems.

The same cache is used for both instruction fetches and data accesses. Separate caches are permitted by HP Precision Architecture and have tremendous advantages when the CPU can issue both an instruction fetch and a data access concurrently. However, the CPU always interleaves these operations on the cache bus, and so separate caches offered no advantage for us. One benefit of a unified cache is that it generally has a significantly lower miss rate than a split cache of the same size, particularly when the associativity is greater than 1.

For the TLB, however, half the array is allocated for instructions and half for data. This prevents thrashing, the situation where a heavily used instruction page and a heavily used data page compete for the same entry. Also, handling TLB misses in software requires that the TLB be able to hold an arbitrary instruction and data page concurrently.

Address hashing is used to improve TLB performance. Instead of using the low-order bits of the virtual page number to select a TLB entry, an XOR of virtual page number and space ID bits is used. This ensures that low-numbered pages from all the spaces do not map to the same entry. This minimizes thrashing because these pages tend to be frequently used. According to preliminary studies performed by our System Performance Laboratory, this improves instruction TLB performance by a factor of 3 and data TLB performance by a factor of 7.

Error Correction

Several features are included to increase product reliability and availability. The CCU implements detection and correction of all single-bit errors in the cache array. Each data word and tag are protected by 16-bit horizontal parity. The CCU also maintains the reference vertical (column) parity in an internal register that is updated on all writes. Copy-in transactions require special handling because they are write-only rather than read-modify-write. The CCU accumulates the vertical parity of the replaced block during copy-out transactions (for dirty misses) or autonomously (for clean misses) before the SIU issues the copy-in transactions.

When a parity error is detected during a read, the CCU hangs the cache bus and initiates error correction. It walks through the entire array (up to 16K locations) to determine the actual vertical parity, and compares this with the reference vertical parity in its register. The bit position in which a mismatch occurs indicates which bit of the data should be flipped. Error correction occurs without software intervention.

After each correction, the CCU issues a low-priority machine check to the CPU so that software can log the error and test the location for hard errors. If a hard error is found, an individual entry or an entire group of the array can be mapped out so that processing can continue (with degraded performance) until the machine is repaired.

The TCU provides the same protection against single-bit errors but in a simpler way. As in the CCU, 16-bit horizontal parity is used to detect errors. When an error is detected, the TCU issues a high-priority machine check to the CPU. The high-priority machine check handler purges the bad entry from the TLB. Upon return from the high-priority machine check handler, a TLB miss trap will occur, and the entry will be inserted into the TLB by the normal TLB miss handler. This simple scheme works because HP Precision Architecture never requires TLB entries to be dirty.

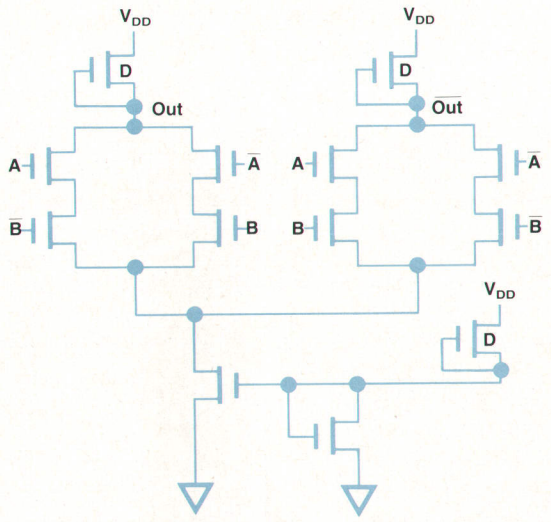
The high-priority machine check handler can also log the error and test the entry for hard errors. If a hard error is found, one half or three quarters of the TLB can be mapped out so that processing can continue despite degraded performance. This is implemented by simply freezing one or two of the RAM address bits.

Internal Circuits

Internal circuits on the TCU and CCU chips are designed to run at 30 MHz worst-case. The chips provide 25-MHz worst-case operation with 25-ns RAMs and 27.5-MHz worst-case operation with 22-ns RAMs.

Writes into the cache array require only one state. A split-phase loop is used to generate timing edges for the NCE and NWE RAM control signals and for the data driver enable and tristate (see "A Precision Clocking System," page 17).

Not surprisingly, transfers of data and addresses to and from the RAMs provide some of the tightest timing paths on these chips. For instance, full 16-bit parity encode and decode must be completed in 7 ns without the advantage of clock edges. This is accomplished by using a tree of XOR gates constructed from analog differential amplifiers of the type shown in Fig. 4. This circuit requires two true/complement pairs as inputs to produce one output pair. The small signal swing this circuit requires allows the result to propagate through the parity tree fast enough to meet its 7-ns budget. The analog differential voltages at the output of the tree are converted to digital voltages by a level trans-



D = Depletion Type MOSFET

Fig. 4. *Analog differential amplifier used as an* XOR *gate.*

lator.

TLB address hashing has been implemented for improved TLB performance. Since the address drive time is in the TLB's critical timing path, the hashing circuits need to be very fast. Fast hashing is accomplished by using the fact that the space ID bits of the virtual address and the control data are available early to set up transfer FETs. The circuit simply steers the late-arriving virtual page number bit to the positive or negative input of the address pad drivers, depending on whether the early space ID bit was a 0 or a 1. This logically XORs the space ID and virtual page number bits with a minimal delay of only one transfer gate.

Three types of control structures are used to provide different combinations of speed and density. Dynamic PLAs (programmable logic arrays) with two levels of logic and a latency of two clock phases are used for the majority of chip control, and smaller two-level static PLAs with one phase of latency are reserved for more speed-critical control. PLA outputs can qualify each other and be wire-ORed together, effectively providing third and fourth levels of logic. It is also possible for static PLAs to generate the bit lines for dynamic PLAs to give an extra two levels of logic. Where speed is essential, a few random logic structures are used.

System Interface Unit

The function of the system interface unit (SIU) is to interface the processor cache bus to the SPU system bus so that it can communicate with main memory and I/O. Since the price/performance goals of the different computers could only be met with two different system bus definitions, two separate SIUs were designed. SIUF interfaces to the 32-bit MidBus used in the Model 825. SIUC interfaces to the 64-bit SMB used in the Model 850S/Series 950. The two SIUs are very similar except that the SIUC implements multiprocessor cache coherency algorithms.

Most of the functions of the SIU involve block transfers to and from the CCUs over the cache bus. A typical cache miss sequence is shown in Fig. 5. It begins with a load or store cache bus transaction in which both CCUs signal a miss. The CCUs send the real address of the requested block to the SIU so that it can begin arbitrating for a read transaction on the system bus. The CCU that is selected for replacement then sends the real address of the block to be replaced along with an indication of whether the block has been modified. If so, the SIU issues eight copy-out cache bus transactions to transfer the replaced block to its write-back buffer.

Sometime after this is completed, the requested block will begin to arrive on the system bus. The SIU issues eight copy-in cache bus transactions to transfer it to the CCU. Finally, the SIU issues a restart cache bus transaction to end the miss sequence, and the CPU resumes issuing transactions. For a load miss, the requested data is transferred from the CCU to the CPU during the restart transaction. The miss penalty is 27 instructions for the SIUF and 16.5 instructions for the SIUC.

The SIU will arbitrate for a write transaction on the system bus to empty its write-back buffer as long as no higher-priority operation is pending. The SIUC has a two-entry write-back buffer for maximum performance.

HP Precision Architecture defines an uncached region of the real address space called I/O space. When a load or store transaction accesses I/O space, the CCUs always signal a miss. The SIU issues a single-word read or write transaction on the system bus since no block transfers are involved. In systems using SIUF, the processor dependent code ROM (which contains boot code) is also uncached. The SIUF accesses this ROM in a byte-serial manner to reduce cost.

The SIU contains some of the HP Precision Architecture control registers. This results in better system partitioning and reduces CPU chip area. On the SIU chip are the temporary registers, the interval timer, and the external inter-

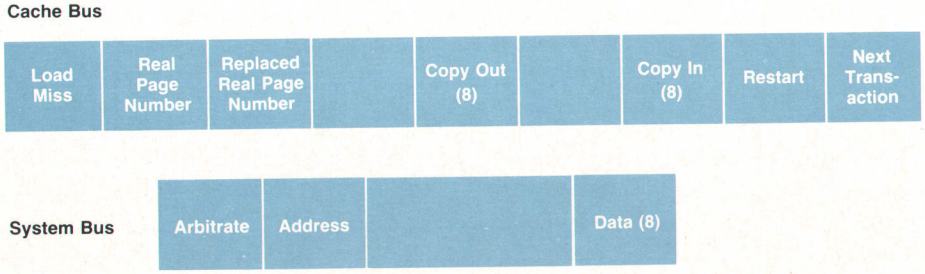


Fig. 5. *Typical cache miss sequence.*

rupt request and mask registers.

The SIU detects a power-up or hard reset operation on the system bus and generates cache bus signals to initialize the other chips. It interrupts the CPU when a powerfail is detected or an external interrupt is pending. Checking is performed for system bus parity errors and protocol violations and errors are reported to the CPU through the high-priority machine check.

HP Precision Architecture requires the SIU to have a few registers accessible to other devices on the system bus for address relocation and receiving interrupts and resets. Additional registers are used for logging system bus errors and other implementation dependent functions.

Many other architectures require the SIU to query the cache during DMA transfers to ensure consistency with main memory. With HP Precision Architecture, it is software's responsibility to flush the affected parts of the cache before starting a DMA transfer. This results in a considerable savings of hardware and complexity.

The processor and memory subsystems of the Model 850S/Series 950 are designed to implement hardware algorithms to ensure full cache and TLB coherency between multiple symmetric processors as required by HP Precision Architecture. Whenever a memory access occurs, all processors must ensure that the processor doing the access acquires an accurate copy of the data.

Each 32-byte block in the cache is marked either clean and private, clean and public, or dirty. If a block is dirty or clean and private, this is the only copy of the block and can be modified without communicating with other processors. If the block is clean and public, it can be read but not written. If the block is absent, it is requested from the memory subsystem.

During the return operation on the SMB, all other processors check their caches for copies of the requested block. If a checking processor discovers a dirty block, the return operation is aborted, the dirty block is flushed to memory, and the return operation is repeated with the correct copy of the data. If a checking processor discovers a clean and public or clean and private copy, it will either delete it or change it to public depending on whether the requesting processor wants a private copy. Similar algorithms are used to maintain coherency during the load, store, load and clear, purge cache, and flush cache operations.

To maintain TLB coherency in a multiprocessor system, the purge TLB instruction is broadcast on the SMB to all processors so that all TLBs are purged simultaneously. The SIU maintains coherency if a purge TLB occurs on the SMB at the same time that the TLB entry is being used to access data from the memory subsystem during a cache miss.

Floating-Point Coprocessor

HP Precision Architecture allows coprocessors to provide hardware assist for complicated operations. Coprocessors have their own set of registers, and once a coprocessor operation is started, the coprocessor can process that instruction concurrently with the CPU. Floating-point operations are well-suited for coprocessing. In general, floating-point operands never need to be operated on by the CPU's integer ALU, so dedicated floating-point registers keep the general registers free. Floating-point operations also tend to take many cycles to complete. While the coprocessor is working on completing a previous operation, the CPU can continue.

The architecture requires the coprocessor to implement full IEEE-compatible floating-point functionality. The coprocessor has sixteen 64-bit registers, of which twelve are floating-point registers and four are status and exception condition registers. Single, double, and quad precision operations are defined by the instruction set. The floating-point hardware actually implements a subset of the standard. The coprocessor redirects to software emulation routines any operations that cannot be handled in hardware. Trapping to software only occurs on infrequent operations and exceptional conditions and thus has a minimal impact on performance.

The floating-point coprocessor is implemented by four chips: the math interface unit (MIU) and three proprietary HP floating-point chips. The three floating-point chips are an adder, a divider, and a multiplier. These same floating-point units are used in the HP 9000 Model 550, the HP 9000 Model 840, and the HP 3000 Series 930 Computers.

Math Interface Unit

Review of the math chips' capabilities showed the benefit of a simple MIU design. The math chips can be configured to allow pipelining on the chips; for example, the add chip can be configured to perform up to five simultaneous independent adds. This increases throughput in those special cases in which many independent adds are needed. However, the penalty for taking this approach is that the overall latency is significantly increased and scalar use of the coprocessor suffers. The determination was made to only allow one floating-point operation at a time to be executed by the math chips. This has the double benefit of decreasing latency and keeping the MIU simple. The result is a very clearly defined and well-partitioned chip that needs a minimum of special circuitry.

The MIU interacts with the CPU over the cache bus. The CPU processes floating-point instructions in its pipeline like other instructions. The CPU determines cache addresses, branch and nullify conditions, etc., before issuing the transaction to the MIU. When the transaction is issued, the cache and TLB report trap conditions and the MIU begins processing the instruction. The cache bus protocol is very flexible, allowing data and trap information to be sent during any state of a transaction. Transactions can also be extended to any length. This interface to the CPU simplifies the communication between the CPU and the MIU, since it removes the requirement that the MIU have knowledge of the CPU pipeline.

The MIU allows operations to overlap with independent loads and stores. Because of this feature, the MIU includes on-chip interlock hardware that checks for interlock conditions and responds properly. Interlock conditions that must be recognized include attempting to start an operation before the previous one has completed, attempting to read a register that is the destination of an executing operation, attempting to load either the sources or destination of an executing operation, and attempting to read the status register while an operation is executing. In each of these cases,

Pin-Grid Array VLSI Packaging

A significant part of the design of the VLSI processor for the HP 3000 Series 950 and HP 9000 Models 850S and 825 Computers was the method of packaging the VLSI chips. A single pin-grid array (PGA) package was designed with enough flexibility and electrical and thermal performance that all six of the ICs on the processor board are able to use it. This same package is also used by three other custom VLSI circuits in the systems.

The package design was constrained by the union of the requirements of all the chips that were to use the package:

- 272 pins, some dedicated to a common ground and others dedicated to three main power supplies. Additional flexibility for separating noisy supplies from internal logic supplies was also required.
- A different chip padout for each of the nine ICs that were to use the package.
- Ceramic package technology for high assembly yield and high reliability.
- Adherence to geometries that can be manufactured using thick-film ceramic technology and assembled using conventional wire-bonding technology.
- Support of high-power (12W dissipation) VLSI NMOS circuits.
- Consistent with through-hole mounting technology on conventional printed circuit boards.

The 272-pin PGA package is shown in Fig. 1. The design makes full use of state-of-the-art multilayer ceramic technology. The PGA has six metallization layers, three of which are ground planes. A fourth layer is a mixed ground and power plane (this plane also acts much like an ac ground). Two layers contain all the signal lines. Each of these two layers has a ground plane above and below it, ensuring a transmission-line environment. Connections between layers are provided by vias punched in the ceramic. These vias also ensure ground plane and power supply integrity.

A few changes in the conventional multilayer ceramic process were required. First, a heat slug made of 90% copper and 10% tungsten is provided for attachment of the IC. This results in a thermal resistance of 0.9°C/W from junction to heatsink, compared with approximately 3 to 5°C/W for conventional die attachment for the same IC and package size. Second, considerable flexibility is provided for pad assignments between ground, power, and signals. Over 90% of the pads have more than one bonding option, and 10% of these have three bonding options. Third, there are three tiers of wire-bond pads on the ceramic and two staggered rows of bond pads on the IC. This provides not only shorter, more dense bonding of the signals and optional power supplies, but also a way to achieve very short wire-bond

the MIU will hang the processor until the interlock is resolved, and then continue. Allowing noninterlocked loads and stores gives the compilers the ability to improve performance significantly with optimized code.

The MIU interacts with the math chips across the math interface bus. The MIU is a complete master of this bus, controlling all the load, unload, and operation timings.

Counters control the number of cycles needed before unloading the result from a math chip. These counters are available to software through an implementation dependent instruction. Since the floating-point math chips' calculation times are independent of any clocks, the number of cycles needed for an operation to complete varies with frequency. Giving software the capability to control the cycle counts allows the coprocessor to run as efficiently as possible at any frequency by changing the number of count cycles as frequency changes. As a debugging feature, another set of machine dependent commands allows software to take direct control of the math bus to facilitate system debugging involving math chips.

Cache Bus Electrical Design

The six chips in the processor communicate via a collection of 127 signals known as the cache bus. Each chip connects only to those signals necessary for its function.

The bus operates in a precharge/pulldown manner. Each signal is designated as either clock 1 or clock 2. Clock 1 signals transmit data during clock 1 and precharge during clock 2, and clock 2 signals transmit data during clock 2 and precharge during clock 1. Each cache bus signal transmits one bit of information per state.

During the precharge phase, all chips connected to a signal help drive it to a high value (2.85V nominal). During the transmit phase, if a chip wishes to assert a zero, it turns its driver on and pulls the signal low. If it wishes to assert a one, it allows the signal to remain high. Any chip connected to a signal may assert a zero independently of the other chips. This wired-AND logic simplifies the functional protocol.

Because each cache access requires a transfer from the CPU to the CCUs and back, short cache bus transmission delays are essential to maximize the processor clock frequency. This was achieved through careful design of the drivers, printed circuit board traces, receivers, and clocking system.

The drivers consist of a large pulldown transistor and a source bootstrapped predriver stage which can apply a full 5V signal to the gate of the pulldown transistor. The delay through the driver is only 2.6 ns worst case. The precharge/pulldown bus results in area-efficient drivers, since the pullup consists only of a precharger. This can be small because several chips plus the terminating resistor help pull each signal high and precharge has an entire phase in which to complete.

The chips are positioned along a linear bus on the board. There are never more than two traces connected to a pin, so the driver never looks into an impedance lower than two parallel traces of 50Ω each when driving the initial edge. The linear bus also reduces undesirable reflections by minimizing impedance mismatches. Considerable attention was given to minimizing the length of traces, particularly those known to be critical. This resulted in a worst-case pin-to-pin propagation delay of only 5.7 ns.

The receivers are of the zero-catching variety. They consist of bistable latches that are reset during the precharge

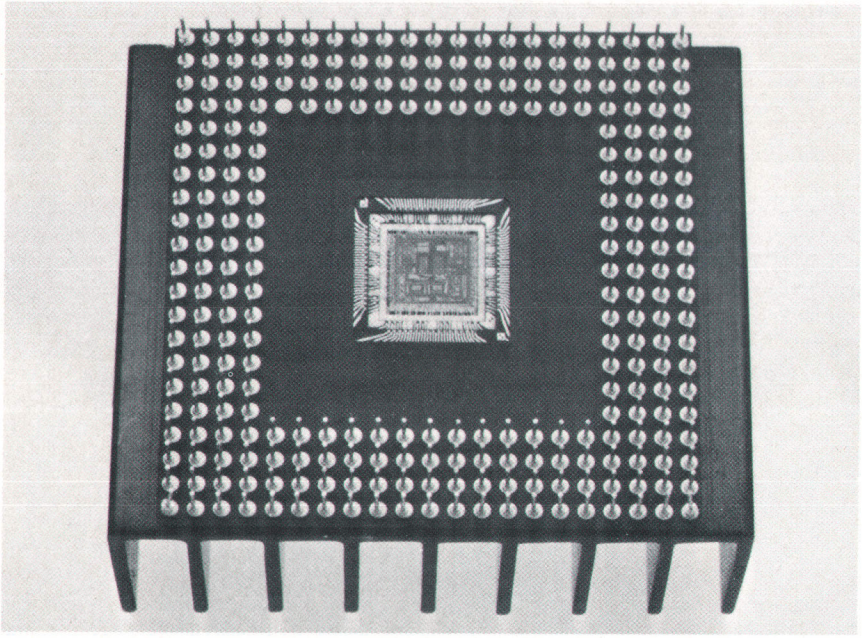


Fig. 1. *272-pin pin-grid array package.*

lengths (less than 0.045 inch) from the pad on the IC directly to a ground plane on the PGA. A constraint on the IC designers was that power supply and ground pads could be located only at certain prespecified locations on the chip.

The electrical requirements for the package were quite aggressive because of the large number of I/O connections and their high switching speeds. Minimizing power supply and ground noise was extremely important. It was also essential to provide a transmission-line environment for the signal lines. The metallization that serves as signal routing is in a stripline configuration. This configuration minimizes signal-to-signal coupling (crosstalk). Power supply routing is designed to minimize the path inductance and maximize the capacitance, thereby providing clean power supplies to the IC. The PGA can have up to thirteen independent power supplies which are isolated on the package and independently bypassed to ground on the package. The ground planes are shorted together through vias to provide a clean ground connection to the IC.

Electrical models for the 272-pin PGA package were extracted directly from the artwork. In lieu of the challenge of verification by direct measurement, these models were verified by comparing measurements from a CPU in a test fixture with Spice simulations that modeled the CPU, PGA, and printed circuit board environment. Good correlation between the measurements and simulation results was obtained. These models were then used by the designers for worst-case system modeling.

Acknowledgments

Thanks to all the people who worked on this project, especially Larry Hall and Guy Wagner.

John E. Moffatt
Asad Aziz
Development Engineers
Colorado Integrated Circuit Division

phase and set during the transmit phase if their input falls below 1.3V for 2.9 ns or longer. Analog design techniques were used to raise and tightly control the trip level. Once tripped, the receivers do not respond to noise or reflections on the bus. This increases noise margin for receiving a zero and effectively reduces propagation delay. The delay through the receiver is only 2.9 ns worst case.

Acknowledgments

The authors would like to thank the many individuals who contributed to the design of the processors discussed in this paper: Paul Bodenstab, Mike Buckley, Dave Conner and Hoang Tran for the SIUF, Tom Asprey, Kevin Ball, Paul Ilgenfritz, Anne Kadonaga, Mark Reed, Francis Schumacher, and Vivian Shen for the SIUC, Dilip Amin, Barry Arnold, Cory Chan, Martha de Forest, Craig Gleason, Ken Holloway, Don Kipp, Rick Luebs, Peter Meier, Tarang Patil, and Will Walker for the CCU, Nick Fiduccia, Harlan Hill, Tom Hotchkiss, Wayne Kever, Gregg Lesartre, Bruce Long, John Scheck, Craig Simpson, Li-Ching Tsai, and Steve Undy for the TCU, Firooz Amjadi, Donald Benson, Annika Bohl, Ken Chan, Pirooz Emad, Dave Goldberg, Sharoan Jeung, John Keller, Willy McAllister, Vijay Madhavan, Gene Mar, Balbir Pahal, R. Ramasubramanian, Jon Sanford, Homy Shahnifar, Charles Tan, and Yeffi Van Atta for the MIU, and Theresa Corsick, Bev Raines, Binh Rybacki, Kathy Schraeder, Dick Vlach, and Valerie Wilson for mask design.

References

1. J.M. Mikkelson, et al, "NMOS-III Process Technology," *Hewlett-Packard Journal*, Vol. 34, no. 8, August 1983, pp. 27-30.
2. D.A. Fotland, et al, "Hardware Design of the First HP Precision Architecture Computers," *Hewlett-Packard Journal*, Vol. 38, no. 3, March 1987, pp. 4-17.

HP Precision Architecture NMOS-III Single-Chip CPU

by Jeffry D.Yetter, Jonathan P. Lotz, William S. Jaffe, Mark A. Forsyth, and Eric R. DeLano

LIKE ALL OF THE CUSTOM VLSI chips designed for the HP 3000 Series 950 and HP 9000 Models 850S and 825 SPUs, the CPU is designed using HP's NMOS-III fabrication process.[1] NMOS-III was a natural choice not only because it affords the density and speed required for the design, but also because of its proven manufacturability. The CPU chip contains 115,000 transistors packed onto a square die measuring 8.4 mm on a side.

Using the NMOS-III technology and a variety of high-performance circuit design techniques, the CPU design achieves a 30-MHz worst-case operating frequency. This exceeds the system clock frequency required by the Model 850S/Series 950 and Model 825 SPUs, allowing the CPU chip to have high yield and operating margins in those environments.

The chip implements the entire set of 140 instructions and 25 trap and interruption conditions of HP Precision Architecture. Instruction execution is pipelined, and 130 of the 140 instructions execute in a single machine cycle. Instruction sequencing and CPU control are implemented in hardwired logic, rather than in microcode. Hardwired control ensures that instructions are executed in the fewest possible cycles and at the highest possible frequency.

The key design goals were to produce a manufacturable HP Precision Architecture CPU design optimized for sustained performance in a realistic high-performance computing environment. Circuit design efforts were concentrated on the critical timing paths. Within the CPU chip, the execution unit presented some of the most critical speed design challenges. The execution unit produces 32-bit results for the arithmetic, logical, extract, and deposit instructions (see "Execution Unit," next page). Many critical paths extend off the chip through its cache bus interface. These paths are optimized to achieve the highest overall system performance.[2]

Fundamental to the performance of the CPU chip is a pair of high-drive clock buffers which produce a pair of nonoverlapping clock pulses (see "A Precision Clocking System," page 17). Each clock signal supports a fanout of over 8,000 gates with a single level of buffering, a feat that is only possible within the confines of an integrated design. Each component of the CPU derives its timing information from these clock signals, yielding a tightly coupled synchronous CPU design.

Architecture Overview

HP Precision Architecture specifies a register-based CPU architecture in which the registers are the source of all operands and the repository for all results. There are 32 general registers, which are used for local storage of operands, intermediate results, and addresses. Instructions interact with the memory hierarchy via the LOAD and STORE

(continued on page 14)

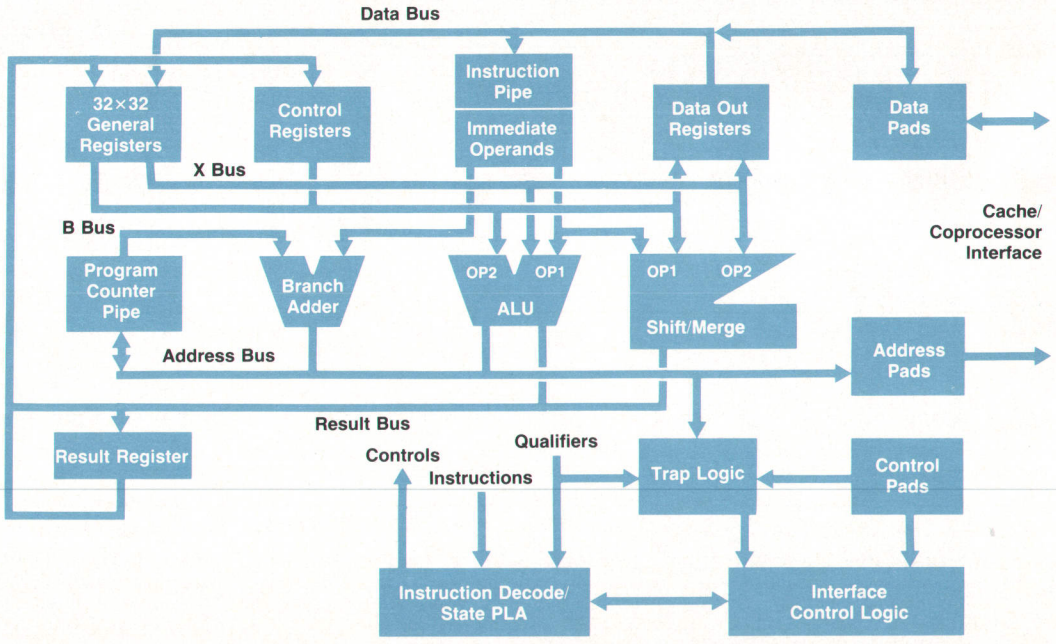


Fig. 1. *HP Precision Architecture VLSI CPU block diagram.*

Execution Unit

The execution unit (E-unit) is the computation engine of the CPU. As its name implies, the E-unit provides the CPU functions of the EXECUTE pipeline stage. Its primary function is the computation of results for the data transformation and branch instructions. Data transformation instructions act on operands selected from the general registers and produce results which are stored back into the general registers. Once a result has been computed, the E-unit must determine whether the condition for branching or nullification has been met.[1] The E-unit contains special hardware which evaluates its result to determine the test condition. The E-unit hardware is outlined in Fig. 1.

Data transformation instructions are of two types: arithmetic/logical and shift/merge. The E-unit contains two specialized components to compute its results: the arithmetic logic unit (ALU) and the shift merge unit (SMU).

The challenge in the ALU design is to provide quick execution of a 32-bit add function. This is the ALU function that requires the most gate delays. Many techniques exist to exploit parallelism to minimize gate delays for addition, but these tend to produce adder designs that are inordinately large. Since the same essential adder that serves the ALU is replicated on the CPU to provide the branch adder, program counter, and recovery counter, such techniques were largely excluded from this design. A relatively simple ALU organization was adopted, and NMOS circuit techniques are employed to provide the necessary speed.

The adder is organized into eight identical 4-bit short carry stages which control an 8-bit-long carry chain. The intermediate carries and the operands are then combined by logic to produce the 32-bit result. Sixteen gate delays are required to transform the operands into a result by this method. Ordinarily, two levels of logic are required to propagate a carry. To speed up the carry propagation, an NMOS series carry chain is employed (Fig. 2). This scheme exploits the noninverting AND function provided by transmission gates, and replaces two levels of logic with a single transmission gate delay. The resulting adder is capable of producing its result within 10 ns of its receipt of operands.

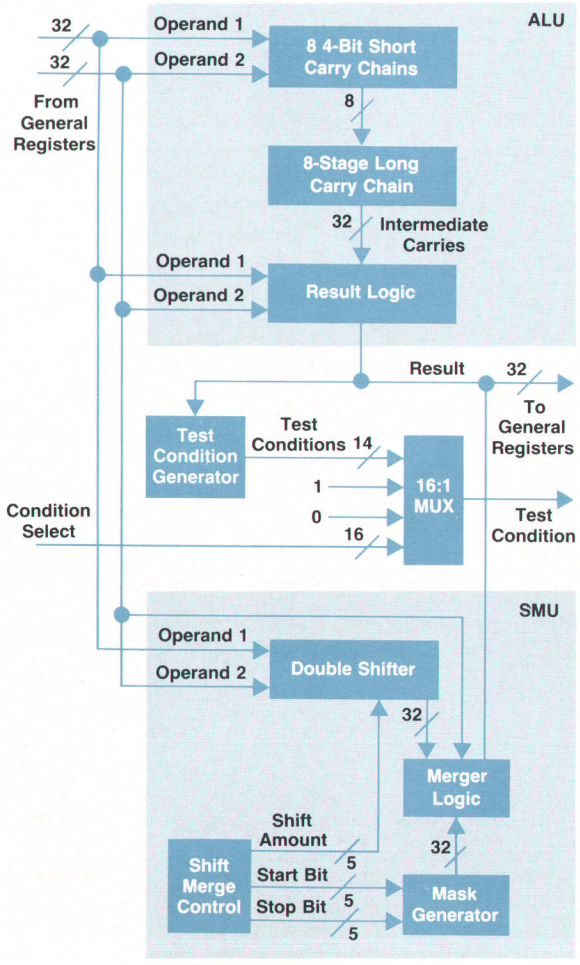


Fig. 1. *Execution unit hardware organization.*

Shift Merge Unit

The SMU consists of three major components: a double-word shifter, a mask generator, and merger logic. The simplest SMU operations are the SHIFT DOUBLE instructions. The source operands are concatenated and then shifted right by a specified amount. The shifter is organized into five stages, each of which is hardwired to perform a partial shift. The first stage optionally shifts 16 bits, the second 8 bits, and so on. Using the shift stages in combination, shift amounts from 0 through 31 bits are accommodated.

The EXTRACT instructions extract a specified field from a single operand, and place it right justified in the result. These use the shifter to perform the justification of the operand. The unused bits of the result are filled with zeros, or optionally sign-extended (filled with the most significant bit of the extracted field.) The mask generator is used to distinguish zero or sign-filled bits of the result from those containing the extracted value. The merger logic then produces the result from the mask and shifter output.

DEPOSIT instructions make up the remainder of the SMU functions. Essentially, DEPOSIT is the inverse of the EXTRACT operation: a right justified field of a specified length is merged into the result register at a specified position. Again the shifter is used to position the specified field into the result. For DEPOSIT, a mask is produced which distinguishes the deposited field from the unaffected portion. The merger logic then assembles the result from the shifter output and the result target (which is itself an operand for DEPOSIT) according to the mask.

The ALU and the SMU share a common result bus which carries their results back to the general registers. Once a result is computed, it is evaluated by the test condition hardware. In

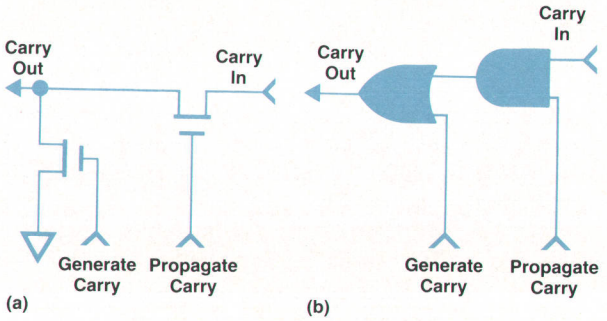


Fig. 2. *(a) NMOS series carry stage. (b) Logic gate equivalent.*

general, 14 nontrivial test conditions may be specified. To speed up the evaluation, all 14 conditions are computed in parallel. The condition specified in the instruction format has been decoded in parallel with the result computation (the code for the condition has been available since the INSTRUCTION DECODE pipeline stage.) The proper condition is selected via a multiplexer circuit, and the E-unit has completed its operation. 40 ns has elapsed since the E-unit first received the operands.

Conditional branch instructions also rely on the E-unit's operation to determine if a branch should be taken. The result of a conditional branch computation is used to determine the branch condition. For ADD AND BRANCH instructions, this result is also stored into the general registers. For other branch instructions, the test condition is computed and the result is discarded.

Reference

1. M.J. Mahon, et al, "Hewlett-Packard Precision Architecture: The Processor," *Hewlett-Packard Journal*, Vol. 37, no. 8, August 1986, pp. 4-21.

Jeffry D. Yetter
Project Manager
Entry Systems Operation

(continued from page 12)

instructions. These instructions have address modification mechanisms which allow code to work on data structures efficiently.

The architecture provides an extensive set of instructions for arithmetic, logical, and field manipulation operations. Primitive operations serve as building blocks for complex operations that cannot execute in a single cycle, such as decimal math and integer multiply and divide. Instructions are also provided to control coprocessors, which perform more complex operations such as floating-point math. Coprocessors can perform their operations in parallel with CPU program execution until data dependencies force an interlock.

The architecture has both conditional and unconditional branches. All branch instructions have the delayed branch feature. This means that the instruction following the branch is executed before the target of the branch is executed. This feature reduces the penalty normally associated with branches in pipelined CPUs. A BRANCH AND LINK instruction is provided to support subroutine calls. This saves the return address of the calling routine in a general register before control is transferred.

The architecture also defines a control feature called nullification. All branch instructions and data transformation instructions can conditionally nullify the next executed instruction. When an instruction is nullified, it executes as a NOP. The effect is logically the same as a skip instruction, except that no branch is required and the CPU pipeline is spared the branch overhead. The architecture also specifies a set of 25 different types of interruptions which cause control to be transferred to interrupt handling routines. Upon completion of interruption processing, a RETURN FROM INTERRUPT instruction returns control to the interrupted routine.

CPU Chip Overview

The major functional blocks of the CPU are shown in Fig. 1 on page 12 and outlined in the chip photomicrograph, Fig. 2. The data path circuitry consists of the functional units required for instruction fetching, operand generation, and instruction execution. All components of the data path are 32 bits wide.

The ALU, test condition block, and shift-merge unit make up the execution unit. The ALU is used to execute the arithmetic and logical instructions, compute the addresses of the LOAD and STORE instructions, and compute target addresses for general register relative branches. In addition, the ALU computes the result to be used in the condition evaluation for most conditional branches. The shift-merge unit is used to execute the set of SHIFT, EXTRACT, and DEPOSIT instructions and to compute the result to be used in the condition evaluation for the BRANCH ON BIT instruction. The test condition block operates on the result generated by the ALU or the shift-merge unit to determine if the condition for branching or nullification is true.

The data path blocks for operand generation are the 32 general registers and the immediate operand assembler. The immediate operand assembler assembles 32-bit operands for the ALU from the immediate fields in the instruction.

The data path blocks for instruction fetching include the program counter and the branch adder. These blocks compute the addresses for sequential and branch target instruction fetches, respectively. Also included in this section is a set of registers to keep track of the addresses and instructions currently in the pipeline.

There is a set of control registers that have a variety of specialized uses. Some are accessed only by software via MOVE TO/FROM CONTROL REGISTER instructions. Some control registers are also controlled implicitly by the CPU hardware. These include the instruction address queue, the interruption parameter registers, the recovery counter,

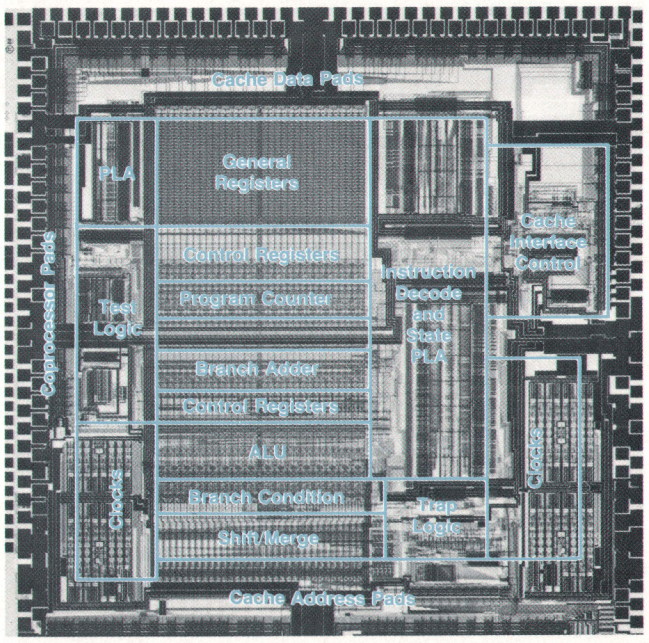


Fig. 2. *Photomicrograph of CPU chip.*

and the interruption processor status word. Other control registers control the generation and servicing of traps and interrupts. The processor status word affects trap generation and other control functions.

The trap logic receives interruption requests from various sources both internal and external to the CPU. Its function is to determine which (if any) of the 25 traps are valid and enabled, and generate a vector address for the highest-priority interruption. It then instructs the CPU to branch to that address, saving the return address in a control register.

The cache bus interface block consists of 32 address pads, 32 data pads, 57 control pads, and the logic required to control them. This interface is used to communicate with off-chip cache, TLB, system interface unit (SIU), and coprocessor chips. Normally, the CPU is the master of the cache bus. An external controller (the SIU) handles copy-in and copy-back traffic between the cache and main memory, as well as multiprocessor cache coherency algorithms (see article, page 4).

Two-phase clock generator circuits capable of driving 500-pF loads with 3-ns rise and fall times are included on the chip. Together, the clock buffers consume nearly 10% of the CPU's available silicon area.

The chip is controlled by a number of programmable logic arrays (PLAs) and a small amount of custom-designed logic. Three large PLAs control the functions of instruction sequencing and decoding, and a fourth PLA aids the CPU in the control of the cache bus interface. Because of the critical timing at this interface, much of its control is delegated to specialized hand-crafted logic.

Testing of the chip is accomplished through a serial diagnostic interface port (DIP). The DIP allows serial shifting of eleven internal scan paths, which access 1,366 bits of internal CPU state. The test logic controls the on-chip scan paths and interfaces to an external tester for serial testing of the chip. The details of DIP operation and the test capabilities it provides are described in "VLSI Test Methodology" on page 24.

Instruction Sequencing and Pipeline Performance

The CPU pipelines the fetching and execution of all instructions. This allows execution of different stages in the pipeline to occur in parallel, thereby reducing the effective execution time of each instruction. Fig. 3a depicts the

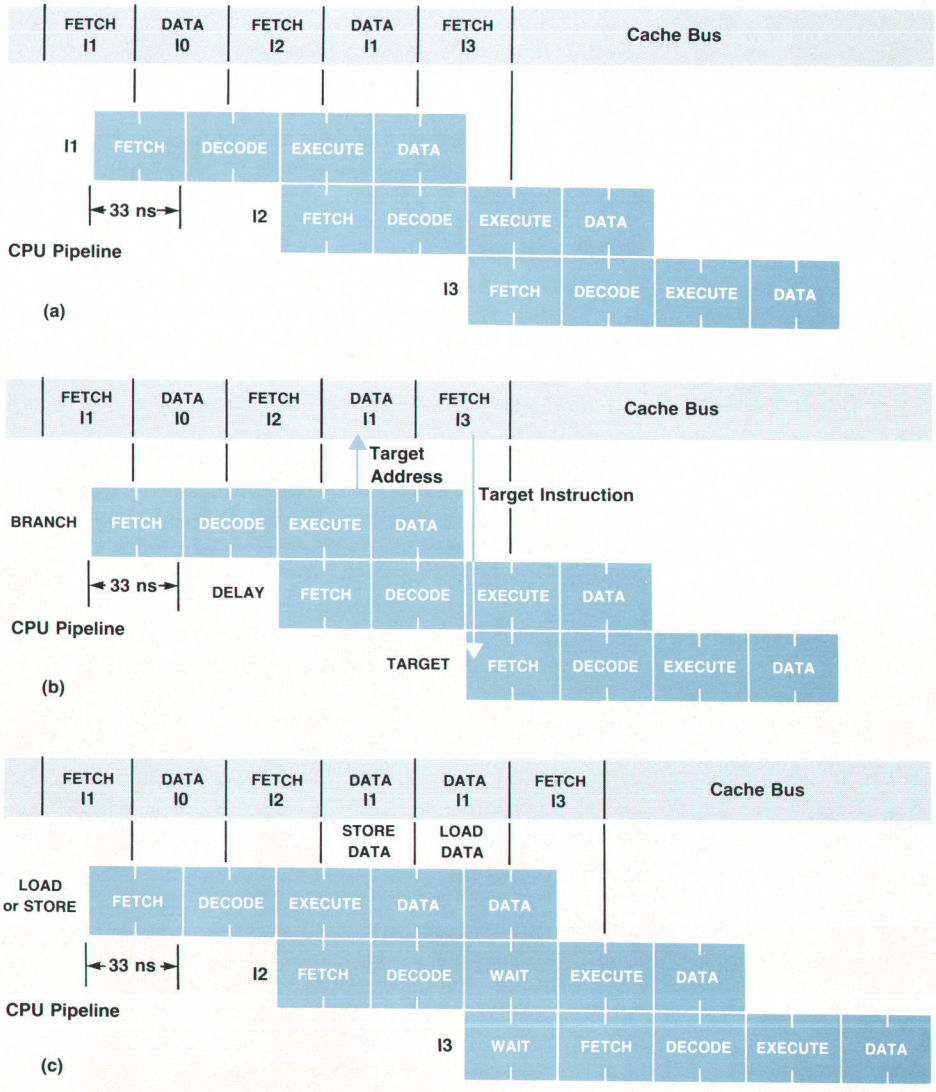


Fig. 3. *Pipelined instruction execution. (a) Sequential execution. (b) Branch execution. (c) Load/store execution.*

pipelined execution of three sequential instructions. One instruction is executed every two clock periods, resulting in a machine cycle time of 66 ns. This allows a peak performance rating of 15 MIPS (million instructions per second). The bandwidth of the cache bus allows one instruction fetch and one data access to be completed each machine cycle.

The pipeline stages executed by each instruction (see Fig. 3a) include instruction fetch (FETCH), instruction decode/operand select (DECODE), ALU operation/data address (EXECUTE), and data access (DATA). An additional one-half clock period not shown in Fig. 3a is required to complete a write to the general registers. Execution of most instructions follows this basic pattern. This allows most instructions to execute in a pipelined manner at the peak rate. Barring pipeline interlocks, 93 percent of all instructions in the set will execute in a single machine cycle. Those that require additional cycles include system control instructions for cache/TLB maintenance, system diagnostics, and interrupt handling.

Instruction address generation is done before the FETCH stage. The address is generated by the program counter for sequential fetches, or by the branch adder or ALU for branch target fetches. Other sources of instruction address include the trap logic and the instruction address queue for vectoring to and returning from an interruption routine. In all cases the instruction address is issued to the cache bus on the phase before the FETCH stage.

Sequential and branch target addresses, however, are actually issued by the cache controller to the cache memory array before the phase preceding the FETCH stage. This allows for a pipelined two-clock-period cache access. Sequential instructions are prefetched by the cache controller which maintains a second copy of the program counter. The sequential instruction address is actually generated three clock periods before the FETCH stage. This allows sequential instructions to be prefetched one machine cycle before they are sent to the CPU. Thus, sequential instruction fetches are completed every machine cycle without a pipeline penalty.

The execution of a branch instruction is shown in Fig. 3b. The branch instruction uses its data cycle on the cache bus to issue the branch target address. This is two clock periods before the FETCH stage of the branch target instruction. Thus, branch target fetches are also pipelined two-clock-period cache accesses. The cache controller receives the computed branch condition from the CPU and uses it to select between the prefetched sequential instruction and the branch target instruction returning from the cache. Hence, taken and untaken branches execute at the peak rate without a pipeline penalty.

On the first phase of the FETCH stage, the instruction and its corresponding TLB miss and protection check flags are driven to the CPU on the cache bus. On the second phase of the FETCH stage the instruction is driven into the chip and set into PLA input latches and the instruction register. During the FETCH stage, the cache controller prefetches the next sequential instruction.

Instruction decoding and operand selection are done during the DECODE stage. During the first phase of DECODE, the PLAs decode the instruction. On the second phase of DECODE, control lines fire to send the general register operands to the execution unit (ALU and shift-merge unit). Also on this phase, the immediate operand assembler sends immediate operands to the execution unit.

The execution unit and the branch adder produce their results during the EXECUTE stage. On the first clock phase of the EXECUTE stage a data address or branch address is valid on the internal address bus. This address is latched at the address pad driver and driven to the cache bus at the beginning of the next clock phase. For conditional branches, the execution unit does a calculation in parallel with the branch target address calculation by the branch adder. The execution unit does a compare, add, shift, or move operation for conditional branches. The result is then tested by the test condition block to determine whether the branch is taken (see "Execution Unit," page 13).

Accesses to the cache bus initiated during the EXECUTE stage are completed during the DATA stage. These occur for loads, stores, semaphore operations, coprocessor operations, and cache and TLB maintenance instructions. Store data is driven to the cache bus on the first phase of the DATA stage. Load data and TLB and coprocessor trap information are received by the CPU by the end of the first phase of DATA. Load data is driven to the CPU data path on the second phase of DATA. Control registers are also set on this state. Load data is set into a general register on the next phase if a trap did not occur.

Execution of LOAD and STORE instructions results in a degradation of performance because of practical limits on the access times that can be obtained with large external cache memories. As shown in Fig. 3c, an additional half machine cycle is inserted into the pipeline to allow sufficient time for the cache memory to complete its access. Additional penalties of one machine cycle are incurred for using data that was loaded in the previous instruction, and for nullification of an instruction.

The total performance degradation incurred from the instruction pipeline (assuming no cache or TLB misses) can be calculated by summing the products of the LOAD, LOAD/USE, STORE, and NULLIFY penalties and their respective frequencies of occurrence. For typical multitasking workloads the average degradation is 0.39 CPI (cycles per instruction), resulting in sustained pipeline performance of 10.8 MIPS. The sustained performance can be increased using an optimizing compiler. All interruptions and degradations to normal pipeline execution are handled completely by CPU hardware in a manner that makes the pipeline transparent to software.

System performance is a function of the operating system, user workload, main memory design, and cache and TLB size and organization. The CPU pipeline and cache implementations are designed to maximize sustained system performance for multiuser computing workloads in a system with a manufacturable and cost-effective large main memory subsystem. In addition, the on-chip 30-MHz synchronous cache bus interface places no constraints on external cache and TLB sizes and organizations. Different main memory systems can also be used with this CPU. The HP 3000 Series 950 (HP 9000 Model 850S) and HP 9000 Model 825 Computers use different main memory, cache and TLB systems for different price/performance trade-offs

A Precision Clocking System

In high-speed synchronous systems, requirements for fast interchip communication are very restrictive. System frequency can be limited by driver speeds, signal propagation time, and chip-to-chip clock skew. System-wide clock skew presents a particularly difficult dilemma. To accommodate the high clock loading requirements of the NMOS chips, a high-gain multistage buffer must be used. Although it is possible to integrate such a buffer system efficiently onto silicon, the buffer delay on each chip, and therefore the system clock skew, can vary widely over the range of possible operating parameters such as supply voltage and temperature. Subtle variations in the NMOS-III manufacturing process compound the problem. It would seem that the high drive requirement is at odds with a low-skew system clock distribution.

To reduce chip-to-chip skew, all chips in our processor system have a local phase-locked clock generator. This circuit ensures that the clock buffers on each chip have a matched delay to within 3 ns over their specified range of parameters.

This circuit is not a phase-locked loop. It does not have a voltage-controlled oscillator. Instead, its operation is based upon a delay element and a comparator. The delay element (Fig. 1a) consists of a capacitor, a precharger, and a pulldown FET. During the precharge cycle, the capacitor is precharged to V_{DL}. When the START signal goes high, the capacitor discharges through the two series FETs. The rate of discharge is controlled by the voltage $V_{CONTROL}$. The lower $V_{CONTROL}$, the lower the gate drive and the longer the delay. Higher $V_{CONTROL}$ produces a shorter delay.

The control voltage is generated by the comparator circuit (Fig. 1b). Its purpose is to move the $V_{CONTROL}$ signal until the SYNC and CK1 (clock 1) edges coincide. If CK1 occurs late, the drain bootstrap circuit that fires DIPUP is allowed to fire. This produces a pulse that allows some or all of the charge on the small dipup capacitor to be shared between the small dipup capacitor and the big $V_{CONTROL}$ capacitor. The voltage $V_{CONTROL}$ will rise and the delay in the delay element will decrease. This means CK1 will occur sooner. If CK1 occurs too soon, the DIPUP circuit will be inactive, and the DIPDN node will pulse. Thus the charge stored on the big $V_{CONTROL}$ capacitor will equalize with the small dipdown capacitor, and the $V_{CONTROL}$ signal will decrease slightly, increasing the delay in the delay element. This circuit is essentially a low-pass switched-capacitor filter.

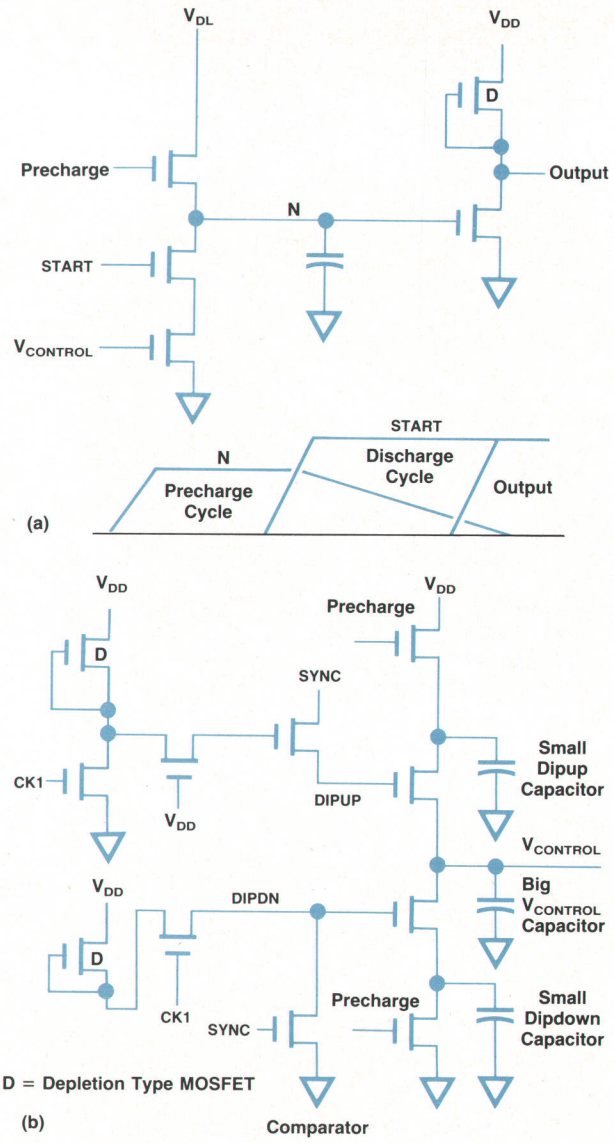


Fig. 1. *(a) Delay element used to reduce clock skew. (b) Comparator circuit adjusts delay element.*

Other Features

Chip capacitive drive is scalable. Each chip has enough output buffers to meet its clock requirements. Each buffer block can drive 75 pF and contains circuitry to reduce ringing caused by path inductance.

Power-up and power-down features are included. The clocks are held low until a circuit detects that the loops are up and stable. Then they are released. This ensures that all chips in a system only have clocks when their loops are correctly placing edges relative to the SYNC signal. This eliminates drive contention on power-up. Similarly, if the clocks fail or SYNC is lost to a particular chip, the clocks are forced low.

William S. Jaffe
Development Engineer
Entry Systems Operation

(see article, page 4).

Summary

The design effort has resulted in a single VLSI component that implements the entire instruction set of a next-generation computer architecture. This CPU can be used in a variety of products spanning a broad range of price and performance by appropriately configuring external cache memory, TLB, main memory, and coprocessors. In addition, high-performance multiprocessor systems can be built using this CPU.

The NMOS-III VLSI process was chosen to implement

this design because of its proven history of high yield and reliability. Manufacturability is enhanced by a worst-case design methodology which ensures reliable operation over the full range of process variation, 0°C-to-110°C junction temperature range, ±10% power supply variation, and worst-case variation of all external components.

Performance was not sacrificed to achieve manufacturability and flexibility. In addition to implementing an instruction set engineered for high throughput and compatibility with next-generation operating systems and compilers, the CPU employs techniques to minimize instruction execution time. Pipelined execution, instruction prefetching, branch target prefetching, multiple internal data paths, and a high-bandwidth external cache interface are some of the mechanisms used to minimize the number of machine cycles required to execute each instruction. A low-skew clocking system coupled with special attention to circuit optimization in the critical timing paths results in a minimum operating frequency of 30 MHz under worst-case conditions.

The design has been proven through extensive functional and electrical characterization at the chip, board, and system levels. Performance has been verified under multiple operating systems running real user workloads.

Acknowledgments

The authors wish to recognize Sean Chapin, Jim Fiasconaro, Dan Halperin, Jerry Kaufman, Joel Lamb, Russ Mason, and Scott McMullen for their contributions to the chip design.

References

1. J.M. Mikkelson, et al, "NMOS-III Process Technology," *Hewlett-Packard Journal*, Vol. 34, no. 8, August 1983, pp. 27-30.
2. J.D. Yetter, et al, "A 15 MIPS Peak 32-bit Microprocessor," *Digest of Technical Papers, 1987 International Solid State Circuits Conference.*

Design, Verification, and Test Methodology for a VLSI Chip Set

by Charles Kohlhardt, Tony W. Gaddis, Daniel L. Halperin, Stephen R. Undy, and Robert A. Schuchard

TEN CUSTOM VLSI CHIPS are used in the processors of the HP 3000 Series 950 and HP 9000 Models 850S and 825 Computers. The complexity of the design, integration, and testing required to deliver ten VLSI chips that meet the functional and electrical requirements for these products demanded close attention to detail throughout the development program.

The strategy used was to set the expectation that the initial lab prototype system built with revision 1.0 silicon was to provide a high level of functionality and electrical quality. A high level of quality implied that the operating systems would boot and that there would be no first-order bugs that would preclude characterization beyond them. With this goal met for revision 1.0 silicon, electrical and functional characterization were to proceed covering all aspects of individual chips, boards, and systems. With the lab prototype characterization complete, the chips would then be released for production prototyping with the expectation that the chips would be production quality.

Tactics to accomplish this strategy were then defined. The first aspect was how to deliver first silicon that would meet the lab prototype quality level. This was the design methodology. The lab prototype evaluation tactics were to define feedback paths required for complete functional and electrical characterization of the chips, boards, and systems. The production prototyping evaluation tactics were to repeat those feedback paths where changes had been made or risks of introducing a new bug were present.

Within the design methodology, heavy emphasis was placed on both worst-case circuit design and functional verification. Functional verification included both FET switch level simulations and behavioral level simulations at the boundaries of each chip. System level models were constructed and behavioral level simulations were performed to verify some of the first-order chip-to-chip transactions. This design methodology will be elaborated on in a later section of this article.

With the first release of the VLSI designs for fabrication, the characterization feedback paths were defined. Project teams were assigned to these feedback paths with responsibility for definition of the plans, development of any special tools, and execution of the plans. We will elaborate on three of the specific feedback paths that were fundamental to the program success.

One of the primary feedback paths was associated with the functional verification of the lab prototype hardware. Since the operating system development was staffed in parallel with hardware development, the hardware design team internalized a significant portion of the functional verification. This allowed the operating system development to proceed relatively independently of the hardware verification. Since hardware/software certification was late in the program schedule, having early functional verification was essential. In addition, special attention could be

placed on corner case (rare event) testing, where operating system tests are written and executed with a different objective. A special operating system called "Khaos" was developed, along with 87K lines of code which stressed the corner cases of this chip set. The tools and verification results will be discussed in more detail in a later section of this article.

Two other areas of feedback were the electrical characterization of the individual chips and the integration and characterization of the VLSI at the processor board level. A scan path test methodology was developed which was applicable to both the chip and board level characterizations. The concepts of that methodology and the results of its application will be presented.

Electrical Design Methodology

One of the key aspects of the design methodology for the chip set was the ability to design for performance. Achieving high system clock rates was an important contribution of the design effort. There are three major facets of the methodology that address high-performance design: structured custom design approach, worst-case design and simulation, and modeling of the chip set in the package and board environment.

The typical chip has a regular structure. This is a result of the use of separate data path and control structures to implement the chip's functionality. The data path consists of registers, adders, comparators, parity trees, and other regular blocks. The control structures are implemented as PLAs (programmed logic arrays). This approach is typically referred to as structured custom design.

Each block of the data path, such as a register or adder, is designed on the basis of a single bit and then is repeated to form a 32-bit-wide block. The global busing passes over the top of each "bit slice" and is an integral part of the structure. Multiple instances of the registers and other blocks are then combined by abutment into the register stack. This arrangement of blocks gives a high packing density within the data path.

The PLA structure makes it possible to implement random logic and state machines in a regular fashion. The control of each chip is specified in a high-level language and is converted into the PLA structure. The control structures were refined and debugged throughout the design phase of the chips and were typically the last structures to change on the chips. This process allowed the PLA structures to change quickly and provided a dense implementation of the control logic.

These regular design structures in the data path and control logic helped maximize density and made it possible to leverage blocks in cases of multiple instances of a single block in the data path. They also offered the capability to design for high performance. With such a regular structure, it was possible to set accurate timing budgets for the critical paths and design the blocks of the data path and the PLAs to meet the timing requirements. Other design styles, such as gate arrays and standard cells, do not offer these benefits since they do not have the same regular structure. By managing the timing on a block level, maximum performance was obtained.

To guarantee that maximum performance is maintained over all operating extremes, it is necessary to use a worst-case electrical simulation methodology in addition to the structured custom approach. The design of the blocks typically started as a schematic which represented the circuit before the artwork design. In addition to the MOS transistors, capacitors and resistors were included to estimate the parasitic effects of the artwork. The schematic was then electrically simulated, and the sizes of the devices were adjusted to meet the timing budgets. The circuit was simulated using worst-case models of the devices based on IC process spread, temperature extremes, voltage margins, and noise effects.

Once the designer was confident in the design, artwork was created for the circuit. The worst-case capacitance for the artwork was extracted and substituted back into the circuit, and electrical simulations were rerun to verify the performance of the artwork. This resulted in blocks for the data path that met performance requirements over the entire range of operation.

The worst-case methodology was effective for the internal blocks of the chip. The pad drivers and receivers were given an equal level of attention.

Package environment effects include the inductance of the bond wires and the noise on the power supply planes caused by the high-current spikes of driver switching. The board environment behaves like a transmission line at the frequency of operation. This environment required construction of a complex model of the chip, package, and board. This model was simulated using the worst-case conditions described above. The effects were accurately predicted and the bus designs met their design specifications.

The structured design approach, worst-case design and simulation methodology, and careful simulation of the system environment were very effective. The first releases of the chips worked at the target frequency in the system with adequate margins using typical parts.

Design Verification Methodology

When a system consisting of ten chips is designed, serious functional bugs will adversely affect the integration schedule. The process of fixing a bug, releasing the chip, and fabrication takes on the order of two months. If several of the chips had had functional bugs that precluded product integration, then the schedule to having working systems would have slipped drastically. Therefore, the goal was set to have first-revision chips that met all of the requirements of the lab prototype system including booting the operating systems. Since the ten chips represent ten individual functional algorithms and there are four major bus protocols connecting the chips, the verification problem presented no small challenge.

To meet the goal, behavioral models of the chips were written to describe their functionality. These models used the same high-level schematics that were being used to construct the chips. For each chip, the same source that was used to generate the control blocks described above was used to generate the description of the control logic for the model. Behavioral descriptions were written for all of the other blocks based on the schematics and functional descriptions. By writing a behavioral model it was possible to have a model for each chip long before the artwork for

the chip existed. The model also made it possible to connect all of the chips into a larger system model.

Because of the difficulty of generating all of the corner cases for a system model including all of the chips, a significant amount of simulation time was spent verifying individual chips. A high-level language was used to specify the corner cases and was compiled into the input source code for the functional simulator. The high-level compilers were Pascal programs written by the verification teams to increase their productivity in generating tests. To ensure good coverage, numerous reviews were held of the corner cases that were executed.

The chip models were put into a complete system model as soon as they were capable of executing their basic functions. The test cases for the system model were written in HP Precision Architecture assembly language. A source of tests was the set of architecture verification programs written at HP Laboratories as part of the architecture definition. These tests covered the functionality of the instruction set. In addition to these tests, other code was needed to provide coverage for areas specific to this implementation, such as the CPU instruction pipeline, the bus protocols, and error handling.

To ensure that the artwork would match the behavioral model, the test cases and results were extracted from the behavioral simulator and run on a switch-level simulator. This simulator represented the transistors as switches, with each switch given a strength rating for drive contention resolution. For each chip, the artwork was used to extract the transistor network for the switch model. When this step of the testing was completed, it was possible to guarantee that the chips would function exactly like the behavioral models.

In the operating system turn-on and reliability testing, only five functional bugs were found, and these required only minor workarounds to complete the testing effort. Fig. 1 is a summary of the resources needed for three of the chips in the system. As can be seen, the effort required for verification was substantial. The resulting accelerated schedule for system integration and time to manufacturing release of the chips more than paid for the time invested.

Hardware Functional Characterization

There were several functional areas where it was impossible to provide extensive coverage during the design phase because of the speed limitations of the simulators. Instruction sequences and system exceptions such as cache misses and traps are examples where the combinatorial possibilities were too high to cover all corner cases in simulation. These cases typically would not be tested until substantial operating system qualification was under way. The schedule made it unacceptable to wait until operating system testing was completed, and this testing would not cover cases such as error recovery in the system, so a different approach was required to ensure that the chip set met manufacturing release quality.

The coverage goal for manufacturing release required a methodology that would accelerate multiple occurrences of corner case events in the system in much the same way that an operating system would do under heavy job loading conditions. To do this, a test operating system, called Khaos, was defined and written. This system consists of a set of multiple-priority scheduling queues and a set of trap handlers that were used by code to handle exception cases in a controlled fashion. With Khaos, test suites could be compiled out of specific test programs and the queues managed to regulate the interaction of the programs to ensure random, thorough coverage of the events of interest. Khaos also provides observability functions for debugging test code and diagnosing hardware errors. These functions were supplemented by the use of logic analyzers.

To test the processor board, the architecture verification programs mentioned above were used, in addition to other code. The code was all written in assembly language to ensure the control needed of the features of the architecture. One group of code that was written exhaustively covered areas such as the code and data protection mechanisms and traps on all instruction classes. Another major group of code was referred to as the "thrashers." These pieces of code went into the cache and TLB subsystem and changed their states so that there would be a high level of misses and traps during the execution of other processes. Still another group of code covered the error recovery capabilities of the cache and TLB subsystem. The code used diagnostic instructions defined for this implementation to seed errors which would appear during the execution of other programs. The programs checked to ensure that error cases were handled properly and that execution resumed normally.

To test the I/O and memory subsystems, two major groups of code were written. One group of code was used to create a high level of traffic through the subsystems and exhaustively execute the associated protocol. HP Precision Architecture assembly code was again used to control the architectural features. The C drivers from the operating system were leveraged to test the I/O channel. The other group of code seeded errors, in addition to using a protocol verification hardware board that interfaced with the bus.

These individual programs were combined into test suites and statistics were gathered to determine a run time for high-confidence coverage of the cases that were being

System Design Statistics

Number of Unique Chips	10
Number of Transistors	758,000
Design Schedule (Start to Last Chip)	28 Months
Resources	150 Engineering Years
Frequency	30 MHz
Technology	1.5-μm NMOS in Pin-Grid Array Packages

Chip Specific Statistics

Chip	Circuit Design Resources*	Verification Resources*	Schedule	Number of Transistors
Cache Controller	70	38	10 Months	72,000
TLB Controller	38	20	9 Months	55,000
Math Interface	50	45	7 Months	66,000

*Engineer Months

Fig. 1. *System design statistics and resources for three of the VLSI chips.*

tested. A statistical model was used and most of the suites were run for a period of 8 to 14 hours on the lab prototype hardware. The results were very successful. All of the bugs found during operating system testing were also found by the verification effort. In addition, corner case bugs involving error recovery were located which would not have been uncovered by the operating system qualification effort. In fact, one of the last bugs found in error recovery was a result of the interaction and sequence of six distinct events.

After all of the other characterization was completed on the lab prototype and the chip set was released with bug fixes, the test suites were reexecuted on the production prototype systems. The results of this testing did demonstrate that the chip set was of manufacturing release quality. The entire testing effort required 110 engineering months and 87 thousand lines of code. The result greatly shortened the time to reach manufacturing release and provided high test coverage of features.

Electrical Characterization

As part of the design of these chips a comprehensive test methodology was developed. This included internal chip components as well as external test hardware and software. The essence of the methodology is the use of serial scan paths. The scan paths are placed at the chip I/O pads and at several other places internal to the chip. Use of this methodology has been applied at the wafer, package, board, and system levels. For a description of the methodology, the hardware, and the software, see "VLSI Test Methodology," page 24.

A wafer and package tester was designed for chip characterization and production testing. A board tester was designed for initial turn-on, integration of the subassemblies, and board level characterization. System testers were also developed for initial system turn-on in the absence of a complete operating system and I/O assemblies.

The following sections will elaborate on the application of the test methodology and the results for the ten VLSI chips and processor board characterization.

Chip Electrical Characterization

A chip electrical characterization methodology was developed to be consistent with the overall strategy of producing production-quality, shippable VLSI parts to meet product schedules. This methodology was standardized and applied uniformly to the ten VLSI components.

The final measure of a chip's manufacturability is its yield in production. The total yield, Y_t, is defined as:

$$Y_t = Y_s Y_f$$

where Y_f is the functional yield, or the percent of all die that are functional at the nominal operating point, and Y_s is the survival yield, or the percent of functional die that survive worst-case operating points.

Y_f is a manufacturing responsibility and is closely linked to defect density. Y_s is the responsibility of the VLSI design team and goals for survival yield were set at the outset of the design effort. The purpose of chip electrical characterization is to demonstrate that the chip meets its survival goal under worst-case conditions.

The range of operating variables to be covered by this methodology included characterization over the system voltage, frequency, and temperature limits with sufficient margin to ensure operation beyond these limits. In addition, the full range of fabrication process deviations was considered.

Voltage, frequency, and temperature can be varied during the test. To provide the process variation, characterization runs were fabricated, with key process parameters varied within a single run. Automatic parametric test data collected from parametric devices located on each die provided the process correlation required to identify worst-case characterization parts to be used for electrical characterization. This allowed correlation of chip yields with process parameters.

The test set for each chip was developed and refined over time to provide tests for all circuits and paths on silicon. The test set had several uses between first silicon and final production parts. It provided initial screening of new silicon to get good lab prototype parts into systems as soon as possible. In general, the time from wafers out of fabrication to assembled and tested packages was less than four days. The test set was instrumental in full characterization of each chip between lab prototype and production prototyping phases. Finally, it has evolved into the tests required for the manufacturing process.

The test set for each chip contained several different types of tests to ensure complete coverage and full characterization. Block tests were generated to test individual circuit blocks on the chip, for example an ALU or comparator within the register stack. Pin vectors were automatically extracted from full-chip simulation models to provide additional test coverage from the chip pads. Ac input timing tests were added to test the speed of input receivers at the chip pads. Output driver and functional tristate tests completed the tests used to characterize the pad driver functions.

Typical chip electrical characterization consisted of exercising a statistically valid sample of parts from a characterization run with the chip specific test set over voltage, frequency, and temperature ranges. The result was a data base of failure information for each chip stored off-line to be analyzed and manipulated later with a software tool designed specifically for that purpose.

Results drawn from the characterization data base were used in locating and diagnosing yield-limiting circuits on the various chips. Fig. 2 shows a block yield histogram for all of the block tests on a given chip. This tool, combined with detailed test vector output, allows the isolation of a circuit problem down to the offending circuit cell. Fig. 3 shows how circuits that were sensitive to process variations were identified. Circuit block failure data is combined with the automatic parametric test data to arrive at block yield versus any process parameter collected from chip parametric devices.

These electrical characterization results for each chip were carefully evaluated and circuit fixes included in subsequent chip releases to meet the production survival goals initially set. In terms of the initial survival yields from the lab prototype releases, improvements of 1.15 to 2.0 times have been observed with production prototyping revisions

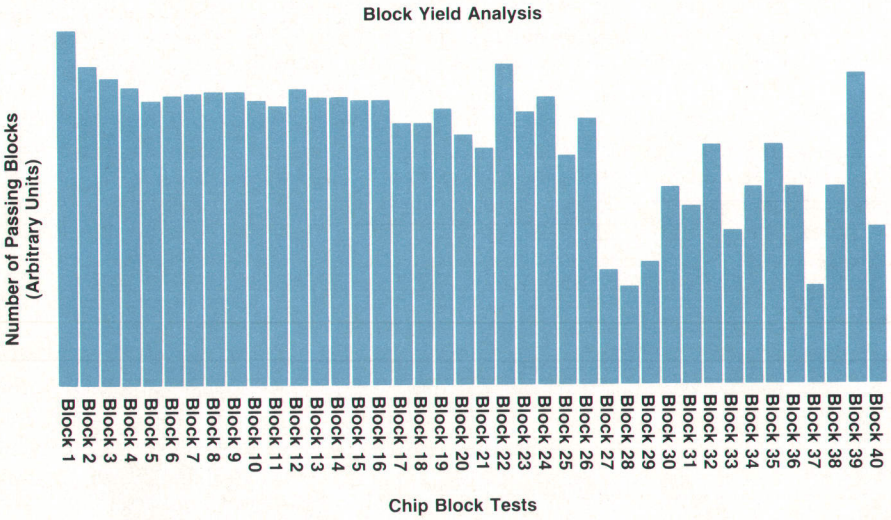


Fig. 2. *Typical block yield histogram for all of the blocks on a single chip.*

of the VLSI chips.

Processor Board Characterization

The basic goal of the processor board electrical characterization effort was to determine the operating range of the processor boards over frequency, voltage, temperature, and VLSI process spreads. In addition, it was desired to provide feedback on any chip deficiencies to the VLSI design groups as quickly as possible. This minimized the possibility of incurring additional VLSI turnarounds to meet performance goals. This effort also supplemented individual chip characterization and provided insight into the interactions between chips, clocks, and boards.

Two different types of tests were developed to stress the VLSI parts during the characterization effort. One type consisted of BVF (block vector file) vectors, which used the chip's scan paths to stimulate the chips and observe the results. The second type consisted of cache-resident tests. The cache-resident tests were HP Precision Architecture code written to test some function, determine if the test was successful, and store the results in the CPU's general registers. To execute a cache-resident test, the instructions were loaded into the cache using the board tester and the CCUs (cache control units). The code was then executed by the processor board and the board tester was used to observe the state of the registers in the processor to determine if the test had passed. Cache-resident programs allow the processor board to run by itself without interfacing to the memory and I/O subsystems.

The tests were run under a variety of conditions. Environmental chambers were used to determine the effect of temperature on processor board performance. A large amount of the testing was done with socketed performance boards so that VLSI parts could be changed quickly. Characterization parts used in socketed boards allowed us to study processor board performance with the full spread of VLSI parts. Once the worst-case combination of parts was determined, boards were built with these parts soldered in to remove the effects of sockets from the results. Voltage-versus-frequency shmoo plots were generated to determine the effects of these parameters on the various tests. Whenever possible, these parameters were varied until the board failed.

The cache bus is the backbone of the processor board, serving as the path for data and control between the VLSI parts. Tests were written to test the noise immunity and speed of the cache bus.

It was determined early in the project that the worst-case cache bus noise occurred on a line that was electrically high while its neighbors were driven low. Therefore, six BVF tests were written, one test for each of the chips on the cache bus. The tests are "walking ones" tests, in which each cache bus line in turn is kept electrically high while all other cache bus lines are driven low. For each of the tests, one chip drives the bus and all chips check that their receivers have the correct value. It is necessary to write and read the contents of the scan paths of all the chips on the cache bus for every vector that is sent across the bus during the test. This effectively limits the time between

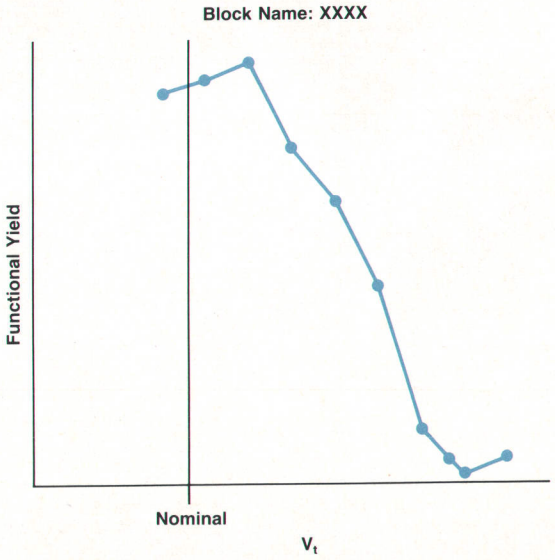


Fig. 3. *Typical plot showing sensitivity of chip yield to parameter variation.*

vectors to many thousands of states. A cache-resident test was written that generated some near-worst-case patterns on the cache bus. The BVF tests allowed us to determine exactly which line or lines failed a test and thus provided good diagnostics. The cache-resident test fired patterns on the cache bus every state and better tested the ability of the power supplies to handle changes in load current.

An additional 46 BVF tests were written to test critical speed paths on the cache bus. For each of the tests, the path is driven electrically low and the resulting value is verified at the receiving chip. The speed path tests allowed us to verify the performance margin of the cache bus. Since specifications for cache bus speed include clock skew, propagation delay, and driver and receiver delay, the cache bus speed tests allowed us to verify the interaction of these specifications.

In addition to testing the chip-to-chip communication paths of the cache bus, it was necessary to test paths involving asynchronous devices, namely static RAM chips. The RAM arrays play a large role on the processor board, forming the cache memory and translation buffer memory. Since the VLSI devices connect to these RAM arrays directly, electrical characterization of the RAM address drivers, RAM data drivers and receivers, and various related internal speed paths was essential.

Two methods were used to test these critical paths for speed and noise margins. The first method used BVF tests to exercise a chip's data paths and control structures along a sensitized asynchronous path. Typically, RAM address drivers were loaded and directed to drive on the first step of the test. The data from the previously loaded RAM array was received and latched in various registers on the clock cycle immediately following. After the test, the contents of the receivers and registers were examined to determine if they were the expected values. The internal paths tested included comparators, parity trees, and PLAs.

The second method used cache-resident code. Programs were written to stress the RAM interfaces. These programs were geared to test either the translation buffer array or the cache arrays.

Typically, alternating address and data patterns were issued to and received from the RAM arrays. After execution of the program, registers on the processor board were examined to determine the results or failure mode of the program. These tests covered circuitry on all of the processor board VLSI chips.

The lab prototype processor boards were fully functional over most of the specified operating range with nominal parts. Nevertheless, fifteen margin problems were uncovered during electrical characterization that could occur with a worst-case combination of VLSI parts, frequency, power supplies, and temperature. Six of the problems were speed problems. The speed problems were evenly divided between the cache bus and the RAM interfaces. Another six of the problems were caused by power supply sensitivities. Two of the problems were caused by non-VLSI circuitry on the processor board. One problem was caused by the on-chip clock circuit which was shared by all the chips.

When a problem was discovered, the information was forwarded to the chip design group. The processor board characterization team worked with the chip group to make sure that the cause of the problem was understood. A bug report was then generated which described the problem along with any pertinent information so that all groups were made aware of the problem. The chip group used this information as well as other feedback paths to ensure that the next revision of the chip was of manufacturable quality. Meanwhile, the electrical characterization team made sure that other problems were not masked by any already discovered problems.

When production prototype boards were available, the full set of tests run on the lab prototype boards was repeated. The operating margins of the production prototype boards were significantly improved over the lab prototypes. Fig. 4 shows an example of the lab prototype electrical quality and the improvement observed with the production prototype version of the processor board. In all cases, the production prototype boards work with worst-case combinations of VLSI parts, frequency, power supplies, and tem-

(continued on page 26)

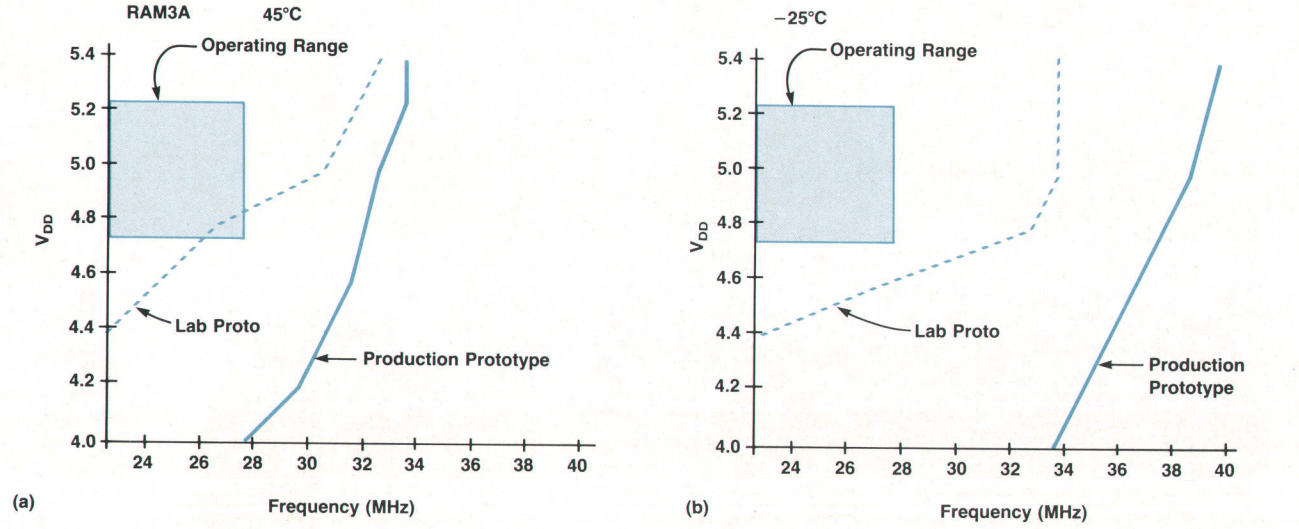


Fig. 4. *Typical improvement of production prototype operating margin over lab prototype.*

VLSI Test Methodology

An integrated test methodology was developed for the rapid turn-on and characterization of the VLSI chips, boards, and systems described in this issue. The methodology provides solutions for each of the three tightly coupled components of the VLSI test problem—design for testability, test systems, and test vector generation.

Design for Testability

The design for testability methodology is a serial test method in which only a subset of the memory elements on each chip are scanned, thereby reducing test circuit overhead. The key aspects of the on-chip design for testability methodology are:

- Common diagnostic interface port (DIP) to provide a uniform interface with the tester independent of the chip's normal system interfaces.
- Access to control and data sections using scan paths placed only at the key interfaces.
- Single-step testable circuits
- I/O pads testable via the DIP.

Fig. 1 shows a simplified block diagram of a typical chip with DIP, test PLA, and scan paths. The DIP and the test PLA are the core of each chip's test circuitry. They multiplex up to 16 serial scan paths and control chip operation. The DIP uses four dedicated I/O pads to implement a common protocol for shifting the on-chip scan paths and for controlling test operations. The important point about the protocol is that one of the pads, the strobe pad, provides the clock for shifting scan paths into and out of the data pads at a rate determined by the tester. This means that the tester data rate does not limit the system clock frequency and permits low-cost tester implementation. A scannable register within the DIP holds a 9-bit command word which specifies a particular test operation to be performed.

The test PLA has two primary functions. First, it controls the DIP hardware to implement the interface protocol. Second, it decodes the DIP command and generates the control signals required to perform the test operation. Each chip must implement basic commands to shift one of up to 16 scan paths, to halt or freeze the state of the chip, and to single-step chip operation. Since the test PLA is automatically generated from a high-level description, additional test operations are easily added for a particular chip.

Most of our chip designs are partitioned into separate data and control sections. The data section or data path consists of custom functional blocks which communicate via local and global buses. The control section is implemented with large synchronous PLAs. Complete testability of the PLA control section is achieved by fully scanning all inputs and outputs. This allows us to halt or freeze the PLA sequencing and to test the PLA array. Testability of the data path is achieved with one or more scannable registers which provide read and write access to each bus in the data path. Any functional block that is not directly scannable is testable because the global buses are controllable from scannable registers and the block's control lines are controllable from the PLA scan paths. Control lines are fired to transfer data from the scannable register to the block under test, perform the desired function, and return the results to another scannable register.

Single-step testing requires that each circuit in the chip be haltable and steppable so that scan operations can be performed without altering the chip state. In NMOS designs with dynamic circuits, it is not possible to stop the clock to halt circuit operation. In this case, each circuit must have an idle state or combination of control inputs that causes the values of any memory element to remain constant. In addition, each circuit must be able to enter and exit that idle state cleanly to ensure that single-step operation is the same as free-running operation. The result is the ability to halt a free-running chip or system. Once the chip or system is in the idle state, the state sequence can be altered to perform

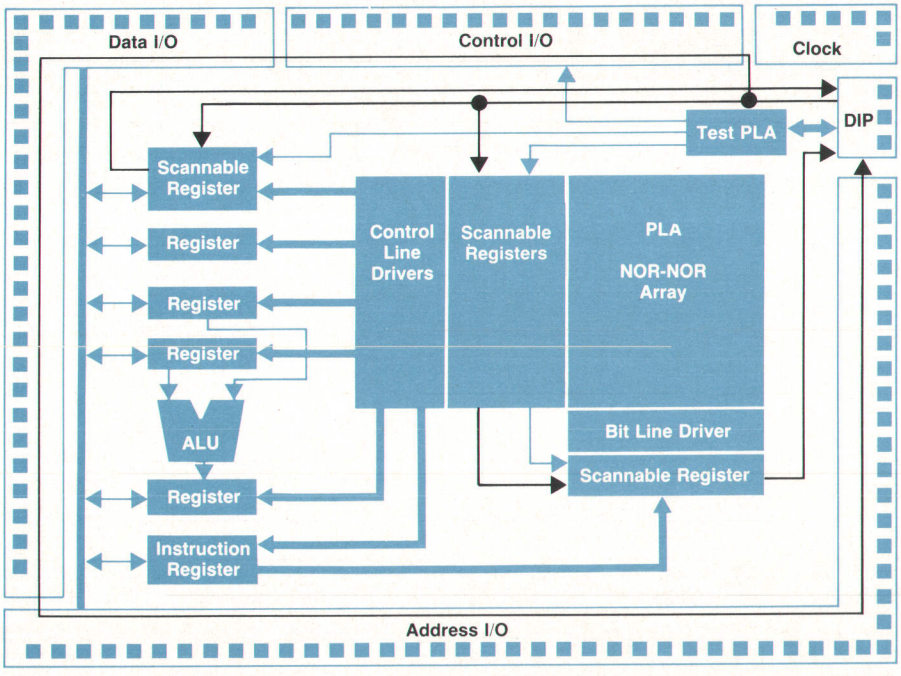


Fig. 1. *Design for testability hardware of a typical chip. Black lines are scan paths.*

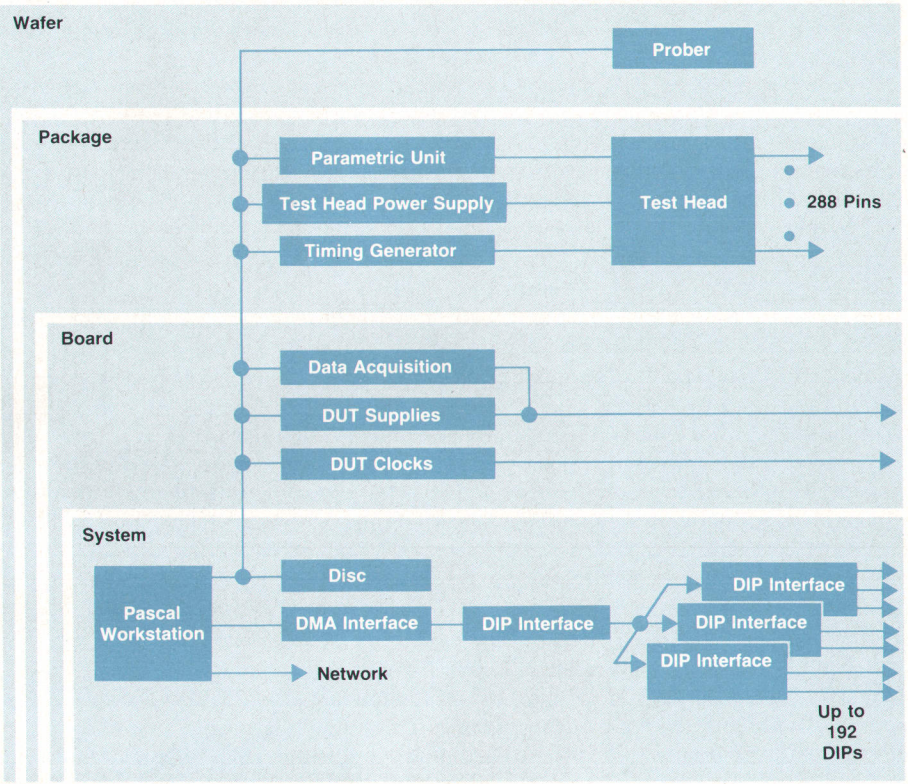


Fig. 2. *Test system family.*

a wide variety of test operations such as testing individual blocks or simply observing the current state and resuming operation.

At-speed functional testing of the pad driver and receiver circuits is controlled by a separate I/O scan path. During test operation, the data from the driver scan latch drives the pad circuits while the receiver scan latch captures the result. In a board or system, the I/O scan path circuits enable both electrical and functional analysis of system bus transactions and the emulation of signal responses from uninstalled chips or subsystems.

The on-chip test circuits require <10% of chip area and <8% of the chip's power. To ensure that the DIP does not limit yield or performance, it is designed using conservative design techniques and for 45-MHz operation.

Test Systems

An integrated family of testers was developed to meet the test system requirements for wafer, package, board, and system testing. (Fig. 2). A common tester operating system developed on an HP 9000 Pascal workstation provides a uniform user and test vector interface with special emphasis on interactive test and diagnostics. Any command can be executed from the keyboard or compiled into a test program. Test vectors can be leveraged at each testing phase.

The test system hardware consists of an HP 9000 Series 200 controller, a set of HP-IB (IEEE 488/IEC 625) instruments, and custom DIP and test head circuitry. The simplest version is the system tester. It consists of the system controller and functional test hardware for the serial test of up to 192 chip systems. The board tester version provides power and clocks for the board under test, and the package tester adds a 288-pin test head with parametric test capability and timing generators for input testing. Finally, the wafer test system adds a cassette-to-cassette wafer prober with optional microprobe capability. The network interface is used to transfer vector files and store functional test data for off-line statistical analysis.

At the board and system level, the test interface is implemented by connecting the DIP signals for each VLSI chip to the tester DIP interface hardware. The job of the DIP interface is to synchronize the DIP operations to make it possible to halt or single-step an entire board or system. This also gives us access to and control of all the buses in the system.

Test Vector Generation

The test vector generation process uses a divide-and-conquer approach to manage the complexity of the problem. The chip is partitioned into independently testable functional units or blocks. A register, an ALU plus the operand registers, or a PLA are examples of blocks. Block tests are the independent tests generated for these blocks in terms of the block ports and are written in a high-level test language. Block tests are generated in three ways: manually generated by the block designer, leveraged from the simulation vectors used in the design phase, or in the case of PLAs, automatically generated using a stuck-at fault model to ensure fault coverage. A set of tools was developed to compile the block test into serial DIP commands in the form required by the tester. These tools also provide translation to and from simulators for the verification and generation of block test vectors.

Don Weiss
Project Manager
Colorado Integrated Circuits Division

(continued from page 23)

perature over the fully specified range of the products with significant margin. The same tests used in the board characterization effort are also used in board manufacturing and failure diagnosis.

Conclusion

The design methodology to achieve the first release of silicon resulted in a design cycle of 10 months or less. Our cycle time response for mask generation, IC fabrication, assembly, and test was less then five weeks. A processor board was integrated and running cache-resident code within two days of delivery of the last component.

The test methodology allowed partial integration and turn-on of the processor board as well as at-speed electrical characterization, independent of the rest of the system components. The level of functionality obtained with the lab prototypes resulted in completion of the HP-UX boot for the HP 9000 Model 825 processor in less then one month from the delivery of the last component. Functional bugs encountered in the evaluation phase were minor. In a few cases the operating systems were required to patch these bugs, but the patches were trivial in nature.

In general, the electrical quality of the lab prototype hardware resulted in systems that operated with margin around the nominal operating point. In characterization, under worse-case conditions of voltage, temperature, and normal process variations, some margin problems were identified. These results were consistent with the original strategy that was set for the lab prototype version of the VLSI.

The above methodology proved powerful, in that five chips of the ten were released to manufacturing with only two revisions. The remaining five chips, which required a third release, provided functional and electrical quality that allowed the system integration to proceed electrically, mechanically, and functionally according to schedule requirements. The CPU chip was released to manufacturing as revision 3.0. The CPU's first revision served as a vehicle to demonstrate the design and test tools. This release was before the time the cache system definition was complete, and as such, required a second release to achieve lab prototype quality. Two of the remaining four chips required a third revision for improved manufacturing margins, while the remaining two required a third revision for functional bugs.

Acknowledgments

Many people contributed to this effort. The following list includes those who have not been recognized in other papers in this issue: Jeff Kehoe, Bob Novak, Paul Perez, Eshan Rashid, Julie Van Gelder, Doug Quarnstrom, John Spencer, Bill Hudson, Bob Naas, Fred Gross, Akshya Prakash, Chick Webb, Hai Vo Ba, Daryl Allred, Keith Erskine, Bill Freund, Mercedes Gil, Susan Hartfeldt, Vinh Nguyen, Gayvin Stong, Leon Cluff, Mike Stahl, Jeff Rearick, Marylou Lillis, Tom Walley, Jeff Beauprez, Don Novy, Matt Reprogle, Greg Halac, Bob Proulx, Virgil Hebert, John Brewer, Frank Galneder, Mary Mirabel, Mark Stolz, Yeffi Van Atta, Dilip Amin, Rick Butler, Ray Gratias, Garry Petrie, Zemen Lebne-Dengel, and Joe Fucetola.

The authors acknowledge Denny Georg for providing the management commitment that was essential to achieving this level of quality. We also acknowledge Mark Canepa, Bob Headrick, Cliff Lob, and John Wheeler for leadership in the definition and execution of this program.

There was a tremendous effort by many other divisions within HP. This includes those that provided the design and verification tools and support, and the manufacturing divisions, which provided mask generation, silicon fabrication, testing, and packaging prototyping.

A Midrange VLSI Hewlett-Packard Precision Architecture Computer

It's designed for mechanical and electrical computer-aided design, computer integrated manufacturing, real-time control, and general-purpose technical applications.

by Craig S. Robinson, Leith Johnson, Robert J. Horning, Russell W. Mason, Mark A. Ludwig, Howell R. Felsenthal, Thomas O. Meyer, and Thomas V. Spencer

THE GOAL ESTABLISHED for HP Precision Architecture computers was to provide a scalable set of hardware and software with the flexibility to be configured for many different applications in a wide variety of market areas. The HP 9000 Model 825 (Fig. 1) is a midrange, compact, high-performance NMOS-III VLSI implementation of HP Precision Architecture. The wide range of system components available in this architecture are all compatible with the Model 825. These include operating systems, languages, graphics, networking, and a wide vari-

Fig. 1. *The HP 9000 Model 825 is a midrange, compact, NMOS VLSI implementation of HP Precision Architecture. It is designed for both single-user workstation and multiuser applications running the HP-UX operating system. The SPU is the middle unit at left.*

ety of peripherals. Also, because adapting to established environments and easy porting of existing applications are of vital import, the Model 825 has been designed in accordance with international standards wherever possible.

User Requirements

The definition of the Model 825 was driven by requirements from several application areas. As a high-performance graphics workstation for mechanical engineering and electrical engineering computer-aided design, small size coupled with high floating-point computational performance for computationally intensive technical applications was required. The size of the configured system needed to be relatively small, since in CAD applications, the SPU is often in the immediate work area of the user. For the same reason, minimizing the level of audible noise was important. As a general-purpose technical computer running the HP-UX operating system, the product required a flexible range of I/O configurations.

Additional requirements were presented by computer integrated manufacturing and real-time control applications, where battery backup of main memory is a requirement. The battery backup option provides at least 30 minutes of backup power during power outages. Also required was the ability to operate over the ranges of temperature, humidity, electrical interference, and mechanical vibration typically encountered on the factory floor.

Overall Design

Fig. 2 shows the major assemblies of the Model 825 Computer, and Fig. 3 shows how the product is organized to meet the user requirements. The enclosure is 325 mm wide by 230 mm high by 500 mm deep, compatible with other HP stackable peripheral components. Within this enclosure is a series of card cage slots capable of accommodating a wide range of user configurations.

Nine card slots are available. The first two hold the processor and system boards. The remaining seven slots can be used for system memory, I/O interface cards, interfaces to high-performance bit-mapped graphics displays including the HP 9000 SRX Solids Rendering Accelerator, and adapters for I/O expansion.

A memory card (8M bytes) and an I/O interface card are half-width cards, and together fill one card cage slot. Graphics interfaces and I/O expansion adapters are full-width cards. The following are possible configurations:

Memory Cards		Graphics Displays	I/O Expanders	I/O Interfaces
number	bytes			
7	56M	0	0	7
6	48M	0	1	14
6	48M	1	0	6
5	40M	1	1	13

The Model 825 is rated at 3.1 MIPS supporting multiple users in technical HP-UX applications and at 8.2 MIPS in single-user applications.

Model 825 Processor

The Model 825 processor consists of two boards. The main processor board contains the core CPU function, including a 16K-byte cache, a 2K-entry translation lookaside buffer (TLB), clock circuitry, and several bus interface circuits. The second board contains most of the floating-point math subsystem, the I/O channel, and the processor dependent hardware. These two boards plug into adjacent motherboard slots and communicate via the math bus and the MidBus.

Main Processor

The Model 825 processor is highly integrated, consisting

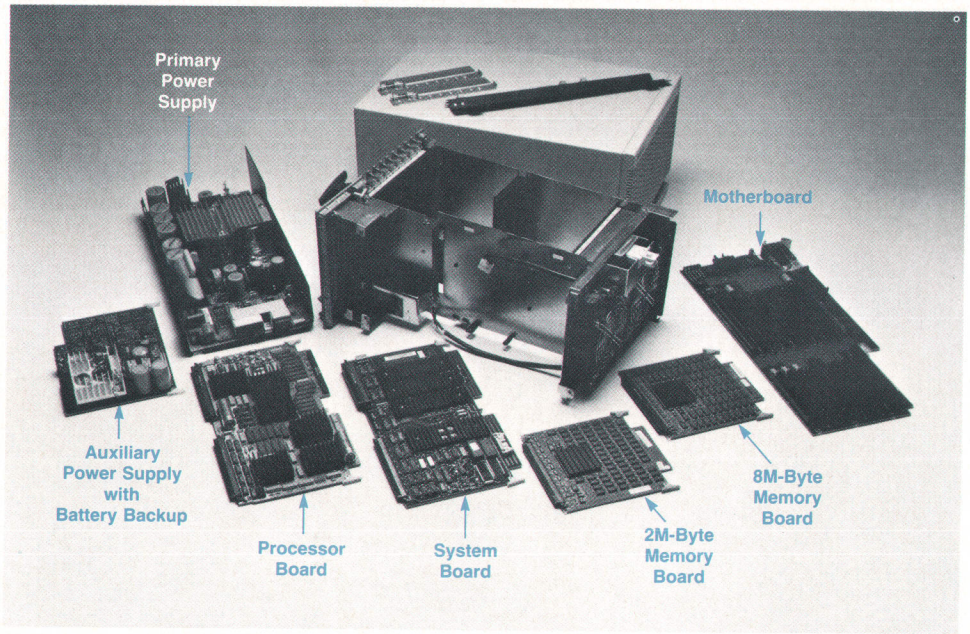


Fig. 2. *HP 9000 Model 825 major assemblies.*

of six high-speed VLSI circuits communicating via the cache bus. These six chips are the CPU, which contains the core CPU function and implements the HP Precision Architecture instruction set, the translation control unit (TCU), which performs virtual-to-real translations and access protection, two cache control units (CCU), each of which controls a static RAM array that makes up a "group" of the cache memory, a math interface unit (MIU), which implements the floating-point math coprocessor function and controls the floating-point math chips, and the system interface unit for the Model 825 (SIUF), which interfaces the cache bus to the main memory bus, the MidBus. Details of these chips are discussed in the paper on page 4.

These VLSI chips are built using HP's high-performance NMOS-III process. They are packaged in 272-pin pin-grid arrays, and consume from 7 to 12 watts depending on the chip type. The basic system frequency is 25 MHz. Providing an environment in which these chips can operate is a significant design challenge.

Cache Bus

Details of the operation of the cache bus are covered in the paper on page 4. In the Model 825 implementation of the cache bus, special attention was paid to propagation delays along printed circuit board traces. For maximum performance, propagation delays were minimized by using

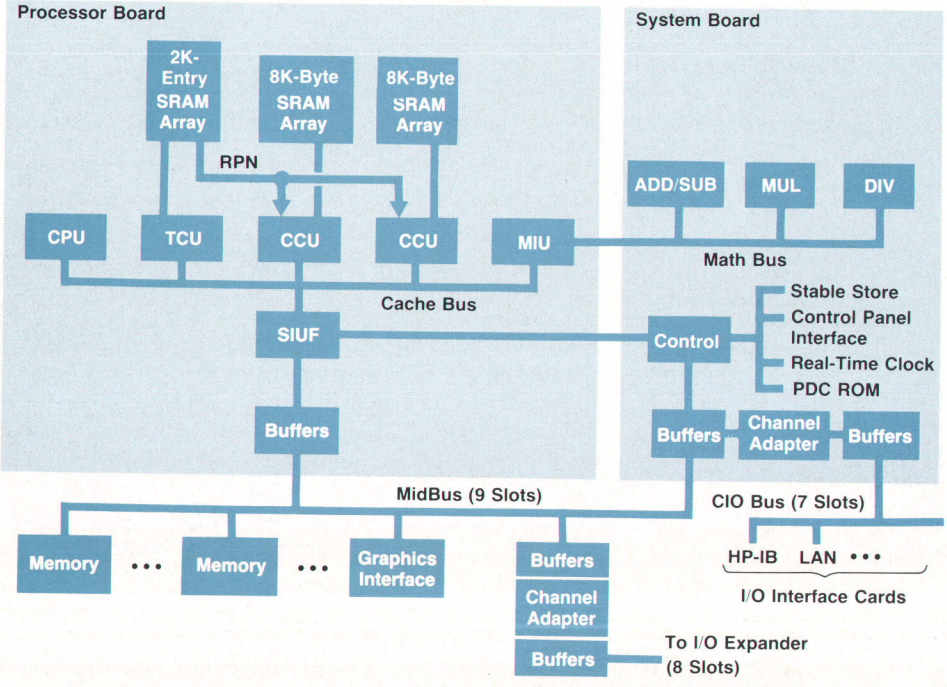


Fig. 3. *HP 9000 Model 825 system processing unit block diagram.*

transmission line techniques[1] and minimum trace lengths.

In addition, considerable effort went into minimizing electrical noise for increased system reliability, as discussed in the next few paragraphs. When worst-case output levels and input thresholds are considered, the noise margin on the cache bus is about 0.5V for both low and high levels. This is similar to noise margins in TTL systems, but because of the high signal transition rates and the receiver characteristics, careful design was necessary to achieve a workable noise budget.

The largest contribution to the noise budget, outside the pin-grid array, is trace-to-trace crosstalk. Typical printed circuit design rules allow traces to be routed with 0.016-in center-to-center distances. Spacing this tight can result in crosstalk of 40% or more. The Model 825 processor board construction and design rules were defined so as to limit trace-to-trace crosstalk to less than 10%.

The next largest possible noise contribution depends on the method by which the bus is terminated. The NMOS-III receivers and drivers are designed such that the bus can operate with no termination at the ends. However, this can effectively double the magnitude of transitions on the bus and therefore double the amount of coupled noise.[2] Terminating the end of the bus effectively reduces the magnitude of reflections, resulting in lower coupled noise. This also helps to absorb the noise that is coupled onto a victim line.

Each cache bus line is terminated by a resistor close to the characteristic impedance of the board traces. Most lines are bidirectional and are terminated at both ends. Some lines are unidirectional and are only terminated at the receiving end to save power and reduce part count.

Special resistor packs are used to terminate the cache bus. These packs are designed for low inductance and low common lead resistance to reduce crosstalk internal to the resistor pack.

One disadvantage of resistor terminators is increased power dissipation. For the Model 825 design there is another problem. Power consumed by the bus depends on bus activity. Under some processing conditions, bus power can change from essentially no load to full load, or vice versa, in one machine cycle. Power supplies are typically not capable of responding to transients of this speed. Power supply droop affects precharge level and consequently reduces noise margin. This was solved by mounting a low-series-resistance, high-valued aluminum electrolytic capacitor directly on the main processor board.

Math Subsystem

The Model 825 has a floating-point math coprocessor. Its interface to the main processor is the math interface unit (MIU) chip, which connects to the cache bus and controls the three floating-point math chips. Ideally, the floating-point math chips should be on the same board as the MIU. However, board space constraints would not allow this. Instead, the floating-point math chips are located on the adjacent system board, and the math bus is run to them through the motherboard. The extra time necessary to transfer data across the motherboard is minimal and does not cause a performance loss at the Model 825's frequency of operation.

An additional constraint on the design of this part of the system was that power supply considerations and power dissipation on the boards made it impossible to terminate this bus. There was also no room on either board for the terminator components. A workable system was devised by building a detailed Spice model of the entire interconnect system. NMOS-III driver sizes were selected such that speed is sufficient, but transition time is maximal. Special treatment was given to critical clock lines that run from the MIU to the floating-point chips.

Cache Array

The Model 825 has a 16K-byte cache organized as two 8K-byte groups. Each group is implemented with a cache control unit (CCU) and eight 2K × 8 25-ns static random-access memories (SRAMs). Five of the SRAMs are used for data and data parity and three are used for tag, status, and tag parity.

The cache access time is in the critical path that determines the clock period for the CPU, which is directly proportional to the performance. The Model 825 clock period is 40 ns. The address is driven on a clock edge, and the CCU must determine if there is a cache hit by the next clock edge. It takes the CCU 7.5 ns to compare the tag to the real address. With 25-ns SRAMs, this leaves 7.5 ns for the address to be driven to the SRAMs and data to be driven back to the CCU. The timing has been verified by the use of Spice simulations and experimentation.

Each RAM address line is a single line with a Schottky diode at the end to clamp undershoot. There is also a 150Ω resistor to 2.85V. The undershoot is clamped mainly to prevent the voltage on the line from ringing back up above 0.7V. The resistor is not for transmission line termination. Its main purpose is to limit the high-level output voltage of the driver. As a result, the high-to-low voltage transition is smaller, giving less ringing in the fast case and making the slow case faster. The slow case model is dominated by the capacitive effects and the limited current that can be provided by the driver, and so a smaller voltage transition will be faster. This can be seen in the basic capacitor equation:

$$I \times dT = C \times dV.$$

Simulations were done to determine the optimal value of resistor to use. A smaller resistor always helps improve the low-to-high transition time because it increases the current. For the high-to-low transition a smaller value helps decrease the transition time by making dV smaller, but also causes an offsetting increase in the transition time because it decreases the current available to change the voltage across the capacitor. The termination voltage could also have been optimized, but this was not necessary because the timing budget was met using the already available 2.85V supply.

Clock Circuit

To meet the system skew budget for the Model 825, each chip must receive a master clock (clock SYNC) that transitions from 0.9V to 4.1V in less than 3 ns. There must be less than 600 ps skew from any chip to any other chip. There are additional specifications for low and high levels,

duty cycle, etc.

To put 600 ps in perspective, electromagnetic waves travel through glass epoxy printed circuit board material at roughly 200 ps/inch. 600 ps is the delay of about 3 inches of printed circuit board trace. Since there are six VLSI chips that require this signal, and each chip is about 2.2 inches on a side, it is not possible simply to connect the clocks in a daisy-chain manner.

The solution is to supply the master clock to each VLSI chip with a separate, equal-length trace. All of these clock supply lines emanate from a single drive point.

It is also desirable that the rise time of the master clock be the same at each VLSI chip. This is a problem because the nominal master clock input capacitance is somewhat different for each chip type. The rise time at the chip receiver is roughly the no-load rise time plus Z_oC, where Z_o is the characteristic impedance of the master clock line and C is the input capacitance. This problem is alleviated by adjusting the master clock line impedance for each chip such that Z_oC is constant for all chip types. Additionally, so that these impedances track as closely as possible, all clock traces are run on the same trace layer.

Since it is important for the chips to receive a clean master clock signal, termination is necessary to reduce reflections. Source termination was chosen for its low power and reasonable drive current levels.

Clock Buffer Circuit. The single drive point impedance is about 7Ω. Combined with the level and rise time requirements of the VLSI chips, this dictated the need for a special clock buffer circuit. The circuit can be split into two pieces: the front end, which generates a signal with the appropriate rise time and high and low levels, and an output section capable of driving 7Ω.

This circuit is implemented in discrete high-frequency transistors. Great care is taken to bias the collector-base junctions to minimize the Miller effect.

The front-end stage takes the TTL-level input signal, sharpens the edge, and produces the correct level for the output stage. The output stage consists of several emitter followers that transform the front end's high-impedance signal to the low impedance necessary to drive the distribution lines.

Printed Circuit Board Construction

Since cache bus design requires transmission line techniques, the printed circuit board itself must be constructed in a controlled-impedance manner. The characteristic impedance of a printed circuit trace is determined primarily by the width of the trace, the dielectric constant of the insulating material, and the geometry of the trace in relation to its adjacent reference plane(s). For reasons related to the board fabrication process, all Model 825 traces are of the stripline variety, that is, the traces are on one board layer (signal layer), and this layer is sandwiched between two board layers with conductive planes on them (plane layers).

High-pin-count, high-speed VLSI created a significant problem for the printed circuit board construction. The two planes that form the stripline configuration should be perfectly ac coupled. If they are not, the signal trace and the planes form what can be viewed as a capacitive divider. When a signal propagates down a trace, some voltage is induced in one plane with respect to the other. Typically, one of the two planes doubles as a power supply distribution layer. The result is noise in the power supply that is also coupled down onto other victim traces.

Normally this problem can be neglected because the plane-to-plane capacitance is much greater than the trace-to-plane capacitance, and transitioning lines are spatially and temporally far enough apart so that the net effect is small.

On the other hand, the high-pin-count, high-speed VLSI used in the Model 825, in combination with relatively high logic swing levels (as much as 3V), is capable of causing as many as 62 closely spaced lines to transition nearly simultaneously. This can result in significant noise coupled into the planes and signal lines.

The obvious solution is to use ground planes between all signal layers. The ground planes would be much closer to the ideal situation, since they are tied together by a large number of vias. Unfortunately, this is not feasible because of board thickness and cost considerations. Careful analysis yielded a board construction with sufficient noise decoupling and reasonable overall thickness:

Layer	Purpose
Top	+2.85V Plane
2	Ground Plane
3	Signal Layer
4	Ground Plane
5	Signal Layer
6	Ground Plane
7	Signal Layer
8	+5V Plane
9	Signal Layer (Uncontrolled)
Bottom	−2V Plane

All signal layers except for layer 9 have 0.008-in traces, impedance-controlled to 50 to 70 ohms. All cache bus traces are on layers 3, 5, and 7. Layer 9 has slightly lower impedance and is used for TTL and other miscellaneous signals.

Printed Circuit Board Layout. One of the most significant challenges of the Model 825 processor was the trace layout of the main processor board. Almost every trace on the board had a length restriction. Cache bus topology had to be rigorously controlled. Clock buffer performance was layout dependent. Thermal and interconnect considerations restricted parts placement. The board contains three major buses and three distinct static RAM arrays. It was extremely important to limit the number of layers and the board thickness for reasons of cost and manufacturability.

Autorouting was out of the question. It also became clear that previously used systems were inadequate. Since we knew that board complexity would require hand layout, what we needed was a powerful graphics editing system. HP's Engineering Graphics System (EGS) is such a system, and it was readily available. The flexibility of EGS, combined with specially written macros and programs, allowed us to build a system tailored to the needs of this layout.

I/O Channel

The I/O system used in the HP 9000 Model 825 is HP's CIO. The system board contains a single VLSI chip that converts from the MidBus to the CIO bus. Details of it are described in the paper on page 38.

Processor Dependent Hardware

HP Precision Architecture allows implementation dependent computer functions, such as the Model 825's control panel interface, stable store, and time-of-day clock, to be realized in a way that is convenient for the particular design. Details of operating these functions are hidden from higher-level software by a layer of firmware called processor dependent code. This code is stored in ROM and begins execution immediately upon power-up.

A primary goal of the processor dependent hardware design was low cost. This hardware communicates with the processor using a special simplified protocol and only the low-order byte on the MidBus. The architecture reserves a certain area of the real address space for processor dependent use. The SIUF chip decodes references to these locations, assembling the bytes into words if necessary for the CPU.

The processor dependent code ROM is simply a large byte-wide ROM. It contains self-test code, boot code, the processor dependent firmware, and other information specific to the Model 825 implementation of HP Precision Architecture.

The processor dependent hardware includes a battery-backed CMOS real-time clock for keeping the correct time and date while the computer is turned off. The batteries are the same as those used in many watches, cameras, and calculators and provide up to a year of backup time.

A stable store is implemented in an EEPROM. Boot path, system serial number, and other similar information is stored here. Constants kept in stable store also assist in more accurate timekeeping and help reduce factory cost. During system operation, real time is kept by software, using timing derived from the 25-MHz main system clock. During power interruptions, real time is kept by the battery-backed CMOS clock circuit. The CMOS clock circuit has its own independent, low-power, low-frequency crystal. The availability of stable store means that crystal correction factors can be stored for both the main system crystal and the backup clock crystal. This allows the use of less expensive crystals and provides more accurate timekeeping. In board manufacture or service, a highly accurate time base is used to measure the crystal frequencies and correction factors are written to stable store. To take full advantage of this scheme, both the 25-MHz main system clock and the real-time clock crystals are located on the system board. This way the correction factors and the correctable devices are installed and replaced as a unit.

Also in the processor dependent hardware is a latch the processor can read to determine the status of the secondary power and other resources. There is also a register the processor can use to control the state of the front-panel LEDs.

Memory Subsystem

The Model 825 supports up to seven memory array boards. Each board contains an 8M-byte memory array which is interfaced to the MidBus by a memory controller. The block diagram of the memory board is shown in Fig. 4.

The memory controller is a custom integrated circuit implemented in HP's NMOS-III technology. It is designed to provide the following features:

- Provide all signals to control 120-ns 1M-bit dynamic random-access memory chips (DRAMs)
- 19-Mbyte/s transfer rate on the 8.3-MHz MidBus
- Error logging
- Correct single-bit errors and detect double-bit errors
- Provide a mechanism to correct double-bit errors with a known hard error
- Support of memory board test and diagnostic features
- Refresh
- Battery backup
- Compact size which, combined with surface mount technology, allows a board size of less than 50 square inches.

Memory Bus Interface

The Model 825 memory board supports 16-byte and 32-byte block read and write operations and a semaphore operation. High-bandwidth data transfers are provided by the 32-byte transactions. The memory bus interface consists of a 72-bit-wide data bus, a 10-bit-wide address bus, two row address strobes (RAS), four column address strobes (CAS), and one write enable signal (WE). Multiple RAS and CAS lines are used to reduce the delays and meet the timing requirements while driving these heavily loaded signals. The memory array is organized as one row of DRAMs with 1M words of 72 bits each. Each memory word is packed with two 32-bit words from the MidBus and eight Hamming check bits.

Fig. 5 shows the timing for memory read and write operations. The 20 address bits required to address the 1M-bit DRAMs are multiplexed onto the 10-bit memory address bus and latched into the DRAM address latches by two negative-going strobes. The first strobe, RAS, latches the row address. The second strobe, CAS, subsequently latches the column address. For a write operation, WE is brought low before CAS, and the data is strobed by CAS. The setup and hold times for data to be written to the memory array are referenced to the falling edge of CAS. For a read operation, WE is held in the high state throughout the memory transaction, and data read from the memory array is avail-

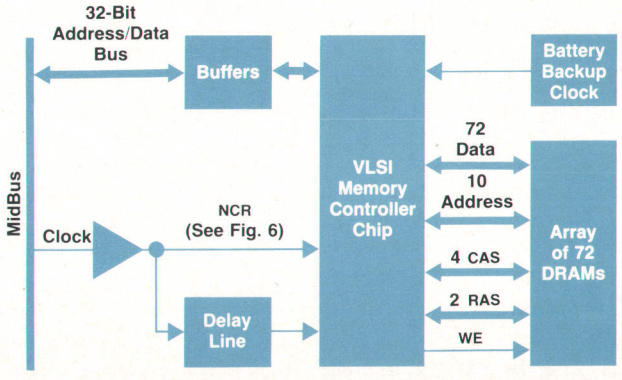


Fig. 4. *Block diagram of the Model 825 memory board.*

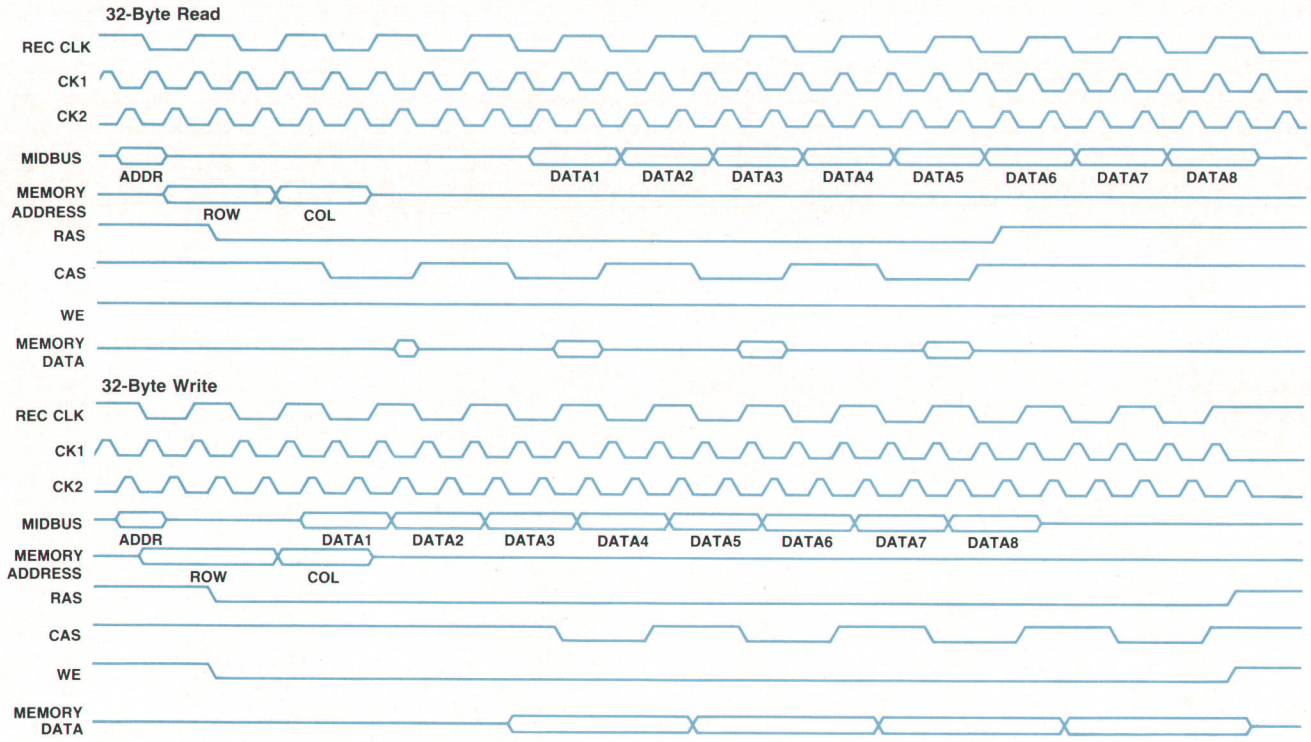


Fig. 5. *Timing for memory read and write operations.*

able within the access time from CAS. The semaphore operation reads 16 bytes from the memory array and clears the first four bytes without affecting the other 12 bytes.

The 1M-bit DRAMs support a feature known as nibble mode. Nibble-mode operation allows high-speed serial access to more than one word of memory. By packing two 32-bit words from the MidBus into each memory word, only two serial accesses are required for a 16-byte transaction and only four serial accesses are required for a 32-byte transaction. The first word is accessed in the normal manner with read data being valid at the CAS access time. The sequential words are read or written by toggling CAS while RAS remains low. The row and column addresses need to be supplied for only the first access. Thereafter, the falling edge of CAS increments the internal nibble counter of the DRAMs and accesses the next word in memory.

Address Strobe Signal Quality

DRAM RAS and CAS inputs function as clock signals and must have clean transitions. The assertion of these signals is also in the critical timing path for accessing data, so minimizing timing skew is important. To ensure that the transitions will be smooth, the clock signals are routed to the DRAMs in a star configuration.

There are four CAS drivers for every 72 DRAM chips and so each CAS driver must drive 18 DRAMs. The CAS line is routed to a location central to the 18 DRAMs. From here it is split into six signals. Each of these is routed to a point central to the location of three of the DRAMs. The six lines are all made the same length to keep the star balanced. Three DRAMs are connected to each of these six signals. Again to keep the star balanced, the lines connecting the DRAMs are all the same length. The four CAS signals coming from the memory controller are routed electrically identically. The CAS signals are all driven directly by the VLSI controller and so each DRAM sees almost exactly the same signal. This allowed the drivers and series termination to be optimized to give a smooth low-skew signal at a single point to ensure that the CAS input signal at all 72 DRAMs would be optimized. There is very little CAS timing skew between DRAMs.

The timing for the RAS signal is less critical than for the CAS signals, but it is important that the transitions be smooth and glitch-free. The signal is connected in a star configuration but there are only two RAS drivers, so the last star connects to six DRAMs instead of three.

Delay Line

To conform to MidBus timing, some cards are required to have a delay line. The value of the delay line depends on the delay and timing of the bus interface circuits and the MidBus buffers.

The delay line for the Model 825 memory array board must be between 21 and 27 ns. At the time of the design, reliable surface mount delay lines were unavailable, so alternatives were investigated. Lumped LC circuits were tried, but it was hard to guarantee the delay with the wide tolerances on the parts (mostly the TTL drivers and receivers). The second alternative was to use a single TTL buffer and a long trace to get the needed delay. It was found that this was feasible.

The signal propagation delay for a long printed circuit

trace between two ground planes is:[1]

$$1.017 \times \sqrt{\epsilon_r} \quad \text{ns/ft}$$

where ϵ_r is the dielectric constant of the insulating medium. The dielectric constant of the printed circuit board material is between 4.4 and 4.8, which gives a delay of 2.13 to 2.23 ns per foot. The trace was made 115.5 inches long to give a delay of 20.5 to 21.5 ns. The TTL driver for the delay line has a delay of 1 to 5 ns, giving a total delay of 21.5 to 26.5 ns.

Two of the potential problems with the long trace are RFI and the signal coupling into itself. To avoid this the trace is sandwiched between two ground planes and runs are spaced at least 0.045 inch apart with a 0.015-in ground trace between runs (sections where the trace loops back on itself).

Powerfail Backup

The power failure backup system in the Model 825 is a RAM-only backup system. If line voltage is lost, the RAM is powered from a backup battery supply while the rest of the system is shut down. Since the dynamic RAMs require little power to retain data, only a relatively small battery and backup regulator are needed to keep the memory system alive. When power is restored after an outage there is enough information available in the memory to resume normal processing after a recovery period.

To support powerfail backup, the RAM board is designed to power down its interface to the rest of the computer cleanly when the failure occurs and to keep the contents of memory refreshed during the outage. Power drain on the backup supply has been minimized for maximum backup time.

To retain the data, each of the 512 row addresses of the DRAM cell matrix must be refreshed within every 8-ms time period. During normal operation the MidBus clock is used to provide the timing for the refresh state machine. However, during a power failure, the MidBus clock is undefined and a secondary refresh clock must be provided on the memory board. This secondary refresh clock is generated with a CMOS 555 timer with a maximum period of 7.5 microseconds.

The powerfail sequence is initiated by the falling edge of POW FAIL L, which indicates that the input to the MidBus power supplies has failed and a powerfail routine should be entered. The power supplies remain regulated long enough after the falling edge of POW FAIL L to guarantee that the cache will be flushed to main memory before the falling edge of POW ON. After POW ON falls and any refresh cycle in progress is completed, the memory controller switches into the battery backup mode of operation.

While in the battery backup mode of operation, the memory controller holds WE and CAS high to prevent inadvertent destruction of the memory contents. In addition, the battery backup circuits are isolated from spurious inputs from the primary control section which occur while power is in transition.

DRAM Error Handling

The memory controller chip incorporates error handling circuits based on Hamming codes to protect the system from errors in the DRAMs on each memory board. The 32-bit words on the MidBus are packed into 72-bit words when written to the DRAMs. The 72 bits consist of two words from the MidBus and eight check bits generated by the Hamming circuit.

On a read from memory, the 72-bit word is presented to the Hamming circuits. If the syndrome word generated is zero, the word from the DRAMs is uncorrupted and the data corrector is told to pass the word unaltered. If the syndrome word generated is nonzero, the condition of the error (recoverable/unrecoverable, mappable/unmappable) will be reported in the STATUS register, the cache line address will be saved in the ERR ADD register, and the syndrome word will be stored in the ERR SYN register. If the syndrome word equals one of the 72 valid patterns, a single-bit error has occurred, and the data corrector flips the bit indicated by the syndrome pattern to recover the data. Detection and correction of single-bit errors are transparent to the system.

If a nonvalid error condition exists, a double-bit (or more) error has occurred. The memory controller has circuits for recovering from many double-bit errors. To use this feature, the system software needs to have identified a troublesome bit (usually a hard failure) in a bank of memory. After identifying it, the system writes the syndrome word of that bit into the MAPSYN register, and by issuing a CMD MAP signal, notifies the memory controller to suspect that bit as bad in a double-bit error. Knowing this, when the nonvalid condition occurs, the memory controller will order its data corrector to flip that bit and recheck the word. If a valid syndrome word is now calculated, the single-bit error routine will be invoked. If the syndrome is not valid, the memory controller will notify the system of an unrecoverable error condition.

Internal Clocks

The internal phase clocks of the Model 825 memory controller are generated by the circuit described in "A Precision Clocking System" on page 17. The SYNC input that circuit requires is generated by an on-chip circuit that effectively doubles the 8.33-MHz MidBus clock frequency to 16.67 MHz. This doubles the number of well-controlled clock phases per bus state for better control of DRAM timing. Fig. 6 is a diagram of the 2× clock generator.

The basic building block of the 2× clock generator is the delay element shown in Fig. 7. The delay element makes use of the voltage-controlled resistance characteristic of MOSFETs. A capacitor in the delay element is precharged when the delay element's input is low. This causes the ARM signal to go high. This ARM signal is NANDed with the input to generate the output. When the input goes high, the output goes low and the capacitor is discharged. When the capacitor voltage drops below the threshold of the sense FET (pulldown of the first inverter following the RC node in Fig. 7), the ARM signal goes low, causing the output to go back high. The capacitor's discharge FET is in series with a FET controlled by a variable voltage (V_{CON}), so the length of time the output is low can be varied.

If the variable voltage is set such that the discharge time plus the delay to disarm the output is one quarter of the

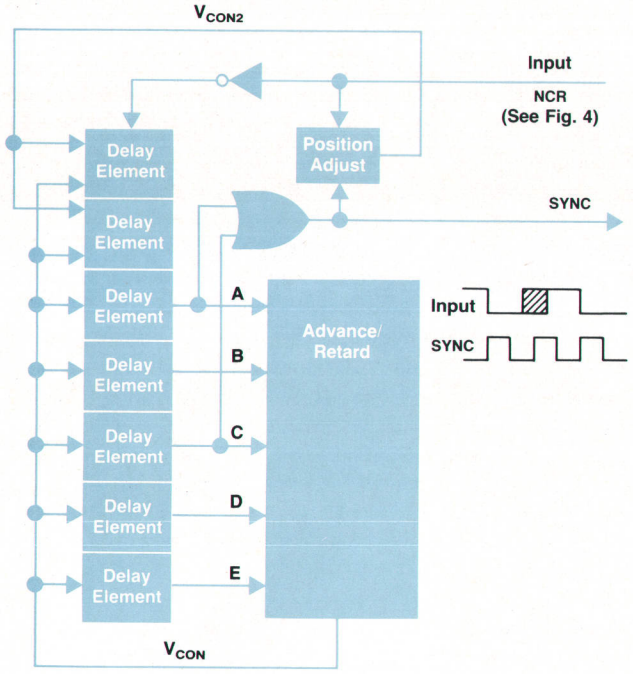


Fig. 6. *2× clock generator generates a sync signal at twice the MidBus clock frequency.*

MidBus clock period, the inverted outputs of alternate elements within a series of delay elements can be ORed together to provide a SYNC signal at twice the MidBus clock frequency.

An advance/retard circuit looks at the inverted (positive pulse) output of five delay elements connected in series. When the output pulse of the second element (B) starts within the output pulse of the fifth element (E), the pulses are longer than desired. The advance/retard circuit increases the variable voltage V_{CON}, which decreases the pulse width. When $\overline{(A+B+C+D)}$ exists, the pulses are shorter than desired, and V_{CON} is decreased, which increases the pulse width.

A position adjust circuit looks at the rising edge of SYNC compared to the falling edge of the input clock. When SYNC is late, the position adjust circuit raises a secondary variable voltage (V_{CON2}) which acts in parallel with V_{CON} to shorten

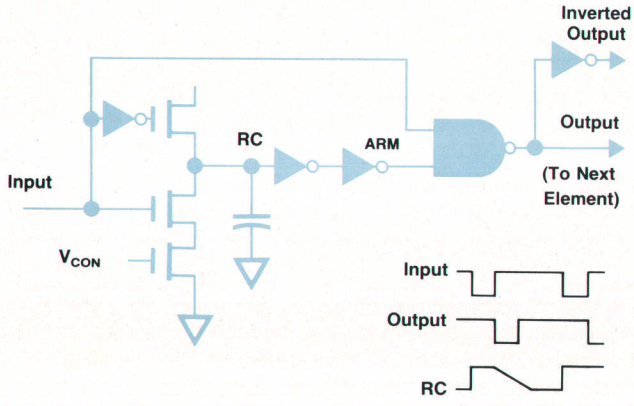


Fig. 7. *Basic delay element of the 2× clock generator.*

two preliminary delay element pulses. These pulses have a width of T/4 − D/2, where T is the MidBus clock period and D is the delay in generating SYNC. Thus one SYNC pulse is placed at T/2 and the next one is aligned with the falling edge of the input clock.

Power Subsystem

Power for the HP 9000 Model 825 Computer, up to 435W total, is provided by a switching mode power supply operating at 29 kHz and using bipolar technology. The basic design of this power system has evolved over the past several years and has been employed with increasing sophistication in a number of products.[3] In each stage of this evolution, improvements have been made in selected areas to increase reliability while reducing complexity and costs. This incremental approach has allowed the crucial compromise between using well understood parts and technology while still exploiting new ideas and developments in the industry.

Six outputs are provided, including an optional battery-backed five-volt output, +5VS. This secondary output provides power to the main memory during a primary power failure, allowing the product to recover and resume operation. The 12-volt batteries and their charger are housed in a separate unit and are cabled to the computer.

Dc fans are used to cool the product and they are controlled by the power supply. The fans are operated at low speed to minimize audible noise in a typical office environment but their speed is increased as external temperatures rise above 30°C. To maintain critical cooling, the fans are also operated while the unit is running on batteries.

Acknowledgments

The authors wish to thank Russ Sparks, R&D section manager, for his leadership and support during the development of the Model 825. We would also like to acknowledge the significant technical contributions made by Sean Chapin, Richard Chou, Jeff Hargis, Dave Hollenbeck, Joel Lamb, Jim McLucas, Floyd Moore, and Tony Riccio for memory system design, Lynne Christofanelli, John Hoppal, and Charlie Shilling for product design and packaging, Russ Brockmann and Jim Murphy for the processor board, Richard Alley and Stuart Yoshida for the expander, Mark Hodapp, Steve McAtee, Rose Melville, Dave Lechtenberg, and Ed Ogle for firmware, diagnostics, and turn-on, and Arnold Gum and Ann Kloster for the motherboard and math coprocessor. Special thanks go to Dana Seccombe, who helped make this program possible.

References

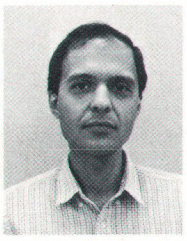
1. W.R. Blood, *MECL System Design Handbook*, Motorola, 1980.
2. A. Feller, H.R. Kaupp, and J.J. Digiacomo, "Crosstalk and Reflections in High-Speed Digital Systems," *AFIPS Conference Proceedings*, Volume 27, Part 1, 1965.
3. J.L. Burkman, et al, "A Compact, Reliable Power Supply for an Advanced Desktop Computer," *Hewlett-Packard Journal*, Vol. 35, no. 5, May 1984.

Authors

September 1987

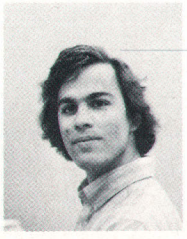
4 VLSI Processor

Darius F. Tanksalvala



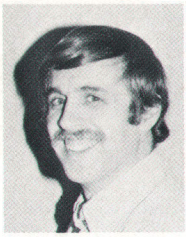
Born in Lahore, India, Darius Tanksalvala received his BTech and MTech degrees from the Indian Institute of Technology in 1967 and 1969. He joined HP in 1975, just after receiving his PhD in electrical engineering from the University of Wisconsin at Madison. He has contributed to the development of HP's proprietary NMOS IC process, has done circuit design for the HP 9000 Series 500 CPU chip, and was project manager for two of the chips in the NMOS-III HP Precision Architecture processor. He's a member of the IEEE and the ACM. Darius and his wife and daughter live in Denver. Backpacking, hiking, and snowshoeing are among his leisure activities.

Steven T. Mangelsdorf



Steve Mangelsdorf joined HP in 1983 during the investigation phase of the NMOS processor chip set project for the Model 850S/Series 950 and Model 825 Computers. He contributed to the definition, control design, and verification of the SIUF and CCU chips, and was project manager for the characterization and second release of the CCU and TCU chips. He has a BS degree in computer science from Princeton University (1982) and an MSEE degree from Stanford University (1983). His professional interests are computer architecture, MOS circuit design, and CAD tools. A resident of Fort Collins, Colorado, he is an audiophile with a preference for alternative rock music. He also enjoys hiking and bicycling.

Charles R. Headrick



VLSI and CPU design specialist Bob Headrick is a section manager with HP's High-Performance Systems Operation. His section was responsible for the math subsystem for the Model 850S/Series 950 and Model 825 Computers. With HP since 1976, he has been a design engineer, project leader, and project manager for various ROM, RAM, and CPU designs. He has a 1976 BSEE degree from Oregon State University and a 1980 MSEE degree from Stanford University. He's a member of the IEEE and has coauthored several papers on VLSI design. Originally from Corvallis, Oregon, he now lives in San Jose, is married, has a son, and collects rare U.S. stamps.

Paul K. French

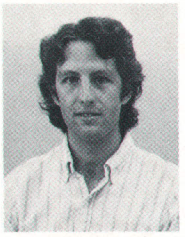
Paul French received his BS degree in engineering from the University of California at Los Angeles in 1979 and his MSEE degree from Stanford University in 1981. Now with HP's Entry Systems Operation, he has designed VLSI cache and address translation chips and has done design and verification of the SIUC chip for the Model 850S/Series 950 and the MIU chip for the Model 850S/Series 950 and Model 825 Computers. He's been with HP since 1981. Born in Torrance, California, Paul now lives in San Jose. He is a founding director and the current president of the HP National Running Club, and has represented HP in many local and national running events.

Darrell M. Burns

A specialist in VLSI computer system design and VLSI device physics, Darrell Burns has been a project manager at HP's High-Performance Systems Operation since 1984, and is currently responsible for the Model 850S/Series 950 SIUC, BC, MCC, and CIO chips. He received his BSEE degree from the University of California at Davis in 1977 and his MSEE degree from the University of California at Berkeley in 1978. After joining HP Laboratories in 1978, he designed custom VLSI memory and logic ICs and did VLSI computer design before becoming a project manager. His work has resulted in a patent on a low-noise CCD sense amplifier. Darrell is married, lives in Los Gatos, California, and likes volleyball, basketball, golf, target shooting, and reading history and fiction.

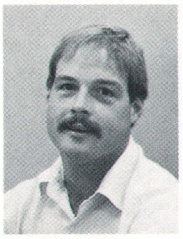
12 NMOS-III CPU

Eric R. DeLano



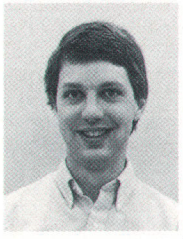
Eric DeLano contributed to the design of the HP Precision Architecture VLSI CPU chip. A native of Los Angeles, he attended the University of California at Berkeley and received a BS degree in 1982 and an MEng degree in 1984, both in electrical engineering. He joined HP in 1984. Eric is married and lives in Fort Collins, Colorado. His interests include climbing, hiking, and travel.

Jeffry D. Yetter



Jeff Yetter was project manager for the NMOS-III HP Precision Architecture CPU and presented the paper describing this chip at the 1987 International Solid-State Circuits Conference. With HP since 1977, he's also done VLSI memory systems design for the HP 9000 Series 500 computer family. Born in Rochelle, Illinois, he received his BS degree in computer engineering in 1977 from the University of Illinois at Urbana. He's a member of the IEEE. He lives in Fort Collins, Colorado and likes skiing, backpacking, and other outdoor activities.

Mark A. Forsyth

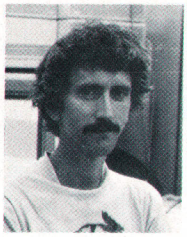


As a project manager at HP's Entry Systems Operation, Mark Forsyth was responsible for the design of the TCU chip and for electrical integration of the processors for the Model 850S/Series 950 and Model 825 Computers. A native of Minnetonka, Minnesota, he graduated from the University of Minnesota in 1978 with a BSEE degree and joined HP the same year. Before becoming a project manager, he helped design the CPUs for Series 500 and HP Precision Architecture processors and did NMOS-III production engineering. A member of the IEEE, he is a coauthor of a 1987 ISSCC paper on the NMOS-III CPU. Besides VLSI design, he's also interested in computer architectures and cache memory design. Mark is married, has a daughter, lives in Fort Collins, Colorado, and enjoys family and outdoor activities.

Jonathan P. Lotz

Jon Lotz is a member of the technical staff at HP's Entry Systems Operation, specializing in IC design. He contributed to the design and verification of the NMOS-III HP Precision Architecture processor chip set. A native of Long Island, New York, he attended Purdue University and received BSEE and MSEE degrees in 1983 and 1984. He came to HP in 1984. Jon lives in Fort Collins, Colorado and enjoys backpacking, bicycling, and skiing.

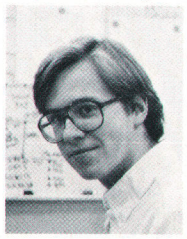
William S. Jaffe



MOS specialist Bill Jaffe was born in New York city. He has a BSEE degree from Arizona State University (1973) and an MSEE from Stanford University (1977). With Sperry Avionics Systems, he designed avionics for the space shuttle trainer aircraft. He came to HP in 1977 and has contributed to the design of the IOP, RAM, and CMC chips for the HP 9000 Series 500 product line and the CPU and CCU chips for the Model 850S/Series 950 and Model 825 Computers. He's a member of the IEEE. A resident of Fort Collins, Colorado, he's married and has a son whom he enjoys watching grow. He also plays piano and likes to go backpacking, hiking, and bicycling.

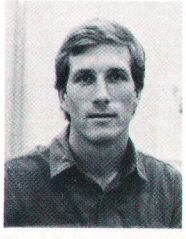
18 Design and Test Methodology

Stephen R. Undy



Steve Undy is a development engineer with HP's Entry Systems Operation, interested in computer architectures and multiprocessing. He has a pair of 1983 BSE degrees from the University of Michigan, one in electrical engineering and one in computer engineering. His MSEE degree is from Purdue University (1985). Since 1985, he's done verification and electrical characterization for the HP Precision Architecture processor chip set, and is now working on system architecture and verification for new VLSI chips. He's a member of the IEEE Computer Society. Steve is a pilot and a bicyclist, born in Detroit and now living in Fort Collins, Colorado.

Robert A. Schuchard



Bob Schuchard began his HP career at HP Laboratories in 1978, where he designed automated IC processing equipment for several years. Now with HP's Entry Systems Operation, he contributed to the design of the SIUF chip for the Model 825 Computer and is project manager for the SIUF and memory controller chips. He coauthored a 1987 ISSCC paper on scan path testing. Born in Wabasha, Minnesota, he received his BSEE degree from the University of Colorado in 1976 and his MSEE from Stanford University in 1978. He's married, has three children, and lives in Fort Collins, Colorado, where he is active in his church's youth ministry. Among his leisure activities are skiing, fishing, hunting, woodworking, and running.

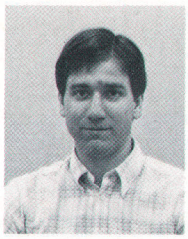
Tony W. Gaddis

Born in Wiesbaden, Germany, Tony Gaddis joined HP in 1978 after receiving his BSEE degree from Arizona State University. He's been involved with HP's NMOS IC process ever since, first in NMOS-II assembly and test, then in fabrication and reliability and yield improvement for NMOS-II and NMOS-III, and more recently in HP Precision Architecture NMOS-III design verification. He's now specializing in CAD tools and is working to improve chip design productivity. He's a member of the IEEE. Tony and his wife and two sons are residents of Fort Collins, Colorado. Tony gets involved in Cub Scout activities, but finds time for sailing and speed roller skating.

Charles Kohlhardt

Charlie Kohlhardt is section manager for VLSI design for the Model 850S/Series 950 and Model 825 Computers. With HP since 1978, he's done memory and processor design for the HP 9000 Series 500 product line, worked on test development, and served as a project manager. A native of Pittsfield, Massachusetts, he attended the University of Wisconsin at Madison, graduating in 1978 with a BSEE degree. He lives in Loveland, Colorado, and enjoys snow skiing, water skiing, and board sailing. He is married and has two daughters.

Daniel L. Halperin



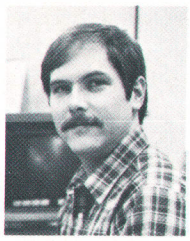
Dan Halperin received his BS degree in electrical engineering from the University of Tennessee in 1978 and his MS and PhD degrees in electrical engineering from the University of Illinois in 1981 and 1984. A native of Oak Ridge, Tennessee, he joined HP in 1983. He's a specialist in computers and VLSI circuits with HP's Entry Systems Operation, and has contributed to the development of the VLSI processor for the Model 850S/Series 950 and Model 825 Computers. He's a member of the IEEE. He and his wife live in Fort Collins and are expecting their first child.

26 Midrange SPU

Thomas O. Meyer

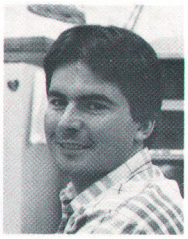
A native of Custer, South Dakota, Tom Meyer graduated from South Dakota School of Mines and Technology with a BSEE degree in 1977, and joined HP the same year as a development engineer. He worked on the memory and remote terminal support for the HP 250 Computer, the power supply for the HP 9000 Model 520, and the battery backup regulator for the Model 825, and served as project manager for the Model 825 power supply and expander. A resident of Fort Collins, Colorado, he enjoys sailing, sailboat racing, skiing, scuba diving, motorcycling, hiking, backpacking, and four-wheel-drive vehicles.

Howell R. Felsenthal



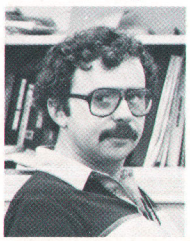
Howell Felsenthal received BSEE and MSEE degrees from the University of Oklahoma in 1978 and 1979. A development engineer with HP's Entry Systems Operation, he has been with HP since 1979 and has contributed to the design of a servo control system, a video display, and the power supplies for the HP 9845 Option 200 and HP 9000 Models 520/30/40/50 and 825 Computers. Born in Ponca City, Oklahoma, he is married, has three children, and lives in Fort Collins, Colorado, where he serves as "visiting scientist" in the local school system. He's also a skier and dirt biker.

Thomas V. Spencer



A development engineer with HP's Entry Systems Operation, Tom Spencer contributed to the design of the HP 9000 Model 825 power supply. He's a graduate of Purdue University, earning his BSEE degree in 1980 and his MSEE in 1982. He joined HP in 1982. Tom comes from Evansville, Indiana. He and his wife, who is also an HP R&D engineer, live in Fort Collins, Colorado. In his spare time, Tom enjoys tennis, softball, basketball, skiing, off-road motorcycling, and white-water rafting.

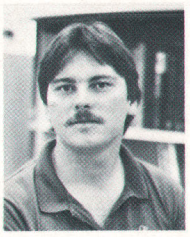
Mark A. Ludwig



Mark Ludwig comes from Chilton, Wisconsin and is a graduate of the University of Wisconsin at Madison. He received his BSEE and MSEE degrees there in 1972 and 1973, and joined HP in 1974. He's done design and production support of an I/O chip for the

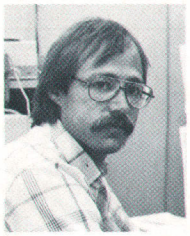
HP 9825 Computer and a memory controller chip for the HP 9000 Series 500, and contributed to the design of several chips for the HP 9000 Model 825. A resident of Loveland, Colorado, he is married and has a son. Besides playing chess and organizing and directing chess tournaments, he enjoys skiing and bicycling.

Russell W. Mason



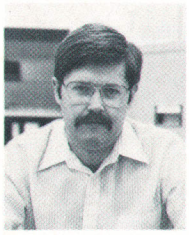
Russ Mason graduated from Mississippi State University in 1984 with a BSEE degree. A development engineer with HP's Entry Systems Operation since 1984, he's interested in VLSI design and digital computer systems, and has contributed to the design of the memory controller for the HP 9000 Model 825 Computer. He was born in Memphis, Tennessee and now lives in Fort Collins, Colorado, where he's involved in church activities. He also enjoys outdoor activities, especially volleyball, bicycling, softball, and photography.

Robert J. Horning



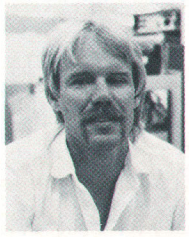
Rob Horning was born in Great Falls, Montana and attended Montana State University, earning BSEE and MSEE degrees in 1976 and 1978. He joined HP in 1978. He's done keyboard and memory design and system testing for the HP 9000 Series 200 product line and memory, system board, and cache system design for the HP 9000 Model 825. A development engineer with HP's Entry Systems Operation, he's interested in memory, cache, and CPU board design. Rob is married, has three children, and lives in Fort Collins, Colorado. His leisure activities include basketball, softball, camping, home improvements, and family bike rides.

Craig S. Robinson



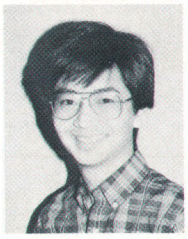
Now a section manager with HP's Technical Workstation Operation, Craig Robinson was project manager for the HP 9000 Model 825 Computer. With HP since 1977, he has contributed to the design of the HP 9835, HP 9845, and HP 9000 Model 520 Computers and served as reliability engineering manager for the Fort Collins Systems Division. A member of the IEEE and the ACM SIGGRAPH, he has coauthored papers on control theory and image array processing. He attended Ohio State University, receiving BSEE, MSEE, and PhDEE degrees in 1973, 1974, and 1977. Craig is a native of Jersey City, New Jersey. He's married and has two children. In addition to various construction projects at his foothills home in Masonville, Colorado, he finds time for beekeeping and skiing.

Leith Johnson



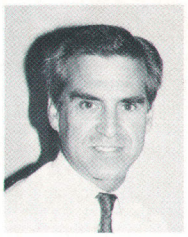
A 1978 BSEE graduate of the University of Nevada at Reno, and currently pursuing an MSCS degree at Colorado State University on an HP fellowship, Leith Johnson is a development engineer with HP's Entry Systems Operation and has professional interests in computer system architectures and high-speed digital design. With HP since 1978, he designed the processor for the HP 9845B Option 200 Computer, did production engineering for the 9845B, and contributed to the design of the CIO channel adapter and the processor board for the HP 9000 Model 825 Computer. A native of Seattle, he lives in Fort Collins, Colorado and likes fast cars, volleyball, boardsailing, and skiing.

Richard Chin



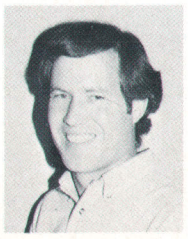
VLSI designer Richard Chin began his HP career in 1978, working summers while he was a student at the Massachusetts Institute of Technology. He's been with HP full-time since 1981, when he received his BSEE and MSEE degrees, and has contributed to the design of a cache subsystem and the memory controller chip for the Model 850S/Series 950. He's a member of the IEEE and the IEEE Computer and Circuits Societies. Richard was born in New York City and now lives in Los Altos, California. He's married and likes to travel.

Gerald R. Gassman



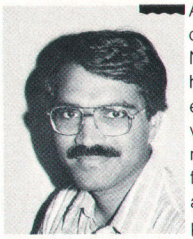
Gerry Gassman is the R&D section manager responsible for the Model 850S/Series 950 SPU. He joined HP in 1977 and has helped in the development of the HP 3000 Series III, 30, 40, and 44 Computers. Before coming to HP, he led Biomation's packaging team. His work prior to HP has resulted in ten design patents on packaging. Gerry is a native of San Francisco, and he and his wife are now residents of Sunnyvale, California. He plays drums, trumpet and tennis, and is interested in art and theater.

Marlin Jones



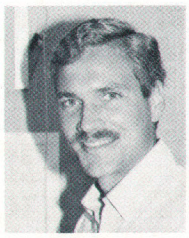
Marlin Jones is a development engineer with HP's High-Performance Systems Operation. He received his BSEE degree from the University of California at Berkeley in 1984 and joined HP the same year. He has provided technical support for HP Spice and was responsible for the memory array and part of the processor and clock boards for the Model 850S/Series 950. His professional interest is high-speed memory system design. Marlin is a native of Oakland, California. He and his wife are residents of San Jose, and he enjoys ultimate frisbee and motorcycling.

Ayee Goundan



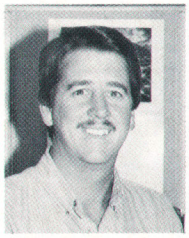
Ayee Goundan is a native of Konganapuram, Tamil Nadu, India. He received his PhD degree in electrical engineering from the University of Southern California in 1978, and designed fault-tolerant controllers and high-speed graphics processors for General Electric before joining HP in 1983. He worked on the fault-tolerance extensions for HP Precision Architecture and was responsible for the Model 850S/Series 950 processor, and is now project manager for the processor, memory, and I/O boards. A member of the IEEE and the IEEE Computer Society, his specialties are architectures, multiprocessors, and high-availability systems. Ayee is married, has three children, and lives in San Jose. He enjoys family activities and traveling.

Robert D. Odineal



Bob Odineal has been doing VLSI design for HP since 1981. Most recently, he contributed to the design of the HP Precision Architecture channel adapter chip and was project manager for the bus converter, memory controller, and channel I/O chips. A graduate of Stanford University, he received both his BS degree in biology and his MSEE degree in 1982. He's a member of the IEEE and a native of Glasgow, Montana, with interests in hang gliding, water skiing, and Shakespeare. He lives in Saratoga, California and serves on the board of directors of a theater group that produces the Saratoga Shakespeare Festival each summer.

Michael W. Schrempp



Mike Schrempp is a project manager with HP's High-Performance Systems Operation, responsible for benchmarking new packaging technologies and design methodologies. He joined HP in 1980 as a product design engineer after receiving his BSME degree from Cornell University. A specialist in thermal design and acoustics, he was project manager for mechanical packaging and product design for the Model 850S/Series 950 SPU. He's a native of Pacifica, California and now lives in Sunnyvale, California.

VLSI-Based High-Performance HP Precision Architecture Computers

The same system processing unit powers two computer systems, one running the MPE XL operating system for commercial data processing and one running the HP-UX operating system for technical and real-time applications.

by Gerald R. Gassman, Michael W. Schrempp, Ayee Goundan, Richard Chin, Robert D. Odineal, and Marlin Jones

THE HP 9000 MODEL 850S and the HP 3000 Series 950 are currently the largest HP technical and commercial computer products, respectively, to use the new Hewlett-Packard Precision Architecture,[1] and along with the HP 9000 Model 825 described in the paper on page 26, are the first to realize the architecture in proprietary VLSI technology. The first technical and commercial HP Precision Architecture systems were the HP 9000 Model 840 and the HP 3000 Series 930, which use commercial TTL technology.[2]

The HP 9000 Model 850S and the HP 3000 Series 950 are both based on the same system processing unit (SPU), which consists of processor, memory, I/O, power, and packaging subsystems. The Model 850S/Series 950 processor uses the NMOS-III VLSI chip set described in the papers on pages 4 and 12.

The differences between the Model 850S and the Series 950 are primarily in the areas of software and configuration. The Model 850S is configured primarily for technical applications. It runs HP-UX, HP's version of AT&T's UNIX® System V operating system with real-time extensions. The Series 950 is configured for business applications. It executes MPE XL, a new version of HP's proprietary MPE operating system. This provides compatibility as well as a performance upgrade for the current HP 3000 customer base.

The Model 850S/Series 950 SPU has a single processor, up to 128M bytes of memory supported from a single memory system, and up to four channel I/O buses. In this paper, references are made to larger memory and I/O capacity, and to the support of multiple processors. The hardware has been designed to support processor, memory, and I/O

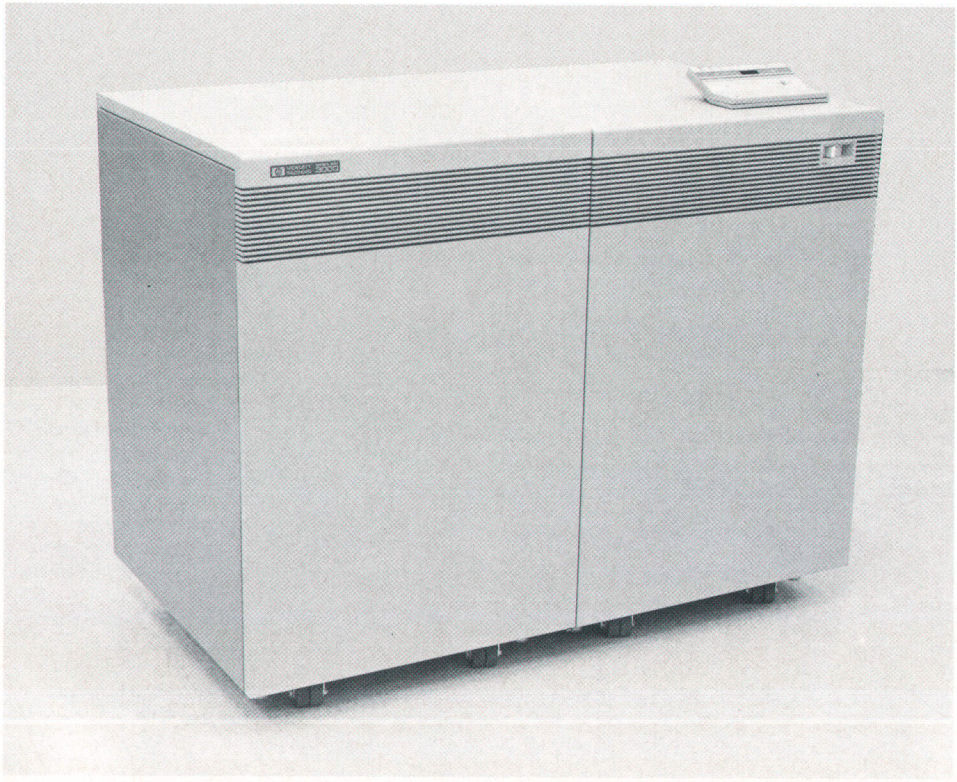


Fig. 1. *System processing unit of the HP 9000 Model 850S and HP 3000 Series 950 Computers.*

extensions beyond those announced to date. These features are described here solely to illustrate why certain technical decisions were made. No implication is made regarding any eventual product that may offer these hardware features.

Delivering 7 MIPS of processing power, the Model 850S/Series 950 SPU (Fig. 1) is the highest-performing HP Precision Architecture SPU developed to date. Containing a single-board VLSI processor capable of running either the MPE XL or HP-UX operating systems, the SPU is designed to fit both commercial and technical computing applications. Performance, user friendliness, system growth, reliability, supportability, and manufacturability were the key design focal points.

SPU Bus Structure

The key to the performance and growth capabilities of the SPU is the bus structure. This determines how and how fast information is passed from one part of the SPU to another. The Model 850S/Series 950 SPU uses a hierarchical three-tier bus structure, as shown in Fig. 2, to achieve high performance and allow for future growth. On the first tier is the fastest bus, a 64-bit-wide bus called the system main bus (SMB).* The second tier in the bus structure consists of a pair of 20-Mbyte/s 32-bit-wide buses called MidBuses.* The third tier in the bus structure is made up of four 5-Mbyte/s HP-standard channel I/O (CIO) buses.*

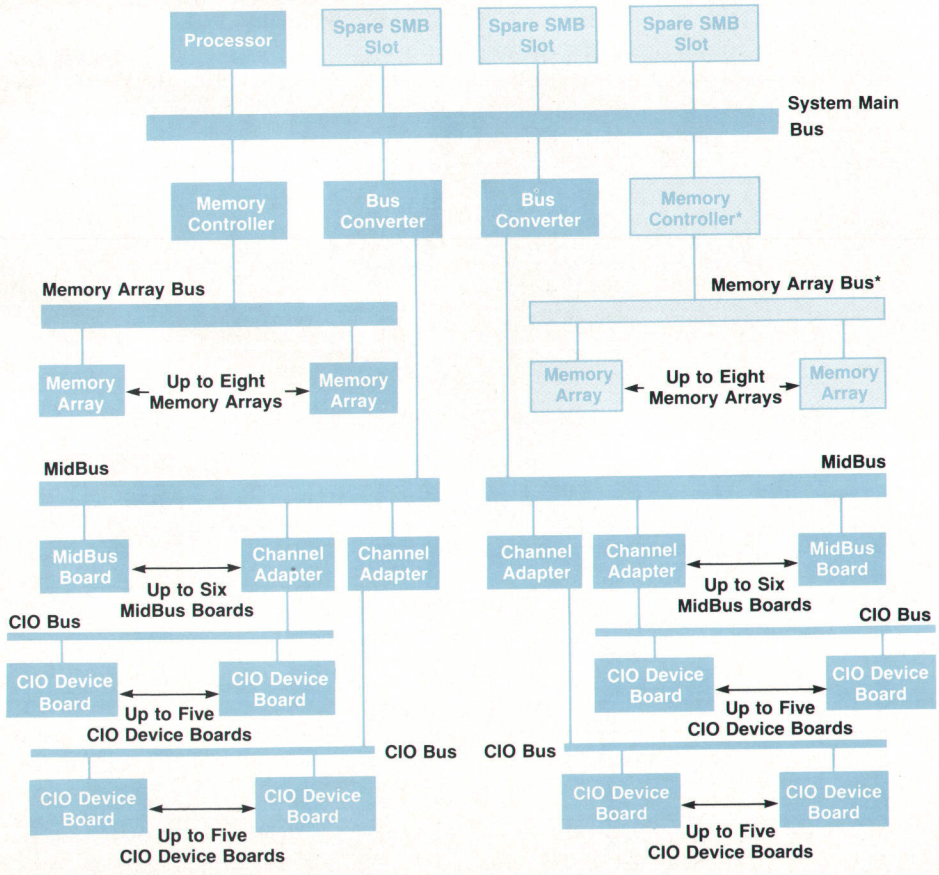
*Other bus names used in some product literature are central bus (CTB) for MidBus, system memory bus for system main bus (SMB), and CIB for channel I/O (CIO) bus.

The Model 850S/Series 950 SPU, as currently released, has four modules connected to the SMB. These are a state-of-the-art NMOS-III VLSI processor with hardware floating-point coprocessor on one board, a memory controller, and two identical bus converters. The SMB is designed to support up to eight modules, leaving room for future product releases. The two bus converters connect the SMB to the MidBuses, providing six slots on each of them. Channel adapters connect the MidBuses to the CIO buses. The main SPU bay supports four CIO buses with five device slots on each.

The hierachical three-tiered bus structure has the power and the flexibility to support higher-performance and larger configurations that may be released in the future. The bus structure also allows the currently released Series 950 and Model 850S configurations to be different, each tailored to its own market requirements. This two-for-one design is a direct result of HP Precision Architecture, which allows a flexible bus structure to be implemented in an architecturally transparent way, and of a well-thought-through design that made flexibility of configuration and smooth growth a very high priority.

Processor Subsystem

The Model 850S/Series 950 processor is a single-board



*Second memory subsystem is a possible future upgrade.

Fig. 2. *Model 850S/Series 950 hierarchical system bus structure. The system main bus (SMB) is on the highest level. Modules on this bus are the processor boards, two bus converters which connect the SMB to the two MidBuses, and memory controllers which connect the memory system to the SMB. Channel I/O (CIO) buses are connected to the MidBuses by channel adapters.*

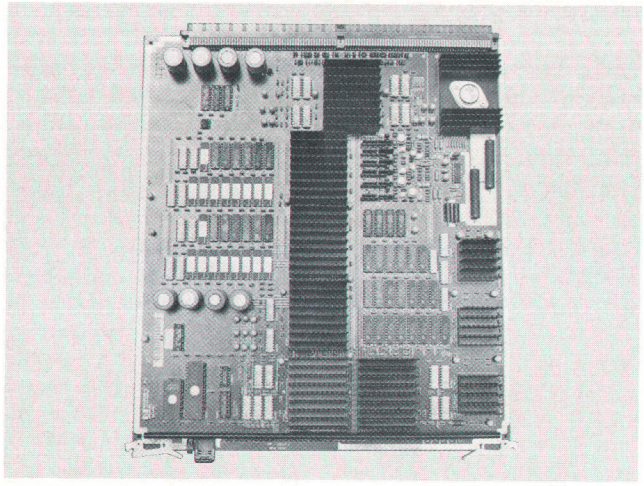


Fig. 3. *Model 850S/Series 950 processor board.*

implementation of the CPU, cache, TLB, and bus interface functions of the level-one HP Precision Architecture processor.[1] A level-one processor supports 16-bit space registers for a 48-bit virtual address space. The processor operates at a frequency of 27.5 MHz and provides an average execution rate of 7.0 MIPS. The Model 850S/Series 950 processor board also contains a floating-point math subsystem capable of providing an average execution rate of 2.8 million Whetstone B1Ds and 0.71 million double-precision Linpacks per second.

Fig. 3 is a photograph of the Model 850S/Series 950 processor board. The board achieves a high level of functional integration and performance by making use of VLSI technology, state-of-the-art commercial RAMs, and precision-tuned clock distribution circuits. The board uses six NMOS-III VLSI chips developed for the Model 850S/Series 950 project: one CPU, one TCU (TLB control unit), two CCUs (cache control units), one SIU (system interface unit), and one MIU (math interface unit). The math functions are implemented using the floating-point math chips developed for the HP 9000 Model 550: ADD (add/subtract), MUL (multiply), and DIV (divide). The Model 850S/Series 950 processor is equipped with a two-set unified data and instruction cache that has a total capacity of 128K bytes. The address translation lookaside buffer (TLB) is split into an instruction TLB with 2K entries and a data TLB with 2K entries. The details of the functional operation of the processor board are described in the paper on page 4.

Processor Buses

Fig. 4 shows the organization of the various functions of the processor board. Three buses are associated with the processor: the math bus, the cache bus, and the system main bus (SMB). The math bus is completely self-contained in the processor board and interfaces the MIU to the math chips. The cache bus is also contained within the processor board and interconnects the SIU, CCU1, CCU0, TCU, CPU, and MIU. The SMB connects the SIU to the memory and I/O subsystems. The SMB is provided on the SMB connector, which also supplies the power, the system clocks, and the interfaces to the support hardware.

The cache bus and the SMB are both precharge-pulldown buses. That is, there are two phases to the bus operation. Each signal is precharged to a nominal 2.85V in one phase (CK2) and conditionally pulled down, or discharged, in the following phase (CK1) (see Fig. 5). All VLSI chips on the bus participate in precharging each signal line. Then one or more chips will drive the signal line low (logical 0 or 1, depending on the signal sense). The precharged buses provide high performance because the NMOS-III drivers can rapidly drive a signal line low, thereby minimizing the data transfer time on the bus. The sender needs to get the bus data ready only just before the drive phase and the

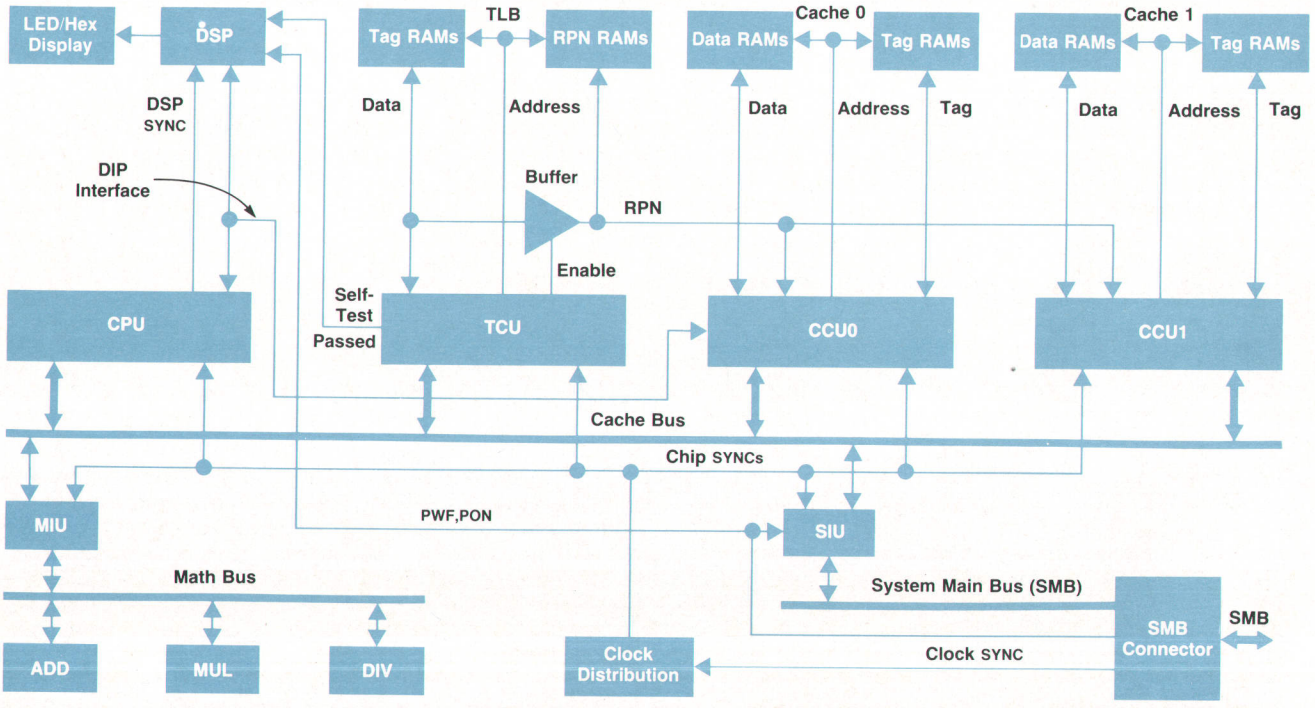


Fig. 4. *Block diagram of processor board functions.*

receiver will have the data immediately after the drive phase. Consequently, the precharge phase causes no performance degradation. To propagate an electrical high value, a signal is allowed to float at its precharge value. The receiver uses a zero-catching design, which latches a logical zero when the signal input falls below the 1.3V minimum trip level for at least 2.9 ns. This receiver is described in the paper on page 4.

Processor Startup

The Model 850S/Series 950 processor uses a diagnostic support processor (DSP) to launch itself at power-up or after a reset. The DSP is implemented using an off-the-shelf microprocessor, and interfaces to the CPU and CCU0 serial ports as shown in Fig. 4. The DSP receives its own clock (the processor clock divided by four). A key challenge for the DSP design was how to synchronize data transfers between the DSP and the CPU or CCU0. Synchronization is achieved by using a double-level clocked register design. The synchronization clock is received from the CPU (DSP SYNC in Fig. 4).

During a processor reset, the DSP sets up the CPU so that the CPU will halt immediately after its internal reset is complete. After successfully completing its internal self-test, the DSP loads part of the processor dependent code (which consists of processor self-test and initialization) into the cache 0 RAMs via scan paths into CCU0 and starts the CPU. Successful completion of the processor self-test transfers control to the rest of the processor dependent code which is located on the processor dependent hardware board. This code completes the self-test and initializes and boots the system.

Locating the DSP on the processor board allows localized self-test that can identify a failing processor board uniquely. Failures in the path from the processor board to the processor dependent hardware board can also be identified. This significantly increases supportability by decreasing the mean time to repair (MTTR).

Processor Board Electrical Design

The Model 850S/Series 950 processor printed circuit board consists of 12 layers: three ground, three voltage, four signal, and top and bottom pad layers. The clock distribution area has extra layers of ground and voltage planes to provide better noise filtering. The signal layers are always placed between ground or voltage planes to provide consistent characteristic impedance. Signal traces are 0.008 inch wide and have 0.025-in pitch to minimize crosstalk. Manual placement and routing were used to route the signals in just four layers, reducing crosstalk and minimizing impedance and propagation delay mismatches on the clock signals. Signal traces typically exhibit about 50Ω of characteristic impedance on an unloaded board. The characteristic impedance is lower on loaded boards because of the input capacitances of the components.

The electrical design of the processor was first based on worst-case circuit simulation using Spice, which provided the initial data for timing budgets and noise margin. The board design was later refined by experiments conducted on revisions of the actual board.

A key challenge was the design of the V_{DL} supply (2.85V nominal), which powers the internal VLSI clock circuitry, all the bus drivers, and the cache bus termination. The processor, memory controller, and bus converter boards have individual V_{DL} regulators.

V_{DL} bypassing was especially a concern on the processor board, where the noise was particularly severe. Several types and sizes of bypass capacitors are used to bypass various frequency bands: 0.001-μF, 0.01-μF, and 0.1-μF ceramics, 22-μF and 68-μF tantalums, and 1000-μF electrolytics. The voltage plane area on the printed circuit board was also maximized.

Since the processor, memory controller, and bus converter boards have separate V_{DL} regulators, any significant offset between the outputs (greater than about 150 millivolts) forces the regulator with the highest value to source the other boards' requirements. Having the clock board supply a common voltage reference for all the local regulators keeps the nominally 2.85V V_{DL} supplies within 50 mV from board to board.

One of the initial design concerns had to do with noise coupling on the cache bus signal lines. The worst-case situation involves an undriven victim trace surrounded by driven traces. Additional coupling is introduced by the cache bus termination scheme which uses a resistor network in a dual in-line package for board density reasons. The effect of the termination resistor package is reduced by mixing signals of different phases in the same package and by using a part that has multiple power and ground pins and built-in bypass capacitance.

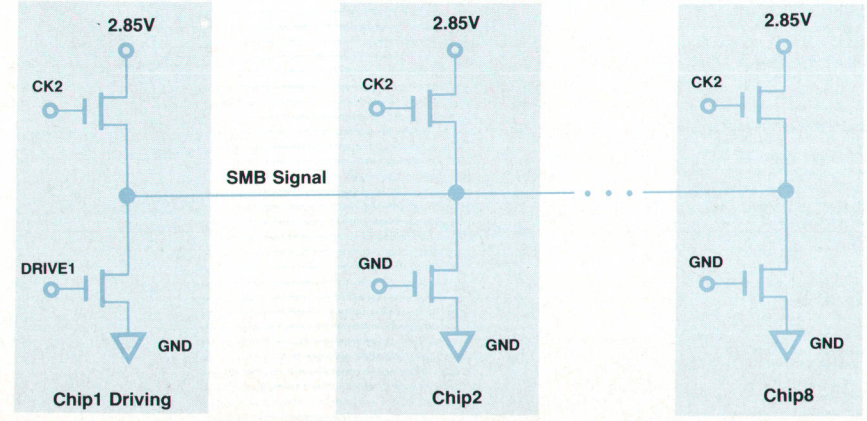


Fig. 5. *Precharge/pulldown structure of the SMB.*

The worst-case design was verified by the construction of slow and fast boards. The slow board was constructed using characterized parts (VLSI chips, RAMs, etc.) whose performance was within specification but close to the minimum acceptable. Similarly, the fast board was constructed using parts that performed near the maximum performance specification. The slow board was used primarily to resolve timing and performance issues, while the fast board was primarily used to verify noise margins and EMI performance. These boards proved to be extremely valuable in establishing confidence in the design, which had very aggressive design goals.

The processor board design incorporates several features to minimize EMI. The outer boundary of the board is provided with window frames which make electrical contact to the chassis through wiping contacts. This helps contain the noise within the board and processor compartment.

Processor Thermal Design

The thermal design of the Model 850S/Series 950 processor board and its integration into the SPU cooling system posed some very challenging design problems. The design parameters were to cool nine VLSI ICs dissipating up to 12 watts each on a single board with a worst-case environment of 15,000-foot altitude and 40°C, plus design margins. It was desirable to use the same airflow used to cool the rest of the electronics in the SPU, components with an order of magnitude lower power than the VLSI ICs. To meet this requirement, the thermal resistance of the VLSI package needed to be an order of magnitude lower than that of the typical package.

The design was started at the chip level, by designing a metal heat spreader into the ceramic package. The VLSI chip is bonded directly to this heat spreader, and a heat sink is attached to the other side. Theoretical and finite element analysis methods were used in the design of the heat sink, and a wind tunnel was designed to test prototypes and correlate the theoretical analysis.

Once the component level design was fully understood, the development was expanded to the board level. The large heat sinks required to cool the VLSI components present enough back pressure to the moving air and their distribution on the board is irregular enough that a uniform airflow across the board could not be assumed. A wind tunnel large enough to hold a single board was built and the airflow studied to determine the exact amounts of cooling air that each VLSI IC would receive. A thermal test board was also designed and built so that the junction temperatures of all the ICs could be directly measured.

Once the design was confirmed at the board level, the thermal test board was put into a complete prototype SPU to confirm the airflow at the system level, and system airflows were measured. This test set was finally put into an environmental test chamber and tested at elevated temperatures to verify the complete thermal design.

System Main Bus

Up to four processor slots, two memory subsystem slots, and two I/O subsystem slots are available for connection to the high-performance system main bus (SMB). The SMB consists of 64 bits of multiplexed address/data and 17 bits of control signals. SMB transactions are split into request and return phases. This allows interleaved requests from multiple masters to be serviced in parallel. The SMB operates at 27.5 MHz and is capable of sustaining a 100-megabyte/s data bandwidth.

Both the processor and the bus converters (which connect the SMB to the two MidBuses) can initiate SMB transactions as masters. The memory controller is always a slave. Upon winning arbitration, a master module asserts a 32-bit address, seven bits of command/size information, and a 6-bit tag that identifies it as the master. SMB read transactions are initiated by a master module to transfer 4, 16, or 32 bytes from the slave to the master. Table I shows all of the SMB transactions. The processor issues a 4-byte read during a load to I/O space and a 32-byte read during a load/store cache miss. The bus converter issues a read when the CIO bus channel requires data from memory. Clear transactions are initiated by the processor or the bus converter to gain access to a semaphore. During a semaphore operation, the memory controller clears the first word of the half line and returns the old half line to the master. Return and return clear transactions are driven by a slave device's returning data following a read or clear operation, respectively. Read or clear transactions that are "smart" are initiated by the processor during a cache miss and require a cache coherency check by any other processors during the return transaction from the memory controller. Read or clear transactions that are "dumb" are initiated by the processor or the bus converter during I/O or DMA operations and do not require a cache coherency check.

During SMB write transactions, the master sends 4, 16, or 32 bytes of data to the slave. Purge cache, flush cache, and purge TLB transactions are broadcast on the SMB to implement these instructions. Five virtual indexing bits are inserted into the 32-bit SMB real address to allow index-

Table I
SMB Transactions

Transaction Type	Size (bytes)	Initiated By	Received By
Read (dumb)	4, 16, 32	SIU, BC	BC, MC,
Read shared (smart)	32	SIU, BC	SIU, MC
Read private (smart)	32	SIU, MC	
Clear (smart)	16	SIU	MC
Clear (dumb)	16	BC	MC
Write	4, 16, 32	SIU, BC	SIU, BC, MC
Return (dumb)	4, 16, 32	BC, MC	SIU, BC
Return shared (smart)	32	MC	SIU
Return private (smart)	32	MC	SIU
Return clear (smart)	16	MC	SIU
Return clear (dumb)	16	MC	BC
Purge cache		SIU	SIU
Flush cache		SIU	SIU
Purge TLB	32	SIU	SIU

SIU = System Interface Unit
BC = Bus Converter
MC = Memory Controller

ing into a 64K-byte cache during cache coherency checking.

To ensure fair access to the SMB for each module, the SMB has a circular arbitration scheme based upon a priority chain with a lockout. The arbitration hierarchy is derived from the SMB backplane wiring that connects the arbitration request output signals (NARB) to the arbitration inhibit input signals (NINH) of the lower-priority modules.

If a module is performing a write, a return of read data, or a purge TLB transaction, data is transferred on subsequent bus cycles. The NBUSY signal locks out arbitration to tie up the bus for multicycle transactions, and the NOK signal is asserted to indicate that the bus address and data are valid. NACK (acknowledge) and NRETRY (retry) are used to indicate acceptance or refusal of transactions. Either NACK or NRETRY is asserted by a cache coherency checking processor during a return transaction if a clean or dirty copy of the line is detected, respectively. The SMB modules log and report parity, addressing, or protocol errors. The SMB address and data are protected by a single odd parity bit.

SMB Electrical Design

SMB electrical design was extensively modeled to predict the interactions of the chips, pin-grid array packages, connectors, and boards. Signal coupling, oscillation, di/dt noise, power supply noise, and maximum frequency of operation were simulated. Worst-case SMB simulations and measurements verified 27.5-MHz operation and an adequate precharge level.

The interconnect model is a complete, unscaled representation of the longest trace path. Propagation delay and impedance values are based on worst-case length and thickness variations. The chip models and power supply net impedences are scaled as a compromise between detail and Spice simulation time. The model is limited to seven signals, two ground pads, and a single power pad. Detailed pin-grid array signal models and inductive coupling are included only in the driving chip.

A new connector and backplane design was required for the 27.5-MHz SMB. The design goals were to maintain a good impedance match between printed circuit boards through the SMB connector, minimize signal crosstalk, and minimize the SMB length for high-frequency operation. The new connector was developed using proven pin and socket technology and our current qualified vendor to reduce program risk. Press-fit technology allows us to load connectors (and therefore boards) from both sides of the backplane to implement a ten-inch-long, 27.5-MHz SMB.

The design goal for the connector impedance was 50Ω to match the 30Ω VLSI package impedance to the 50Ω circuit board and backplane impedance. Impedance matching is important to minimize signal reflection and oscillation. However, typical pin and socket connectors exhibit 80-to-100-ohm impedances. To reduce connector impedance and signal crosstalk, the number of ground pins was increased, resulting in a 1:1 ground-to-signal pin ratio. Also, a low-impedance ground return path was added: a fifth row was added to the 4-row, 440-pin connector in the form of small plates and flat spring contacts. This creates a very short, low-inductance path with a relatively wide, flat surface area. The new connector's impedance of 60 to 65 ohms allows the SMB to run at 27.5 MHz and prevents excessive crosstalk during multiple switching operations.

Memory Subsystem

The main memory system of the Model 850S/Series 950 is designed for high bandwidth, large maximum memory size, and reliable operation. The main memory controller can support from 16M to 128M bytes of memory. Provision has been made to allow the addition of a second memory controller with its own memory bus and 16M to 128M bytes of memory; however, this configuration is not supported at this time.

Memory Controller Board

The memory controller board provides the interface between the system main bus (SMB) shared between the processor and I/O and the memory array bus shared between the memory controller and the memory array boards. The memory controller communicates with from one to eight 16M-byte memory array boards over the ASTTL memory array bus.

The heart of the memory controller board is the memory controller chip, a proprietary HP VLSI IC fabricated in NMOS-III. This IC incorporates the logic to interface the SMB to the memory array bus, provide error detection and correction, conform to HP Precision Architecture requirements, and generate DRAM control timing, all in one 272-pin pin-grid array package. The high density of this controller contributes to the reliable operation of the memory system.

During power failures, data in memory is maintained by external batteries for at least fifteen minutes. To maximize the amount of battery backup time available, the memory controller IC is not battery-backed and TTL on the memory controller board handles all memory refresh operations.

Memory Array Boards

Each memory array board provides the basic control logic and buffering to drive 144 dynamic random-access memory (DRAM) ICs. The DRAMs are arranged as two banks, each 1M words by 64 bits (plus 8 bits for error correction). Memory data access interleaves between these banks and uses the nibble-mode function of the DRAMs. The combination of bank interleaving, nibble-mode DRAMs, and a wide data bus provides the high bandwidth of the memory system.

Memory can be accessed in either 16-byte or 32-byte transactions. A 32-byte transaction can come from either a processor cache miss or the I/O system and requires 17 cycles for read operations and 16 cycles for write operations from the memory controller to the selected array. A 16-byte transaction comes from the I/O system and requires 12 cycles for either read or write operations. This timing allows a maximum sustained performance of 51 megabytes per second during 32-byte read cycles and 55 megabytes per second during 32-byte write cycles.

To maximize performance, careful attention was paid to board and backplane layout, the choice of TTL devices used, and the internal design of the memory controller IC. Spice simulations were done on the backplane and DRAM drivers to minimize both delay and skew. Careful analysis of the DRAM and bank switching control signals was done

to optimize performance while still permitting a large memory size of 128M bytes per memory controller.

Memory Controller Chip

The memory controller chip employs a data path organization with a central register stack containing the address/data path of the controller. The design, implemented with 80,000 transistors, consists of two I/O bus interfaces, three control PLAs, and a register stack containing nine circuit blocks. Of the 272 pins in the package, 84 are used for the SMB interface, 93 for the memory array bus interface, four to connect to nine serial scan paths, and 91 for power, ground, and clocks.

A number of circuit blocks are unique to the memory controller. There are comparators to map incoming addresses to various address ranges, queues to buffer addresses and data, and error detection/correction circuitry for the SMB and memory array bus addresses and data. To sustain maximum throughput to memory, 64-bit buses are widely employed in the data path and interfaces.

The memory controller services reads and writes to main memory as initiated on the SMB. In addition, the memory controller responds as an I/O device to configure memory and return status. The controller can buffer two read transactions and one write transaction. Each transaction can be either a 16-byte or a 32-byte data transfer. The controller buffers the data for one read request and one write request. The controller's write queue is logically an extension of the processor's cache. Internally, the controller initiates memory refresh sequences to main memory.

The memory controller operates as two independent units controlling the SMB and memory array board interfaces, respectively. Internally, the controller arbitrates for use of the queues and buffers within the chip. The controller can simultaneously process an SMB read/write request or return read data while processing a memory array bus read/write to memory or memory refresh sequence.

This partition ideally maximizes memory throughput during heavy traffic conditions. When two memory reads are buffered, the controller is able to start the second memory read even though the first memory read's data is still in the internal data buffers. The second transaction is allowed to proceed if the first read transaction is able to return data to the SMB before the second read's data becomes available. If the read buffer remains occupied, the second read is aborted and restarted.

In any computer, data integrity is paramount. Each 64-bit memory word has eight check bits associated with it. These bits allow the memory system to detect and correct all single-bit errors in any word of memory. Should a single-bit error occur, the incorrect data and location are stored in a memory controller register for operating system use or diagnostic logging. These eight extra check bits also allow for the detection of all double-bit errors in any 64-bit word. Double-bit errors are not correctable.

Parity is generated and checked between the memory controller and the selected memory array on memory addresses. This will detect otherwise unreported transient errors that could destroy data at random memory locations.

I/O Subsystem

The Model 850S/Series 950 I/O subsystem is designed to the specifications of HP Precision I/O Architecture.[3] The main design feature of this architecture is its transparency at the architected level. To software, the I/O system is a uniform set of memory mapped registers independent of which bus they are physically located on.

Although the architecture prescribes a simple and uniform I/O system software interface, the hardware is allowed tremendous flexibility. In particular, the I/O system can include any number of dissimilar buses interconnected by transparent bus converters. The transparent bus converters make the boundary between adjacent buses invisible to software, automatically compensating for differences in bus speed or protocol. The mapping between buses on the Model 850S/Series 950 is accomplished primarily through the use of bus converters.

The architecture differentiates between HP Precision Architecture buses and other buses. An HP Precision Architecture bus supports the HP Precision Architecture standard transactions and can be connected to other HP Precision Architecture buses through transparent bus converters. Other buses can be connected to an HP Precision Architecture system through foreign bus adapters, which are not transparent, but instead have an architected software interface. The Model 850S/Series 950 takes advantage of bus converters where an interface to an existing bus (such as HP's CIO bus) is required.

The I/O system of the Model 850S/Series 950 relies heavily on the same custom NMOS-III VLSI technology used in the system's processor. Two custom ICs were developed for the I/O subsystem: a bus converter chip, which implements a subset of the bus converter functionality, and a CIO bus channel adapter chip, which implements the complete translation of the MidBus protocol to the CIO bus protocol. The use of NMOS VLSI technology in these circuits made possible their implementation in a reasonable size and at a much lower cost than alternative technologies.

Bus Converter

The function of the SMB-to-MidBus bus converter is to convert transactions between the SMB and the TTL-signal-level MidBus. The bus converter consists of a single board containing one custom NMOS-III VLSI bus converter chip and several TTL buffer chips.

As the first bus converter to be implemented in an HP Precision Architecture system, the Model 850S/Series 950 bus converter had a large influence on the development of the bus converter definition for the architecture. Much of the bus converter architecture was developed in parallel with the bus converter design, and the architecture benefited from the insights and experience of the bus converter implementation team.

Since transactions can originate on either the SMB or the MidBus, there are two sets of control centers within the bus converter chip, each of which is associated with one of the two bus interfaces. Communication between the two interfaces is facilitated by data/address queues and an array of transaction state latches. The algorithms of the bus converter are designed to maximize throughput of the most

frequent operations, especially writes from I/O modules on the MidBus to memory on the SMB.

Testability is an important aspect of the bus converter design, and the test engineer was an integral member of the design team. The bus converter follows the conventional NMOS-III VLSI test methodology using shift chains, single-step, and a debug port to allow diagnostic access to almost every storage node within the chip. The bus converter is capable of scanning 977 bits of state information from seven scan paths routed through the functional blocks.

Channel Adapter

The channel adapter is a bus adapter module that interfaces the MidBus to the CIO bus, which is a standard HP I/O bus. The channel adapter performs all the necessary protocol conversions between the MidBus and the 16-bit, five-megabyte/second peak bandwidth CIO bus. The channel adapter consists of a single board containing one custom NMOS-III VLSI channel adapter chip, several TTL buffer chips, ROMs containing code for I/O drivers and self-test, and miscellaneous support logic.

The channel adapter allows full compatibility with all existing HP CIO I/O cards, as well as additional HP CIO cards presently in development. Although the CIO bus protocol differs from HP Precision I/O Architecture in many ways, the foreign bus adapter maps all of the necessary CIO functions into the standard register interface through which it communicates with the I/O system. In accordance with the CIO bus protocol, the channel adapter serves as a central time-shared DMA controller on the CIO bus. The channel adapter is the initiator of all CIO bus transactions, and it is the arbitrator that maximizes the efficient use of the CIO bus bandwidth. The channel adapter provides data buffering and address translation as it transfers data between the I/O modules on the CIO bus and the memory modules on other buses within the system. The channel adapter also translates interrupts and error messages into the protocol used by the HP Precision Architecture I/O system. By handling all normal DMA transfers and the majority of error conditions in complete autonomy, the channel adapter can greatly reduce the processor overhead required to operate the CIO bus. Except in the rare error case that requires software intervention, the channel adapter appears to the system as a set of standard DMA adapter modules conforming to the HP Precision Architecture specifications for an I/O module.

System Clock

The accurate generation and distribution of a 27.5-MHz clock signal is crucial to the performance of the Model 850S/Series 950. The clock signal originates on the clock board and is shaped, amplified, and distributed to eight individual slots on the system main bus (SMB). While the Model 850S/Series 950 can use up to four SMB slots, the clock distribution network must support the full eight SMB slots to provide for future expansion. Each SMB board has its own local driver circuitry which then distributes the clock signal to individual VLSI ICs on the board. The reduction and control of skew in the clock system was a major challenge to the design team and required tight tolerances on many aspects of the design.

The system clock originates on the clock board at TTL levels. A single hexadecimal NOR TTL buffer is used to drive the discrete transistor clock sync circuits and the TTL refresh clocks. The use of a single buffer minimizes skew between the VLSI memory controller and TTL logic on the memory controller board which provides battery-backed refresh signals. The discrete clock sync circuitry provides the signals that become the main system clock.

Conversion between the the TTL clock levels and the analog levels of the clock sync circuitry is done by an npn differential amplifier (Fig. 6). The output of the differential amplifier is then fed to an emitter follower, which drives three npn/pnp push-pull pairs which drive the clock SYNC signals out to receiver circuitry on up to eight different boards on the SMB. The emitter follower's output is the one absolute time reference point for the clock generation and distribution system.

The clock SYNC signals enter the 50Ω backplane traces through source-terminated transmission lines. Each re-

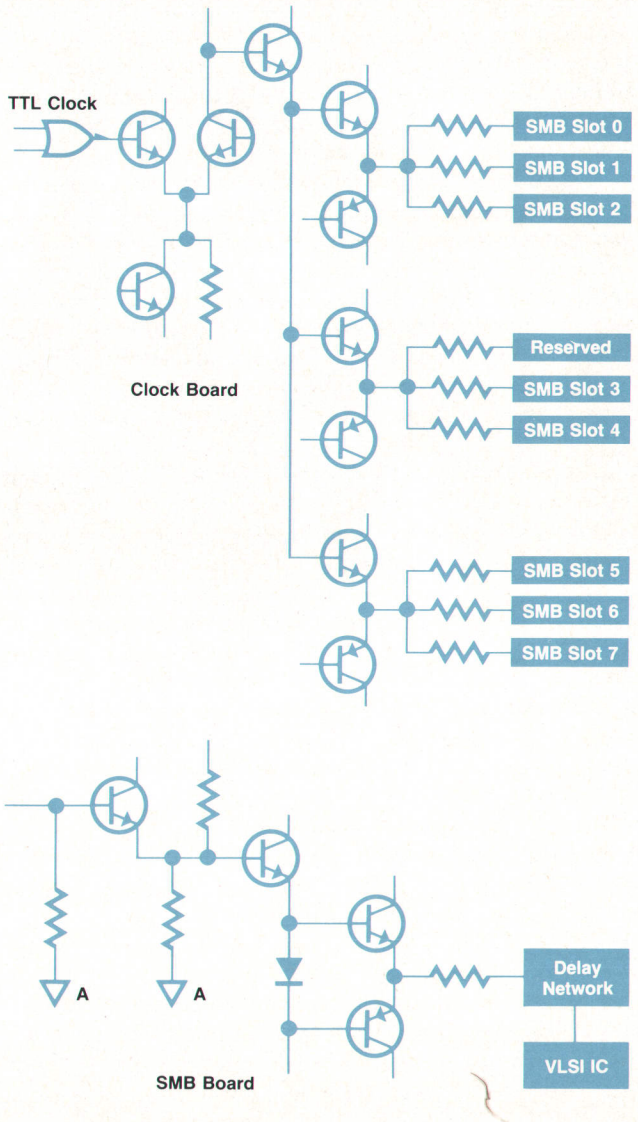


Fig. 6. *Clock distribution system.*

ceiver board has an npn emitter follower receiver which then drives an npn emitter follower predriver. The predriver controls an npn/pnp push/pull pair which drives the clock signal through a termination and deskewing network to each IC.

The clock deskewing network minimizes circuit variations and increases performance. Each network is adjusted at the factory by adding or subtracting delay lines by means of jumpers. All boards are set to the same standard, so the customer can expect a clock system that is optimized for high performance but will not require periodic field tuning and will accept boards manufactured at any HP site.

Each VLSI IC has its own internal circuitry that develops two phases of nonoverlapped clock signals for internal use. The SYNC signal from the discrete clock circuitry is only used as a timing reference; hence the name "clock SYNC." In the Model 850S/Series 950 clock system only the rising edge of the SYNC signal has any significance.

For a given IC process and machine layout many variables, such as device and backplane propagation delays, are fixed. Thus the difference in time between the clock seen at one point in the system and the clock seen at another point, or clock skew, can limit a given machine's maximum frequency of operation. In the Model 850S/Series 950 the limiting effects of clock skew showed up on both the cache bus and the system memory bus. Reduction of this skew was crucial to system performance.

Skew was reduced by careful design, layout, and testing of backplanes and printed circuit boards. Time intervals of tens of picoseconds were calculated and measured. High-frequency second-order effects such as the effects of via capacitance had to be understood to minimize the differences between clock circuits on different boards. The solution has resulted in a clock system that can route a clock SYNC signal to three to eight SMB boards and a maximum of 28 different VLSI ICs with a total skew of less than 800 ps.

Control, Diagnostic, and Power Subsystem

One of the CIO bus channel adapter slots has been customized to allow the installation of the access port card. This card provides operator access to the SPU, remote support of the SPU via modem connection, and remote operator access to the SPU. Operator control of the SPU is provided through an operator's console. This console can be located away from the SPU for efficient computer room layout. A control panel located on top of the SPU contains a subset of the operator controls and provides a keyed switch for operator access security.

The control panel provides a visual indication of the SPU's operational status and an uncomplicated, physical interface to start and control the system. It provides system self-test information during the boot process and system functionality during normal operation through a four-digit hexadecimal display. Power system diagnostic information is supplied by LEDs. All diagnostic information is combined into three basic machine conditions (normal operation, operation with an operator warning, and nonoperating) which are displayed through three indicators (green, red, and yellow) visible from all sides of the SPU. An additional high-visibility display indicates whether remote operator access is enabled.

Design of the control panel for electromagnetic compatibility was especially difficult since the control panel must contain any electromagnetic interference (EMI) generated in the high-performance logic system, and must protect the circuitry from electrostatic discharge (ESD). Mechanical and electrical design teams worked together to meet these stringent requirements. Interface circuits are filtered to remove normal-mode noise before cabling to the control panel. The filters also guard the internal circuits against ESD-induced transients. The mechanical design minimizes the control panel's ESD entry points by using light pipes and a molded plastic enclosure. The inner surface of the enclosure is zinc-coated and electrically connected to the chassis to act as an EMI shield and a conductive path for ESD.

The Model 850S/Series 950 power system delivers 2.4 kilowatts of dc power to the processor, memory, I/O, and support subsystems. Ac power is filtered and converted to 300Vdc in the ac unit. This 300V dc power is then used to power eight dc-to-dc switching power supplies, which provide the dc voltages required by the electronics. High-current power is delivered through a solid copper bus bar, and lower-current voltages are supplied via cables. Some critical voltages are regulated on the boards where they are needed.

SPU Product Design

Packaging

The Model 850S/Series 950 is packaged for the international EDP room environment, where operators work on the SPU from remote terminals and have only intermittent contact with the SPU itself. When there is contact with the SPU, the operator is usually standing. For this reason, the SPU height and control panel location were designed to be a good fit for the middle 90% of the international population. Additionally, the primary status indicators and the SPU itself are designed to be viewed from all sides.

The enclosures (front and back doors, side panels, and top covers) of the SPU are made of injection molded structural foam plastic. This provides an excellent appearance, with crisp lines and consistent detailing. The use of molded plastic also allowed the design of features for quick, easy access to the machine, aerodynamic venting for more efficient cooling, elimination of sightlines into the machine, and sound absorption.

The processor and I/O cardcages incorporate RFI shielding, fixtureless backplane assembly, air filter, fan plenum attachment features, and space for the maximum number of boards to be installed. Similarly, the molded fan plena, system frame, and power supply rack all integrate many functions. This level of integration allows very efficient manufacturing and simple field installation and access.

The package is key to providing the growth and upgrade potential of the product. Using two back-to-back cardcages allows the devices on the SMB to be placed close enough together that space is left for additional SMB devices to be added in the future. All the major subsystems in the SPU are near other similar subsystems. This allows the use of common cooling and EMI shielding systems, and

minimizes the number of parts required in the package.

Cooling is simplified by the dual-cardcage design. All of the boards in the system, including the processors, are arranged vertically and side by side, like books on a bookshelf. These are in the middle of the SPU and cooling fans are underneath. The fans are mounted in plastic fan trays and pull cool, clean air across the components in the system. The plastic trays and simple card arrangement combine to provide easy manufacturing and servicing of the cooling system.

This complex integrated package design was possible because of extensive interaction between the packaging engineers and the electrical engineers early in the development of the product. The result is an integrated package that minimizes floor space and complexity, without sacrificing manufacturability or serviceability.

System Growth

A design goal was that the basic Model 850S/Series 950 SPU hardware design be able to support at least two substantial performance upgrades, including larger memory and I/O configurations, for possible future HP product releases. The electrical, firmware, and product design elements have been designed to support this goal without adding a significant factory cost burden to the initially released product.

The performance levels of the buses (Fig. 2) and their support capabilities are key to providing growth potential. The high SMB bandwidth can support four additional modules of equal or higher performance, ensuring possible processor performance and memory expansion growth paths. The two MidBuses can support high-performance I/O devices or additional VLSI channel adapters, allowing the possibility of external I/O expansion. The bus structure was thoroughly analyzed, modeled, and margin tested to ensure the long-term stability required for growth.

Careful attention to configuration and bus length requirements produced an internal configuration that is logical and supports upgrade goals. The SMB has to be short for performance reasons, so SMB devices are on both sides of the CPU/memory backplane, cutting the maximum bus length in half. The bus converters distribute the MidBuses on a separate I/O backplane parallel to the CPU/memory backplane. A power distribution bus bar is sandwiched between the two backplanes. This efficient basic layout provides a very robust configuration with minimum space and cost penalties. Slots for three possible future additional SMB devices are found on the CPU side of the cardcage. Slots for a second memory controller for a second memory bus supporting eight more memory arrays are there for possible future releases. This design meets the high performance and upgrade goals important to our customers.

Reliability

The reliability of the Model 850S/Series 950 SPU is expected to be greatly improved over previous SPUs. The VLSI technology contributes to this improvement by a large reduction in the number of parts in the CPU area alone. For comparison, the HP 3000 Series 70 takes eight Model 850S/Series 950-size boards for the processor, whereas the Model 850S/Series 950 uses only one. The VLSI memory controller and bus converter greatly reduce parts count. Parts counts are also reduced in other areas.

The design also includes features to improve reliability further as the product matures. An excellent example is that the memory system was designed to take advantage of 1M-byte DRAMs for performance and reliability reasons from the beginning. By the time the cost of those chips made a 16M-byte board economical, the board design was already done. There are many other areas in the current design that will enable further reliability improvements.

Supportability

SPU support is composed of two categories: initial and add-on installation, and on-going support. Both of these categories are addressed by the design.

Supportability goals were set early in the development process and reviews were conducted during all phases of the development cycle to evaluate the ability of an HP Customer Engineer to isolate and replace failed assemblies quickly and accurately. For example, there are no configuration switches or jumpers on any assemblies. Instead, all assemblies are configured automatically during power-up and those required for initial system boot are tested automatically during the boot process. Captive fasteners are used extensively to speed up unit removal and replacement. Where captive fasteners could not be employed, the number of screw sizes was minimized to simplify the reassembly process. Front connectors have been eliminated from all assemblies, and cabling is reduced to a minimum.

Fault diagnosis uses a four-level approach, with each level capable of diagnosing all hardware necessary to load the next level of diagnostics. At power-on, the processor board initiates the first level by running a self-test on itself. After this test is passed, a second level of self-test is invoked, testing all the hardware in the boot path. The third level of diagnostics can now be run, and all the boards in the system can be tested. After this, the operating system can be loaded. Then the fourth level of diagnostics, a set of on-line diagnostics, can be run.

Manufacturability

Throughout the development of the Model 850S/Series 950 SPU, design engineers and manufacturing engineers worked together to optimize the manufacturability of the SPU. Early in the project, the SPU design team set goals for part count, part number count, number of vendors, assembly time, and percent autoinsertability of components. These goals were chosen to minimize overhead cost and cycle time and increase quality. In addition, standard shipping packages and shipping methods were targeted and used to help set the external size constraints for the machine.

Design for manufacturability guidelines were used by all engineers, and design reviews were conducted to improve the design. As the development of the SPU progressed, these critical parameters were tracked and the designs reviewed with the intent of improving the performance in these areas. Progress toward the goals was reviewed and tracked on a monthly basis, and a complete assembly analysis was performed before each major design phase of the project. The result is a product with high manufacturing

efficiency, which translates into lower prices and better value to the customer.

Shipment methods and their effect on installation time were important considerations. Two methods of shipment were developed. For shipments within the continental United States, the Model 850S/Series 950 is shipped as a single unit in a padded van. For overseas shipments, the Model 850S/Series 950 SPU is put on a single pallet which fits into standard aircraft cargo containers. This two-method shipping strategy reduces shipping costs for domestic shipments by eliminating parts.

Installation is now very simple. No mechanical or electrical assembly is required at the customer's site. The installation time is reduced to less than half that of the HP 3000 Series 70.

This commitment to designing for manufacturability resulted in a significant improvement over existing products. All measures show improvement over current high-end products. Part count, part number count, and assembly time have been reduced by 50%. Manufacturing cost has been reduced by 25%, and autoinsertability has been increased by 33%.

Acknowledgments

The Model 850S/Series 950 SPU development team was composed of design, manufacturing, support, and product marketing engineers, all working together from the initial concept and throughout the development process. In a project of this magnitude it is difficult to acknowledge everyone who has contributed. First, we must thank the other contributors to this paper in addition to the main authors listed: Don Rogers on the control diagnostic and power subsystem, Paul Rogers and Kim Chen on the processor subsystem, Paul Rogers, Darrell Burns, and Elaine Christiansen-Thomas on the SMB, Brian Boschma and Chuen Hu on the memory system, Ed Holland, Allan Strong, Tim Rueth, André Wiseman, and Elio Toschi on the I/O system, and Bill Berte on the clock system. Next we must thank the people who worked with us on the project: Steve Brown, Karl Hassur, Harold Kaye, Peter Lee, and Jim Smeenge on packaging, Norm Galassi, Norm Marschke, Ed Miller, and Jim Schinnerer on the power system, Sheri Abrams, Bob Lundin, Sue Skinner, Vilma Di Mariano, Nobuo Ikeda, Jeff Zimmerman, and Jim McGovern on transition and technical support, David Means, Klaus Tippelt, Alfred Haaga, and Eric Decker on bring-up, Sui-Hing Leung, Charles Tan, Stewart Lee, and Ken Holloway on the memory system, Rich McClosky, Ilan Krashinsky, Donald Benson, Bob Brooks, Yeffi Van Atta, Dilip Amin, Jon Sanford, Cy Hay, Nick Fiduccia, Ken Chan, and Ed Holland on the I/O system, Linley Gwennap, Francine Nunley, and Carol Flint on processor dependent code and hardware, Sam Leung on the power system monitor, John Delaney, Ron Guffin, Dean Lindsay, and Bill Winchester on diagnostics, Bryan Hoover and Wayne King on the access port card, Sharoan Jeung and Cory Chan on tools, Dick Vlach, Valerie Wilson, and Theresa Corsick on IC layout, and Cliff Lob, who managed the end of the VLSI effort. Finally we must thank those people that made this program possible: George Bodway, Dana Seccombe, Peter Rosenbladt, Denny Georg, Mark Canepa, Jack Anderson, and Joel Birnbaum.

References

1. M.J. Mahon, et al, "Hewlett-Packard Precision Architecture: The Processor," *Hewlett-Packard Journal*, Vol. 37, no. 8, August 1986, pp. 4-21.
2. D.A. Fotland, et al, "Hardware Design of the First HP Precision Architecture Computers," *Hewlett-Packard Journal*, Vol. 38, no. 3, March 1987, pp. 4-17.
3. D.V. James, et al, "Hewlett-Packard Precision Architecture: The Input/Output System," *Hewlett-Packard Journal*, Vol. 37, no. 8, August 1986, pp. 23-30.

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

HEWLETT-PACKARD JOURNAL

September 1987 Volume 38 • Number 9

Technical Information from the Laboratories of Hewlett-Packard Company
Hewlett-Packard Company, 3200 Hillview Avenue
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
P.O. Box 529, Startbaan 16
1180 AM Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

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

5953-8564