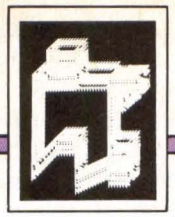

A MULTIPROCESSOR MINICOMPUTER DESIGNED FOR UNIX

Using industry standard design, challenging memory requirements of the UNIX operating system are met by a multiprocessor equipped minicomputer

by Curtis Myers
and Grant Munsey

A multi-user minicomputer designed specifically for the UNIX operating system incorporates industry standard design elements. The architecture's sole purpose is to optimize UNIX by exploiting its strengths and designing around operating system characteristics that hinder performance in multiprocessing environments.

Designing system hardware for a specific operating system is not a new concept; neither is designing according to readily available and understood industry standards. However, a number of factors, such as the growing cost of software development for both the system integrator and the end user, have combined to make these two design criteria essential for future systems. Additional factors include the falling price of hardware, standardization of buses and interfaces, leveling of technical capabilities, and growing availability of special purpose integrated circuits. These factors have diminished the advantages of proprietary architectures and operating systems.

In short, market forces have played the largest part in system design. Providing high system performance and throughput was the first design criterion. The second was to provide interfaces for the widest range of peripherals possible, and the third was to provide a growth path through a regular, uniform architecture.

The strengths of UNIX

UNIX is a clean, simple operating system with comprehensive capabilities. Highly modular, it consists of many separate processes that could fit within a limited hardware address range. Program development consists of small single-task modules that are later linked together; its interprocess controls allow very large applications to be developed and tested in modules. While a full UNIX system occupies up to 6M bytes of secondary storage, most modules are very compact.

Because the total UNIX system is large, it must be disk resident, and the system designer must provide fast and efficient disk access for the required memory swapping. This is particularly important in view of the tree structure of the UNIX file system which involves many indirect accesses. In a multi-user commercial environment dominated by file transfers and inquiry/response tasks, this directory structure poses a challenge. As system loading increases, the performance of an unaided central processing unit (CPU) quickly degrades with the overhead of handling interrupts and the associated

As cofounder and vice president of corporate development for Plexus Computers, Inc, 2230 Martin Ave, Santa Clara, CA 95050, Curtis Myers is responsible for engineering and strategic and product planning functions. Mr Myers has a BSEE and an MBA from the University of California, Berkeley.

Grant Munsey is a senior software engineer at Plexus, where he heads the operating system development. Mr Munsey has a BSEE from the University of California, Irvine.

memory swapping; this results in response times that suffer dramatically. Intelligent input/output (I/O) controllers and an efficient common memory manager can address some of these problems.

Design overview

The Plexus P/40 is a 16-bit multiprocessing minicomputer designed to rapidly integrate hardware and software. Its hardware essentially consists of off the shelf microprocessors, while its system design is a proven, high performance minicomputer architecture. Moreover, the I/O design, based on Intel's MULTIBUS, is easily understood and interfaced. Using industry standards, including the MULTIBUS backplane, the device has a storage module disk (SMD) interface, UNIX, and supports asynchronous, bisynchronous, and high level data link control/synchronous data link control (HDLC/SDLC) communications (Fig 1).

In addition, the P/40 provides intelligent I/O controllers, an efficient common memory manager, fast access through fast disks, and an intelligent SMD type disk controller supported by a direct memory access (DMA) channel to main memory. The technical design goal in using multiprocessors was to eliminate traditional bottlenecks on data transfer rates and interrupt handling so that the system performance would have only two technical limitations: disk access time (20 to 40 ms) and overall main processor performance.

The main processor, designated the job processor, is built around a 16-bit microprocessor and is responsible for executing user programs. Because its shorter instructions meant fewer fetches per instruction and in most cases faster execution, the Z8000 series was chosen over other microprocessors with larger address space. The nature of the design is such that, by distributing processing power throughout the system, a job processor can be chosen for its efficiency at particular tasks. The system's 8M-byte physical address space was considered more than sufficient for the initial markets; the present model supports 4M bytes.

At least two other types of processor, also built around Z8000 microprocessors, assist the job processor—the intelligent communications processor (ICP) and a mass storage controller. These peripheral processors handle all interaction with terminals, printers, and disk and tape units independent of the main processor. Information is transferred between the peripheral processors and main memory using DMA techniques. Since each processor has its own DMA channel, the job processor is rarely interrupted. Multiple ICPs, job processors, and/or mass storage processors can be plugged into the MULTIBUS backplane.

The major issues affecting system design were how and what the auxiliary processors would communicate with the job processor, and how memory management would efficiently maintain control. UNIX made it fairly easy to resolve these issues since it is built on a set of small, relatively independent modules.

Job processor

Nearly all features of the Z8001 are used, including separate instruction and data spaces, user and system modes, and vectored interrupts. With 16 general purpose registers and 110 distinct instruction types, the job processor uses four address spaces. The normal I/O space is used to communicate with I/O ports on the MULTIBUS while special I/O space (SIO) is used to control the processor. Memory instruction and data spaces are used to access the corresponding locations in main memory.

An intelligent controller helps maintain high rates by offloading the main processor.

The job processor maps all normal I/O references to the MULTIBUS. When executing an I/O instruction, the processor requests control of the MULTIBUS through its bus arbiter. When control is granted, the processor places the port address on the lower 16 address lines (A0 through A15) and asserts the appropriate read or write control line (IORC or IOWC). After the addressed MULTIBUS port executes the command, it returns an acknowledge (XACK) to the processor, which allows the I/O instruction to complete execution and releases control of the MULTIBUS. If either the MULTIBUS is busy when the processor tries to execute a normal I/O instruction or if the port is slow in returning its acknowledge signal, wait states are automatically inserted to extend the I/O instruction. The job processor uses only word I/O instructions when accessing MULTIBUS ports. If the port only supports byte operations, the high order byte is undefined when the port is read and ignored when the port is written.

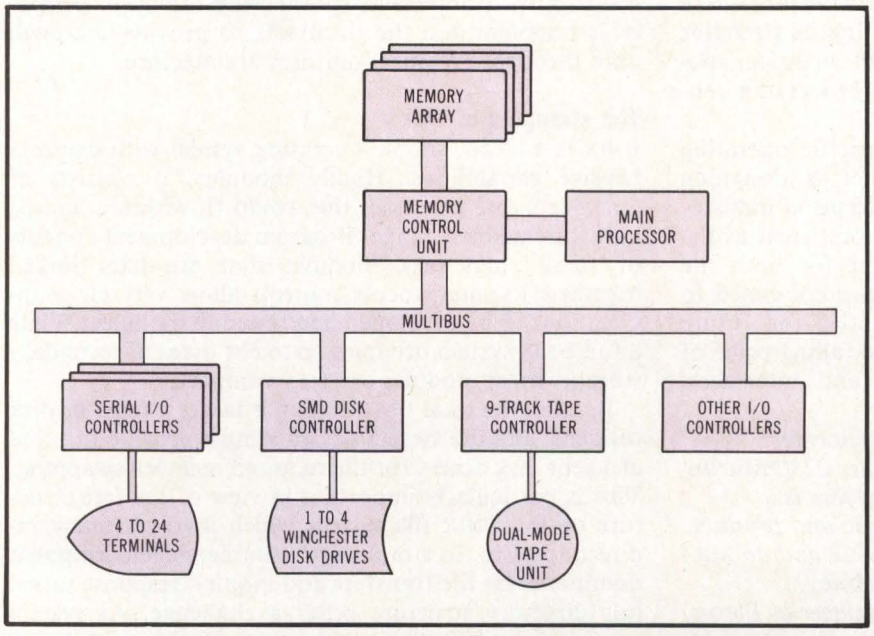

Fig 1 Plexus P/40 block diagram. System relies on IEEE 796 bus (MULTIBUS) to communicate with its I/O device controllers. For greater speed, system memory is on separate bus.

Job processor's SIO is used for system control, diagnostic, and housekeeping functions. Because efficient diagnostics are critical in a multiprocessor design, a special serial I/O port is used to connect a diagnostic terminal directly to the processor. A universal asynchronous receiver/transmitter (UART) with a programmable baud rate generator is used to create a single vectored interrupt to the job processor. To aid in software debugging, the job processor has a trace mode capability that generates a vectored interrupt whenever the processor fetches the first word of an instruction and is in normal mode. This feature can be used to single-step through programs.

In addition, the job processor has access to a counter/timer circuit (CTC). In normal operation, the CTC is used to divide the system clock down to 50 Hz to provide the job processor with realtime clock interrupts. A battery powered clock/calendar gives the job processor access to the time of day in hours, minutes, and seconds and will operate up to forty days without system power. This feature is critical as many UNIX utilities require the time of day to be correct. Additionally, the clock/calendar allows the system to go from power-up to a fully operational state without operator intervention. Power-up sequences are automatic and are initiated with an on/off switch.

To augment its capabilities, the job processor also contains an arithmetic processing unit (APU), which performs floating point addition, subtraction, multiplication, and division on 32- and 64-bit numbers in American National Standards Institute (ANSI) standard format. Software options are being explored for even greater speeds.

The job processor uses two Intel 8259A programmable interrupt controllers (PICs), each with eight interrupt request lines. The PICs are cascaded together in a master/slave configuration to give a total of 15 interrupt lines with distinct vectors. This may seem extremely low for a processor of this size, but only a few interrupts per second actually reach the job processor—the intelligent communications processors handle the rest locally while the DMA techniques further reduce job processor involvement in interrupt handling. When considering the auxiliary processors and the MULTIBUS interrupts, the system has hundreds of distinct interrupt vectors.

Here, a small design problem had to be overcome. Because the processor requires that all vectors be on an even boundary (bit 0 must be 0), the PICs cannot be connected in a normal manner. Instead, data line 0 from the PIC is connected to data line 1 of the job processor bus, and so on with PIC line 7 to processor line 8. Processor data line 0 is pulled low, which means that data must be shifted one line to the right when the PIC is read or written to.

Memory control unit

Efficient and proven memory management and error handling and correction are critical in a multiprocessor system, particularly since UNIX is rather intolerant of errors. To break a large amount of memory into small segments that enable the processors to find information faster, and to include error checking and correction is the system's concept. The memory control unit (MCU) takes requests for main memory, from both the job processor and the peripheral processors, and generates the proper signals to the memory array boards. Controlled by the job processor with SIO instructions, the MCU contains address mapping circuitry (Fig 2), error correction logic, and the dynamic random access memory (RAM) controller.

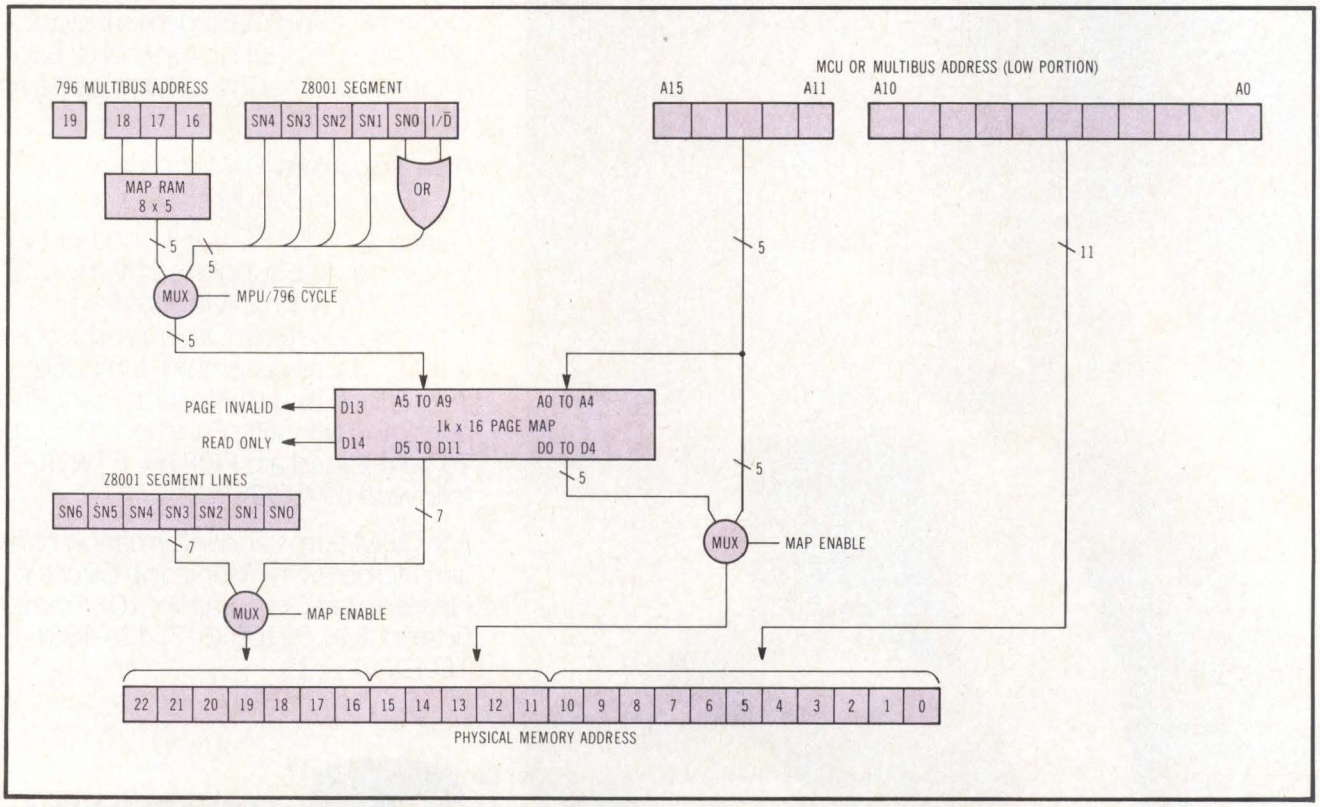

Fig 2 Memory control unit translates MCU and MULTIBUS logical data addresses into physical memory addresses.

In normal operation the MCU takes 21 address lines from the job processor or the MULTIBUS, then maps them into a 23-bit (8M-byte) physical address space. When the job processor accesses memory, five segment address lines and the 16 address lines form the address to the MCU. Segment line 0 is ORed with the instruction/data (I/D) line; this implies that the job processor can access data memory in segments 0 to 31, but instruction references can be made only to odd numbered segments (I/D bit = 1). Thus, running in segmented mode, the operating system has access to all memory segments, whereas user programs, which run in unsegmented mode, have access to one code and one data space.

Logically, the five highest MCU address lines select 1 of 32 map sets (Fig 3). A map contains from 1 to 32 2k-byte pages of memory. A typical process or program requires one map set for instructions and one for data. Minimum memory for a single program is 4k bytes, and the maximum is 128k bytes. Therefore, maps for up to 16 processes can be simultaneously resident in the mapping RAM, and if a process is resident in memory without a map slot, only one map slot for that process must be changed for the process to run. In other words, frequently called processes can be stored in memory without occupying map slots and be run very quickly without swapping the process itself in from secondary storage.

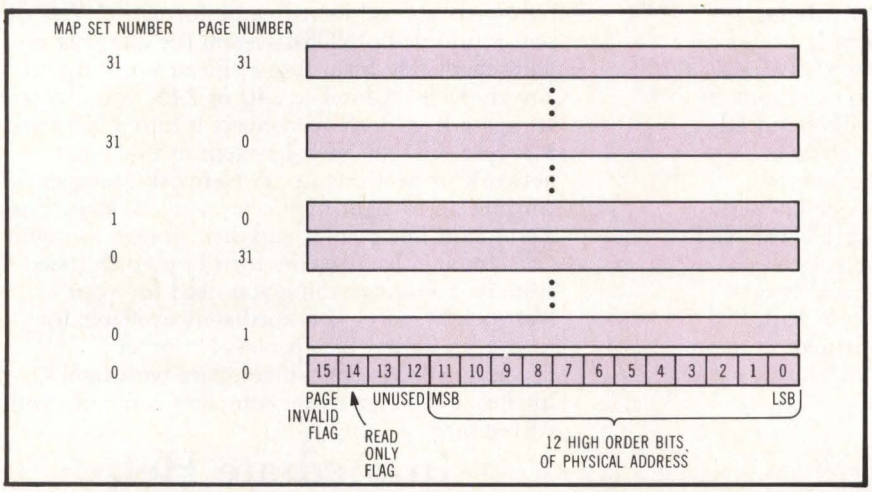

Fig 3 Page map RAM consists of 1024 16-bit entries. Map contains information used to translate between logical and physical memory addresses.

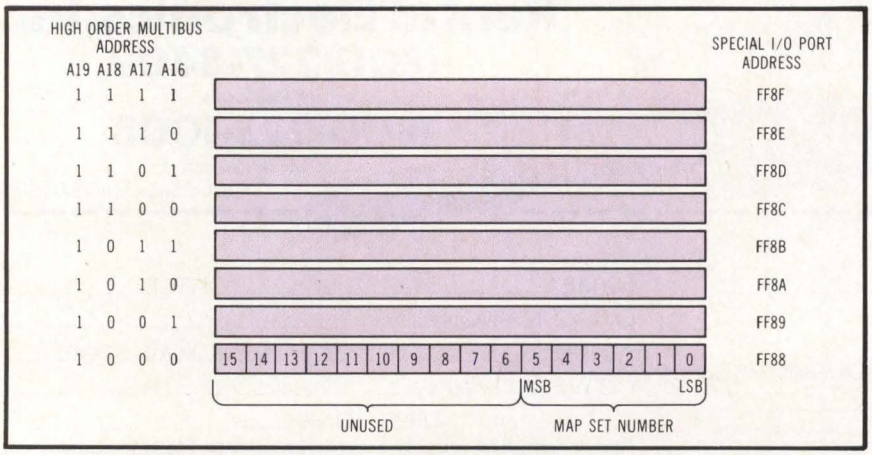

Fig 4 796 bus map contains eight 16-bit entries. Map translates data addresses on MULTIBUS into system memory addresses.

In addition to the page map, a MULTIBUS map RAM allows processors on the MULTIBUS to access memory through the MCU (Fig 4). This RAM can store eight map set numbers, allowing processors to simultaneously access eight of the 32 map sets.

Moreover, the MCU is responsible for error handling, checking each reference to memory for consistency with the attribute bits assigned to each page. Each page can be assigned a read only or invalid status. When a violation occurs, the MCU latches the address and map set number that caused the violation; the status information; and, for an error generated by the job processor, the address of the first word of the last instruction fetched before the error. The MCU determines whether the job processor or a peripheral processor generated an illegal reference and takes appropriate actions. If the job processor caused the violation, the MCU asserts the job processor segment trap line. For a MULTIBUS processor error, the MCU enables the job processor to take control of the MULTIBUS and notify the processor that an error occurred before the processor tries to transfer another block.

MCU arbiter

The MCU arbitrates between MULTIBUS processor and job processor requests for main memory. When a device wants to request a memory cycle, it asserts its memory request line to the MCU arbiter, which, in turn, asserts the memory grant line to the winning device. The grant line enables the device's address, data, and control lines, using a first come, first serve protocol when memory is not in use and interleaving requests when memory is active. For example, if the memory is inactive and the job processor requests memory, it immediately receives a grant and the cycle begins. If a MULTIBUS processor requests a cycle during the job processor cycle, it will not receive a grant until the job processor cycle is complete. Cycles are interleaved when the MULTIBUS port and the job processor issue a continuous stream of requests.

MULTIBUS memory port

The MULTIBUS memory port maps the upper half of the 1M-byte MULTIBUS memory address space through the MCU. When the port detects a MULTIBUS memory request with the high address bit (A19) active, the port requests control of the MCU through its arbiter. Then the port initiates the memory cycle and generates the acknowledge signals (AACK and XACK) when the cycle is complete. Generated a fixed time before the cycle finishes, the AACK is used by the peripheral processors to increase throughput. The port supports both word and byte memory accesses.

Certain measures are taken when the job processor and the MULTIBUS processors share buffers in memory, because addressing conventions used by the job processor and the MULTIBUS byte are different. The job processor expects the high order byte in a word to be at an even address, while the MULTIBUS uses the opposite convention. Data can be accessed from the MULTIBUS or by the job processor without further consideration, provided they use independent sections of memory. When main memory is used as a common area for passing data between MULTIBUS devices and the job processor, data are stored in memory. (See Table, "Memory addressing.")

Memory addressing

	Even address	Odd address
MULTIBUS	RAM low bank data (0 to 7)	RAM high bank data (8 to F)
Job Processor	RAM high bank data (8 to F)	RAM low bank data (0 to 7)

When the job processor writes a byte into a location using an even address (eg, 1000H), it is placed in the high bank. If data from the same location are read from the MULTIBUS, low bank data are actually read. To read the correct byte, the next address location (1001H) is read from the MULTIBUS.

Dynamic RAM array. The dynamic RAM boards contain 128k or 512k 22-bit data words, each logically made up of a 16-bit data word and a 6-bit error correction field (modified Hamming code). The error correction scheme detects and corrects single-bit errors and detects double-bit errors. In addition, the arrays contain basic support functions such as address decoders, buffers, and row address strobe/column address strobe multiplexers. All control, address, and data lines come from the MCU over a dedicated bus, taking power and ground from the MULTIBUS connector.

Intelligent communications processor

A powerful, intelligent device, the ICP contains a processor, memory, eight serial ports, one parallel port, and DMA channels associated with each port. Typically, it handles all low level buffering and processing necessary to support a variety of terminals, modems, and printers. The eight RS-232 serial ports are implemented with universal synchronous/asynchronous receiver/transmitters (USARTs). Asynchronous baud rates from 50 to 38.4k are selected for each channel by programming a CTC counter/timer. Character length, parity, and the number of stop bits are also programmable. Each serial port can support asynchronous, bisynchronous, and HDLC/SDLC protocols.

The ICP communicates with the job processor using command and status blocks located in main memory. Data transfers between the ICP and main memory are controlled by the ICP's processor, thus further freeing the job processor of any time-critical interrupt handling.

The ICP uses a Z8000-series 16-bit microprocessor. Always operated in the system mode, there is no separation of instruction and data or special and normal I/O spaces. The ICP contains 16k bytes of programmable read only memory and 32k bytes of RAM with parity arranged in two 16k x 9 banks. The processor can directly address 64k bytes of memory, with the lower 48k resident on the ICP and the upper 16k mapped into main memory via the MULTIBUS and the system MCU. Parity generation and checking are enabled or disabled by the processor.

DMA techniques. The ICP uses a combination of software and hardware to simulate a DMA. Data are moved to and from main memory via block move instructions executed by the ICP processor. Since the upper 16k bytes of the ICP's address space are mapped into main memory, the ICP processor executes a block move from the middle 32k bytes of memory to the upper 16k bytes when it wants to move data into main memory. It programs a CTC channel to provide a pulse every n seconds, nominally 1 to 2 ms but tunable, depending on the application. This pulse goes to the MULTIBUS arbiter and causes it to request the bus. When control is obtained the arbiter sends an interrupt to the ICP processor, which initiates a block move of approximately 32 words. The size of the block is also tunable depending on the application. After the move is complete the processor signals the arbiter to release the bus. When no blocks remain in the DMA queue, the ICP processor disables the CTC channel so that no more interrupts are generated.

The nature of the design is such...that a job processor can be chosen for its efficiency at particular tasks.

This technique moves 64k characters/s, or the equivalent of eight terminals running in burst mode, and uses less than 10% of processor bandwidth. It allows a DMA task to run in the background without monopolizing either the MULTIBUS or local bus, and it leaves a great deal of the ICP processing capabilities free for tasks such as executing segments of UNIX, table conversions for data communications protocols, terminal handling programs, and other local processing tasks.

Port handling is also done with DMA channels within each ICP, which allows all eight ports to output simultaneously at a 19.2k-baud maximum asynchronous rate. The DMA function is provided by three Intel 8237 DMA chips, which are programmed by the processor to transfer one byte, then release the local bus to the processor. This ensures that the ICP processor will get at least every other memory cycle.

Based on the Z80A PIO, the parallel port is also DMA supported. Control logic associated with the port generates all control and handshake signals necessary for the DMA to transfer data to a line printer without processor intervention. The port sends a vectored interrupt to the processor if the printer asserts the fault line.

MULTIBUS arbiter and interface. Using the parallel arbitration technique, the MULTIBUS arbiter on the ICP requests the MULTIBUS every time a processor controlled timer circuit reaches a 0 count. When control of the MULTIBUS is obtained, the arbiter locks BUSY and interrupts the ICP processor, informing the processor that it can begin a transfer to main memory. The processor

resets the arbiter and releases the MULTIBUS when the transfer is complete.

A master that supports only 16-bit memory read and write operations, the MULTIBUS interface allows the processor to assert one of the eight MULTIBUS interrupt lines (INT0 to INT7). It initiates a MULTIBUS memory cycle when the processor performs a memory operation in the upper 16k bytes of its address space, which are mapped into the system's main memory. Addresses are formed by linking a 6-bit port with the 14 least significant processor address lines, forming the 20-bit MULTIBUS address. Nine peripheral chips in a daisy chain generate vectored interrupts. A translation circuit makes the interrupt vector produced by the peripheral chips compatible with the ICP processor.

Disk and tape controllers

The disk controller is an intelligent device that supports up to four drives. High performance SMD disks were chosen for the system because a wide range is available from many vendors (20M bytes to 1G byte), allowing systems integrators to precisely match disk capacity with applications, and also, because of its size and flexible file system, UNIX requires frequent disk accesses. As the UNIX file system is tree structured and file space is not preallocated, most accesses consist of several indirect accesses through directories. Thus, a very fast disk and disk controller supporting high transfer rates are desirable for UNIX based minicomputers. DMA channels provide the 3-MHz bandwidth for these rates.

An intelligent controller helps maintain high rates by offloading the main processor. The job processor simply sends the disk addresses and lengths of the blocks to be transferred; the controller's intelligence allows it to send a very large number of blocks, provided the MULTIBUS is not busy.

Built around an Intel 8089 16-bit I/O processor with 10k bytes of memory, the controller is capable of performing multiple-sector operations that span tracks, as well as automatic error detection and correction. It uses a 32-bit fire code to detect 22-bit burst errors and correct 11-bit errors in a way that makes its operation transparent to the job processor. Like the ICP, the controller receives commands from the job processor by reading a control block in main memory. But unlike the ICP, which controls the MULTIBUS for transfers of about 32 words, the disk controller can be interrupted after each word by higher priority MULTIBUS interrupts. While preventing data transfers from tying up the MULTIBUS, it allows the full 1.2M-byte transfer rate of the SMD disks.

The technical design goal... was to eliminate traditional bottlenecks on data transfer and interrupt handling.

The magnetic tape controller supports up to four industry standard 9-track, 0.5" tape drives. These are dual-mode units to support streaming as well as start/stop operation. Like the disk controller, it uses an 8089 I/O processor and has its own local memory. Since it maintains record buffers in main system memory, there is no limitation on tape block length.

Conclusion

While the job processor can be expanded from 256k bytes to 4M bytes and beyond, up to eight job processors, ICPs, and mass storage controllers can be configured in a single system. This results in a mid-range minicomputer with expansion to 24 terminals and 580M bytes of disk storage. Another growth path is through networking—not by adding even more power to the processor, but by adding another processor. This multiprocessor approach of using industry standards makes networking extremely easy. Since the ICP is a general purpose computer in its own right and can be downloaded, it can be programmed to handle Ethernet, X.25, or other network protocols. Its overall performance is roughly that of a PDP-11/70 or the high end of the 16-bit Eclipse systems, at approximately one-third to one-half the cost.

Please rate the value of this article to you by circling the appropriate number in the "Editorial Score Box" on the Inquiry Card.

High 707 *Average 708* *Low 709*