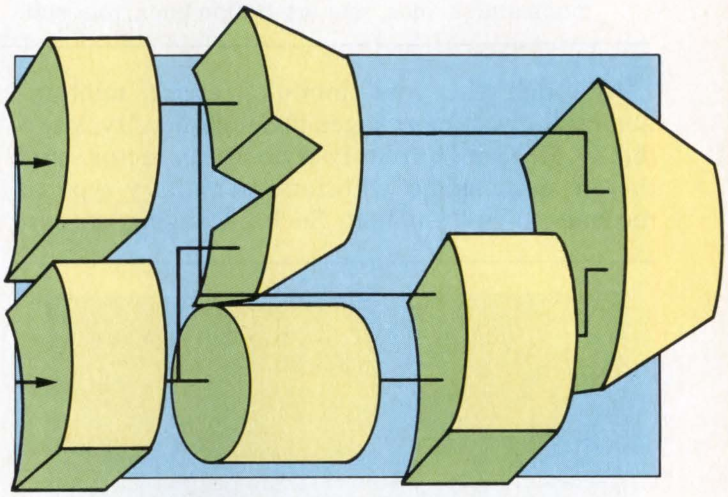
Single-chip processor runs Lisp environments

With the new dedicated Lisp chip, intelligent Lisp/AI workstations can be made much smaller, easier to use, and more affordable as either applications development tools or delivery vehicles.

A dedicated single-chip Lisp processor has been designed by Texas Instruments to replace multiple boards of MSI- and SSI-based parts. With an architecture that's optimized for Lisp/artificial intelligence applications, the new Lisp chip is one of the first of a generation of chips that will use the combination of VLSI processes and procedures that TI calls Megachip technologies.

The chip represents what can be accomplished with VLSI technology. Using 1.25-micron CMOS design rules, designers fabricated more than 553,000 transistors on a 1-cm^2 chip that implements 60 percent of the functionality of the original, full-size processor of the Explorer AI workstation.

Providing a powerful Lisp environment for the rapid development of AI software, the Explorer was one of the first 32-bit workstations suitable for the office environment. Explorer's genesis began with the introduction of the first Lisp machine, the CADR, developed by the Massachusetts Institute of Technology (Cambridge, MA) in 1978 under sponsorship of the Defense Advanced Research Projects Agency (DARPA). As a result of the new Lisp chip and Megachip technologies, Explorer performance will increase by about five times and its IC count will be significantly reduced.



The Lisp chip will be the basis for a Compact Lisp Machine (CLM) that will be used in defense systems. Also DARPA-sponsored, the CLM will let the powers of Lisp and AI be put in a package small enough to fit in fighter aircraft cockpits. The CLM consists of four high-density circuit modules that can be mixed and matched. Intended for system-development applications, the modules comprise a processor, a cache, a mapper, a memory and an outside bus interface.

The four modules interconnect via the 32-bit, MIT-developed Nubus. The Explorer, the CLM and several other commercial computer products use the Nubus because of its processor independence, high speed, and ability to handle almost any combination of disparate processors in a multiprocessor environment.

Gene Matthews, Robert Hewes and Steve Krueger

Matthews is director of the Symbolic Computing Laboratory of the Computer Science Center at Texas Instruments (Dallas, TX). Krueger is a senior member of that laboratory's technical staff, and Hewes is associate director of the company's VLSI Design Laboratory at the Semiconductor Process and Design Center.

CLM takes the Nubus

The Nubus, operating with a 10-MHz clock to achieve a maximum bandwidth of 37.5 Mbytes/s, is a synchronous, high-performance, processor-independent, 32-bit bus designed to support a multiprocessing environment such as the Compact Lisp Machine (CLM). It's optimized for block transfers of 2, 4, 6, 8 or 16 words, but it also supports word, halfword and byte transfers. Reading from or writing to its 4-Gbyte physical address space are the sole operations necessary and require only a simple interface protocol.

The Nubus' arbitration protocol is a combination of priority encoding and round-robin techniques. When the bus is lightly loaded, this combination provides fast access for high-priority devices. When heavily loaded, however, the combination provides equal access. Nubus also supports bus locking for indivisible operations.

The bus requires 49 signals as well as power and ground. It comprises 32 multiplexed address/data lines (with parity optional), two multiplexed mode/status lines, four arbitration lines, four slot-identification lines and seven control lines for clock, reset and so forth.

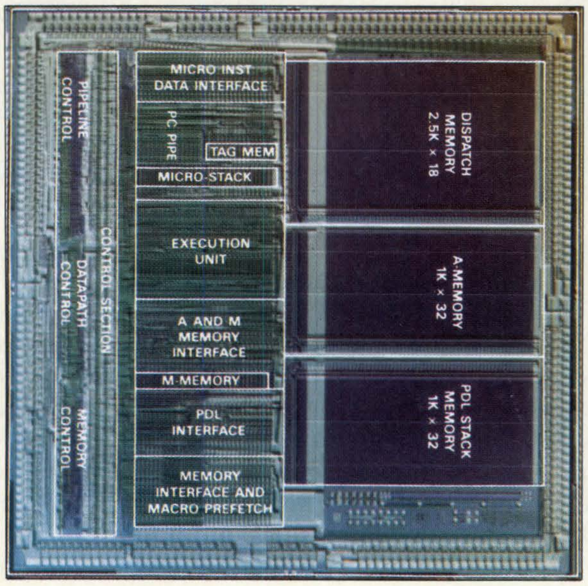
Each bus module receives its slot identification from the backplane, identifying it by its physical location. Because the ID lines are part of the address decoding, there's no need to set switches or jumpers to identify the modules.

A fixed area in the address space of each slot defines a configuration ROM on each bus module. By reading those ROMs, the bus master can easily determine the configuration of the entire system.

In Nubus, a write operation called an event can generate an interrupt. Address-sensitive hardware detects those events. Because any event can be signaled by software or hardware, this simple protocol of events provides a flexible and powerful symmetry between hardware and software.

In addition to being used in Texas Instruments' CLM, Explorer and various business system product families, the Nubus is also used in the new Macintosh II, recently introduced by Apple Computer (Cupertino, CA).

Although the four modules aren't military-qualified, they're packaged in a high-density, size 6 (6.58×5.875-in.) military-style form factor, and they're designed to withstand a military type of mechanical environment. Such packaging lets system designers fit a CLM system into a volume as small as a shoe box. This size contrasts sharply with the several cubic feet that today's Lisp systems occupy, a fact that makes the current systems unsuitable for use as Lisp/AI applications delivery vehicles in aircraft or tanks or even for use in many commercial AI applications.



On the new dedicated Lisp processor, some 553,000 transistors are implemented in 1.25-micron CMOS design rules on the new dedicated Lisp processor. These transistors implement about 60 percent of the processor-chip circuits of a high-end Lisp workstation.

In each module, two circuit boards are bonded to a metal thermal core. The circuit boards are ceramic with copper thick-film interconnection on a cofired ceramic substrate. A surface-mounted high-density connector at one end lets the board be plugged into the bus socket. Wrap-around conductors provide connections between the boards. Optional metal covers protect the components.

The new Lisp chip is at the heart of the processor board of these four modules. Certain system designs—a plug-in board for a personal computer, for example—could use the processor board in conjunction with other boards that aren't designed for the CLM.

TI is developing system software and development tools for the CLM without government funding. The software is based on the industry-standard Common Lisp environment with extensions for Lisp machines. Object code-compatible with Explorer, the software and development tools include a set of CLM diagnostics and an Explorer-compatible interactive software development/debugging system.

Since microcode executes many of the CLM's instructions, the CLM processor module contains a write-controllable microcode-store 16k×64-bit static RAM for the Lisp processor itself. Packaged in a custom 264-pin grid array, the processor is memory-intensive and contains over 100 kbits of RAM. A 45-kbit dispatch memory contains multiway branch tables. One 32-kbit memory acts as a top-of-stack cache, and another 32-kbit memory provides a general-purpose scratch pad.

The processor module also contains a clock generator and an interrupt interface for the Lisp processor chip. Local software-controlled timers connect to the processor via its 32-bit virtual memory address and data buses. These buses also connect to booting and configuration ROMs and to a Nubus interface circuit. Because the Lisp chip is new, a debugging interface provides access for testing and development of the system software and microcode on existing Explorer machines and provides the system designer with a productive development environment.

Maximizing memory bandwidth

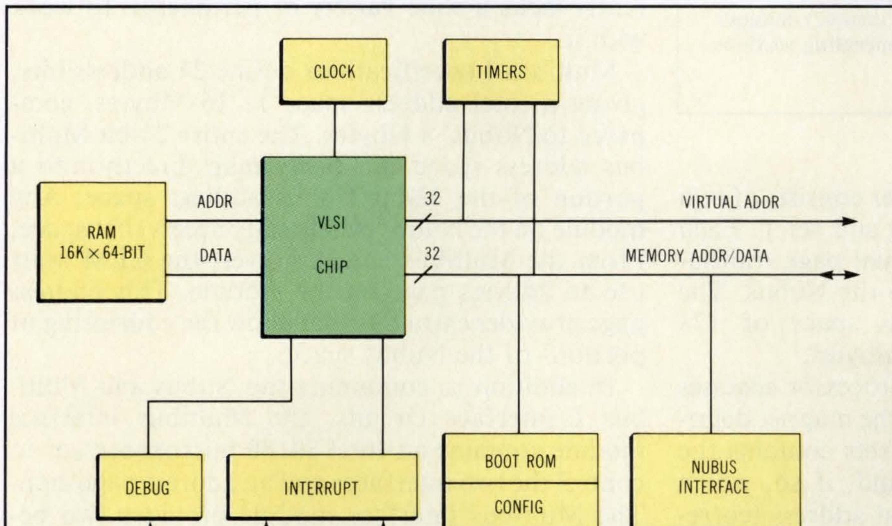
The CLM main memory is contained on one or more memory modules. Each module has a 2-Mbyte capacity and consists of an array of 72 dynamic RAMs, each configured as 256k × 1-bit. Each of the 4 bytes that make up the system's 32-bit word has a separate parity bit generated and checked by parity circuitry. The status-logic circuitry records errors detected by the parity circuitry. The control logic implements the address multiplexing, timing and control signals for accessing and refreshing the DRAMs.

The Nubus interface of the memory module supports block as well as byte and half-word transfers. To speed updates to system cache memory, the transfers of commonly used four-word blocks are optimized.

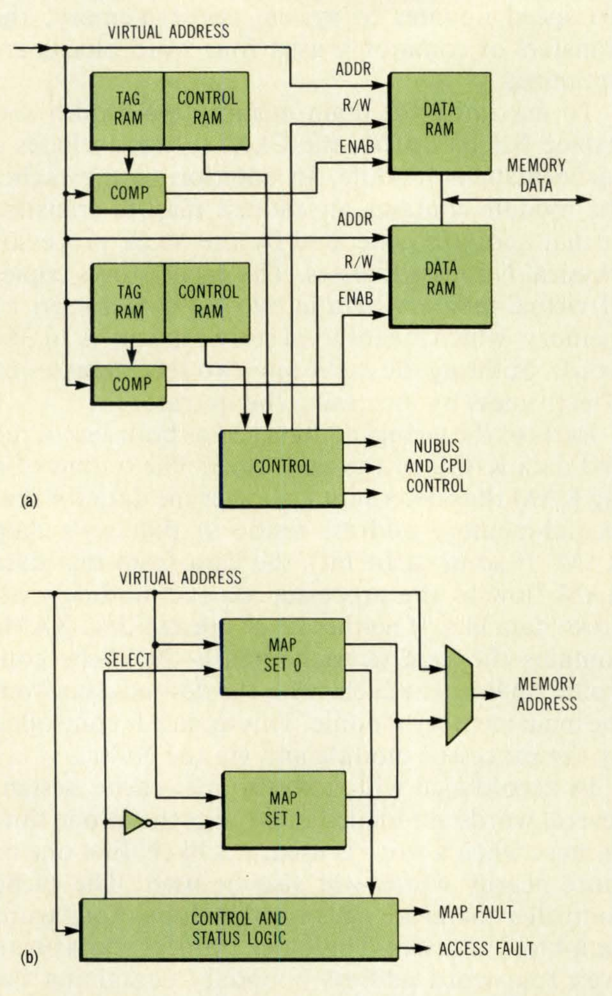
To maximize the main memory bandwidth and reduce Nubus traffic, the CLM system includes a cache/mapper module. In addition to the cache, the module contains an address map to translate virtual addreses generated by the VLSI processor physical Nubus addresses. The cache stores copies of virtual-memory data in two 8,192-word sets of memory, which means total store capacity is 16,384 words. Splitting the cache into two sets increases its effectiveness by increasing the hit rate.

Parts of the virtual address access both sets of tag and data RAMs at the same time. The output of a tag RAM thus determines whether the data for that virtual-memory address reside in that set's data RAM. If so (a cache hit), the data from that data RAM flow to the processor via the module's address/data bus. If neither set of tag and data RAMs contains the data (a cache miss), the cache controller updates the cache with the desired data from the main memory module. This update is controlled by the processor module and via the Nubus.

In accordance with conventional cache design, several words are loaded into the cache at one time because when a word is used, it's likely that one or more nearby words will also be used. The cache controller therefore calls for the Nubus four-word data-block transfer. This block transfer starts on an even four-word address boundary containing the missed word and adjacent words.



In addition to including the new VLSI Lisp processor chip, the Compact Lisp Machine processor module has an off-chip 16k × 64-bit write-store RAM for processor microcode. Also included off-chip are the clock, timers and other circuitry to operate and interface to the processor. System designers can use the chip by itself or use the processor module.



To maximize memory bandwidth, the mapper/cache module (a) contains a data cache divided into two parallel sections. Like the cache, the memory mapper (b) also is divided into two parallel operating sections to increase the hit rate.

Mapping the memory

Like the cache, the memory mapper consists of two performance-speeding sets—set 0 and set 1. Each set can store 2,048 256-word virtual page translations from the CLM processor to the Nubus. The processor has a virtual address space of 128 Mbytes; Nubus has a space of 4 Gbytes.

Also like the cache, when the processor accesses a virtual address, control logic in the mapper determines whether either of the two sets contains the translation to the desired page and, if so, which one. A map hit sends the physical address (corresponding to the virtual address) via the address/data bus to the Nubus interface residing on the processor module.

Address translation and cache hit/miss determination proceed in parallel. If the cache declares a miss, the Nubus' physical address of the desired data in the CLM main memory could be immediately available. If the cache declares a hit, the processor doesn't need the address.

After a cache miss, it's possible that neither set of the map contains the translated address of the desired page. Such a map miss requires that the processor, under microcode control, determine the address translation and enter it into one set of the map. Addresses don't always need to be translated. The mapper can also use the physical addresses directly without translation.

The mapper contains a separate address-space status map. It divides the 128-Mbyte virtual address space capability of the processor into 4,096 regions of 32 kbytes each. Eight status bits for each region and additional status logic provide hardware support for the CLM garbage-collection function, separating some of that function from the virtual address translation.

Interfacing to the world

System designers who opt for the CLM or the Lisp chip will need to connect their designs to the outside world. Because the CLM designers expect that initial applications of the modules will be designed to widely varying requirements, there's a need for an interface—the Multibus interface module to the popular Multibus I commercial bus.

Many different types of I/O, graphics and other peripheral products are available for this bus. The Nubus, which is being considered as an IEEE standard, is ideal for supporting a Lisp system but currently lacks a wide variety of peripherals to work with it.

Multibus I specifications define 24 address bits, giving a total address space of 16 Mbytes, compared to Nubus' 4 Gbytes. The entire 24-bit Multibus address space can easily map directly into a portion of the 32-bit Nubus address space. Any module on the Nubus can directly access that space. From the Multibus end, however, the CLM must use an address page on the module. This address page provides extra bits that allow the addressing of portions of the Nubus space.

In addition to containing the Nubus and Multibus I interface circuits, the Multibus interface module contains an Intel 80188 microprocessor to control the two interfaces and an address page map. The Multibus interface module provides two between-bus transfer modes—immediate and wait.

The Lisp processor chip implements a microcoded architecture compatible with earlier Lisp machines. Because the chip's active area didn't contain enough room for the 32k × 64-bit writable con-

Symbolic languages fill AI needs

Although all programming languages use symbols, conventional languages such as Basic, Pascal and C are intended primarily for numerical operations; numbers are the symbols they manipulate most readily. In contrast, artificial intelligence applications need to actually manipulate abstract, nonnumeric symbols.

To meet this need, purely symbolic languages such as Lisp and Prolog and their dialects were developed. To create a standard, a new dialect called Common Lisp has been developed and widely adopted. Incorporating the richness and power of its many predecessors, Common Lisp provides true portability for AI applications and thus has found strong commercial and military support.

Lisp differs from most programming languages in three important ways. First, Lisp has flexible data structuring in lists and structures. Any element of a list or structure may be any type of Lisp data. Lisp data may be structured naturally to match complex applications.

Second, Lisp's syntax is relatively simple. It has just two types of program elements—atoms and expressions. Atoms are variables, constants and functions; expressions consist of a function atom and its operands, which may be either kind of program element. Expressions are represented as lists, so it's easy for programs to construct or modify expressions.

Lisp functions can be interpreted from this list form or compiled into machine language, and the two types of functions can be freely intermixed. Because Lisp functions can run in an interpreted mode and because of the uniform list structure of the data, the programmer can interact closely with the code under development. This flexibility also allows extensive and speedy debugging.

The basic unit of data is the fixed-size cell containing three fields—a data type, a data value and CDR code. When data are too big to fit in the data value field or are of variable size, the cell's data type indicates that its data value is an address or pointer to where the data are located in a block of storage located elsewhere.

The Explorer, the Compact Lisp Machine (CLM) and the Lisp chip support a wide variety of data types, including many numeric data types. In addition to these "programmer-visible" types, several internal types are used by Lisp. These internal types serve the operation of both the Explorer and the CLM.

Because the data carries its type with it, many of Lisp's basic operations are generic; they apply to multiple data types. For instance, one ADD operation works for all number representations and accepts floating-point and even complex numbers as well as integers. Attempts to operate on invalid data types, such as adding a "list" to an integer, however, cause the operation to abort with an error message. Checking for invalid data types is done both at compile and run time.

In manipulating lists, Lisp typically uses many large and small chunks of memory for short periods of time. This activity, unfortunately, can quickly fragment and exhaust the memory space. Techniques for periodically reclaiming the chunks of memory no longer used are collectively called garbage collection. Both the Explorer and the CLM have built-in garbage collection transparent to the user. This garbage collection works while the computer is used for other purposes.

The key to speedy garbage collection is the typed data concept. Typed data let the garbage collector circuitry recognize all pointers in the data. Data addressed by some nongarbage pointer aren't garbage, but data that aren't addressable are garbage.

The garbage collector rearranges the blocks of data to remove fragmentation and restore the memory to a linear, contiguous order. This automatic memory management requires overhead. For maximum system throughput, garbage collection is accomplished in hardware wherever possible. The Lisp processor enjoys the services of hardware-assisted garbage collection.

trol store, the microcode resides off-chip.

The Lisp chip supports many Lisp typed-data operations, including multiway branching for data-type checking; additional data-type checking and result tagging; and bit-field depositing or extracting for manipulating data objects internally. It also provides hardware support for garbage collection.

The reliability and quality of such a chip are major considerations. The processor is based on a 100 percent scan design, for example, and includes a signature-analyzer feature and a 16-kbit microcode ROM for self-test. But just 2.5 percent of the chip is devoted to these capabilities, which are essential to ensuring a testable Lisp system design. Futhermore, the processor was designed for zero static-power consumption. Faults involving dominance are detected by monitoring the processor power supply current.

On-chip memory dominates

Since Lisp is memory-intensive, on-chip memory dominates the chip's area. There are three larger memory units—two 1k × 32-bit and one 2.5k × 18-bit RAMs—and three smaller ones. A 256 × 64-bit microcode ROM provides the code for both booting and self-test. The chip's data paths take most of the remaining chip area, except for a small part for the control logic function.

Individual refresh counters and a master refresh timer support the RAMs. When the RAMs aren't engaged in other operations, the system refreshes them. If the refresh period expires before all RAMs can be refreshed, no more instructions are issued, so refreshing can be completed. Access time of the large memories is 17 ns; cycle time is 25 ns. This type of performance enables the processor's fast cycle time.

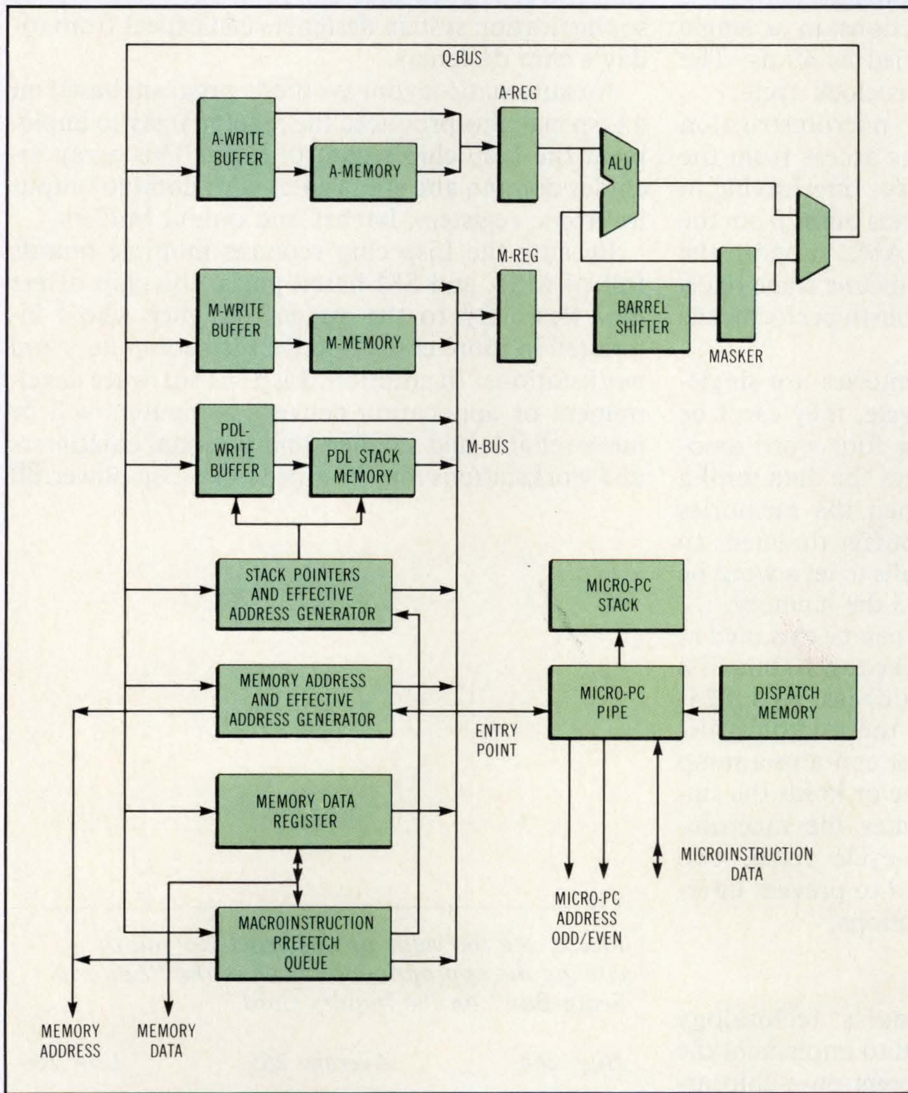
One of the 1k × 32-bit RAMs (known as the A memory), a microcode scratch pad, acts as a single input source for the chip's arithmetic logic unit. The other 1k × 32-bit RAM, a stack memory (called the PDL), and a small 64 × 32-bit scratch pad (M memory) provide the other input. The PDL stores the top 1,024 words of stack on-chip. This design eliminates memory references that would be required to access or store these variables during function calls.

A 32-bit masker/barrel-shifter adds a great deal of flexibility and performance to the ALU's data-processing ability. With its modified Booth's algorithm, the ALU can compute a 2 × 32-bit multiply with one microinstruction. A normalization counter counts leading 0s or 1s to support floating-point calculations. A novel carry-select, lookahead circuit in the ALU lets the unit quickly compute the ALU = 0 state required for branching.

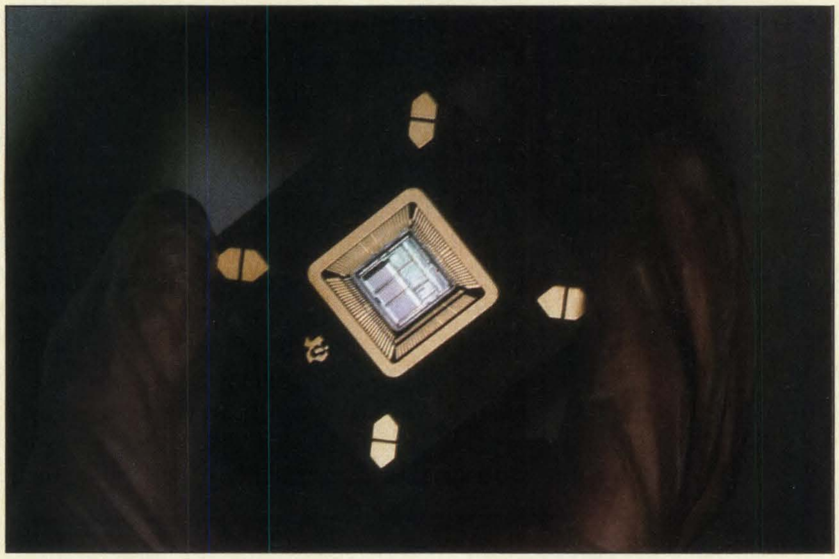
The remaining large memory, the 2.5k × 18-bit dispatch memory, contains branching tables. These tables let up to 128-way microinstruction branches run as quickly as a simple jump. A macroinstruction prefetch unit prefetches up to four 16-bit macroinstructions, and statically follows the branches. This action speeds system throughput.

Other unusual features

The Lisp chip has many other unusual architectural features. The chip can pipeline to implement one microinstruction per system clock. High through-



On the Lisp processor, the masker and barrel shifter add flexibility to and increase performance of the ALU. The A memory acts as the ALU input source, and the PDL and M memories provide the other input. The other large memory, the dispatch memory, has branching tables that let up to 128-way microinstruction branches run as quickly as one jump.


A full computer system on a chip, the Lisp processor is the basis for the processor board in the Compact Lisp Machine. Occupying 1 cm^2, it also could form the heart of other Lisp systems, such as plug-in boards for conventional workstations and personal computers.

put is the goal here. A four-stage pipeline allows the execution of most microinstructions in a single clock cycle conservatively specified as 40 ns. The design goal for the chip is a 25-ns clock cycle.

Two pipeline stages for the microinstruction fetch provide enough time for its access from the off-chip microinstruction memory. Interleaving in that memory and two sets of address buses from the processor let available static RAMs support the very high clock rate. The third pipeline stage reads the internal memories while the fourth performs the actual computations.

Since the A, M and PDL memories are single-ported and can be read every cycle, they can't be written into directly. Instead, a four-word associative memory receives and stores the data until a time slot becomes available when the memories aren't being read. If this write buffer threatens to overflow, the instruction flow stalls to let a word be read out of buffer and written to the memory.

Simple macroinstructions also can be executed at the rate of one per clock cycle. The top 10 bits of a macroinstruction directly address a 1,024-instruction entry-point table in the microinstruction memory. That location either contains a jump to the required microcode routine or holds the single microinstruction that emulates the macroinstruction. Because of the two-cycle fetch, two copies of this table are interleaved to prevent interference between adjacent instructions.

Lisp chip offers new flexibility

The Lisp chip's internal process technology features and the design tools used to implement the chip are as significant as its system-on-a-chip architecture. The features and tools show the level of sophistication system designers can expect from today's chip designers.

An automatic layout synthesis program based on a Lisp machine produced the regular array to implement the Lisp chip's control logic. This array includes domino and static gates with domino output inverters, registers, latches and output buffers.

Because the Lisp chip replaces multiple boards full of MSI- and SSI-based parts, the chip offers new flexibility to the system designer who's interested in more cost-effective Lisp computers and workstations. In addition, Lisp/AI software development or application delivery computers will be more reliable and smaller, and personal computers and workstations may be able to use Lisp power. **CD**

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

High 264 *Average 265* *Low 266*