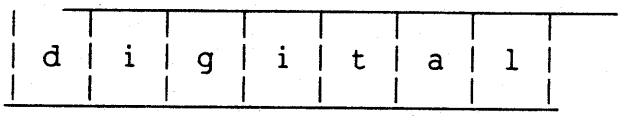
The Firefly Workstation

Charles Thacker



Systems Research Center
130 Lytton Avenue
Palo Alto, Ca., 94301

1. Introduction

Firefly is a personal multiprocessor workstation being designed at the DEC Systems Research Center (SRC). The system consists of five Motorola 68010 processors, each with an associated sixteen kilobyte cache and address translation unit, eight megabytes of main storage, and controllers for disks, bitmap display, Ethernet, and other miscellaneous input-output devices. The Firefly is packaged in a floor pedestal cabinet 8" wide by 24" high by 28" deep, and is intended for use in an ordinary office environment.

Firefly is being built for two reasons: First, it provides a significant amount of computing power to a single user in a compact and affordable package. Most SRC staff members are engaged in research in the areas of operating systems, user interfaces, programming languages and hardware design; these activities require large amounts of computing. Second, we want to experiment with a system of tightly coupled multiprocessors. If a modest number of processors can be efficiently applied to the computing needs of a single user, multiprocessors provide an attractive alternative to a high speed time-shared processor, since high speed technologies tend to have low packaging density and significant power and cooling problems. Although the

present Firefly is implemented with off-the-shelf components, if our initial experience with the system is positive we plan to build a more ambitious version using LSI.

The most noteworthy architectural feature of the Firefly is that the caches associated with each processor are consistent, in that stores done by one processor are reflected immediately in all other caches. This makes it possible to construct parallel programs that share memory directly without being concerned with issues of stale data in other caches. The method we have chosen to ensure consistency, referred to as "conditional write- through", is efficient both in the amount of logic required for its implementation and in the bus bandwidth required to support it.

The remainder of this report describes the Firefly hardware, with emphasis on the cache and its consistency protocol.

2. Hardware Overview

The Firefly, shown in block form in figure 1, consists of eight printed circuit cards. Each card with the exception of the Ethernet controller is 8" by 10" in area and can hold approximately 160 dual-inline integrated circuit packages.

The Ethernet controller is half this size. Standard DEC controllers are used for the disks (RQDX1) and Ethernet (DEQNA). The system can be configured with two 5.25" full-height rigid or floppy disks in any combination. The I/O controllers are connected via the standard DEC Q22 bus to a card that contains a Q bus DMA interface, a single processor and its associated cache and address translation unit, a controller for a mouse, keyboard, and RS232 communications channel, a DES encryption chip, a battery powered time-of-year clock, and a high resolution interval timer. This card also contains the system bootstrap code in EPROM. The Q bus processor is responsible for initializing the system and starting the other processors.

A second card type contains two processors, their caches and address translation units. The two processor/cache units are identical to each other and essentially identical to the processor and cache on the Q bus board. There are two of these cards in the system, for a total of five processors.

The Q bus is used only for input/output transfers. A private 10 Mbyte/second bus (the M bus) connects the processor/cache cards to each other and to two 4 Mbyte storage boards. This bus is carried across the rear edge of the cards on flat ribbon cable.

2.1. Storage Cards

The Firefly contains up to eight Mbytes of storage, packaged on two four Mbyte boards. 256Kbit dynamic RAMs are used as the storage element. One of the boards contains the timing and control logic for both boards, as well as a parity generator/checker and the system clock. The second board contains only storage chips and data and address receivers and drivers.

The caches always transfer four bytes at a time on the M bus, and this is the width of a storage word. Each storage word also contains a single parity bit. Parity is checked only on the contents of storage; transfers on M are not checked, nor are the contents of the caches. Parity errors are not reported with interrupts. Instead, the address of the 1Mbyte bank in error is saved in a register when an error occurs. It is the responsibility of the software to poll this register frequently and take action on an error.

The last 4Kbyte page of the real address space is reserved for interprocessor and processor-to-storage communication. A processor can read the storage error register (and simultaneously clear it) by doing a read to either of two locations in this page. The location used determines whether even or odd parity will be subsequently generated

and checked (this is provided for diagnostics).

2.2. Display Control

3. Cache Structure

The Firefly cache is direct mapped, and contains 4096 four-byte lines. The principal memories comprising the cache are the A (address) and D (data) memories (see figure 2). The virtual memory of the system consists of 128 address spaces, each containing 4096 pages of 4096 bytes. The address translation units support paging - any additional structuring is transparent to the hardware.

In a direct-mapped cache, there is a one-to-one correspondence between a line in main storage and a line in the cache. A cache entry contains the line's data, the high order address bits that do not participate in the selection of the line or the data within the line, and flag bits indicating the status of the line. A single comparison between the requested address and the high order address bits contained in the entry is sufficient to determine whether the cache contains the requested line of storage. Since the 68010 provides byte addresses, the two low order address bits select the byte within a line and bits 13..2 select the line in the A and D memories. The A memory holds

the real address of the line, as well as two status bits. The ASh (AShared) bit is used by the consistency mechanism, and indicates that this line may be present in one or more other caches. The ADirty bit indicates that the line contains the only valid copy of the datum, which necessitates writing the line to main storage before reassigning it to some other address.

The consistency mechanism requires that the cache be accessed using real, as opposed to virtual addresses. This means that the virtual addresses emitted by the processor must be translated before being used to access the data cache. The translation is done by the AS (address space) and T (translation) memories, which together constitute a 4096 entry direct-mapped cache for address translations. The translation is done in parallel with the access of the A and D memories. Unlike the data cache, a translation miss does not cause automatic reloading of the entry. Instead, a trap is caused, and reloading of the appropriate T entry and restarting the reference is the responsibility of the system software. The hardware has no knowledge of the data structures used for address translation.

The first stage in the translation is done by the AS memory, which maps the function codes (FC2..0) emitted by the 68010

into an address space number (AS6..0). The correspondence between address spaces and function codes is maintained by the system software; the AS memory must be reloaded by the CPU when it switches processes. Each AS entry also contains a bit (Transp') that indicates that the address provided by the 68010 is to be treated as a real address and the translation ignored - this is "transparent mode".

The T memory contains the actual translations. T is addressed by the virtual page number (V23..12), exclusive-ORed with three bits of the address space number. The address space number is included so that translations for up to eight address spaces that contain identical page numbers but do not use all of the virtual space can coexist in T simultaneously. Each entry in T is composed of three parts: An address space number (TAS6..0) that is compared with AS6..0 to form the translation hit (THit) signal indicating that the requested translation is present, the real address corresponding to the requested virtual address (TR22..14), and three flag bits. The TWE (write enable) bit indicates that the page may be written, the TTU (trap user) bit causes a trap if the 68010 attempts to access the page from user mode (FC2 = 0), and the TNP (not present) bit causes a trap if any access to the page is attempted.

Note that a T memory entry does not contain the full real page number - the two low order bits are elided. Similarly, the A memory entries contain only the nine bits of the real address that are not used to address the D and T memories. Since the processor indexes A and D with the low 14 bits of the virtual address while accesses done to maintain cache consistency use real addresses, the implementation requires that the two low order bits of the virtual and real page numbers must be identical. This requirement establishes a constraint on the software that manages real memory: A page with virtual page number V must be placed in a page frame with real page address R such that V MOD 4 = R MOD 4. Since we do not anticipate that software will need to map the same page to different virtual addresses, this constraint, which simplifies the design, is not considered serious.

The T, A, and D memories are high speed (45ns) 4K by 4 bit static MOS RAMS. The AS memory is composed of 25ns 16 by 4 bit bipolar RAMS. In addition to the sixteen chips of memory, the cache has thirty-three chips of logic in its data paths, plus twenty chips of control logic.

3.1. Busses

The processor sends twenty-four bit virtual addresses to the cache on the V bus, and transfers data on the sixteen-bit

bidirectional D bus. Communication between the caches and main storage takes place on the thirty-two bit M bus. M is synchronous and is time-multiplexed between data and addresses. The width of a cache line, the data bus, and a main storage word are identical, so only one bus cycle is required for a data transfer (although four cycles are required for the entire M operation). The M bus consists of the following forty-four signals:

Thirty-two data and address lines,

six Request lines, one for each cache (plus a spare),

an MBusy line that indicates that a memory cycle is in progress,

a Read/Write line that specifies the request type,

the MShared line used to maintain cache consistency,

an IORef line, asserted by the storage boards when a reference is made to the last page of the real address space,

a global Reset line, asserted by the master processor during power-up, and

the system clock.

The M bus protocol is discussed in section 5.

3.2. I/O Addressing

Addressing is slightly more complex on the board containing the Q bus interface. The 32 Kbyte bootstrap EPROM, the I/O

devices on this board, and the upper 8 Kbytes of the Q bus address space are mapped into two identical 64 Kbyte regions of the 68010 address space. The first region is at location zero of supervisor program space, so that the 68010 reset vector will be in the proper location. The second region starts at location 10000 (hex) in supervisor data space, so that the 68010 can access the I/O devices with normal data access instructions. Since the 68010 byte ordering is reversed from that of the PDP-11, for which Q bus I/O controllers were designed, the Q bus interface swaps the bytes of the data path.

3.3. DMA

The system supports word references to I/O registers in the last 8Kbyte page of Q bus address space, as well as DMA transfers done by Q bus devices into main storage. DMA devices acquire the V and D busses using the 68010 BG/BGACK protocol provided for this purpose, and do their data transfers through the cache. Since the 68010 is not prepared to handle a translation fault during a DMA transfer, devices must use real addresses and operate the cache in transparent mode. Because the Q bus is only capable of addressing 4 Mbytes of storage while the system contains 8 Mbytes, an extra address bit must be provided.

This is done by using the eight unused locations in the AS memory as a set of mapping registers. The top three bits of the Q bus address are used to index this memory during a DMA access, and four bits of the output are used as the top four bits of the real address. The Q bus interface is composed of approximately thirty integrated circuits.

4. Cache Operation

The cache and address translation unit respond to Fetch and Store requests from the processor. During a Fetch, the cache supplies a 16-bit datum in response to a the request. During a Store, the cache accepts a 16 bit datum plus two signals, UDS and LDS, that indicate the validity of the two bytes of the datum. In both operations, a three-bit function code indicates the address space to be referenced. Defined spaces are User Program, User Data, Supervisor Program, Supervisor Data, and CPU Space. The defined spaces are referenced as a part of normal program execution, but the 68010 has a privileged instruction that allows it to generate references using any function code value. CPU space references are treated specially, in that they are used to read and write the registers of the cache itself.

A processor operation proceeds as follows: The function

code is used to index the AS memory to obtain the address space number, and three bits of this value are XORed with bits 23..12 of the virtual address and used to index the T memory. If TAS6..0 (from T) are equal to AS6..0 (from AS), or if the AS entry or a cache mode control bit (described later) specifies transparent mode, the THit signal is generated. THit indicates that the real address on TR22..14 is a valid translation. In normal mode, TR22..14 and the TWE, TTU, and TNP flag bits come from the T memory. In transparent mode, TR22..14 are taken from V22..14, and the flags are forced to TWE=1, TTU=0, TNP=0, allowing the CPU to access real addresses directly.

If the translation fails or if the T flags prohibit the reference, a bus error trap is started immediately, aborting the operation. Otherwise, TR22..14 is compared with AR22..14, which was fetched from the A memory in parallel with the AS and T accesses. If they are equal, the cache contains the requested datum. If not, the datum must be read by doing an M access, but before the read, if the ADirty flag in the selected cache entry is true, the non-matching entry must be written back to main storage.

Cache consistency is maintained using the ASh bit. Whenever an M bus operation is done, all caches inspect the A memory

entry corresponding to the request. If they contain the referenced line, they assert a signal, MShared, that is part of the M bus, and simultaneously set the ASh bit in their A entry. The cache that originated the request sets the ASh bit in the referenced line to the value on the MShared line. If the M operation is Read, any cache that hits supplies its data on M. The MShared line inhibits main storage during a Read, so in this case the data comes from one or more other caches. More than one cache may supply the data, but since they do so at the same time and supply the same value, this is not a problem. If the M operation is Write, any cache that hits replaces its data with the data supplied by the cache that originated the request. During a Write, main storage also takes the data, so all caches that hit clear their dirty bits in the matching entry. This arrangement insures that when a cache line is first assigned to a line of main storage, all caches will be informed, and will set their ASh bits if they also contain the line.

In the originating cache, if the operation is Read, no special action is taken if the ASh bit is on in the referenced line. The cache supplies the data to the processor and the reference terminates normally. On the other hand, if the original processor request was a Store, when the requested line is present in the cache (either

because the request hit or after the line is read by an M Read operation), the action taken depends on ASh. If ASh = 0, then no other cache contains the line. The D memory entry is updated with the datum supplied by the processor and the reference is terminated. If ASh = 1, some other cache may contain the line, so an M Write is done and the data supplied by the processor (as well as the balance of the line contained in the D memory) is written to the other caches and to main storage. The D memory is updated with the data from the processor as a part of the M access, so at the end of the Write, all caches will contain the same data.

Table I summarizes the actions taken by a cache as a result of processor requests (PFetch, PStore) and M bus operations (MRead, MWrite). The figure shows the four states that a cache entry can assume, as well as the transitions between the states. From the figure, it can be seen that in the absence of sharing, the caches use write-back, but when data are shared by more than one cache, the operation switches automatically to write-through for the shared items. When a datum is no longer shared (because the shared line was reassigned in all but one of the caches), the last cache that holds the datum will do only one unnecessary write-through, at which point is will clear its ASh bit and return to a write-back strategy. This conditional write-through

scheme is efficient in that it uses only slightly more bus bandwidth than the minimum amount necessary to communicate changes between the caches, and no additional communication paths other than the single MShared line.

Table I
Cache States

	Operation:	PFetch	PStore	MRead	MWrite
Initial State:	0: not Dirty, not Shared	0	2	1	- *1
	1: not Dirty, Shared	1	0/1 *2	1	1
	2: Dirty, not Shared	2	2	3	- *1
	3: Dirty, Shared	3	0/1 *2	3	1 *3

The MRead and MWrite transitions only apply to entries that match the M address.

*1: An MWrite will not hit in a cache unless Shared is true, since an MRead must have preceded the write and set Shared.

*2: A write-through is done. If another cache asserts MShared, State := 1, otherwise State := 0.

*3 Dirty is cleared when a write-through is done, since main storage takes the data.

4.1. Synchronization

The 68010 provides a test-and-set instruction as a synchronization primitive. This instruction does a read-modify- write reference (this is the only situation in which

the 68010 uses RMW), but it is not possible for the cache to determine that an RMW is being done until the store is started. This means that atomicity cannot be provided by acquiring and holding the M bus between the fetch and the subsequent store. Instead, the cache uses the normal consistency-maintaining mechanism to determine if the atomicity of an RMW has been compromised, and causes a trap if it has. Since the 68010 restarts an RMW from the beginning after a trap, the operation will eventually succeed. The implementation is as follows: Each cache contains a flag that is set when an operation completes, and is cleared when the 68010 deasserts its address strobe (AS) signal. AS is asserted continuously between the fetch and the store portion of an RMW, so at the start of a store if this flag is set the operation is known to be an RMW. The cache also contains a second flag, set if the first flag is set and an M bus operation hits in the cache, and cleared whenever AS is deasserted. When a store is done to a shared line and both these flags are true, another cache may have modified the target line since this cache read it. In this case, a trap is started and the write is aborted. The trap-handling software will retry the reference until it succeeds. The test done by this logic is conservative, since there is no check that the M operation was done to the

same address as the CPU reference, but since such conflicts are expected to be rare, the conservative (and inexpensive) approach is satisfactory.

4.2. Cache Modes

Each cache in the system contains a four-bit mode register that is accessed as a location in the last page of real memory. Any processor can set or clear the mode bits in any cache including its own. The mode register is write-only, and is used primarily to allow the master processor to bring the other processors and caches into a known state during initialization. The mode bits are:

Transparent mode: This mode causes the cache to bypass the TLB and to treat an address presented by the CPU as a real address. A bit in the AS memory can also force individual address spaces into transparent mode.

Miss mode: This mode causes references done by the CPU to miss in the cache, thereby reloading the entry from main storage. Miss mode also inhibits write-back, so that main storage will not be corrupted during initialization, when the contents of the cache D and A memories are undefined.

Reset mode: When this mode is set, the associated CPU

is reset. In the master processor, reset is done differently, since this processor is the source of reset for the slaves, and is responsible for initializing the system. In the master, reset is caused by the restart button on the front panel, by the power supply, or by the keyboard microprocessor (so that a sequence of keystrokes can bootstrap the system). After a reset, the master processor begins executing code from its EPROM, and all other processors are halted. All processors including the master are reset to Transparent = TRUE, Miss = TRUE. Software in the master initializes its own cache, sets up main storage, then releases reset in the slaves. Each slave (which can now execute code from main storage) then sets up its cache and waits for an interrupt indicating that it has work to do.

Interrupt mode: When this bit is asserted, a CPU interrupt is requested. This bit cannot be cleared via an M bus reference; the interrupted CPU clears the request flag by referencing a location in CPU space. In the slave processors, this is the only source of interrupts. In the master, the I/O devices can also initiate interrupts.

5. Cache Timing

A Fetch can cause zero, one, or two M bus references, since it may have to reassign a cache line and may also have to write the line's original contents back to main storage. A store can cause up to three references - the two associated with a fetch plus a write-through. If a request hits in the cache, the 68010 proceeds at full speed, without wait states. Three wait states are added by the first M operation involved in handling a miss, and if more than one operation is required, each additional M reference adds four wait states to the operation.

An M Read operation requires a minimum of four clock cycles: one for bus arbitration and address transmission (this cycle is repeated if a higher priority requester acquires the M bus), two cycles to access the data, and one cycle to transport the data to the requester. An M Write requires only three cycles: The arbitration cycle plus two cycles to store the data. Write data is transported during the first cycle after acquisition of the bus. During M operations, all caches inspect their contents during the first cycle of the operation. Accesses to the A memory done on behalf of an M operation have priority over processor cycles - if there is a conflict, the processor must wait for the next

cycle. Normally a 68010 memory access requires four clocks, but the A memory is required during only one of these cycles, so in the case of a conflict the processor access will take five rather than four cycles. If M operations are taking place at full speed, conflicts will occur in an average of one-fourth of all cycles, so the processor access time will be increased from four cycles to (.75*4 + .25*5) = 4.25 cycles due to M conflicts. Things are slightly worse than this simplified calculation implies, since some fraction of the M operations will be Writes which take only three cycles, thereby increasing the number of conflicts. In addition, if an M operation causes a change in the state of a cache entry, an extra A cycle must be taken to record the change. These effects are difficult to quantify, since they depend on the cache miss rates, the fraction of the accesses that are writes, and on the degree of data sharing. They are not expected to be significant.

Table 2
Actions resulting from an A cycle initiated by P

	THit	TNP	TTU and Umode	TWE	AHit	A Dirty	ASh	Action
P Read:								
	0	X	X	X	X	X	X	Trap
	1	1	X	X	X	X	X	Page Fault
	1	0	1	X	X	X	X	Protection Fault
	1	0	0	X	0	1	X	Write, ADirty := 0
	1	0	0	X	0	0	X	Read, P := MD *1
	1	0	0	X	1	X	X	P := MD
P Write:								
	0	X	X	X	X	X	X	Trap
	1	1	X	X	X	X	X	Page Fault
	1	0	1	X	X	X	X	Protection Fault
	1	0	0	0	X	X	X	Write Fault
	1	0	0	1	0	1	X	Write, ADirty := 0
	1	0	0	1	0	0	X	Read *1,2
	1	0	0	1	1	X	1	Write *3
	1	0	0	1	1	0	0	D := PD, ADirty := 1
	1	0	0	1	1	1	0	D := PD

*1: ASh := MShared;

*2: IF MShared THEN [ADirty := 0; retry] (the retry will cause a Stor
ELSE [ADirty := 1; D := PD];

*3: Write-through to update memory and other caches that share the line.

ADirty := 0 in all caches that match the address, including the originating cache.

Other caches that match assert MShared, and the originating cache sets ASh := MShared (this will clear ASh in the originating cache if the line is no longer shared).

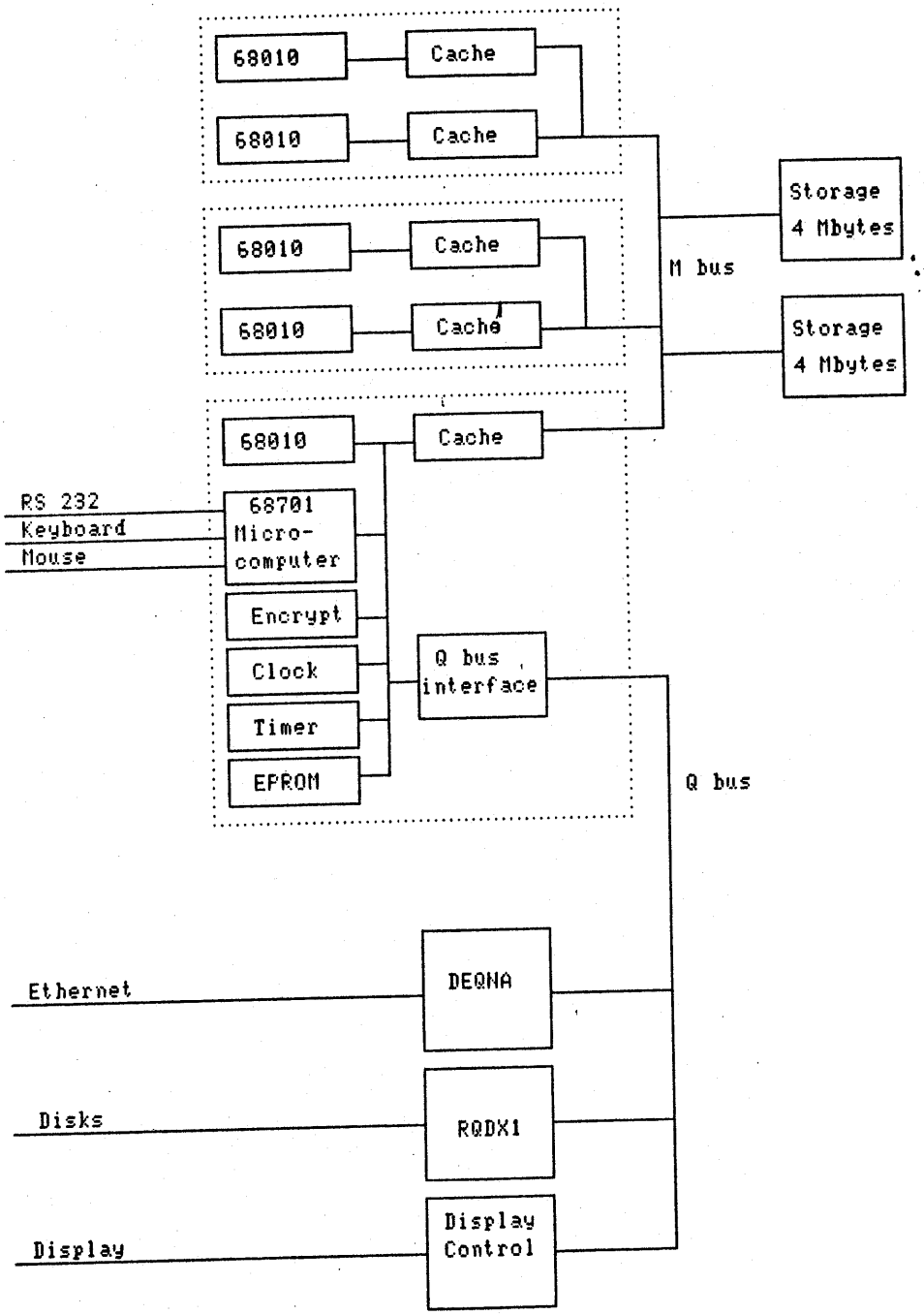


Figure 1: Firefly System

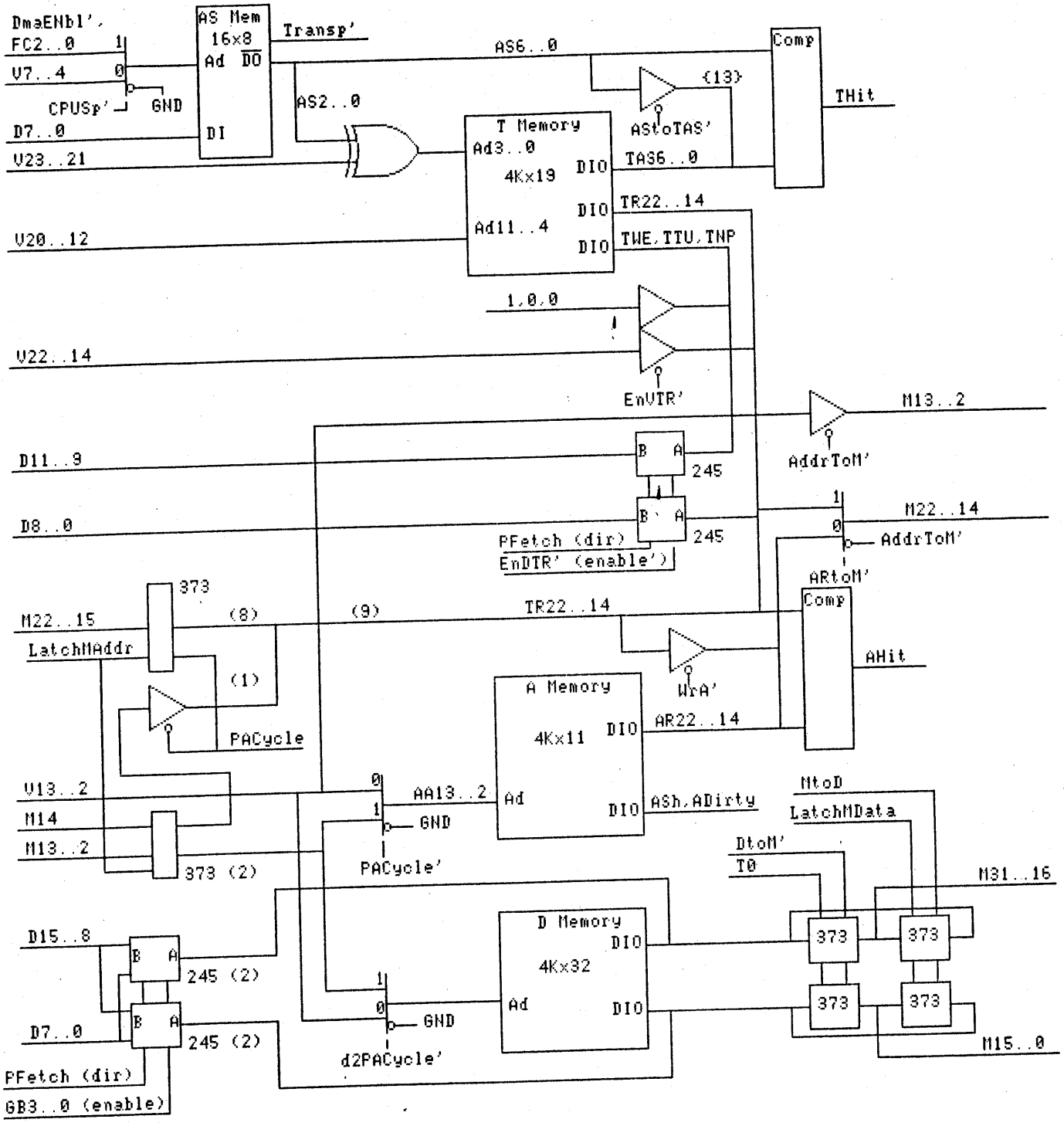


Figure 2: Firefly cache and TLB