Firefox Workstation Dual-CVAX Processor Module Functional Specification

Revision 3.0

David Smith (DECWSE::DSMITH)
Tom Furlong (DECWSE::FURLONG)

Workstation Systems Engineering
Digital Equipment Corporation
100 Hamilton Avenue
Palo Alto, CA 94301
415-853-6730

December 21, 1987

RESTRICTED DISTRIBUTION

Blank Page

Table of Contents

6. Dual-CVAX Processor Module

Revision History

Date	Version	Content/Changes
12 Nov 87	3.0	Updated KA600 to KA60; corrected initialization code; Operations, BOM, and Placement moved to Engineering Specification
14 Aug 87	2.1	Updated BOM and unqualified parts list; added new power requirements and placement
01 May 87	2.0	Updated functionality, implementation Added programming information Revised as per Revision 2 M-Bus Revised as per Revision 2 FBIC
30 Jan 87	1.1	Integrate with System Specification
10 Jan 87	1.0	First external release
21 Dec 86	0.0	Preliminary draft

Blank Page

6. Dual-CVAX Processor Module

This functional specification for the Firefox dual-CVAX processor module, KA60, describes the implementation of the modular CPU element of a Firefox multiprocessor. It does not describe the detailed function of the VLSI devices on the module, pertinent functional specifications for which are included in the Appendices to the *Firefox System Specification Designer's Guide*.

The functionality provided by the KA60 forms the basis of the expandable CPU architecture of the system. The circuitry required to implement two CVAX CPU elements of the Firefox multiprocessor is provided on this single, quad-height module, which contains all per-processor functions necessary to construct a symmetric multiprocessor (SMP). Supported functions include the following:

- Two CVAX CPU/FPA chipsets
- Two 64-Kbyte level-2 snoopy caches
- Two sets of SMP-required IPRs
- Firefox M-bus interface
- Two sets of diagnostic/boot ROMs
- Test mode inputs and status indicators

At least one KA60 module is required in all Firefox configurations. Certain configurations specify two processor modules, providing four CVAX processors. The architecture of the KA60 does not preclude configurations consisting of more than two processor modules. The factors that limit the number of CPUs in a Firefox multiprocessor are the backplane slots, M-bus bandwidth, and I/O throughput. Every effort has been made to ensure that as many KA60 modules as other system components are able to support can be configured in a Firefox.

Two versions of the KA60 are planned; they will differ only in the speed sort of the CVAX chipset and the cache RAMs that support it. Although the KA60 is being designed to support a CMOS II CVAX chipset with a 60-ns cycle time, the initial version of the KA60 will use a binned CMOS I chipset running at an 80-ns cycle time. As necessary throughout this document, the different versions will be referred to as KA60-60 and KA60-80 respectively.

6.1. Functionality

The KA60 provides two CVAX CPUs in a Firefox SMP environment. The system's M-bus and snoopy caches are designed to present each processor a consistent view of a single, global memory and I/O space. This feature, along with an interprocessor interrupt capability, provides the hardware functionality necessary to support an SMP operating system, under which the addition of each CPU improves the computational performance of the entire system.

Figure 6-1 is a functional block diagram of the KA60, with its two functionally independent CPUs that share a single set of bus transceivers, clocks, and test-mode connections.

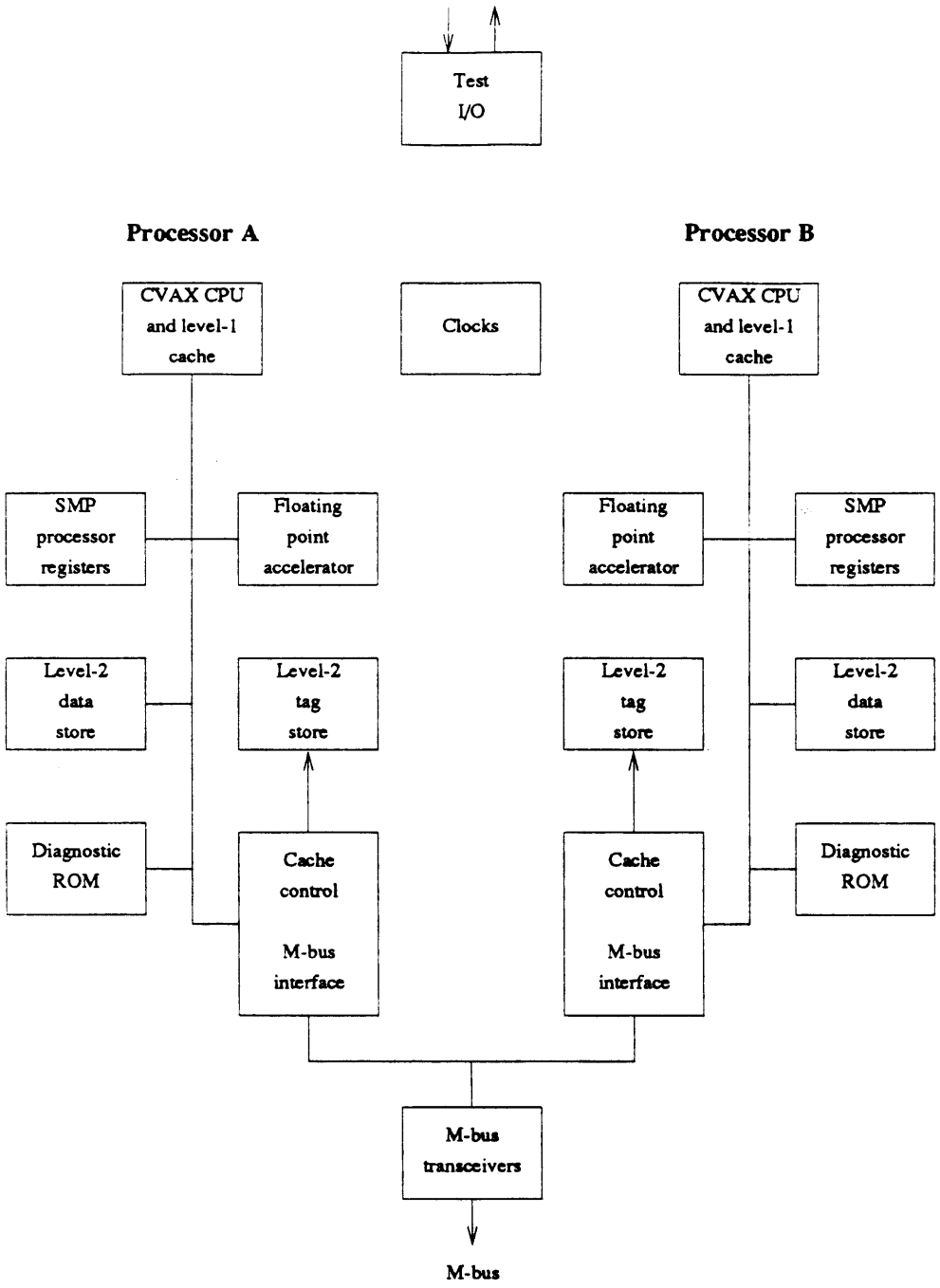


Figure 6-1: KA60 Processor Module

Because of the symmetry of the KA60, the following sections describe the functionality of either processor A or processor B without specific reference to implementation.

6.1.1. CVAX CPU/FPA

The CVAX CPU is a 32-bit, virtual-memory microprocessor that is implemented in custom CMOS technology and incorporates the following features:

- A subset of VAX data types: byte, word, longword, quadword, character string, and variable-length bit field

- The MicroVAX subset of the VAX instruction set: integer and logical, address, variable-length bit field, control, procedure call, queue, MOVC3/MOVC5, and operating system support
- Full VAX memory management
- An on-chip, 1 Kbyte, 2-way associative mixed instruction- and data-stream cache
- A 32-bit standard interface (CVAX pin-bus)
- Large virtual- and physical-memory address spaces: 4-Gbyte virtual- and 1-Gbyte physical-address space

To accelerate floating point operations, each CVAX has a companion CFPA that performs the following:

- Operates on F_floating, D_floating, and G_floating VAX data types
- Supports VAX floating-point instructions for F_floating, D_floating, and G_floating
- Accelerates execution of MULL, DIVL, and EMUL integer instructions

Complete specifications for the CVAX and CFPA chips are available in the appendices to the *Firefox Designers' Guide*.

VAX SRM values are as follow:

- System ID for the CVAX processor: 10
- System type for the KA60: 3

6.1.2. Two-Level Caches

Firefox CPUs implement two-level caches. The level-1 cache is internal to the CVAX processor. The level-2 cache is the external snoopy cache. The Firefox snoopy caches both service their attached CVAX processor and monitor the system M-bus to provide other caches with the contents of memory locations they have cached. Because all caches monitor all memory-space bus transactions, they can continually determine whether cache entries are exclusively in their cache or are shared by two or more caches. The caches use this information to maintain systemwide cache consistency and to dynamically minimize M-bus bandwidth, utilizing a write-back policy for unshared entries and a write-though policy for shared entries.

6.1.2.1. Level-1 Cache

Each CVAX CPU chip contains an internal, 1-Kbyte, 2-way, set-associative level-1 cache, with a quad-word line size. The level-1 cache, which is always write-through, can be independently enabled/disabled for instruction- and data-stream references.

In Firefox, the level-1 cache is enabled for both instruction- and data-stream references. To ensure system-wide cache consistency the level-1 cache is maintained as a subset of the level-2 cache. This is accomplished through CVAX cache-invalidate cycles in the following manner:

- A line removed from the level-2 cache is invalidated in the level-1 cache.
- A line updated from the M-bus in the level-2 cache is invalidated in the level-1 cache.

To maintain cache consistency, the level-1 cache must not be enabled while the level-2 cache is disabled.

6.1.2.2. Level-2 Snoopy Cache

All CVAX references to global memory or I/O space must go through a level-2 cache that implements the snoopy cache protocol as defined by the *Firefox System Specification*. The level-2 cache provides long-word access to a direct-mapped cache that has an octaword line size. At system powerup, a single-entry, level-2 snoopy cache present in the Firefox M-bus interface is utilized. After self-test and proper software initialization, a 64-Kbyte, level-2 snoopy cache is enabled for each processor on the KA60.

CVAX references to global I/O space are never cached and always generate M-bus cycles. CVAX memory space references are always cached by the level-2 cache and generate M-bus transactions

dependent upon the values of state variables stored with each line.

Each entry in the level-2 cache consists of an octaword data store, a tag store, and two state variables. The state variables represent the two independent properties of each entry: shared and dirty.

The shared property indicates whether the entry is present exclusively in this cache or resident in two or more system caches. If a cache entry is unshared, writes to this location do not generate bus traffic. If a cache entry is shared, write-hits initiate an M-bus write-through transaction to update the shared entries in other caches.

The dirty property indicates that the entry has been modified in the level-2 cache but has not been updated in memory. Dirty entries must be written back to memory if a miss requires use of the dirty entry's location in the cache; this is termed a *victim write*.

The snoopy caches monitor all M-bus memory space transactions and probe their tag store at the indicated address. If a hit results, they assert the MSHARED signal on the M-bus, thus allowing update of the shared status flags. For a write transaction, all monitoring caches with a hit update their data entries. For a read transaction, data will be supplied either from memory or from a cache that shares the entry. Data is supplied by a monitoring cache only when it hits on a dirty entry.

The combination of shared and dirty properties results in four possible states for each cache entry. Figure 6-2 diagrams the transitions among these states.

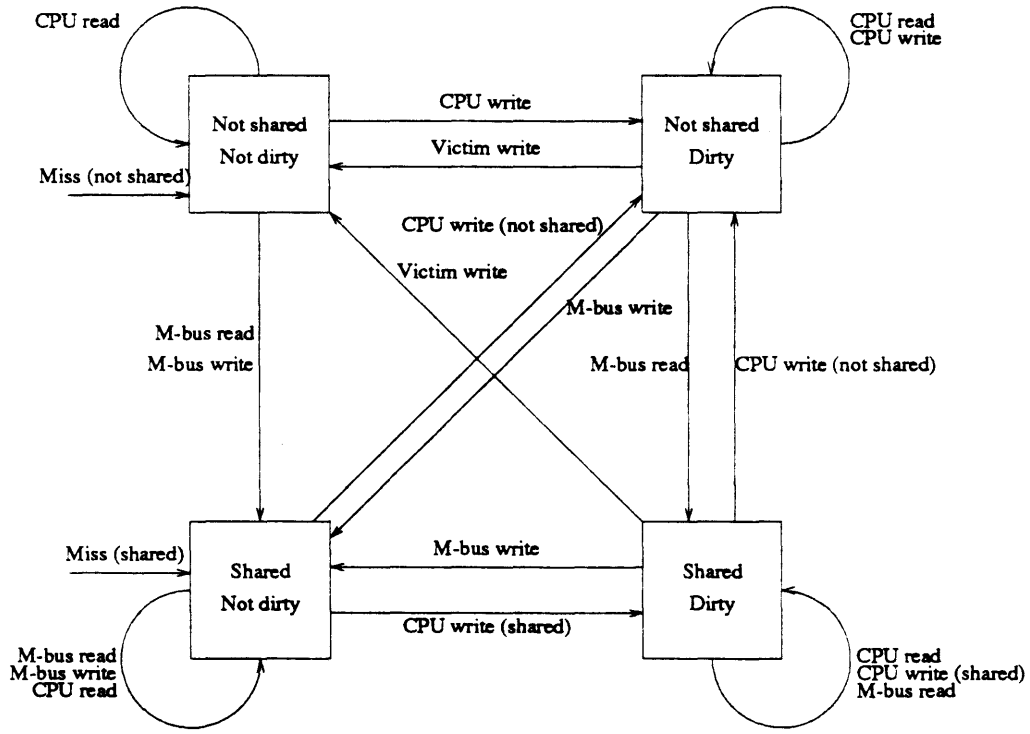


Figure 6-2: Snoopy Cache State Transitions

6.1.2.2.1. Cache Reads

When the CVAX issues a memory-space read, the level-2 cache probes its tag store to determine if the requested location is present in the cache. If the tag probe results in a hit, the level-2 cache presents the data from the level-2 data store to the CVAX. This is accomplished within one CVAX external cycle.

In the event of a tag-probe miss, the level-2 cache retries the CVAX, performs victim processing on the data store, and fills the data store with the requested line. The CVAX hardware will retry the read cycle, which now results in a read hit.

6.1.2.2.2. Cache Writes

When the CVAX issues a memory-space write, the level-2 cache probes its tag store to determine whether the requested location is present in the cache. If the tag probe results in a hit, the level-2 data store is updated by the CVAX, and the dirty bit is set for the cache line.

If the tag had the shared bit asserted, a masked octaword write-through transaction is generated on the M-bus using the data and byte masks from the CVAX. When the write through is completed, the level-2 shared bit for the line is updated with the value of MSHARED. That is, if the line is no longer in other caches, it reverts to being an unshared line.

If the tag probe results in a miss, the level-2 cache retries the CVAX, performs victim processing on the data store, and fills the data store with the requested line. The CVAX hardware will retry the write cycle, which now results in a write hit.

6.1.2.2.3. Victim Processing and Cache Fills

When a read or write miss occurs, victim processing is initiated before the data store is filled with the requested line.

If the victim line is clean, the level-2 cache immediately issues an M-bus memory read to obtain the requested line. While reading the data from memory, the cache controller performs a CVAX octaword cache invalidate cycle to (possibly) remove the victim line from the level-1 cache. When the M-bus read is complete, the data store is updated with the requested line. The tag store is updated with the new address, the shared bit is set to the value of the MSHARED signal, and the dirty bit is cleared.

If the victim line is dirty, the level-2 cache performs a CVAX octaword cache invalidate cycle to (possibly) remove the victim from the level-1 cache. The cache controller issues an M-bus memory victim write to flush the dirty line back to memory. The cache controller issues an M-bus memory read to obtain the requested line and fill the data store. The tag store is updated with the new address, the shared bit is set to the value of the MSHARED signal, and the dirty bit is cleared.

6.1.2.2.4. Shared Reads

Whenever the level-2 cache observes the start of a memory read on the M-bus, it does a tag-store probe to determine whether its data store contains the requested line. If a hit results, the cache controller sets its shared bit and asserts the MSHARED signal so the module that issued the memory read can also mark the cache line "shared." If the tag hit references a dirty line, the cache controller arbitrates to provide its data to the M-bus master. This ensures that modified data resident in caches is used in place of stale memory data.

6.1.2.2.5. Shared Writes

Whenever the level-2 cache observes the start of a memory write through on the M-bus, it does a tag-store probe to determine whether its data store contains the requested line. If a hit results, the cache controller sets its shared bit and asserts the MSHARED signal so the module that issued the memory write can also mark the cache line "shared." The slave tag-store dirty bit is cleared because the initiator of the write through has set its dirty bit.

6.1.2.3. Order for Enabling/Disabling Caches

To maintain systemwide cache coherence, the following procedure must be followed.

The system powers up with CVAX level-1 cache disabled and, by default, uses the hardware-initialized FBIC single-entry level-2 snoopy cache. The CPU begins executing out of ROM and turns on the level-2 cache to enable access to its cache tags. The tags are initialized to 00000000#16. When this is completed, the data entries are initialized to 00000000#16, following which the tags are rewritten to reflect shared, dirty status 00006000#16. At this point, all caches contain the same entries and are consistent, and the level-1 cache can be enabled as an instruction and data cache. The level-1 cache must not be enabled until the level-2 cache is operational.

Once the external cache has been enabled, it must not be turned off without first being manually flushed, or modified cache data will be lost. Should the level-2 cache need to be disabled, the level-1 cache must first be disabled, and the external cache manually flushed. To flush the external data store, read all tags through I/O space and force victim writes of any dirty lines by reading memory at a congruent address without generating memory writes in the process. Turning off the external cache is not recommended.

6.1.3. M-Bus Interface

Each CPU implements an M-bus interface that conforms to the *Firefox M-Bus Specification*. The M-bus provides a view of memory and I/O space that is consistent to all processors.

In addition to the M-bus snoopy cache protocol, the M-bus interface provides support for the following:

- VAX interlocked instructions
- VAX vectored interrupts
- Slot-specific I/O mapping

The diagnostic/boot ROMs and FBIC device registers are mapped to the 32-Mbyte, M-bus slot-specific I/O space. Processors A and B on the KA60 module further subdivide the slot-specific I/O into two 16-Mbyte regions.

M-bus arbitration and priority are managed in the M-bus interface. Subarbitration and priority determination between the two processors on the KA60 module are supported by the interface.

6.1.4. SMP Requirements

The KA60 supports required SMP functionality in the following manner:

- A global memory and I/O space is implemented through the hardware level-2 snoopy caches.
- A single console is provided that can independently halt any processor. Console input can be directed to any halted processor.
- External processor registers (EPRs) that contain the processor-specific data structures CPUID and WHAMI are implemented.
- The KA60 supports interprocessor interrupts at four interrupt-priority levels (14, 15, 16, and 17).
- A mechanism is provided to determine the number and status of processors in the system.

6.1.5. Diagnostic/Boot ROM

Each CPU has its own diagnostic/boot ROM. The ROM resides in local CVAX I/O space in the VAX restart address range of 2004000#16 to 2007FFFF#16. The ROM is also mapped to global slot-specific I/O space.

6.1.6. Test-Mode Inputs and Status Indicators

The value of two manufacturing-mode input pins is available to both CPUs for use by test software. Both CPUs on the KA60 share a set of test-mode input and output connections. Each CPU is provided a four-bit bank of red LEDs used to display per-processor status. A single bit from both CPUs' LED output latch is logically ANDed to provide module-OK test output and drive a single, green LED status indicator.

6.1.7. External Connections

The KA60 module has no external connections. Two manufacturing-mode input pins and one module-OK output pin are provided for attachment to manufacturing testers.

6.2. Implementation

The KA60 design makes extensive use of Surface Mount Device (SMD) technology to fit two VAX CPUs onto a single-sided module with 70 square inches of logic area. The KA60 uses the comet-on-a-quad PC board outline defined for Firefox.

A block diagram of the KA60 module appears in Figure 6-3.

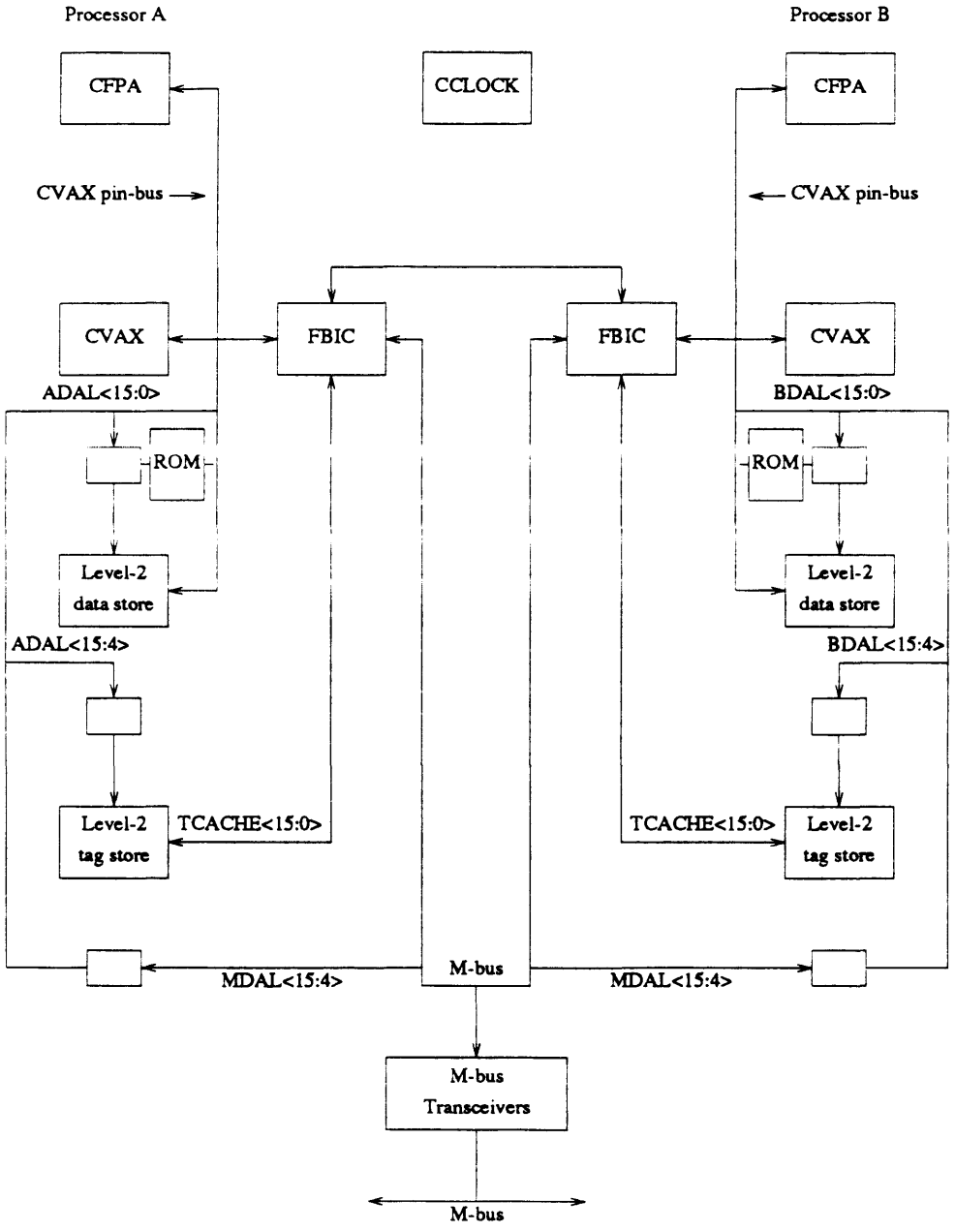


Figure 6-3: KA60 Block Diagram

The sections that follow describe the hardware implementation of the KA60. The FBIC is the multi-purpose bus interface and cache controller for Firefox. It connects a CVAX pin-bus to the system M-bus and supports the level-2 snoopy cache, SMP registers, and the manufacturing mode interfaces for the KA60. Subsets of the FBIC features are used in the implementation of each function described. The FBIC device features used in support of a given function are described in the context of that functional unit.

6.2.1. CVAX CPU

The CVAX CPU functionality of the KA60 is implemented in three custom CMOS chips produced by the Semiconductor Engineering Group in Hudson:

- CVAX CPU chip
- CVAX Floating-Point Accelerator chip (CFPA)
- CVAX Clock chip (CCLOCK)

Most of the circuitry required to configure the CVAX chipset as a Firefox SMP computer is provided through a single Firefox Bus Interface Chip (FBIC), which is an ASIC chip designed by the Workstation Systems Engineering Group in Palo Alto.

The KA60-80 uses binned CMOS I chips with an 80-ns cycle time. KA60-60 uses CMOS II chips with a 60-ns cycle time.

The CCLOCK chip is a CVAX support chip that generates all the precision MOS clock signals necessary to operate the CVAX CPU, CFPA, and FBIC. One CCLOCK chip generates clocks and for both CPUs. The different versions of the KA60 use oscillators of either a 50-Mhz, KA60-80 or a 66.6 MHz, KA60-60. For module-level testing, the CCLOCK oscillator input can be provided externally. The CCLOCK chip contains no software-configurable registers.

A complete description of the chipset can be obtained by referring to the CVAX, CFPA, and CCLOCK chip specifications provided in the appendices to the *Firefox System Specification Designer's Guide*.

Two CVAX CPU circuits are provided on the KA60, they share one CCLOCK chip.

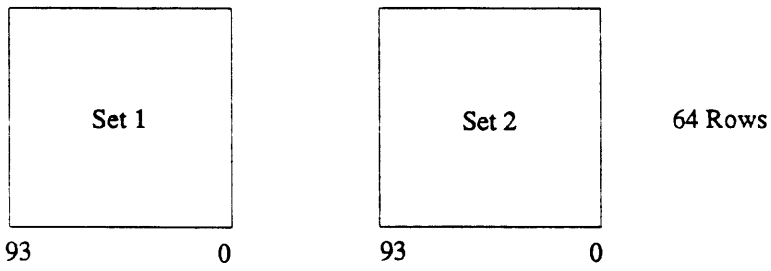
6.2.2. Caches

The KA60 level-2 cache is physically implemented as eight 16K x 4-bit SRAMs for data store, four 16K x 1-bit SRAMs for data parity, and four 4K x 4-bit SRAMs for the tag and tag parity. Six 74FCT373As provide the necessary bus isolation and address latching. A 74FCT191A and a 74FCT374A provide the longword addressing of the octaword data store. The cache controller and CVAX pin-bus to M-bus interface function is provided by the FBIC. A level-2 snoopy cache is provided for each CPU implemented on the KA60.

6.2.2.1. CVAX Level-1 Cache

The CVAX level-1 cache is a 1-Kbyte, two-way set-associative, write-through cache with a quadword line size. The level-1 cache stores both instruction and data-stream references. Level-1 cache organization is depicted in Figure 6-4.

Figure 6-4: Level-1 Cache Organization



A line in the level-1 cache is shown in Figure 6-5.

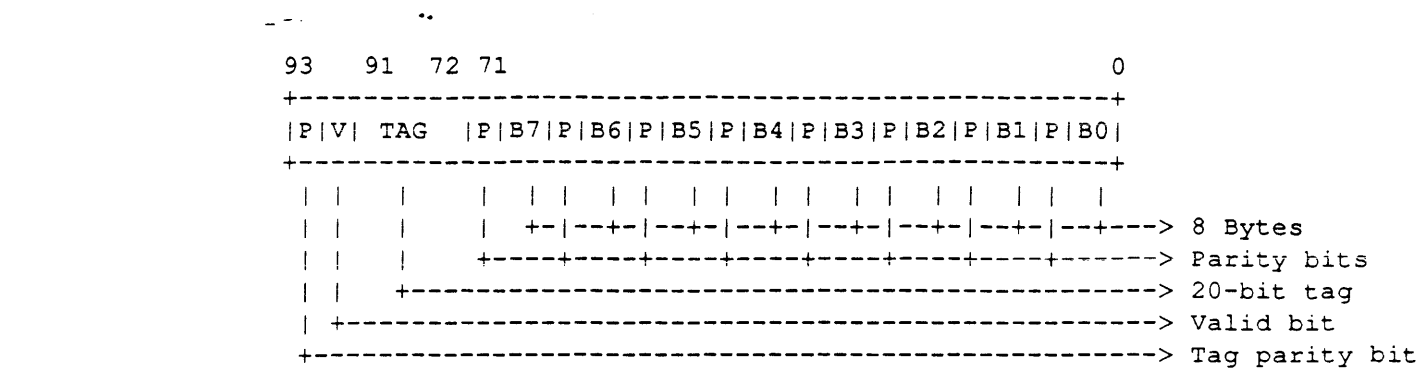


Figure 6-5: CVAX Level-1 Cache Line

In Firefox, the level-1 cache is enabled for both instruction- and data-stream references. This is accomplished by programming the CVAX cache disable register bits CADR<5> and CADR<4> to 0s. Under no conditions are I/O-space references stored in the level-1 cache.

The level-1 cache is protected by byte parity on the data store and by a single parity bit on the tag store. Internal CVAX error-checking signals parity errors on the level-1 cache.

The level-1 cache in Firefox is maintained as a consistent subset of the level-2 snoopy cache. Whenever a line is removed from the level-2 cache, or whenever a shared entry in a level-2 cache is updated from the M-bus, level-1 cache entries may become inconsistent with level-2 and must be invalidated. Level-1 cache invalidate cycles are performed by the FBIC in response to either of these conditions.

6.2.2.2. Level-2 Cache

Both CVAX CPUs on the KA60 are supported externally by a level-2 snoopy cache. The level-2 cache is a 64-Kbyte, direct-mapped, write-back snoopy cache with an octaword line size. The level-2 cache stores both instruction- and data-stream references. All CVAX instruction- and data-stream references that miss the level-1 cache will generate level-2 cache cycles. I/O-space references are never cached.

The level-2 cache has a data store 64 Kbytes in size and configured as 4K lines. Each cache entry has a 128-bit (octaword) data store, a 15-bit tag store, 16 data parity bits (byte parity), and 1 tag parity bit. The level-2 cache is always longword accessed; M-bus accesses always reference four sequential longwords. Figure 6-6 shows the organization of the level-2 cache.

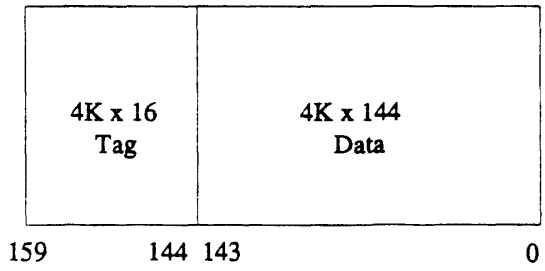


Figure 6-6: Level-2 Cache Organization

6.2.2.2.1. Level-2 Data Store

The data store consists of 16K longwords organized as 4K octawords. Each byte in the data store is error-protected by a parity bit. Parity is calculated/evaluated either by the CVAX processor or by the FBIC, dependent upon whether it is a CVAX pin-bus- or an M-bus-generated cycle.

The level-2 data store resides on the CVAX pin-bus side of the FBIC. Whenever an access to the data store from the M-bus is necessary, a CVAX DMA cycle is performed to get control of the CVAX pin-bus. The FBIC performs this function.

Figure 6-7 shows an entry in the level-2 data store.

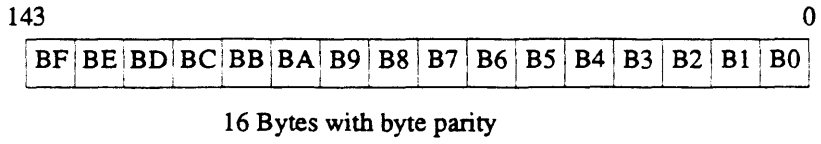


Figure 6-7: Level-2 Data-Store Entry

6.2.2.2.2. Level-2 Tag Store

The tag store is comprised of the address tag field, the snoopy cache dirty and shared flags, and a tag parity bit. A level-2 tag-store entry appears in Figure 6-8.

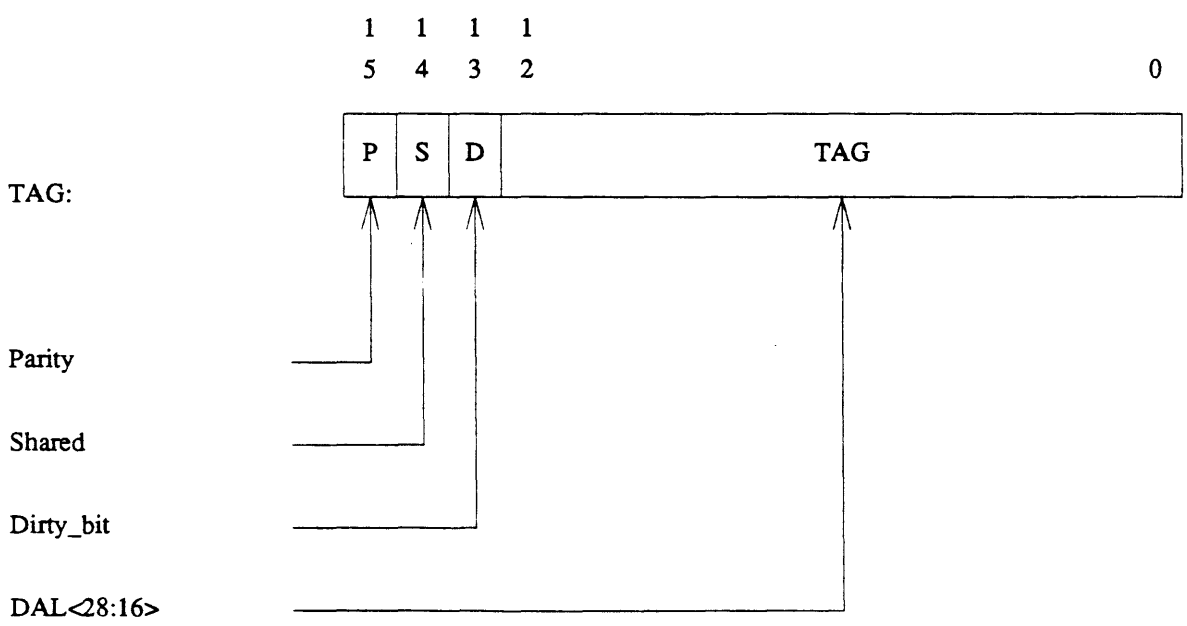


Figure 6-8: Level-2 Tag-Store Entry

Only the lowest 512 Mbytes of M-bus memory space are cached in the level-2 cache. This represents the full physical address space of a VAX processor. The address tag field corresponds to address bits <28:16> of the physical address of the current cache entry. The level-2 cache is a direct-mapped physical cache; address bits <15:0> represent the index into the 64-Kbyte cache.

The physical address of the entry in the data store is written to the address tag field whenever a line is allocated. The dirty flag is updated whenever a write occurs to the data store. The shared flag is updated whenever there is a read or write hit in the level-2 data store. New tag parity is calculated/evaluated

whenever the tag store is accessed. The FBIC is responsible for all access and updating of the level-2 tag store.

Physical address bits CDAL<15:4> are used to index this field. The contents of this field are compared against physical address bits CDAL<28:16> for equality. If there is a match, a cache hit occurs, and the FBIC takes appropriate action. The physical address bits CDAL<3:2> are used to index the appropriate longword in the cache. See Figure 6-9.

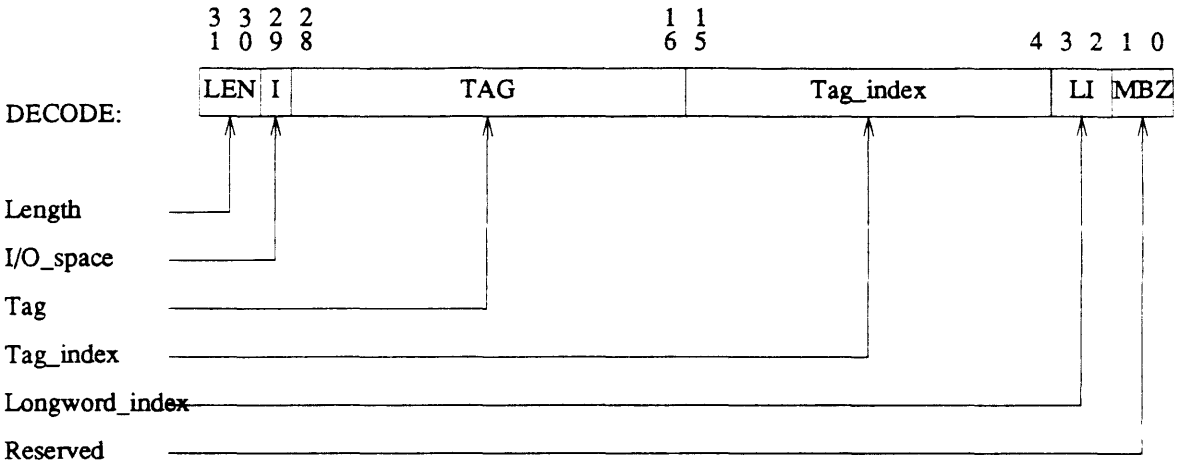


Figure 6-9: Level-2 Cache Address Decode

6.2.2.3. FBIC Support of the Level-2 Cache

The FBIC implements both a single-entry level-2 snoopy cache to be used at powerup and the controller for the external level-2 cache. The FBICSR <26> EXCAEN bit controls enable/disable of the level-2 cache.

6.2.3. M-Bus Interface

The M-bus interface consists of one FBIC per processor. Circuitry shared between the two CPUs includes seven 74F245 M-bus transceivers, one 74AS760 open collector M-bus driver, and one 74AS808B for on-board arbitration.

The FBIC registers that support access to the M-bus and error logging are MODTYPE, BUSCSR, BUSCTL, BUSADR, BUSDAT, FBICSR, RANGE, IADR1 and IADR2.

6.2.4. SMP Requirements

All support for SMP is provided through the FBIC. Some of the registers that support the SMP functions are implemented as external processor registers. All SMP registers are mapped to slot-specific I/O space. The console program initializes the SMP-specific registers of the Firefox.

6.2.4.1. Halting Processors

Individual processors can be halted by writing to FBICSR<25>, HALTCPU, in each CPU's slot-specific I/O space. To enable halting, FBICSR<7> HALTEN must be set. All processors in the system may be halted simultaneously by depressing the Halt switch on the RF distribution panel of the Firefox workstation.

6.2.4.2. Interprocessor Interrupts

Delivery of interprocessor interrupts is controlled by the FBIC IPDVINT register. Programming IPDVINT<16>, IPUNIT, to a 1 with an appropriate vector in IPDVINT<15:0>, VECTOR, causes the FBIC to function as an interprocessor interrupt unit.

Any processor in the system can interrupt another processor by writing to the target processor's IPDVINT<27:24>, IPL, field in the other processors' slot-specific I/O space. The FBIC will generate a vectored interrupt to occur on the target processor at the requested IPL level. The level of an interprocessor interrupt is passed to the target processor via the bits <3:2> of the vector.

6.2.4.3. SMP Processor Registers

The FBIC implements the SMP CPUID and WHAMI external processor registers. CPUID is a read-only register that specifies the hardware CPU identifier. CPUID is accessible as EPR 14 and through slot-specific I/O space. WHAMI (WHo AM I) is a read/write register that specifies a software CPU identifier, typically a pointer to a CPU-specific data structure. WHAMI is accessible as EPR 15 and through slot-specific I/O space.

6.2.4.4. SMP Console Support

The FBIC implements the SAVGPR (Save General Purpose Register) external processor register for use by the console program. SAVGPR is a read/write register that allows the console program to push the contents of one of the VAX GPRs without affecting processor context. SAVGPR is accessible as IPR 41 and through slot-specific I/O space.

6.2.4.5. SMP CPU Census

The FBIC provides the hardware necessary to determine both the number of CPUs present in an SMP configuration and their statuses.

At M-bus initialization, all FBICs assert their M-bus MBRQ lines. This provides a positive option-present determination for all M-bus slots. The value of the MBRQ lines is logged in the FBIC BUSCTL <6:0>, MBRM, register in each FBIC. By reading the value of the MODTYPE register of each FBIC for which a positive option present was received, the number of CPUs in the system can be determined.

The console program is able to interrogate the status of each CPU. CPUs that fail the self-test can be removed from the SMP system by the console program. This is accomplished by setting the TSTFNC bits of the FBICSR <5:1> register to logically isolate the processor on that FBIC from the M-bus. In that all CPUs run the self-test in parallel, the primary CPU needs only poll the status of each processor's CPU-OK signal in FBICSR <12>, LEDS<4>, to determine whether a given processor is functioning.

6.2.5. Diagnostic/Boot ROM

Each CPU is configured with one 27210 64K by 16 EPROM. A total of 128 Kbytes is provided to each CPU. The FBIC controls reading of the EPROMs that physically reside on the CVAX pin-bus. These EPROMs are socketed.

Diagnostic/boot ROM is mapped to the CVAX in local I/O space in the VAX restart address range of 2004000#16 to 2007FFFF#16. To the M-bus, the ROM appears in slot-specific I/O space defined by the FBIC.

Per the SRM, the system type identifier (value = 3 for the KA60) is programmed into the ROM at address 20040004.

6.2.6. Test-Mode Inputs and Status Indicators

Each processor implements a red quad LED indicator on which to report per-processor status. A single green LED indicator and a 74F00 provide the module-OK signal and indicator.

The FBIC provides the interface through which software reads the test-mode inputs and sets the indicator outputs. The FBICSR<11:8> LEDS are connected to each processor's red quad LEDs. The FBICSR<12> LEDS output of each CPU's FBIC are ANDed together externally to drive the single green LED. Setting FBICSR<12> in both FBICs will illuminate the green LED.

FBICSR<31:30> reflects the current status of the manufacturing-mode input pins to the module. The manufacturing-mode inputs of both FBICs are wired together.

6.3. Programming

This section describes both required initialization of the KA60 and the address spaces seen by the processor. The examples used here are based on a KA60 in M-bus slot 1. All addresses are VAX physical addresses.

6.3.1. Address Map

The Firefox M-bus supports a 2-Gbyte memory address space and a separate 2-Gbyte I/O address space. VAX processors support a 512-Mbyte memory address space and a separate 512-Mbyte I/O address space. Mapping of the 30-bit VAX address space into the 32-bit M-bus address space is provided by the FBIC. In this section, this mapping is ignored, and address maps are presented as they are seen by the CVAX processors. The KA60 physical address map is shown in Figure 6-10.

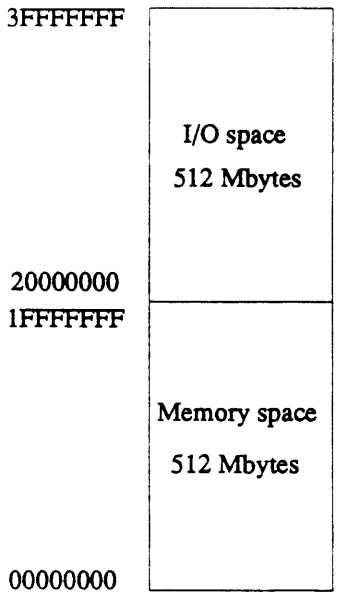


Figure 6-10: KA60 Physical Address Map

Memory-space references are decoded for CVAX physical addresses in the range of 00000000 . . . 1FFFFFFF#16. I/O space is accessed by addresses in the range of 20000000 . . . 3FFFFFFF#16.

6.3.1.1. Firefox I/O Space

Firefox I/O space, as seen by the KA60, is broken into the following four sections:

- M-bus global I/O
- Module-specific ROM
- Processor-local I/O
- Slot-specific I/O

M-bus global I/O provides approximately 128 Mbytes of systemwide I/O space. This is implemented to allow mapping of VLSI controllers with hardwired address decoders and encompasses the range of addresses assigned to Unibus and Q-bus controllers. M-bus global I/O locations can be mapped to any system module through the programming of the FBIC range registers.

Module-specific ROM is decoded locally on the KA60 in the VAX restart address range 20040000 . . . 2007FFFF. This allows each processor in a Firefox system to perform a boot and self-test without requiring any M-bus references. The module-specific ROM is also mapped by the FBIC to the slot-specific I/O region.

Processor-local I/O is provided for I/O-mapped coprocessors. Addresses in this range are fixed for each processor and are not globally mapped. This allows processor access to coprocessor registers at a fixed address that is independent of M-bus slot. Symmetric configurations of CPU/coprocessor pairs require this address mapping to allow the code that services them to be independent of the pair on which it executes.

Slot-specific I/O provides a 32-Mbyte region for module-specific I/O. This region contains FBIC registers, module ROM, tag stores, module CSRs, and I/O mapped buffer memory. At most, eight modules can reside in an M-bus backplane. Each module is assigned a 32-Mbyte region of I/O space, as determined by the module ID decoding of its backplane slot.

Figure 6-11 shows the Firefox I/O space as seen by a CVAX processor.

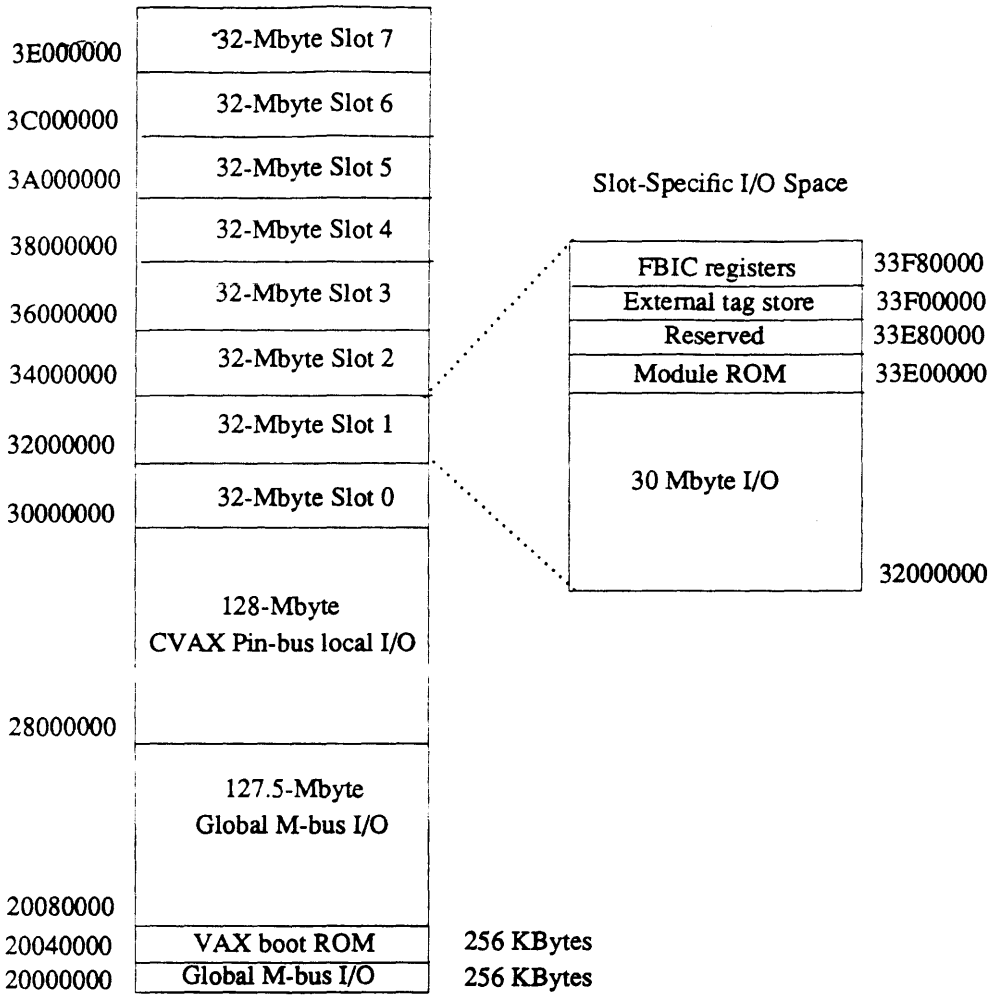


Figure 6-11: Firefox I/O Space

6.3.1.2. Slot-Specific I/O

Table 6-1 lists the base addresses for the various M-bus backplane slots. Accesses to devices mapped to slot-specific I/O must add the device offset to the slot base address.

Table 6-1: M-Bus Backplane Slot Base Addresses

Slot	M-Bus Address	VAX Address
0	90000000#16	30000000#16
1	92000000#16	32000000#16
2	94000000#16	34000000#16
3	96000000#16	36000000#16
4	98000000#16	38000000#16
5	9A000000#16	3A000000#16
6	9C000000#16	3C000000#16
7	9E000000#16	3E000000#16

The FBIC maps the upper two Mbytes of slot-specific I/O into four separate 512-Kbyte regions that contain the following:

- FBIC registers
- External tag store (processor modules only)
- Reserved
- Module ROM

Table 6-2 lists the offsets for the upper two Mbytes of slot-specific I/O space.

Table 6-2: FBIC-Managed Offsets

Slot	Address Range
FBIC registers	XXF80000 . . . XXFFFFFF
Tag Store	XXF00000 . . . XXF7FFFF
Reserved	XXE80000 . . . XXEFFFFF
Module ROM	XXE00000 . . . XXE7FFFF

FBIC registers require only a 64-byte portion of the 512-Kbyte portion. The registers appear 8192 times within this region. For compatibility with possible future extensions, software must access the registers only at their designated addresses. Table 6-3 provides a description of the FBIC registers and the functions they support.

Table 6-3: FBIC Register Map

Name	Address	R/W	Description
MODTYPE	XXFFFFFC#16	R	Module-type register
BUSCSR	XXFFFFF8#16	R/W	M-bus error status register
BUSCTL	XXFFFFF4#16	R/W	M-bus error control signal log register
BUSADR	XXFFFFF0#16	R/W	M-bus error address signal log register
BUSDAT	XXFFFFEC#16	R/W	M-bus error data signal log register
FBICSR	XXFFFFE8#16	R/W	FBIC control status register
RANGE	XXFFFFE4#16	R/W	I/O-space range decode register
IPDVINT	XXFFFFE0#16	R/W	Interprocessor/device interrupt register
WHAMI	XXFFFFDC#16	R/W	Unique software ID register
CPUID	XXFFFFD8#16	R	Unique hardware ID register
IADR1	XXFFFFD4#16	R/W	Interlock 1 address register
IADR2	XXFFFFD0#16	R/W	Interlock 2 address register
SAVGPR	XXFFFFC4#16	R/W	Scratch register for halt code

Each external cache store tag appears in I/O space four times at each longword within the naturally aligned octaword. That is, the tag for external cache line 0 is accessible at offsets XXF00000, XXF00004, XXF00008, and XXF0000C. Software can access the tag for a cache line at any of the four addresses. The entire set of tags reappears eight times within the 512-Kbyte tag space. For compatibility with larger external caches, software must access only the tags in the XXF00000 ... XXF0FFFF region.

The ROM appears at both the VAX restart address of 20040000 and XXE00000. The FBIC responds only to addresses 20040000 ... 2007FFFF (256 Kbytes) due to address-space assignments of the CQBIC. Modules with 512 Kbytes of ROM can access only the upper 256 Kbytes of the ROM at XXE40000 ... XXE7FFFF. A ROM smaller than its assigned region reappears multiple times within the region.

Multiple processor modules in a system communicate through slot-specific I/O space. Because there are two CVAX processors on each KA60 module, the slot-specific I/O space for a processor module is further subdivided into two 16-Mbyte regions. Figure 6-12 shows the I/O address map of another KA60 module in M-bus slot 1.

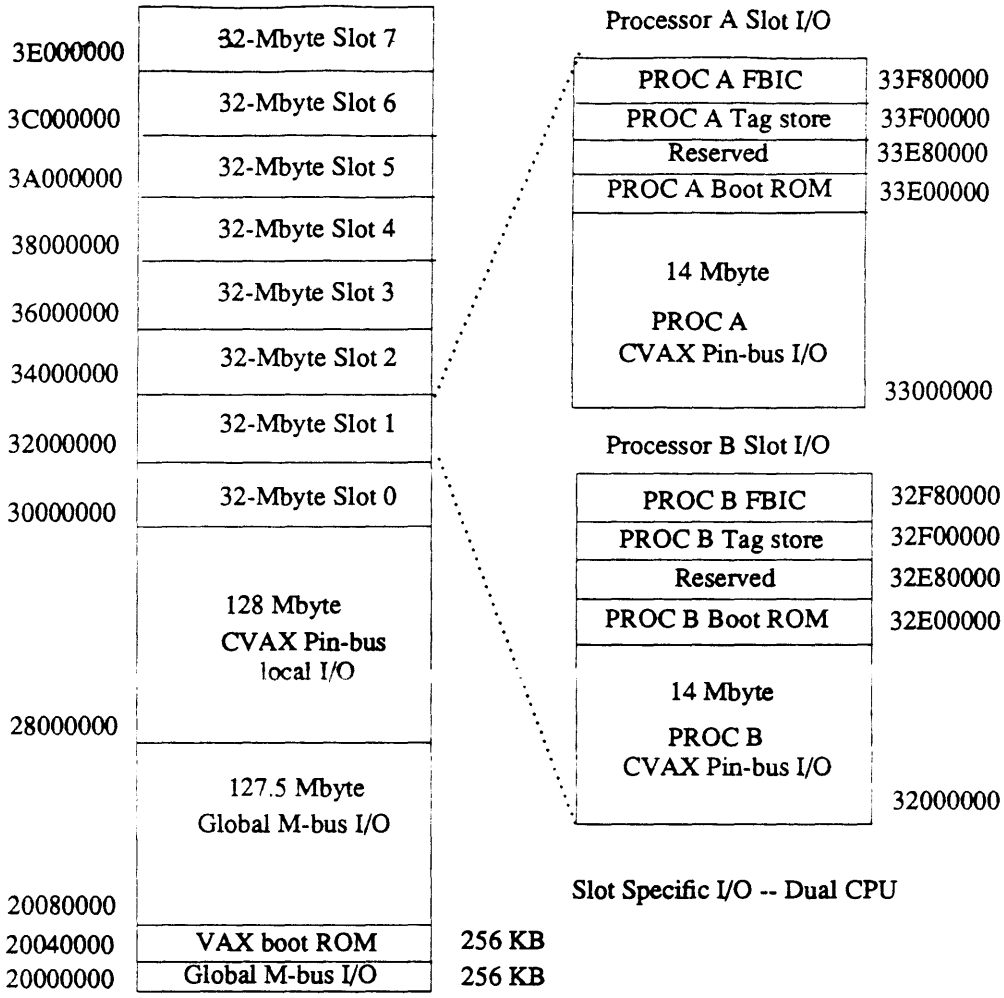


Figure 6-12: KA60 Slot-Specific I/O-Space Map

6.3.1.3. External Processor Registers (EPRs)

KA60 modules provide external processor register access to the SMP registers in the FBIC. Table 6-4 summarizes the FBIC EPRs.

Table 6-4: KA60 External Processor Registers

Name	EPR Address	R/W	Description
CPUID	14	R	Unique hardware ID register
WHAMI	15	R/W	Unique software ID register
SAVGPR	41	R/W	Scratch register for halt code

6.3.2. Locating Modules

After a workstation reset (M-bus MRESET asserted) console and diagnostic software must determine the workstation configuration. During MRESET, the FBIC saves the value of the M-bus MBRQ signals in its BUSCTL register. Software may use this as a module-present indication to identify backplane M-bus slots that contain Firefox modules. To interpret the BUSCTL<MBRM> bits, a module must first determine its

own M-bus slot by reading its FBIC CPUID<MID> register field. The FBIC CPUID register is mapped as a CVAX EPR and is read through a move-from-processor-register instruction. At powerup, each CPU must perform an EPR read of CPUID to determine the base address of its slot-specific I/O space. Table 6-5 lists the interpretation of BUSCTL<MBRM> as a function of a module's M-bus slot (from its CPUID<MID>). Software must use or save the value of BUSCTL<MBRM> before it enables FBIC error logging, or the information will be lost.

Table 6-5: Interpretation of the FBIC BUSCTL<MBRM> Field

CPUID<MID>	<6>	<5>	<4>	<3>	<2>	<1>	<0>
0	7	6	5	4	3	2	1
1	7	6	5	4	3	2	0
2	7	6	5	4	3	1	0
3	7	6	5	4	2	1	0
4	7	6	5	3	2	1	0
5	7	6	4	3	2	1	0
6	7	5	4	3	2	1	0
7	6	5	4	3	2	1	0

If a module reads its CPUID<MID> register field and obtains 4, it is in M-bus slot 4. If it then reads its BUSCTL<6:0> register field and obtains 1011001#2, there are modules in M-bus slots 0, 3, 5, and 7.

To confirm presence of a module in each slot, software should read the MODTYPE register of each slot. In the example just given, the MODTYPE registers would be at VAX addresses 31FFFFFC#16, 37FFFFFC#16, 3BFFFFFC#16, and 3FFFFFFC#16.

Computation of I/O address is performed as follows:

```
MFPR    #cpuid,rn          ;read cpuid
ASHL    #23,rn,rx          ;shift left 23
BICL2   #^X800000,rx       ;drop LSB of cpuid
ADDL2   #^X30000000,rx     ;= SLOT IO space add
```

6.3.3. Initialization

After workstation reset, the KA60 requires initialization of the FBIC and level-2 cache before normal operation can commence. Firmware will perform the required initialization sequences necessary to get to base system ROM on the I/O module and execute the console code. Details on the initialization sequence can be found in Chapter 13, "Diagnostics."

6.3.4. Base Workstation ROM

The base workstation ROM is located on the workstation I/O module. It contains system self-test code and workstation disk/tape/network bootstrap code. Access to the ROM from other modules is relatively slow--typically two microseconds per access. It is recommended that frequently used or time-critical code be copied to memory.