Origin™ and Onyx2™ Programmer's Reference Manual

Document Number 007-3410-001

CONTRIBUTORS

Written by Joseph Heinrich
Illustrated by Dan Young and Cheri Brown
Edited by Christina Cary
Production by Linda Rae Sande
For engineering contributions, please see *References and Source Material*.
St Peter's Basilica image courtesy of ENEL SpA and InfoByte SpA. Disk Thrower image courtesy of Xavier Berenguer, Animatica.

Origin™ and Onyx2™ Programmer's Reference Manual
Document Number 007-3410-001

Contents

List of Figures

List of Tables

About This Guide

This manual includes the following sections (each is a separate file):

- Front matter
- Contents
- List of Figures
- List of Tables
- *About This Guide*, which describes the organization of the manual, references, and typographical conventions
- Chapter 1, which describes the Origin™ Family memory map, including the physical and virtual address spaces
- Chapter 2, which describes the *M Mode* physical address space, including the *Cached*, *Hub Special*, *I/O*, *Memory Special*, and *Uncached* spaces
- Chapter 3, which describes the processor's view of *I/O* space
- Chapter 4, which describes the XIO view of *I/O* space
- Index

The figure on the next page illustrates this organization pictorially.

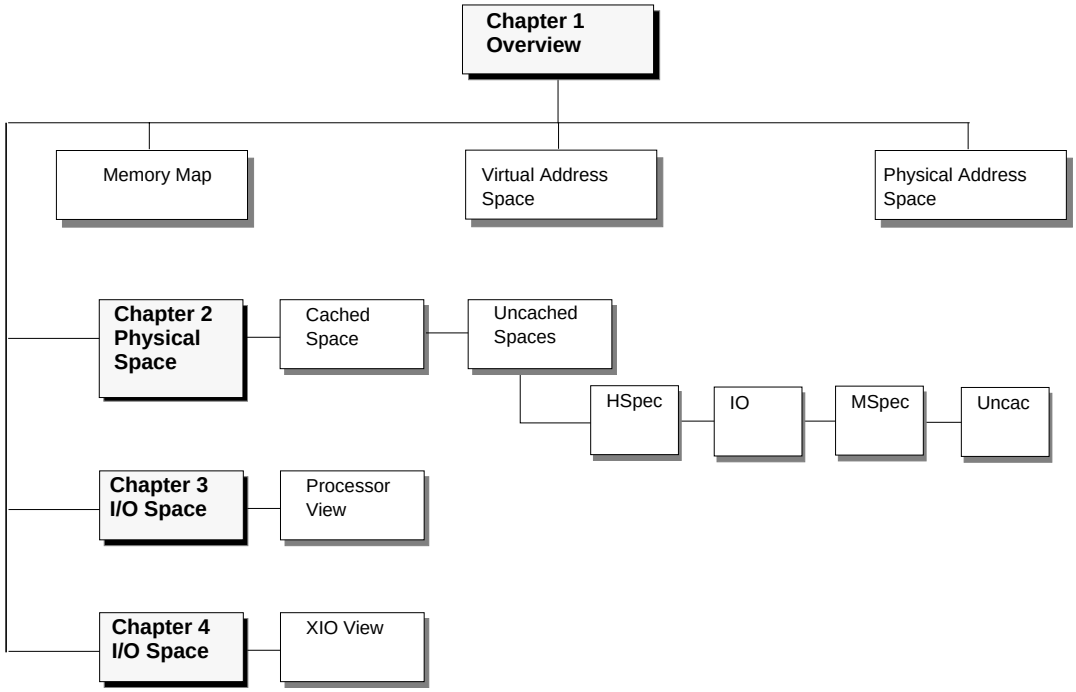

Figure i Organization of this Manual

References and Source Material

Much of the explanatory material in this book was taken from the following canons:

- Lenoski, Daniel and Weber, Wolf-Dietrich. *Scalable Shared-Memory Multiprocessing* San Francisco: Morgan Kauffman, 1995.
- Hennessy, John and Patterson David. *Computer Architecture: A Quantitative Approach* San Mateo, California: Morgan Kauffman, 1990
- Schimmel, Curt. *Unix Systems for Modern Architectures* Menlo Park, California: Addison Wesley, 1994

James Laudon's *System Specification* and *Cache Coherence Protocol Specification* provided a wealth of information. Both are internal Silicon Graphics® documents.

The following information was also relevant:

- Kourosh Gharachorloo, Daniel Lenoski, James Laudon, Phillip Gibbons, Anoop Gupta, and John Hennessy. *Memory Consistency and Event Ordering in Scalable Shared-Memory Multiprocessors*. Proceedings of the 17th International Symposium on Computer Architecture, pages 15-26, May 1990.

  `ftp://www-flash.stanford.edu/pub/flash/ISCA90.ps.Z`

- Kourosh Gharachorloo, Anoop Gupta, and John Hennessy. *Revision to Memory Consistency and Event Ordering in Scalable Shared-Memory Multiprocessors*. Technical Report CSL-TR-93-568, Computer Systems Laboratory, Stanford University, April 1993.

  `ftp://www-flash.stanford.edu/pub/flash/ISCA90_rev.ps.Z`

- Daniel Lenoski, James Laudon, Truman Joe, David Nakahira, Luis Stevens, Anoop Gupta, and John Hennessy. *The DASH Prototype: Implementation and Performance*. In Proceedings of the 19th International Symposium on Computer Architecture, pages 92-103, Gold Coast, Australia, May 1992.

  `http://www-flash.stanford.edu/architecture/papers/paperlinks.html`

Thanks also to **Ben Passarelli**, **Rick Bahr**, **Rich Altmaier**, **Ben Fathi**, **Ed Reidenbach**, **Rob Warnock**, **Jim** "Positive-ECL" **Smith**, **Sam Sengupta**, **Dave Parry**, **Robert A. dePeyster**, **Mike Galles**, and **Luis Stevens**.

Finally, thanks to **John Mashey** (mash@mash.sgi.com) for making himself iteratively available during various emergencies.

Typographical Conventions

Italic

- is used for *emphasis*
- is used for *bits*, *fields*, and *registers* important from a software perspective (for instance, *address bits* used by software, *programmable registers*, etc.)

Bold Text

- represents a term that is being **defined**
- is used for **bits** and **fields** which are important from a hardware perspective (for instance **signals** on the backplane, or **register** bits which are not programmable but accessible only to hardware)

For More information

The following documents provide additional information about the Origin family of systems:

Origin and Onyx2 Theory of Operations Manual, part number 007-3439-*nnn*

IRIX Device Driver Programmer's Guide, part number 007-0911-*nnn*

Origin2000 Deskside Server Owner's Guide, part number 007-3453-*nnn*

Origin2000 Rackmount Owner's Guide, part number 007-3456-*nnn*

Onyx2 Deskside Workstation Owner's Guide, part number 007-3454-*nnn*

Origin200 Owner's Guide, part number 007-3415-*nnn*

Chapter 1

Overview of Origin Family Memory Map

An Introduction to Origin Family

The Origin family of multiprocessor systems includes the entry-level Origin200™ system, and deskside and rackmounted Origin2000™ systems.[1]

The Origin family is both modular and scalable; that is, it can be increased in size (**scaled**) by adding **nodes** (or node boards) to the interconnection fabric. Each **Node board** can contain up to two R10000™ processors, with accompanying cache, directory, main memory, and interfaces to both I/O devices and the interconnection fabric.

The **interconnection fabric** (called the **CrayLink™ Interconnect**) replaces the shared bus of the Everest architecture with a web of point-to-point links that simultaneously connect the nodes to each other and present a multitude of paths from one node to another. For instance, as shown in Figure 1-1. R1 can communicate with R0, R2 to R3, R4 to R6, and R5 to R7, all without having to interface with any other node.

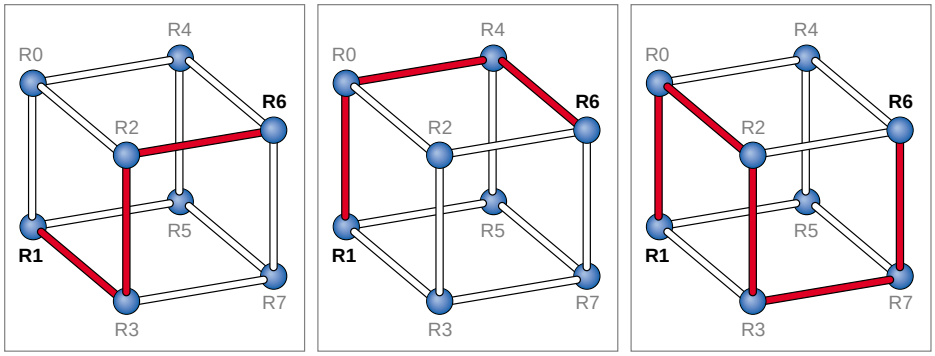

Figure 1-1 Origin System Datapaths

[1] For more information on the hardware aspects of the Origin family, please refer to *Origin and Onyx2 Theory of Operations Manual*, document number 007-3439-*nnn*.

The Origin200 comes in a server module, while the Origin2000 has several types of modules:

- graphics
- server
- peripheral

These Origin2000 modules are used in two types of systems:

- deskside
- rackmounted

This hierarchy, in order of increasing complexity, is shown in Figure 1-2.

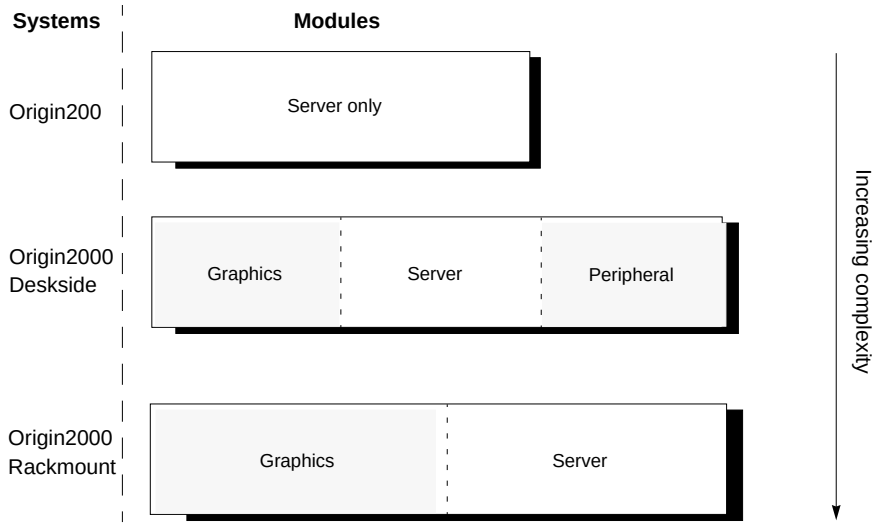

Figure 1-2 Origin Family Components: System and Modules

An Origin2000 system can be a single node or it can consist of a number of nodes mounted inside a deskside enclosure. Combinations of these deskside enclosures can be combined in a rack, and a system can be made up of a number of racks. Presently, the largest system available has 128 processors (a **128P** system).

Virtual Address Space

The MIPS® family of 64-bit processors provide a single uniform virtual address space for user processes. The Origin family uses the R10000 processor which defines a 2^{44}, 16 terabyte (TB) user-addressable virtual address space labelled *xuseg*.

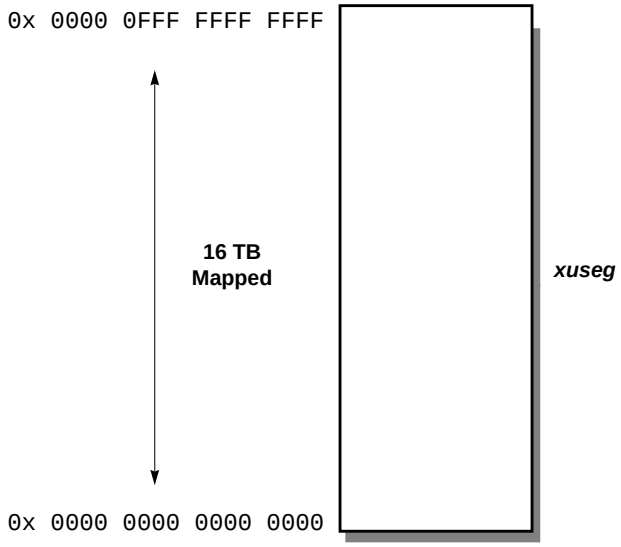

Figure 1-3 User-Addressable Virtual Address Space

The Origin family has a **distributed shared-memory** architecture, in which shared main memory is distributed amongst the nodes. This shared memory is accessible to every processor in the system.

Following is a mapping of the virtual address bits as they are decoded in Origin family. Detailed descriptions of the R10000 address spaces are given in the *MIPS R10000 Microprocessor User's Manual*.

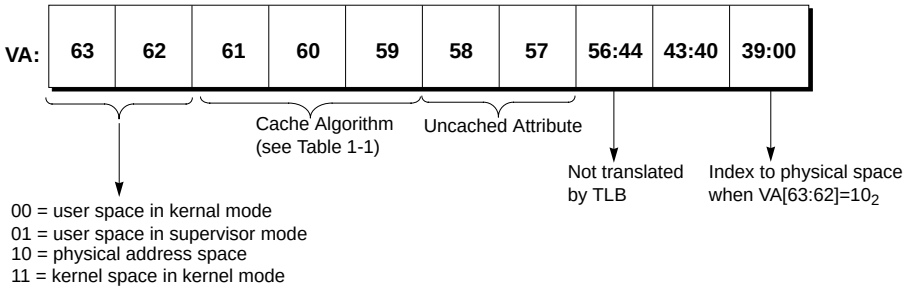

Figure 1-4 Virtual Address Bit Mappings

VA[61:59] *Cache Algorithm* bits set the behavior of the processor when executing load and store instructions. There are five cache algorithms, as listed in Table 1-1:

Table 1-1 Origin Family Cache Algorithms

VA[61:59]	Cache Algorithm
000	Reserved
001	Reserved
010	Uncached
011	Cacheable, noncoherent
100	Cacheable, coherent exclusive
101	Cacheable, coherent exclusive on write
110	Reserved
111	Uncached accelerated

In kernel mode, when VA[63:62] are 10_2, the *Uncached Attribute* bits on VA[58:57] select among the four uncached spaces, as described in Table 2-1 in Chapter 2.

Physical Address Space

To a processor, main memory appears as a single address space containing many individually-addressable **blocks**, or *pages*. Each node is allotted a static portion of this address space — which means there is a gap in the address space if a node is not present. Figure 1-5 shows an address space in which each node is allocated 4 GB of address space, and Node 2 is missing, showing a gap from address space 4G to 8G.

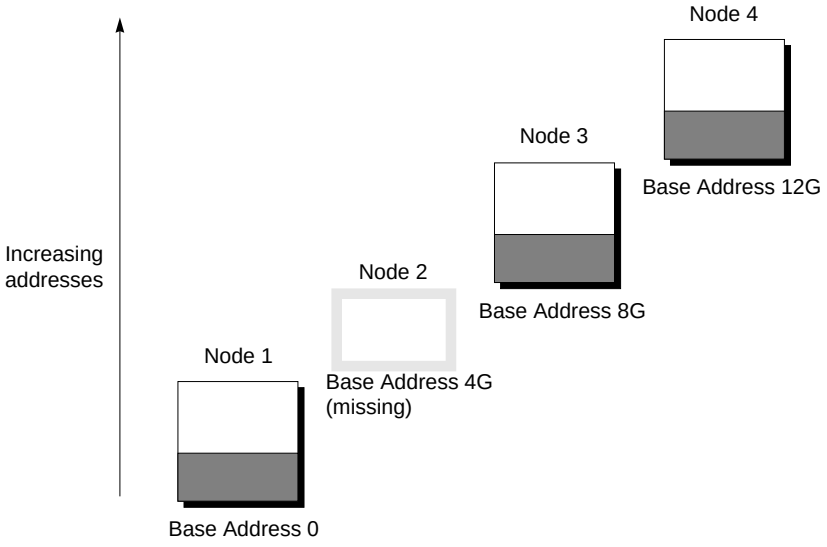

Figure 1-5 Physical Address Space

Secondary cache lines are fixed in size at 32 words, or 128 bytes. Main memory page sizes are multiples of 4 KB, usually 16 KB. These two configurations are shown in Figure 1-6.

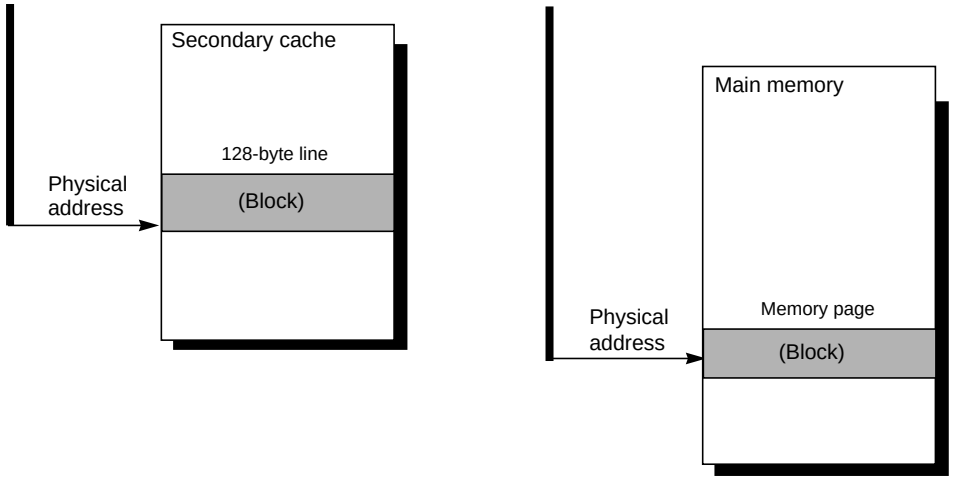

Figure 1-6 Cache Lines and Memory Pages

Architecturally, the physical address is divided into the fields shown in Figure 1-7. The upper 12 index bits of the **NUMA Address Space Identifier** (NASID) are used to select the node that contains the addressed physical memory. The lower 36 **NASID offset bits** are used to address memory blocks within a NASID's physical memory space.

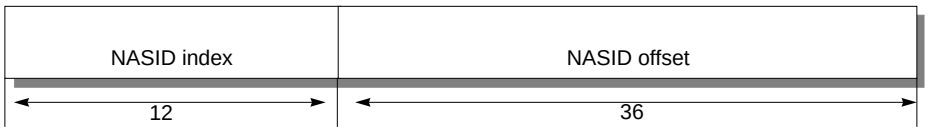

Figure 1-7 Physical Address Space Fields

The architectural limit of physical addressing is:

- 12 bits of NASID index (architecturally it is possible to address 2^{12} or 4K nodes)
- 36 bits of memory per NASID (architecturally it is possible to address 2^{36} or 64 GB of main memory in each NASID).

Note that Origin systems do not use this entire address space, but are implemented as *subsets* of these architectural limits, as described in the next section.

Node Physical Address Space

Although the architecture specifies a 48-bit address space, the initial implementation of the Origin family does not support this entire address range. Instead, the M Mode configuration, shown in Figure 1-8, is supported.

M Mode places an 8-bit NASID index in the upper 8 bits of the physical address. The remaining 32 bits of the physical address are used as offsets within each NASID. The NASID index addresses up to 256 nodes (512 processors), and each node addresses up to 4 GB of main memory.

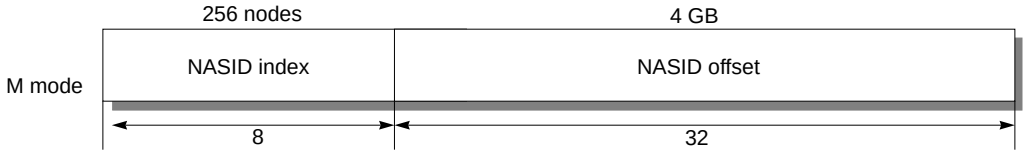

Figure 1-8 *M Mode* Physical Address Fields

Hub Physical Address Space

Inside the Hub ASIC, a 41-bit format is used to address physical memory. An extra bit is added to the *M Mode* NASID index, while the lower 32 bits are used as the address offset. Figure 1-9 shows how the system converts the 40-bit *M Mode* address to the 41-bit Hub address: in *M Mode* an extra address bit is added to the NASID index, making it 9 bits wide.

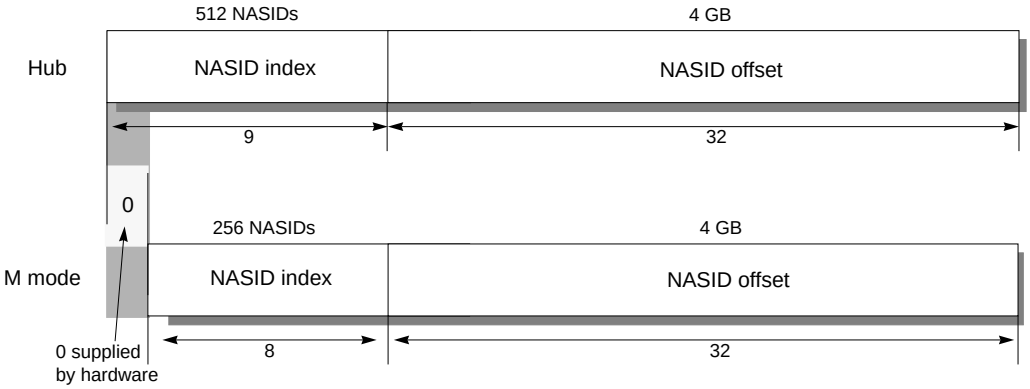

Figure 1-9 Conversion of NASIDs to Hub Internal Addresses

M Mode operations are described in the next chapter.

Chapter 2

M Mode Operations

Figure 2-1 illustrates *M Mode* physical address space. There are five different address spaces:

- *Cached* space
- *Hub Special* space
- *I/O* space
- *Memory Special* space
- *Uncached* space

All spaces but *Cac* are uncached spaces.

Each of these spaces are 1 TB, within which an *M Mode* node is allocated 4 GB. As shown in Figure 2-1, the entire 1 TB range of each address space is divided equally among the 256 nodes (`256*4 GB = 1 TB`).

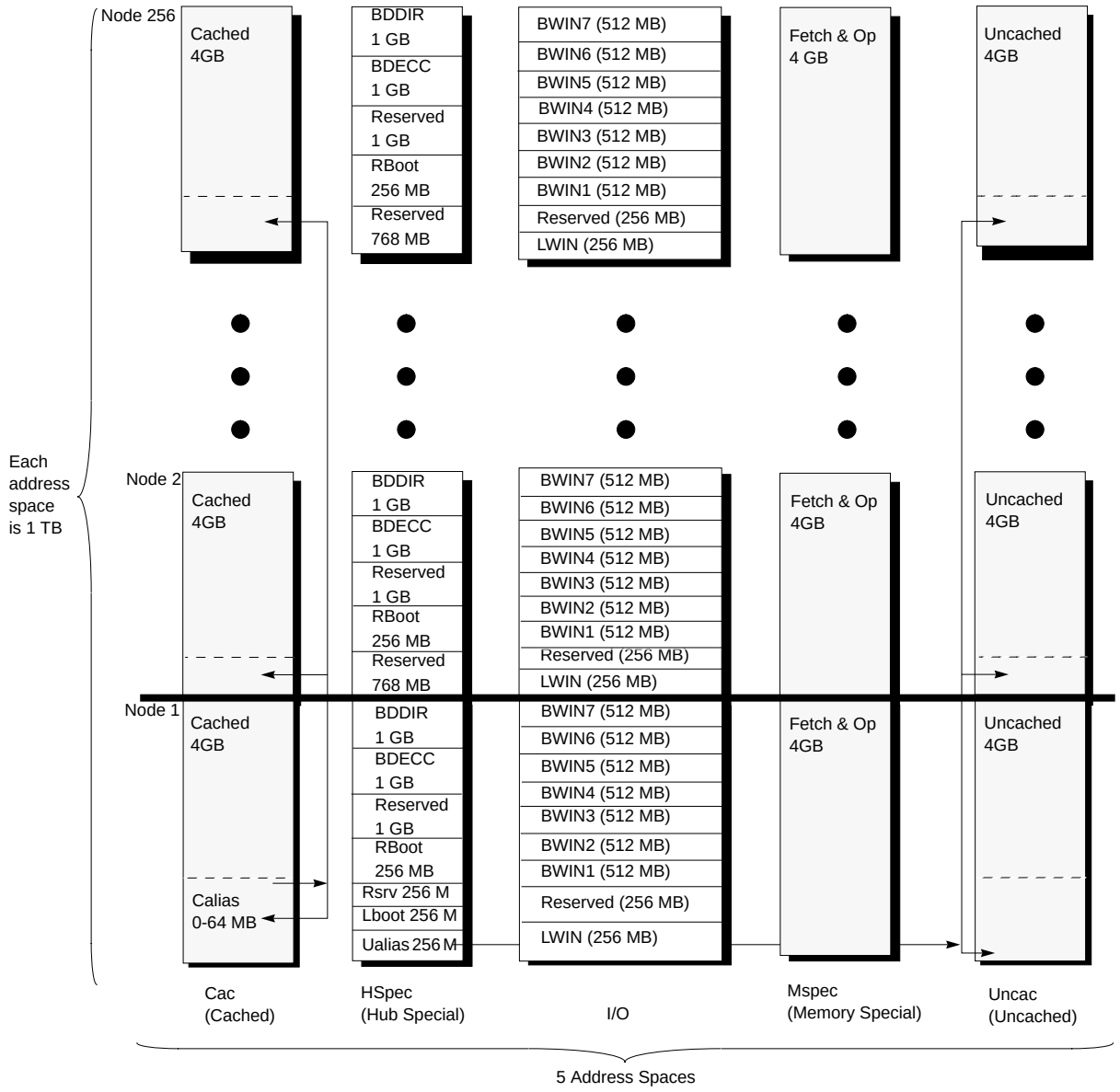

Figure 2-1 M Mode Physical Address Space

The first of these spaces, *Cac*, is described in the next section. The four remaining uncached spaces are described in succeeding sections.

The four uncached spaces are accessed by using uncached operations, and are differentiated by encodings of the 2-bit *Uncached Attribute* field of the *EntryLo0* and *EntryLo1* registers on the R10000 processor. These four spaces and their *Uncached Attribute* field encodings are listed in Table 2-1:

Table 2-1 *Uncached Attribute* Field Encoding

***UC* Field Encoding**	**Space**	**Name**
00	*HSpec*	Hub Special Space
01	*IO*	I/O Space
10	*Mspec*	Memory Special Space
11	*Uncac*	Uncached Space

Cached Space (*Cac*)

All cacheable loads and stores operate in the *Cac* physical address space. Secondary cache lines are 128 bytes.

The *Cac* space is flat, except for the lowest portion of *Cac* space memory, which may be used as a local alias. This lowest portion is referred to as **Calias**. It may range in size from 0 to 64 MB, as determined by bit settings in the *CALIAS_SIZE* register in the Hub ASIC. Encodings of the *CALIAS_SIZE* register are shown in Table 2-2.

Table 2-2 *Calias_Size* Register Encoding

Register Entry	Calias Space Size	Register Entry	Calias Space Size
0000	0 bytes	1000	512 KB
0001	4 KB	1001	1 MB
0010	8 KB	1010	2 MB
0011	16 KB	1011	4 MB
0100	32 KB	1100	8 MB
0101	64 KB	1101	16 MB
0110	128 KB	1110	32 MB
0111	256 KB	1111	64 MB

With *Calias* set to 0, *M Mode* memory accesses to the *Cac* space map from node to node in 4 GB increments, as shown in Figure 2-2:

- accesses to physical address 0 map to the base address of Node 1 (address 0 GB)
- accesses to physical address 4G map to the base address of Node 2 (address 4 GB)
- accesses to physical address 8G map to the base address of Node 3 (address 8 GB)
- accesses to physical address 12G map to the base address of Node 4 (address 12 GB)

And so on.

If a particular node is missing, it leaves a gap in the address map and any attempted access to the missing address space returns an address error.

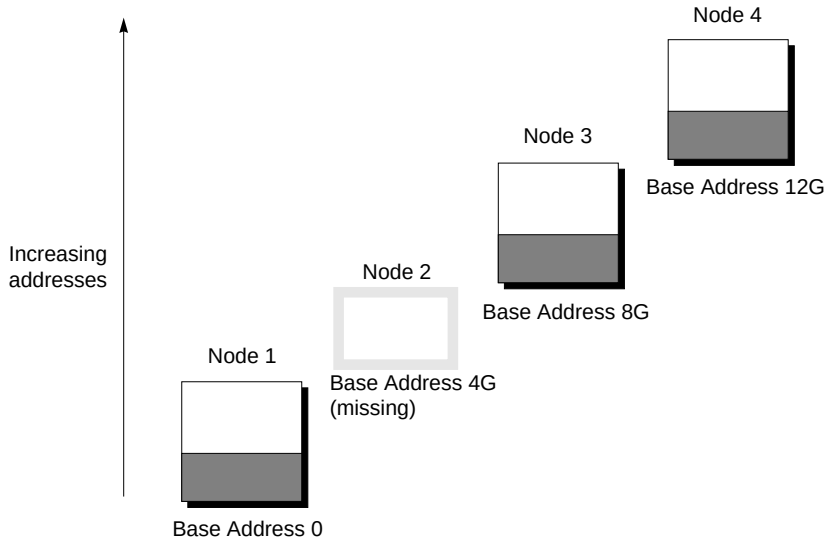

Figure 2-2 *Cac* Space Addressing

Calias memory space for all nodes is only accessible by the local processor, at physical address 0 up to *Calias* size.

Hub Special Space (*HSpec*)

The uncached space designated when *Uncached Attribute*=0 is the Hub Special space, or *HSpec* space. It is divided into the following subspaces:

- Backdoor Directory (*BDDir*) space, 1 GB
- Backdoor ECC (*BDECC*) space. 1 GB
- Local Boot (*LBoot*) and Remote Boot (*RBoot*) spaces, 256 MB apiece
- Uncached Alias (*UAlias*) space, 256 MB

Backdoor Directory (*BDDir*)

As shown in Figure 2-1, the uppermost 1 GB of each node's *HSpec* space is a backdoor access to the directory entries and protection information, labelled the *BDDir* space. Backdoor access is available through this separate memory space for diagnostics.

As shown in Figure 2-3, directory entries are indexed by address bits *A[11:00]* and protection information indexed by region bits *R[5:0]*. There is a directory entry for each 4 KB page in the node's main memory, and an Origin system has 64 regions, one for each two-processor node. These directory entries and regions are grouped together as 128 consecutive words.

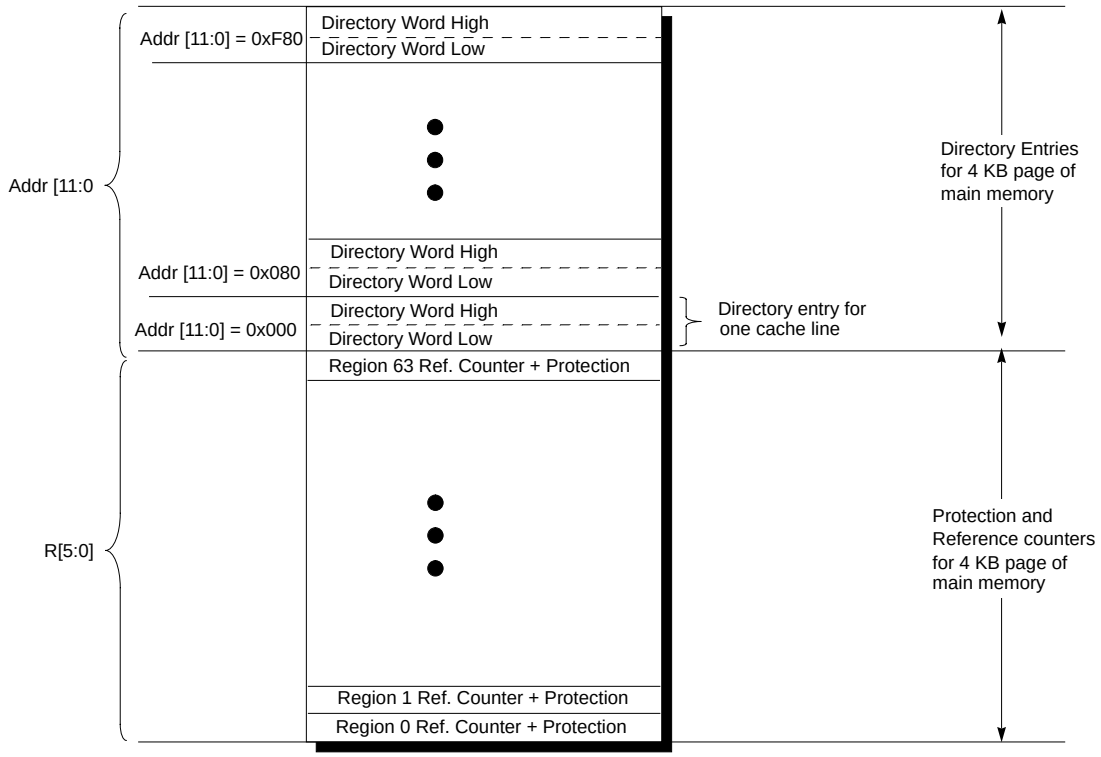

Figure 2-3 *BDDir* Space Map

Directory Widths

The directory comes in two widths, extended and regular.

- The width of the **regular directory** is 16 bits. For configurations up to 32 processors (**32P**), regular directory memory is included in the main memory DIMMs.
- The width of the **extended** or **premium directory** is 48 bits. For configurations larger than 32P, an extended directory memory must be used. The extended directory is contained in dedicated directory DIMMs which are installed in dedicated slots on the Node board.

BDDir Reads

Only doubleword reads are allowed from the *BDDir* space.

The directory word contains an ECC code, and a *BDDir* read returns both the directory information and this ECC code. Read data is carried on bits *[47:0]* for the extended directory, on bits *[15:0]* for the regular directory.

BDDir Writes

Only doubleword writes are allowed to the *BDDir* space.

Backdoor directory writes contain the ECC code to be written. Write data for an extended directory is on bits *[47:0]*, and on bits *[15:0]* for a regular directory.

If bit 63 of the write data is cleared (a zero), the ECC value generated by the Hub chip is written into memory. If bit 63 is set (a one), the ECC write data is used in the backdoor write. Bit 63 should only be set to force an incorrect ECC value into the directory.

Indexing a Directory Entry

The *HSpec* address *A[39:32]*, 1, 1, *A[31:12]*, 1, *A[11:7]*, *DWBit*, 000 is used to index the directory entry for a given 128-byte cache line *A[39:7]* in *M Mode*. This mapping is shown in Figure 2-4.

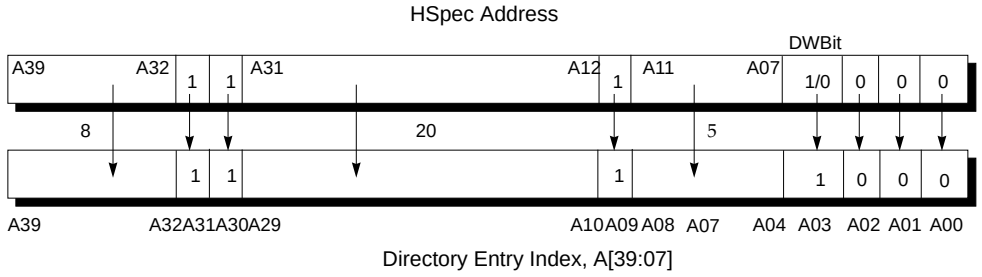

Figure 2-4 *HSpec* Address Used for Accessing a Directory Entry in *M Mode*

Accessing the Regions, *R[5:0]*

When a system has 128 or fewer processors, a **region** is defined as a single node. If a system has more than 128 CPUs, a region consists of 8 nodes.

Accesses to the protection and page reference count for a given region, *R[5:0]*, and the page *A[39:12]*, in *M Mode*, are made using the *Hspec* address bits *A[39:32]*, 1, 1, *A[31:12]*, 0, *R[5:0],* 000. This mapping is shown in Figure 2-5.

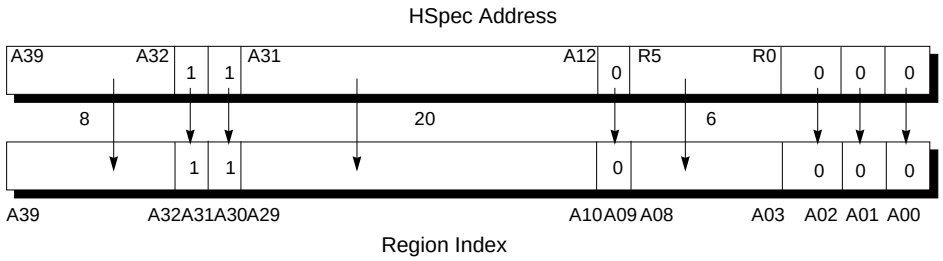

Figure 2-5 *HSpec* Address for Accessing the *M Mode* Protection and Page Reference Count

On backdoor writes to the protection and page reference entries, bit 63 is ignored, since these entries do not use ECC.

Backdoor ECC (*BDECC*)

As shown in Figure 2-1, the 1 GB address space below *BDDir* is the Backdoor ECC space, *BDECC*. Memory ECC entries are read from and written to this space.

Accessing the backdoor ECC for a given address *A[39:3]* is done by mapping the following *HSpec* address: *A[39:32]*, 1, 0, *A[31:5]*, 0/1, *A[4:3]*.

A[4:3] is an endianness indicator; for Little Endian, *A[4:3]* is replaced with its complement. Backdoor ECC mapping is shown in Figure 2-6.

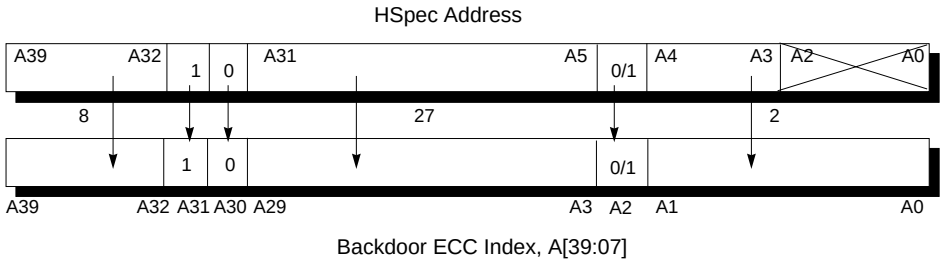

Figure 2-6 *HSpec* Address Used for Accessing the Backdoor ECC (*BDECC*)

BDECC Reads

A read of the *BDECC* retrieves the 8-bit ECC codes for 4 consecutive doublewords. Reads can be byte, halfword, or word operations. 64-bit doubleword operations are supported by the hardware, but simply return the same data duplicated on the high and low 32 bits.

The hardware does not use address bit 2 when designating the ECC values to be read or written; this means the 32-bit data can be accessed with two addresses, one with bit 2 set and one with bit 2 clear.

BDECC Writes

Byte, halfword, and word operations may also be used to perform *BDECC* writes. The hardware does not use address bit *A[2]* when designating the ECC values to be read or written; this means the 32-bit data can be accessed with two addresses, one with bit *A[2]* set and one with *A[2]* clear.

Local and Remote Boot Spaces (*LBoot* and *RBoot*)

All nodes have an *Rboot* space allocated, however only Node 1 has an *LBoot* space. Refer to Figure 2-1 for the locations of *LBoot* and *RBoot* space.

Accesses to *LBoot* space map to the node's local boot PROM and any other devices that are associated with the local boot PROM. The R10000 boot vector resides in *LBoot* space.

The 256 MB *RBoot* space is located at an offset of 768M within each node. This space is an alias for the *LBoot* PROM of that node, and is intended to be used for diagnostics by remote processors.

Flash PROM (FPROM)

The 1 MB Flash PROM is accessible in both the *LBoot* and *RBoot* spaces. The Hub allows three operations to the Flash PROM:

- doubleword (64-bit) reads
- word (32-bit) reads
- doubleword writes (for which only the lower byte is used)

The Hub sequences reads of the PROM in Big Endian order. Doubleword reads are the same for both Big and Little Endian modes, but the layout of the FPROM is different for word reads in Big Endian and Little Endian modes, as shown in Figure 2-7. In Little-endian mode, all word pairs are swapped. The Hub sequences reads of the Flash PROM in Big Endian mode only.

The Hub does not decode the *LBoot* or *RBoot* space for address errors, so the FPROM can be accessed in any window that is mod 0 of the FPROM size.

Read and write images of the FPROM are shown in Figure 2-7.

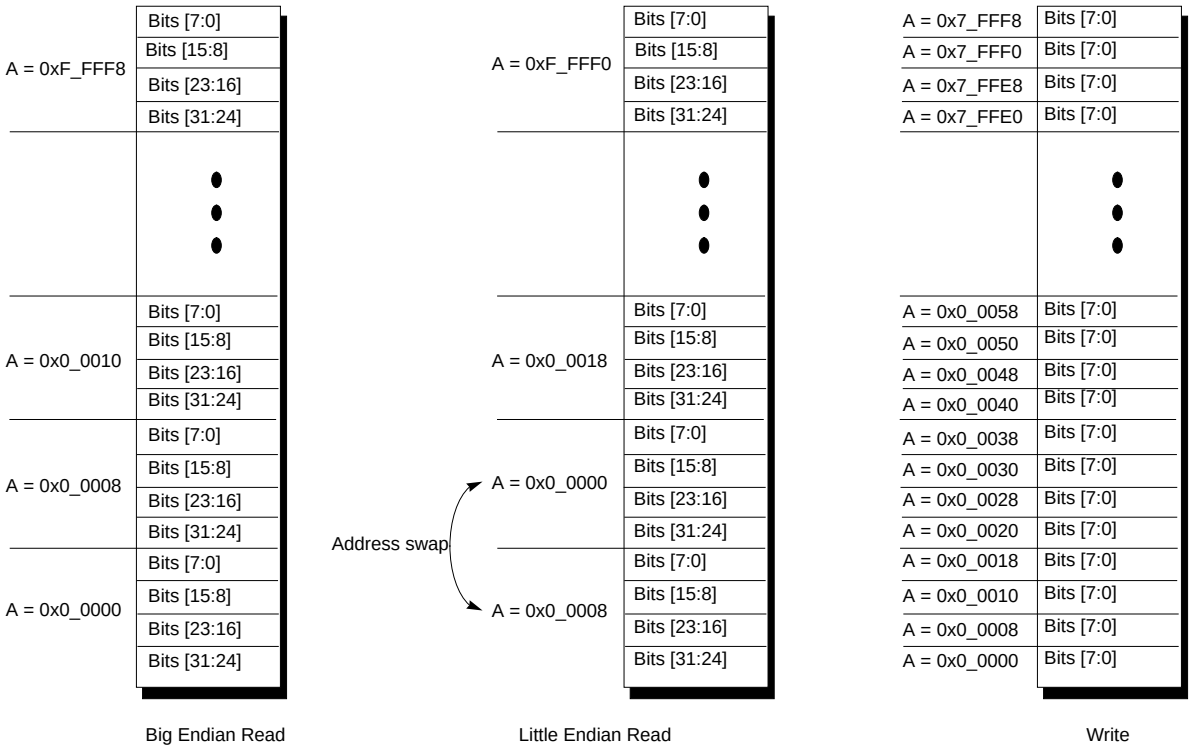

Figure 2-7 Read/Write Organization of the Flash PROM

FPROM Reads

Reading the FPROM requires a 1 MB window in *LBoot* or *RBoot* space.

FPROM Writes

Writing the 1-MB FPROM requires an 8 MB window in *LBoot* or *RBoot* space, since writes are doubleword operations with respect to the processor, but only byte operations into the FPROM.[1]

[1] The write address given is the final write address of the Flash PROM programming sequence.

Uncached Alias (*Ualias*)

With the exception of Node 1, the lowest 768 MB of each node's *HSpec* space is reserved; refer to Figure 2-1 for an illustration of this mapping.

In Node 1, the lowest 256 MB of the *HSpec* space is used as an alias for uncached access to low local memory. This uncached alias space is labelled *Ualias*, and the low local memory it aliases is in *Uncac* space (the *Uncached Attribute*=11_2).

Similar to *Calias* space, an access by Node 1 to *Ualias* indexes the lowest 256 MB of its *Uncac* space. An access by Node 2 to *Ualias* addresses the lowest 256 MB of its *Uncac*, which has a base address of 4GB. An access by Node 3 to *Ualias* addresses the lowest 256 MB of its *Uncac*, which starts at 8GB.

However, *Ualias* differs from *Calias* in regards to the two R10000 processors, processor 0 (A) and processor 1 (B), on its Hub.

- For processor 0 (A), the address mapping is done as normal: accesses to address 0 map to address 0, address 64K maps to 64K.
- However, when processor 1 (B) accesses the lowest 64K, its access starts at base address 64K, and when it accesses the next 64K, its access starts at base address 0K; that is, the lowest and next lowest 64 KB pages are flipped.

The vector for the cache error handler is located in *Ualias* space. The address swapping described above provides the cache error handler of each processor with an individual portion of memory it can access relative to *r0*, to store processor state. The swapping is shown in Figure 2-8.

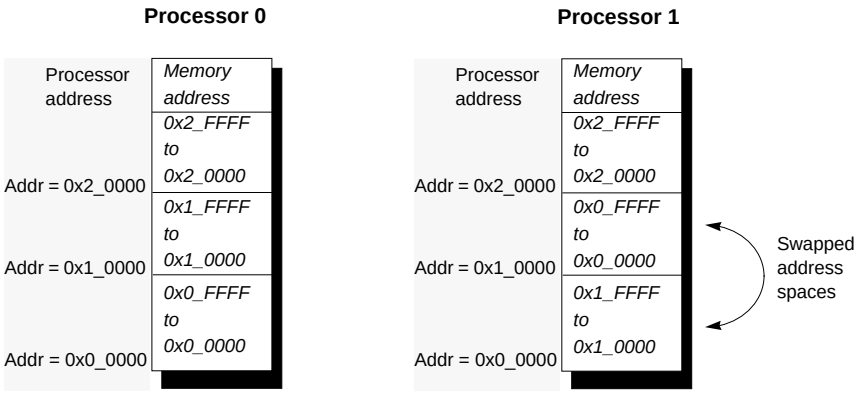

Figure 2-8 Processor 0/1 Address Mapping in *Ualias*

Accessing I/O Space (*IO*)

IO space is accessed by an uncached operation with an *Uncached Attribute* of 01_2 (refer to Table 2-1 for the *Uncached Attributes*).

Each node's portion of the *IO* space is divided into seven 512-MB subspaces and two 256 MB subspaces, as shown in Figure 2-1.

- The lowest 256 MB subspace is referred to as a "little window" (*LWin*) into *IO* space. *LWin* consists of sixteen 16-MB direct-mapped windows, one for each of the 16 XIO devices that can attach to Node (`16*16 MB = 256 MB`).
- The next highest 256 MB subspace is reserved, and aliases to *LWin* space.
- The remaining seven 512-MB spaces within *IO* are referred to as "big windows," and are labelled *BWin1* through *BWin7*. Each *BWin[7:1]* provides a 512 MB window that can be mapped to a 512 MB-aligned block of any I/O device's address space.

I/O Space is described further in Chapter 3 and Chapter 4.

Memory Special Space (*MSpec*)

Uncached operations with an *Uncached Attribute* of 10_2 (refer to Table 2-1 for a list of *Uncached Attributes*) perform fetch-and-op operations on the memory space referred to as memory special, or *MSpec*. Any page used for a fetch-and-op cannot also be used for a regular cached access. The backing store for *MSpec* is regular memory.

As shown in Figure 2-1, *MSpec* is a flat address space.

Uncached Space (*Uncac*)

Uncached operations with an *Uncached Attribute* of 11_2 (refer to Table 2-1 for a list of *Uncached Attributes*) perform uncached reads and writes of the memory referred to as uncached, or *Uncac*. As shown in Figure 2-1, *Uncac* is a flat address space.

Processor View of I/O Space

I/O uncached space is divided into 256 equal segments of 4 GB each.

Each 4 GB I/O space is divided into seven 512 MB segments (*BWin[7:1]*, a single 256 MB segment (*LWin*), and the remaining 256 MB aliases to *LWin* (see Chapter 2).

The *LWin* segment is further divided into sixteen 16 MB spaces, each directly mapped to one of the sixteen XIO devices, called **widgets**. The *I/O* space map, from the view of the SysAD bus on the processor, is given in Figure 3-1.

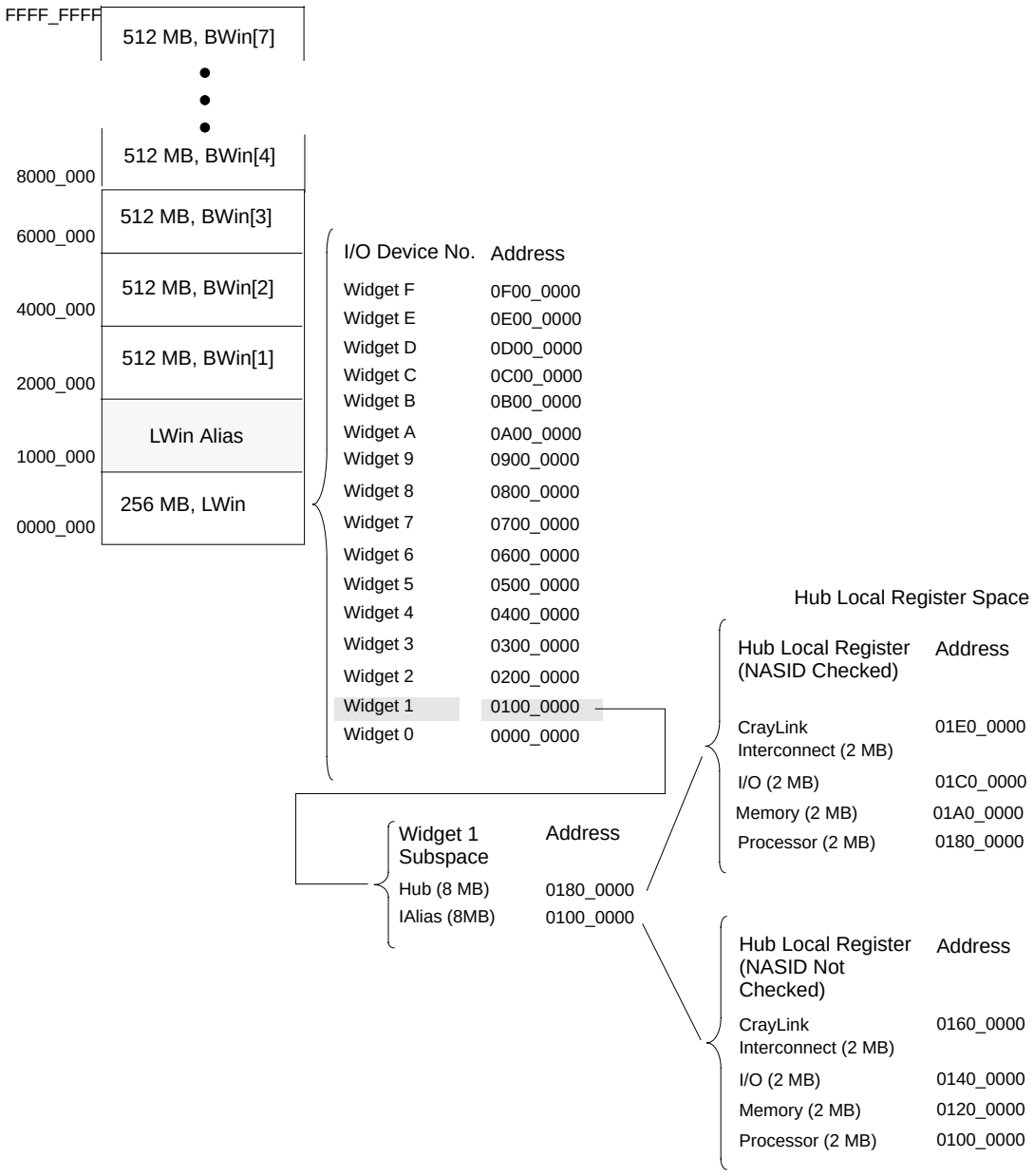

Figure 3-1 I/O Map (Per Node) as it Appears to the Processor

Little Window (*LWin*) Map

Big Windows (*BWin*) are set at 512 MB. *LWin*, however, is set at 256 MB and the remaining 256 MB region immediately above *LWin* aliases to *LWin*.

The *LWin* space contains sixteen 16 MB subspaces which map to the sixteen XIO devices or "widgets." These sixteen subspaces are labelled *Widget 0* through *Widget F*, and their base addresses are shown in Figure 3-1.

Widget 1 space is subdivided into two 8 MB spaces: *Hub* and *IAlias*, and these two 8 MB spaces are each subdivided into four 2 MB regions, mapping to the four Hub ASIC interfaces, as shown in Figure 3-1.

All of these spaces and subspaces are described below.

IAlias and *Hub* Spaces

The Hub's local registers reside within the *IO* uncached space, in *Widget 1* portion of *LWin* space, as shown below in Figure 3-2.

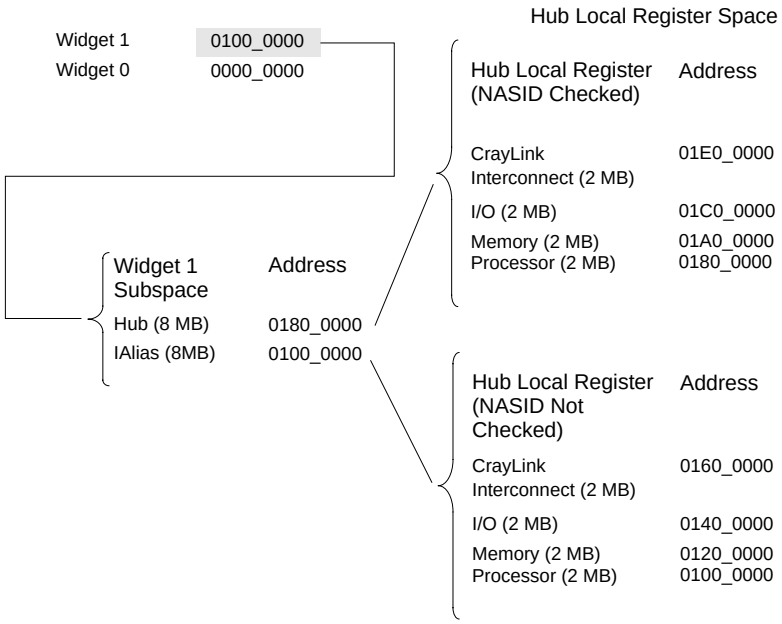

Figure 3-2 Hub Local Register Space

When the NASID (the node ID) is not known — such as at boot time or during diagnostics — it is necessary to unambiguously access all the registers that are locally available to a node through a local R10000 processor.

To accommodate this access, the 16 MB *Widget 1* space assigned to the Hub is further divided into a pair of 8 MB regions, *IAlias* and *Hub*, as shown in Figure 3-2.

- Accesses to the lower 8 MB region labelled *IAlias* are not checked for a NASID. A processor cannot directly address a remote *IAlias* space, which means the *IAlias* space is reserved for the use of processors local to it. *IAlias* space is used to avoid having to load a NASID when running code such as an interrupt handler and communicating with local registers.
- Accesses to the upper 8 MB region labelled *Hub* are checked for a match with the correct NASID. If there is a match, these accesses map to the *IAlias* space. Since the NASID is checked, remote accesses to the *Hub* space are permissible.

Each 8 MB space is further divided into regions corresponding to the Hub interfaces, as described in the next section.

Hub Local Register Regions

The Hub chip has four major interfaces:

- the IO interface (*II*)
- the CrayLink Interconnect (*NI*)
- the memory/directory interface (*MD*)
- the processor interface (*PI*)

Each 8 MB *IAlias* and *Hub* subspace is further divided into four 2 MB regions, as shown in Figure 3-1 and Figure 3-2. These four 2 MB regions represent local register spaces for the four interfaces of the Hub (NI, PI, MD, II), shown in Figure 3-3. There are a small number of Hub crossbar registers, and these reside in the MD region. The next section describes the conditions that must be met for accessing these local register regions.

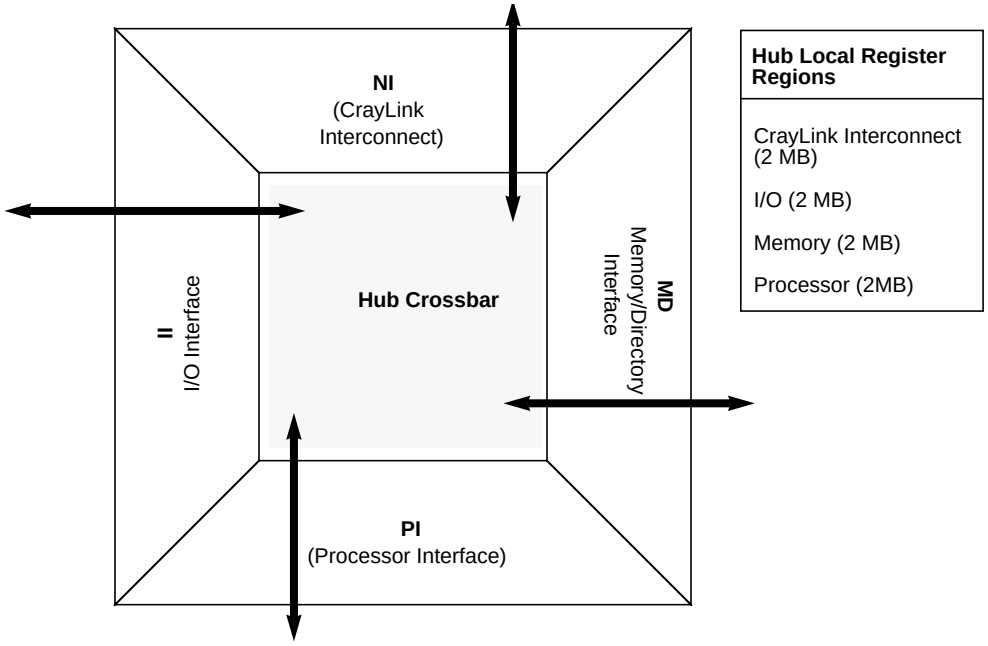

Figure 3-3 Hub Chip Interfaces

Accessing the Hub Local Registers

In accessing the Hub's local registers, four separate conditions must be met before a target (*NI, PI, MD, II*) can recognize a valid address:

1. The access must be to *LWin*; that is, *A[31:28]*, must be zero.
2. *A[27:24]* must be equal to 0001 (indicating *Widget 1* space in *LWin*).
3. If *A[23]* is set (indicating an access to *Hub* space), the *Source[10:2]* field of the incoming request must match this node's *NASID*. If *A[23]* is clear, *Source[10:2]* is ignored, allowing access to the *IAlias* space.
4. *A[22:0]* is decoded to determine which 2 MB local register space is being addressed (CrayLink Interconnect, I/O, memory, or processor).

SysAD[2:0] are converted to byte-enable bits to allow Little and Big Endian accesses that are not doubleword aligned.

Figure 3-4 shows *LWin* address mapping in *M Mode*.

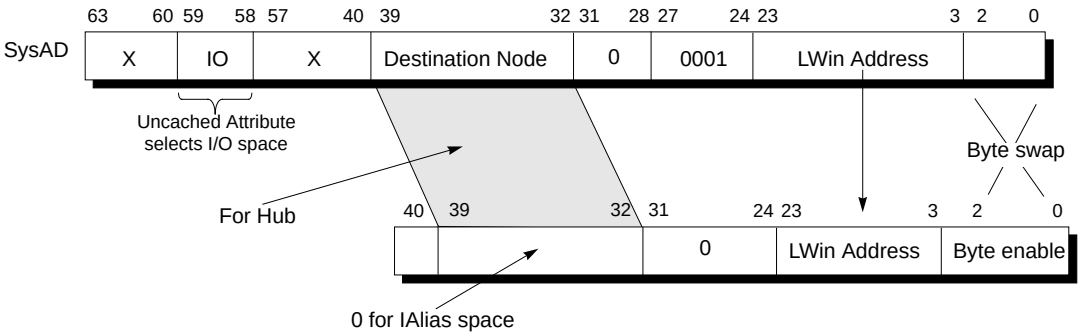

Figure 3-4 Accessing the Hub Local Registers in *LWin* Space

The kernel transparently converts the Hub and XIO addresses, and handles Node register accesses.

Big Window (*BWin[7:1]*) Map

The remaining seven big windows can be mapped to any region within the XIO address space through a translation table. There are always only seven big windows available per node. In *M Mode*, up to 4 GB of addressable space is allotted to each node, so each *BWin* is 512 MB.

In the present implementation of Origin2000, *Widget 1* space hosts the Hub registers and currently only widgets 0, and 8 through F are addressable in the Crossbow. Their addresses are shown in Figure 3-1.

XIO uses a 48-bit address. When converting a 40-bit SysAD bus address to XIO format, only 5 of the offset bits are programmable. In M Mode a 16 GB address per widget is provided to the processor. This space is sufficient for the current range of devices; the 512 MB big window can be located anywhere within the 16 GB window available per widget.

Figure 3-5 shows the address mapping of the big windows space, *BWin*. *M Mode* addressing capability per widget is 16 GB, *A[33:0]*.

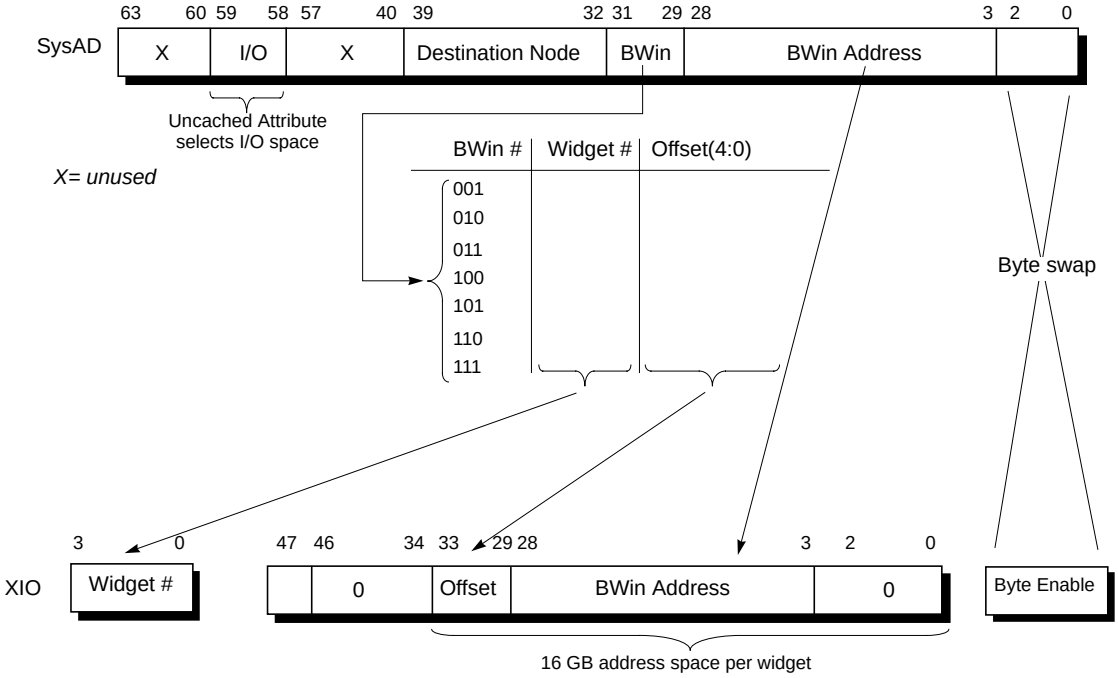

Figure 3-5 *BWin* Space Access in M Mode

Chapter 4

XIO View of I/O Space

The XIO bus gives a widget either a memory view or an I/O view of the Origin2000 system. The *XIO_Address[47]* bit selects the type of view (I/O, memory) seen by the widget.

- If *XIO_Address[47]* is set, the XIO accesses the I/O view. This view is primarily designed to allow access to the Hub interrupt registers.
- If *XIO_Address[47]* is cleared, the XIO accesses the memory view.

In normal operation, DMA operations use the memory view.

XIO Memory View

The XIO memory view is similar to the memory address map available to a processor, in that XIO's 40-bit memory space is divided into equal regions per node. The Hub ignores *A[46:40]* in the memory view, effectively creating an alias space with these bits.

The *M Mode* address map for the XIO memory view is shown in Figure 4-1.

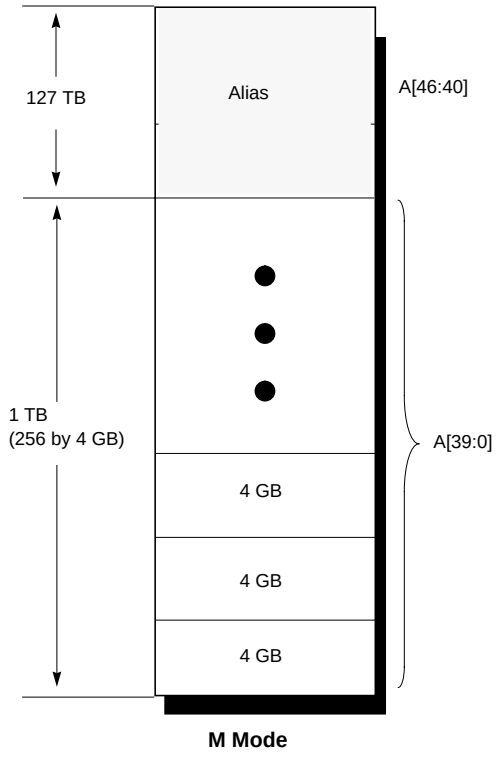

Figure 4-1 XIO Memory View Address Map

M Mode memory view address mappings are shown in Figure 4-2. *XIO_Address[47]* is always set to a 0, indicating a memory view.

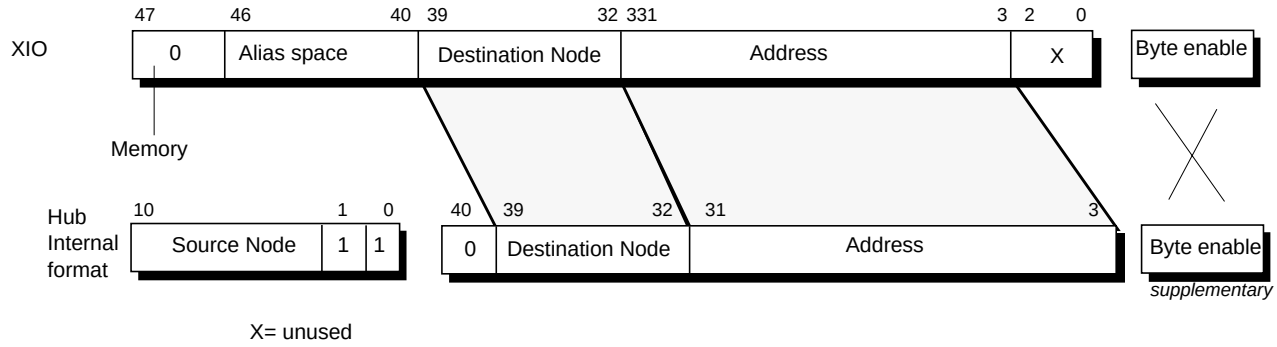

Figure 4-2 Memory View Access in *M Mode*

XIO IO View of Origin2000

The XIO's IO view of the Origin2000 system is slightly different from the XIO memory view. The 9-bit *NASID* is merged with *XIO_Address[46:38].*[1] This means the 4-GB Hub region occurs somewhere within a 256 GB space.

The Hub region is indexed by the sum of *NodeID* plus *XIO_Address[46:38]*, and offset is *XIO_Address[31:0]*, as is shown in Figure 4-3.

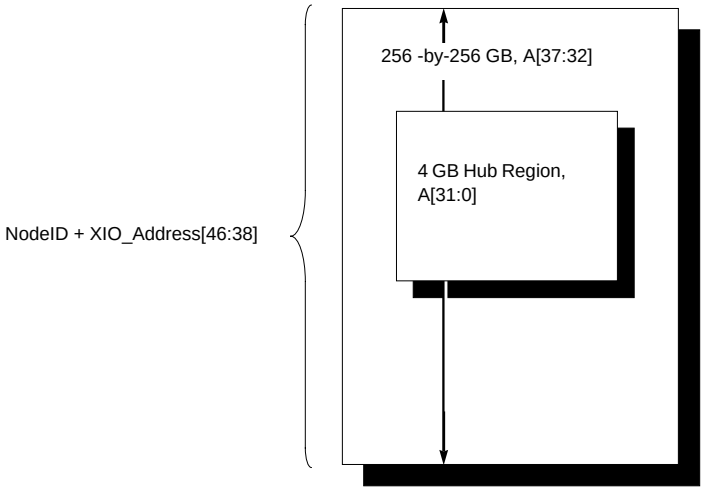

Figure 4-3 *M Mode* Addressing in an XIO IO View

A series of Hub regions within their encompassing 256 GB spaces are shown in Figure 4-5.

[1] Software has the responsibility for zeroing bit[46], the upper bit of the NASID index.

As shown in Figure 4-4, the Hub ignores *XIO_Address[37:32]*. *XIO_Address[47]* is always set to a 1, indicating an IO view, as shown in this figure.

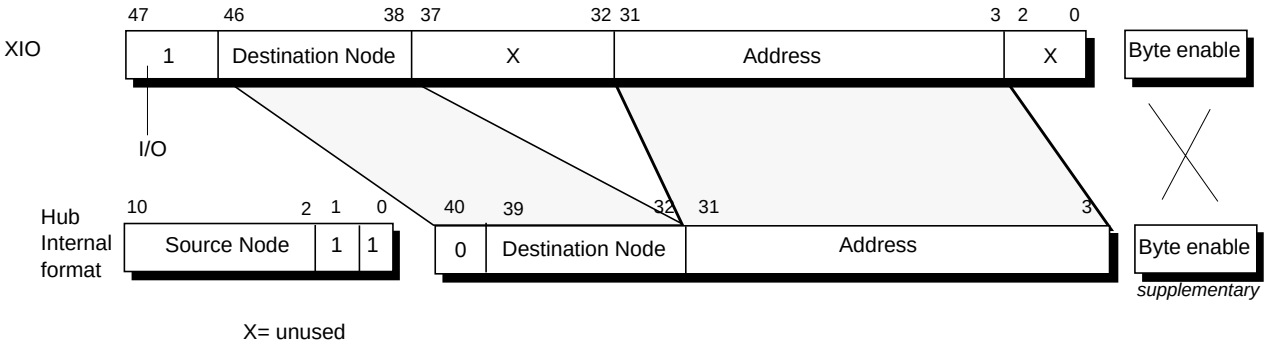

Figure 4-4 I/O View Access in M Mode

The remainder of the XIO address is similar to the memory view model, which means the view within a node is similar to the processor's view of memory, as shown in Figure 4-5.

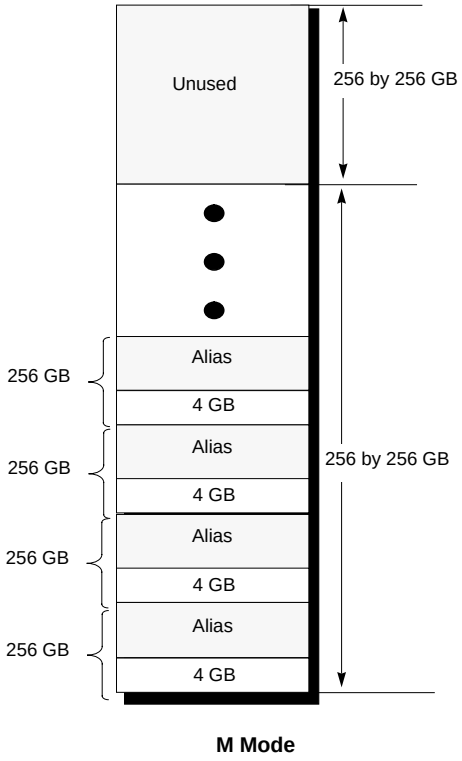

Figure 4-5 XIO IO View Map

Index

H

I

L

M

N

O

P

Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-3410-001.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:
  - On the Internet: techpubs@sgi.com
  - For UUCP mail (through any backbone site): *[your_site]*!sgi!techpubs
- To **fax** your comments (or annotated copies of manual pages), use this fax number: 650-932-0801
- To send your comments by **traditional mail**, use this address:

  Technical Publications
  Silicon Graphics, Inc.
  2011 North Shoreline Boulevard, M/S 535
  Mountain View, California 94043-1389