AR3-00

Sun-4D Architecture

Michel Cekleov, Jean-Marc Frailong, Pradeep Sindhu

Written: January 30, 1990 **Version:** 1.4

Revised: June 22, 1992

Abstract: This description of the Sun-4D architecture is intended for kernel and diagnostic programmers. It discusses the memory model and its implications, the address spaces organization, the processor unit architecture, the main memory organization, the I/O subsystem architecture, interrupt handling, reset, and interactions with JTAG.

An important goal of this document is correctness. Please report any error, omissions or oversights immediately so they can be corrected in future revisions.

Keywords: Architecture, Programming Model, Memory Model, Viking, Cache, Main Memory, I/O, Boot, Sunness, JTAG, Interrupts

Maintained by: Michel Cekleov

sun microsystems

Revision History

Revision 0.1 **12/22/89**

Limited distribution for review purposes

Revision 1.0 **1/30/90**

First general release
Viking chapter removed

Revision 1.1 **4/25/90**

Changed CC Error register format, error bits made write-one-to-clear
Added internal errors of BW
Clarified BW time-outs
Added bit to BW control register
JTAG control register & command encoding changed
Added LEDs status bits in BootBus Status_2 register, modified register encoding
Specified MQH SIMM ID register format
Added Refresh Enable to MQH Control & Status register
Documented change in ECC structure for Memory subsystem
Changed polarity of ECC enable bit in MQH Diagnostic register
Explained memory initialization & scrubbing
Clarified stream write DVMA model
Added RSB bit to Stream Buffer Control registers in SBI
Specified Loopback mode for SBI
Board slot number of control board changed form 0xA to 0xF
Additional status bits in BARB shadow ring 0
Changed order of bits in CARB shadow ring 0
Added JTAG specification for CC
Added INIT bit to MQH DynaBus Control and Status register

Revision 1.2 **6/12/90**

Modified semantics of Local Software Reset (SI bit of CC reset register)
Modified INTSID for MQH & SBI errors
Added TLD mode bit to DynaBus CSR in BW & IOC
Changed encoding of IOC cache flags
Added RDTOL to IOC DynaBus CSR
Added IOC internal errors
Clarified MQH ECC error interrupts, added SWAP to ECC error registers
CC reference counters modified

Revision 1.3 **3/29/91**

Various sections: explanations related to dual-processor system board
CC: Warning that CC LVL15 is level-sensitive
CC: Warning that Local reset disables Viking parity, but not CC parity
BW: Added chip test bits in DCSR

BootBus: Reorganization of the control & status registers, introduction of dual-processor board BootBus
MQH: Clarification of address generation
MQH: Added section on column and row address generation
MQH: Specified RequestDelay and MCRAM information for Mitsubishi & TI SIMMs
MQH: Updated Client Device Errors table
IOC: Cache reduced to 4 lines instead of 8
IOC: Added chip test bits in DCSR
SBI: IMS bit removed, Interrupt Diagnostic register now write-only
SBI: BA64 and BA32 are now supported
SBI: LBE bit is cleared on reset
SBI: Writing a 0 into FWB & IRB bits has no effects, streaming buffers need to be initialized even if not used
Error: Clarified early vs. late store errors
Error: Documented DeMap time-out error case
JTAG: Scan chain changes to BW, IOC, SBI due to register changes
JTAG: Corrected Viking and CC JTAG Component ID
JTAG: Added scannable registers to system board & control board chains

Revision 1.4 6/22/92

Removed single-processor BootBus description
Renamed document to Sun-4D architecture
Incorporated Scorpion features in all chapters
Introduction chapter includes a physical description of Scorpion.
Address Spaces chapter has been completely rewritten for clarity
Address tables have been moved to the beginning of each functional units chapters (processor, memory and I/O).
Processor Unit includes BWP description.
Power Supply Failure interrupt has been replaced by the Ethernet Link Test feature on the Scorpion BootBus.
Memory Unit chapter includes MQHP description
More detailed description of refresh variables (RFEN, RCNT).
NV_SIMMs documented as part of the Memory Unit chapter
Programming notes on NV_SIMMs.
Programming notes removed from Reset chapter. Obsolete due to POST FCS Specification.

Table of Contents

Chapter 4

Chapter 1
Introduction

1.1 Scope

This document describes the Sun-4D system architecture. The Sun-4D Architecture defines the programmer's model of systems build around Dynabus. It is intended to be the *reference* for software and firmware. The first two Sun hardware platforms based on this architecture are **SunDragon** and **Scorpion**. The Sun-4D Architecture provides a great deal of flexibility. Systems based on this architecture may include one, two or four Dynabusses, from 1 to 20 processors and up to 40 SBus boards for I/O. Only built-in I/O devices are described in this document. The usual I/O interfaces found in other Sun systems like SCSI, Ethernet and VME are not described here as they are all attached to an SBus through a combination of I/O Cache and SBus Interface chips. They are described in companion documents.

Although, this document is intended to encompass all system implementations using the same Dynabus chip set, it focuses specifically on SunDragon and Scorpion.

This document also describes the diagnostic features that are accessible from processors. However, diagnostic interfaces that must be accessed from a remote service processor are not described here.

1.2 Hardware Architecture Overview

Sun-4D defines a shared memory multiprocessor architecture. Sun-4D systems are composed of three types of units: the Processor unit, the Main Memory unit and the I/O unit. All these units are interconnected through one, two or four Dynabus(ses). Dynabus is a high-speed, consistent, packet-switched bus [4]. In a given implementation, the total number of units and the number of units of a given type is dictated mainly by the following considerations: bandwidth of the Dynabus(ses), overall computing throughput and I/O connectivity.

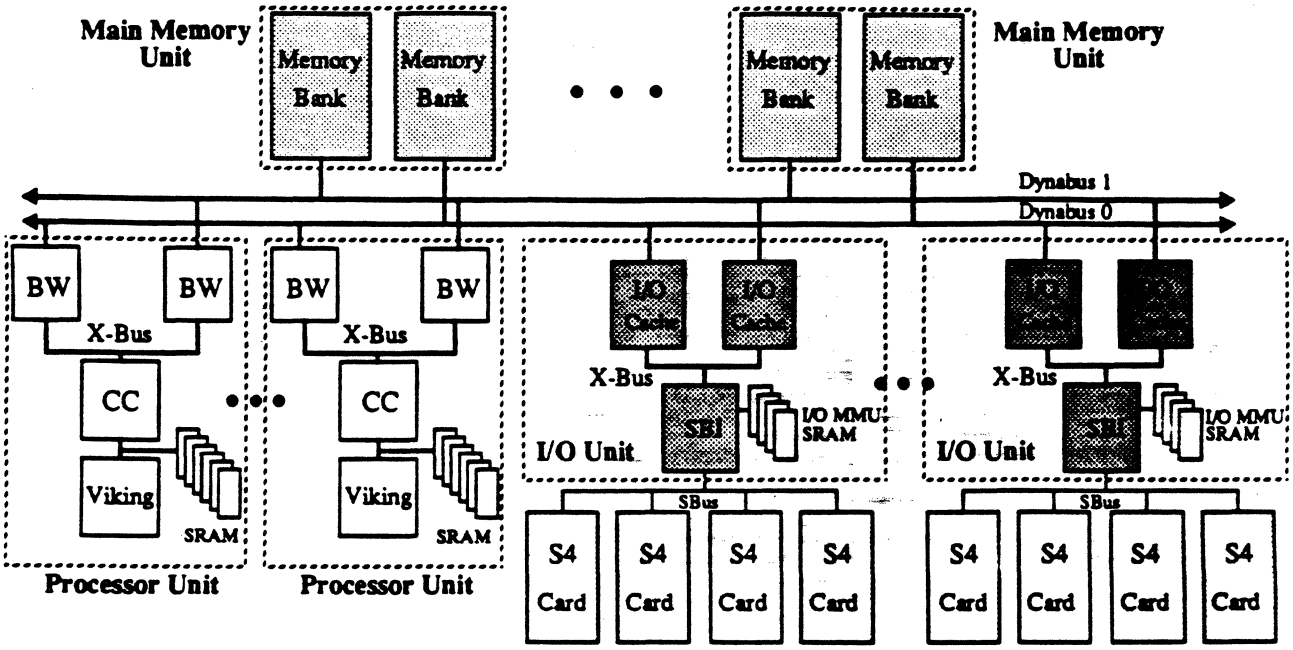
1.2.1 SunDragon

SunDragon is built around two Dynabusses, and all units have two Dynabus ports. Physical addresses are interleaved on a 256-byte boundary across the two busses. For practical purposes, the two Dynabusses appear to programmers as a single bus with twice the bandwidth.

SunDragon can have between 1 and 20 Processor units, namely a SunDragon system can support between 1 to 20 Viking processors. SunDragon can include between 1 and 10 Main Memory units.The minimum main memory capacity is 64 M-bytes (using 4 M-bit DRAM chips) and the maximum 5 G-bytes (using 16 M-bit DRAM chips).

SunDragon uses SBus as I/O bus. A SunDragon system can have between 1 and 10 SBus (i.e. 1 to 10 I/O units), each supporting four slots.

The Figure below illustrates how the various units are interconnected.



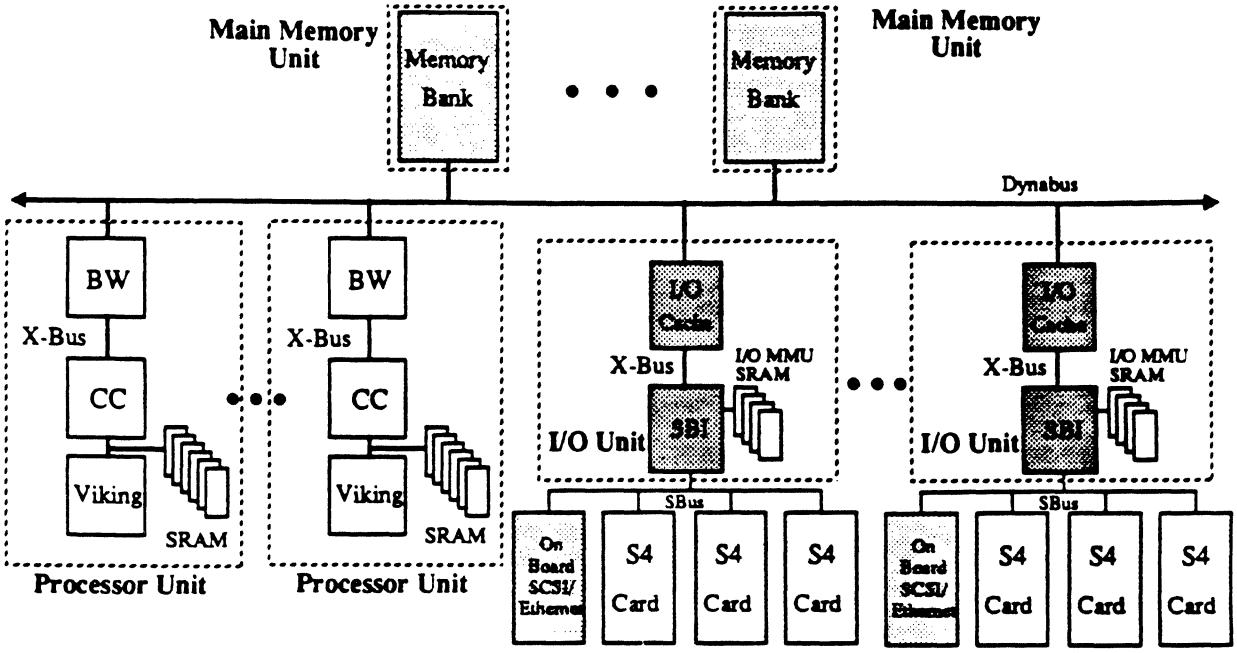
SunDragon Logical Organization

1.2.2 Scorpion

Scorpion is based on the same chip set as SunDragon, but the system is more restricted in term of flexibility and expendability. Scorpion is built around a single Dynabus. Because of this single Dynabus architecture, all Processor, Memory and I/O units in Scorpion have a single Dynabus interface.

A Scorpion system can include 1 to 8 processor units, 1 to 4 I/O units and 1 to 4 main memory units. Therefore, Scorpion can support between 1 to 8 Viking processors, from 1 to 4 SBus and a main memory capacity between 32 M-bytes and 2 G-bytes. Each I/O unit interfaces an SBus with three slots. The fourth slot is used by an on-board SCSI and Ethernet interface.

The Figure below describes the Scorpion system logical organization:



Scorpion Logical Organization

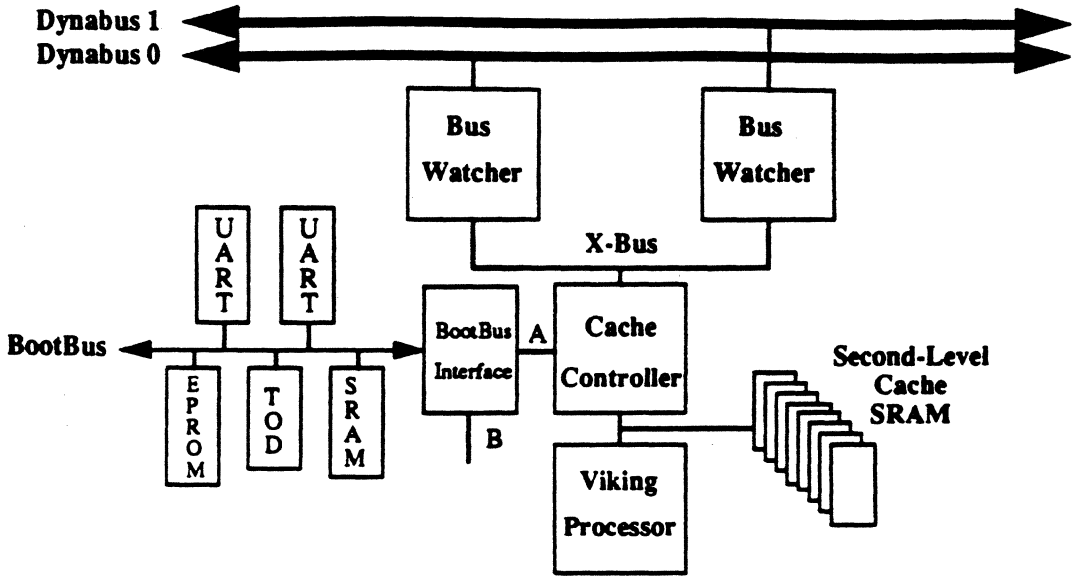
1.2.3 Processor Unit

Each processor unit consists of a Viking processor, a second-level cache and the Dynabus interface(s). The Viking processor chip includes: an Integer Unit (IU), a Floating Point Unit (FPU), a SPARC Reference Memory Management Unit (SRMMU), a 16 K-byte Data Cache and a 20 K-byte Instruction Cache. Viking is fully compatible with the SPARC architecture [1].

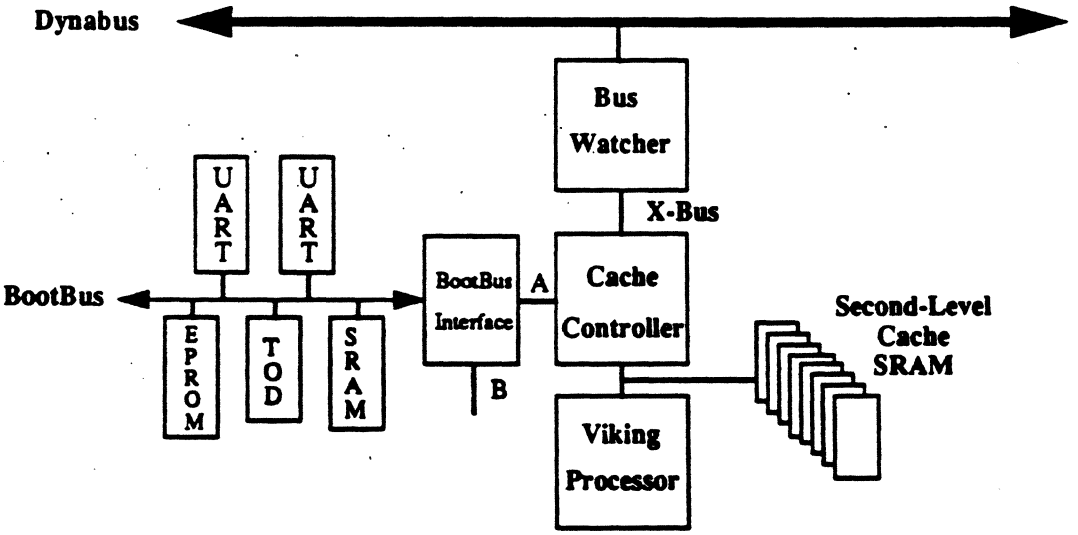
The second-level cache is a one M-byte, direct-map, physical address cache. It consists of a Cache Controller chip (CC), eight 128 Kx9 SRAM chips and either one or two Bus Watcher chips (BW). Each Bus Watcher links the Cache Controller to a Dynabus. In SunDragon, the processor unit comprises two Bus Watchers in order to interface the Cache Controller to the two system Dynabusses. In Scorpion, the processor unit includes only a single Bus Watcher. The Bus Watcher(s) and the Cache Controller cooperate to support the cache consistency protocol. All processor caches in Sun 4-D systems are kept consistent by the hardware.

An 8-bit local bus, called the BootBus, is supported by the Cache Controller. It is used to attach I/O devices that comprise the "Sunness" part of a processor unit, including the EPROM used for bootstrap and Power-On Self Tests. A single BootBus may be shared among multiple processor units. For packaging reasons, two processor units share the same BootBus both in SunDragon and Scorpion.

The Figures below illustrate how the main components of the processor unit are interconnected.



SunDragon Processor Unit



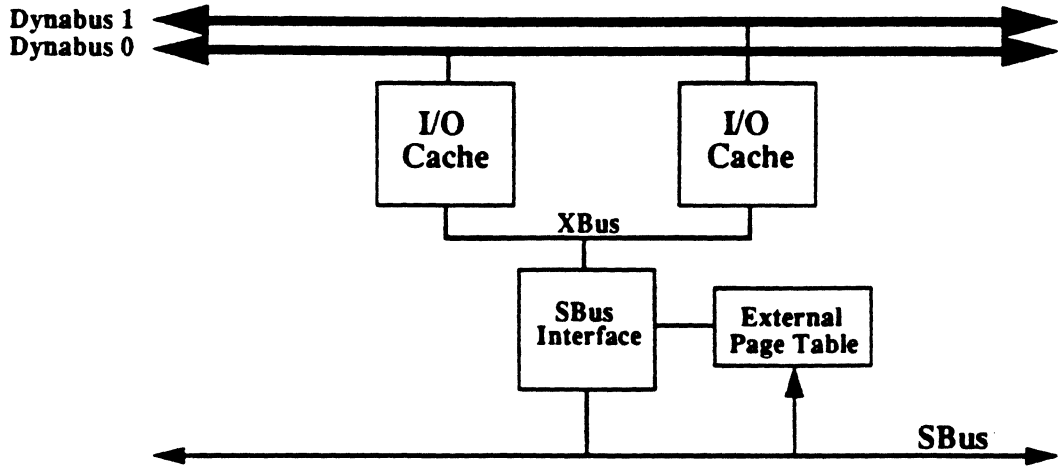
Scorpion Processor Unit

1.2.4 I/O Unit

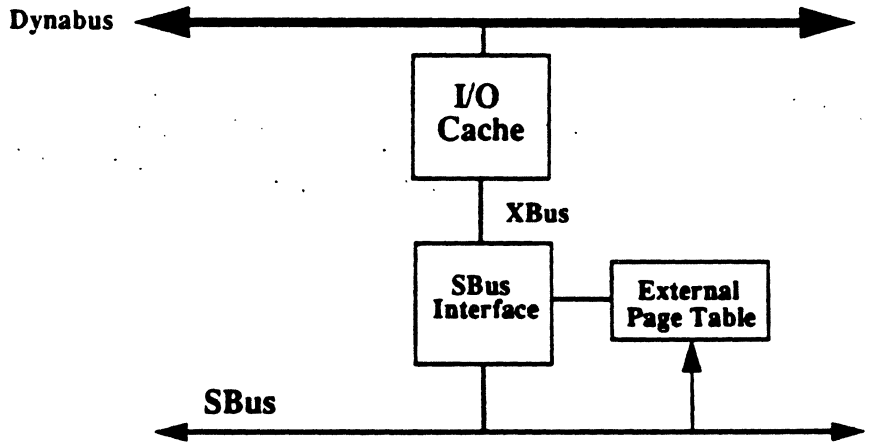
All "regular" I/O devices are attached via SBus. In SunDragon, an SBus is interfaced to both Dynabusses through an SBus Interface chip (SBI) and two I/O Cache chips (IOC). In Scorpion, the I/O units comprises an SBus Interface chip and a single I/O Cache chip. IOC chips contain a small I/O cache that is kept consistent with the processor caches. SBus addresses are translated into memory and I/O addresses

with an I/O MMU dedicated to this SBus. The I/O MMU is implemented with an external SRAM array, called the External Page Table (XPT), which is controlled by the SBI. Each SBus provides 4 SBus slots.

The Figures below describes how the components of the I/O unit are interconnected.



SunDragon I/O Unit



Scorpion I/O Unit

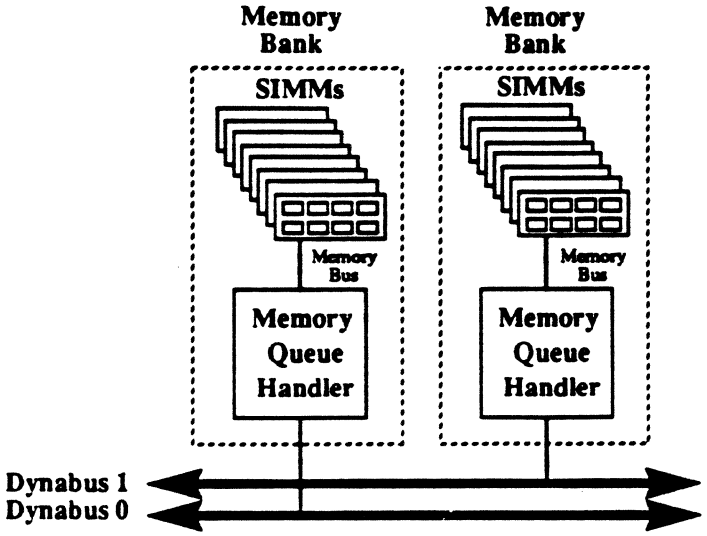
1.2.5 Main Memory Unit

The main memory in SunDragon is interleaved across both Dynabusses on a 256-byte boundary. A SunDragon main memory unit consists of two memory banks, one per Dynabus. A memory bank is a DRAM memory array controlled by a Memory Queue Handler chip (MQH). A SunDragon main memory unit can have a memory capacity between 32 M-bytes (with 4 M-bit chips) and 512 M-bytes (with 16 M-bit chips.

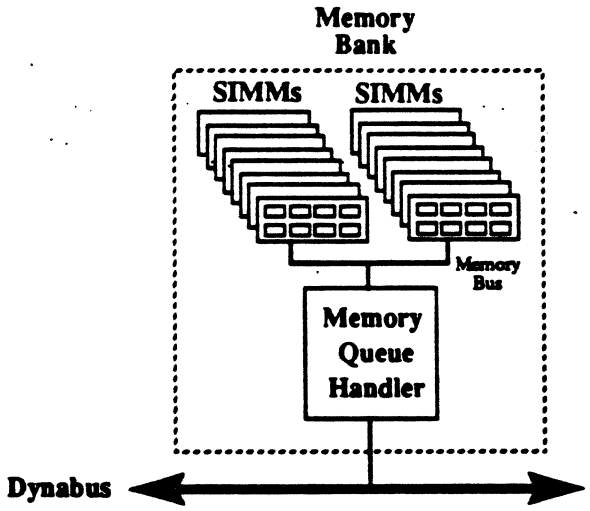
A Scorpion main memory unit consists of a single memory bank. As in SunDragon, a memory bank consists of a DRAM array controlled by a Memory Queue Handler

chip. In Scorpion, a memory array associated with an MQH is twice the size of the memory array of a SunDragon memory bank. Therefore, a Scorpion memory unit has the same capacity as a SunDragon main memory unit (between 32 and 512 M-bytes).

The Figures below describe the components of the Main Memory unit:



SunDragon Main Memory Unit



Scorpion Main Memory Unit

1.3 Implementation Overview

1.3.1 SunDragon Implementation

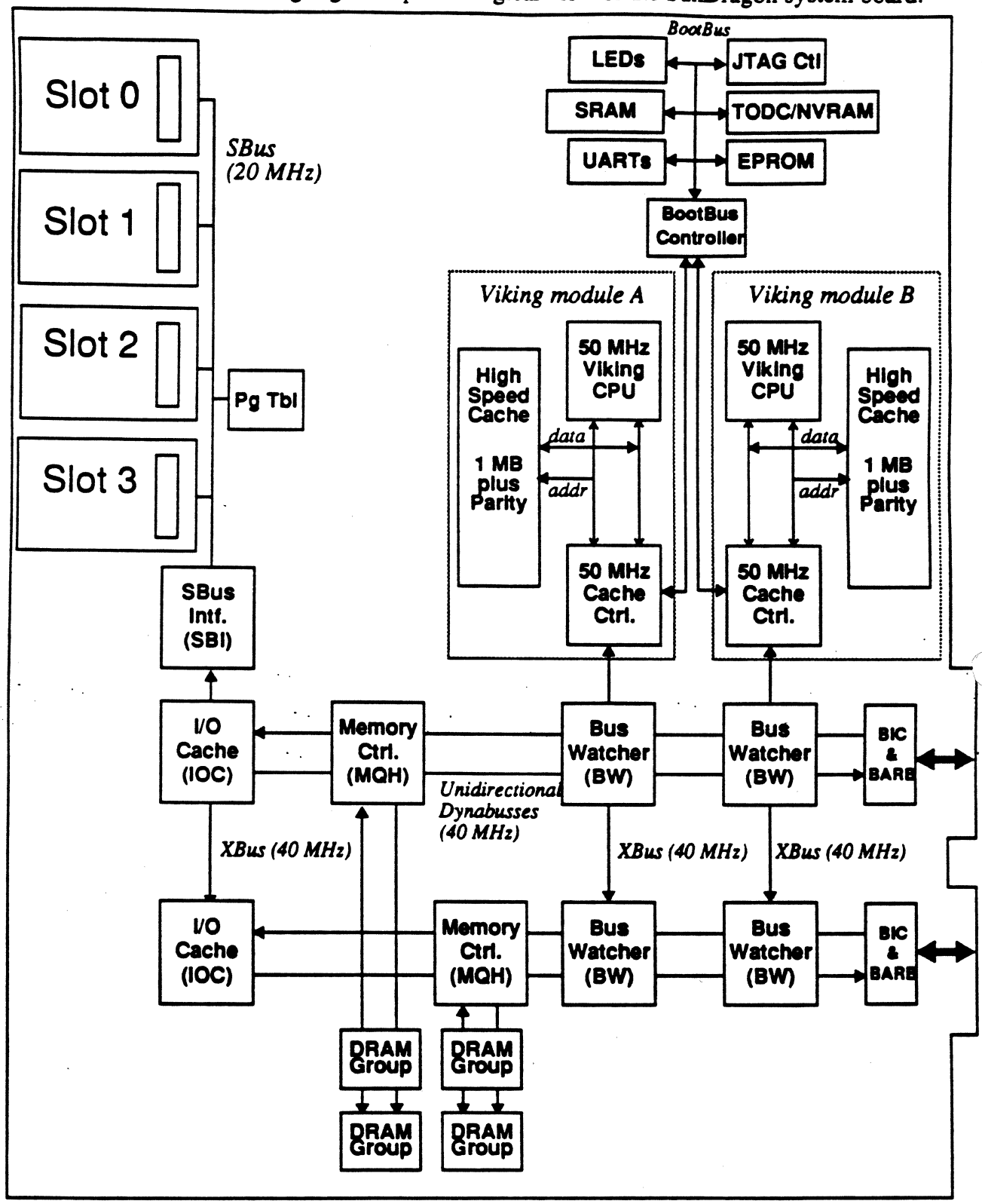
The mix of different types of units on the same board is only the result of packaging considerations. All units of the same type are identically accessible (with the same

access time) whatever the board on which they are located. The main memory access latency is the same for all processors whatever the physical location of the accessed memory bank. The access latency to an I/O device is also the same for all processors regardless of the particular SBus to which the device is attached.

There is only one type of System Board. The SunDragon system board contains two processor units, one main memory unit and one I/O unit. The SunDragon rack can accommodate up to 10 system board.

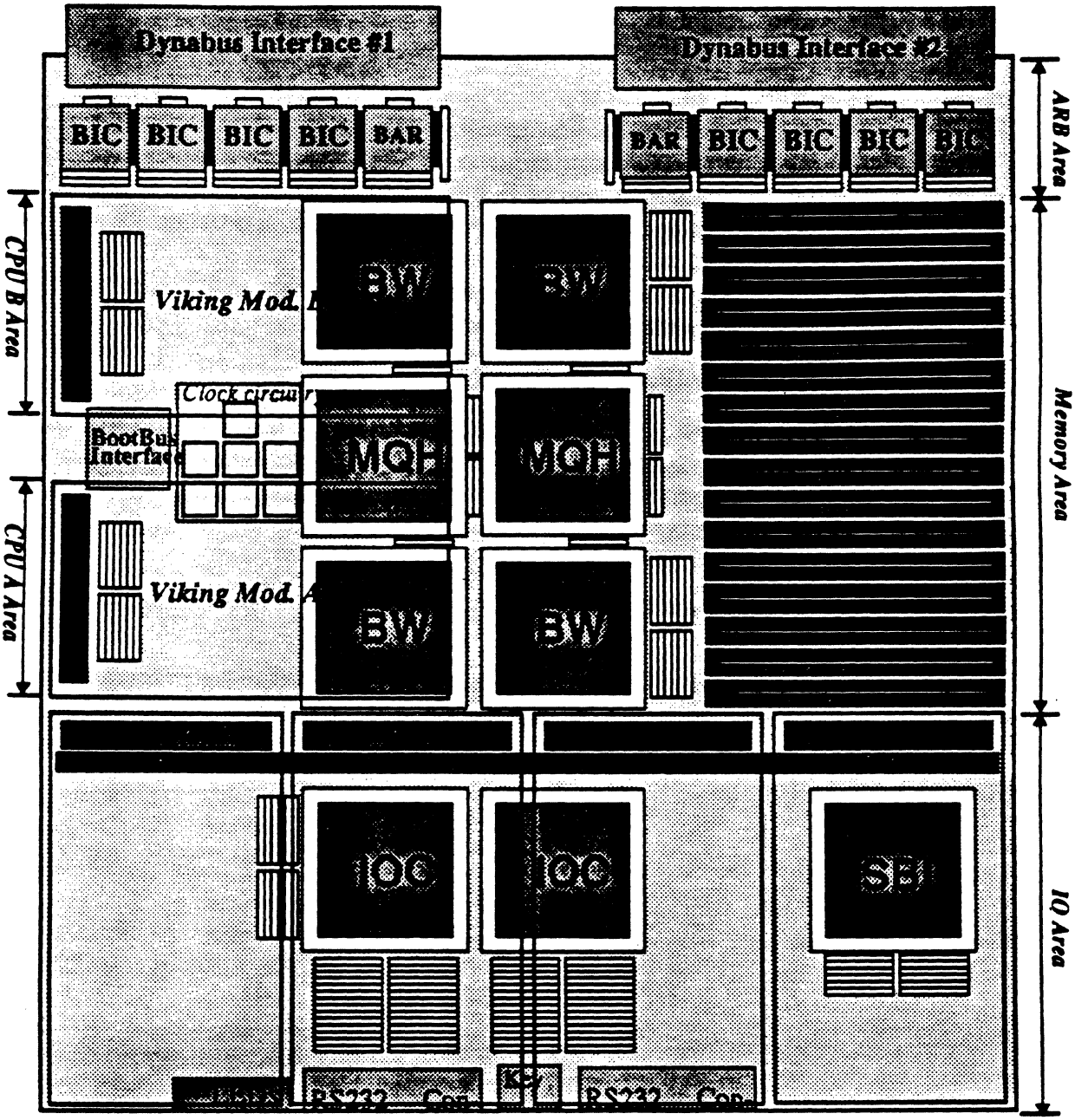
Processor units on a system board are pluggable components (Viking module). As a result, a system board may contain 0, 1 or 2 processors.

The following Figure depicts a logical view of the SunDragon system board:



SunDragon System Board Logical Diagram

The SunDragon system board approximate layout is depicted by the following Figure.



SunDragon System Board Layout

The SunDragon backplane contains the two Dynabusses. The interface between the on-board Dynabusses and the backplane Dynabusses is done with Bus Interface chips (BIC). BICs are16 bit-sliced pipeline registers (4 per Dynabus).

The Dynabus arbitration is done at two levels with Board Arbiter chips (BARB) and Central Arbiter chips (CARB). A BARB can serve four Dynabus devices. There is one BARB per bus on the System Board. The CARB can serve 10 BARBs.

Two CARBs, one per Dynabus, and the clock generation circuitry are mounted on a specific board called the System Control board. This board is attached to the rear of the backplane.

All ASICs support JTAG for testing and configuration purposes. JTAG provides a serial data path to shift information inside the chips. At the system level, SunDragon implements an extended version of JTAG, called JTAG+, that allows to select a particular board and a chip on that board. There are no jumpers on the System Board as all initialization parameters can be loaded in the chips with JTAG. The JTAG scan bus is accessible by the processors and, for manufacturing purposes, by an external controller attached via the system backplane.

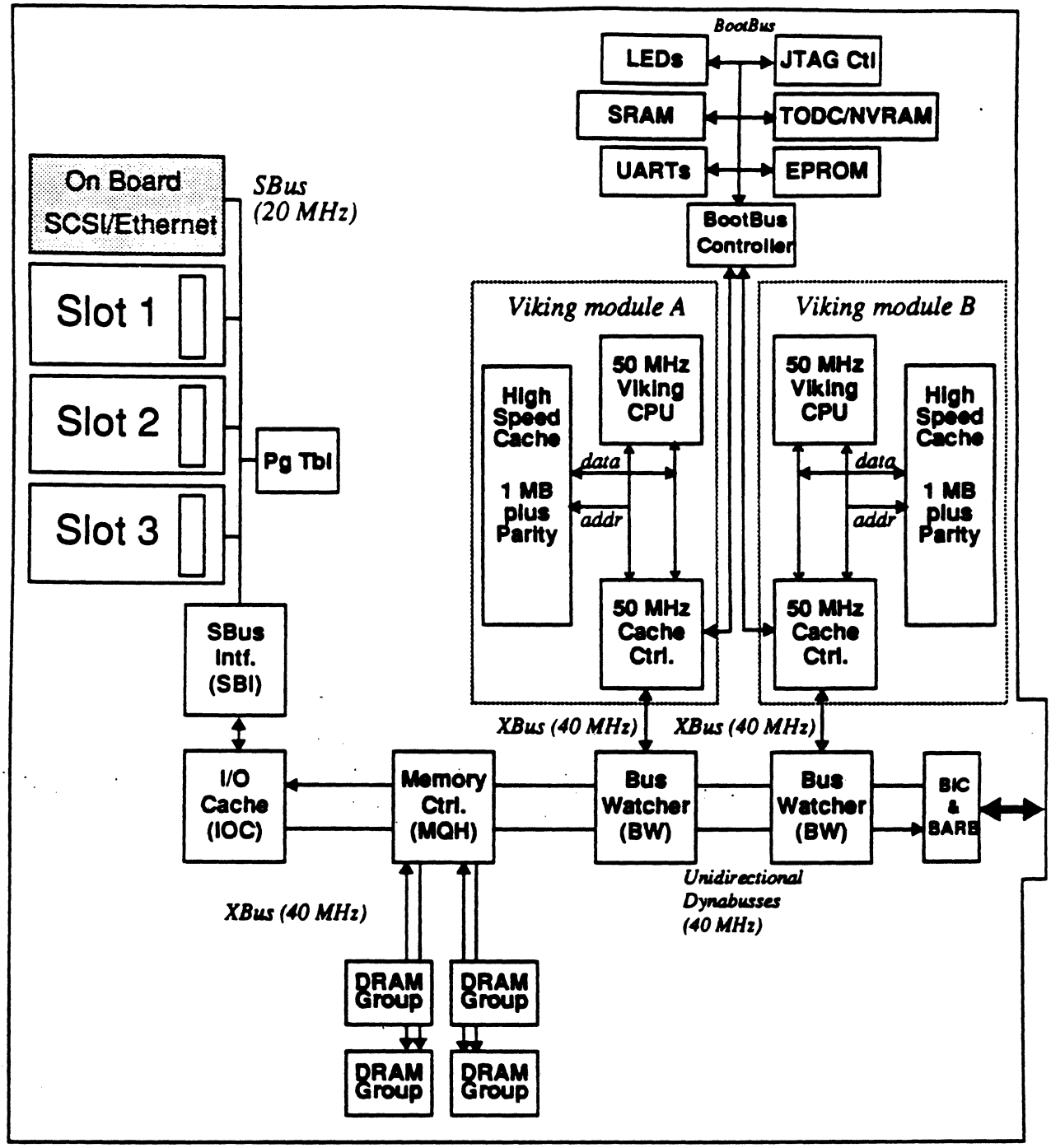
1.3.2 Scorpion Implementation

Scorpion, like SunDragon, uses only one type of system board. The mix of different units on the system board was dictated by packaging consideration. The Scorpion system board comprises two processor units, a main memory unit and an I/O unit. Scorpion backplane supports a single Dynabus. A control board is connected to the back of the backplane.

The processor modules and the main memory SIMMs are identical for SunDragon and Scorpion.

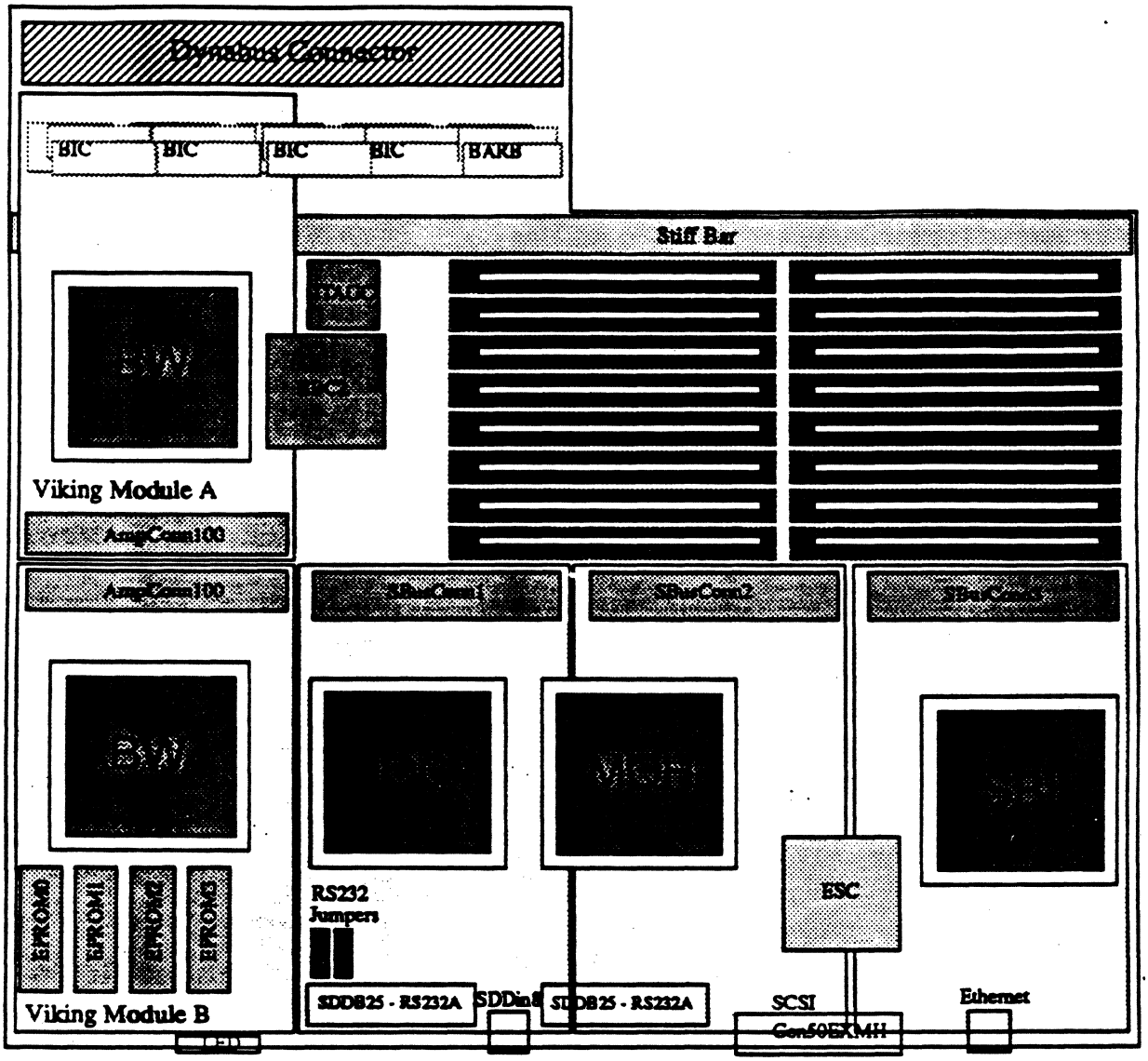
There are only three SBus slots (i.e. three SBus connectors) due to the physical size of the board. However, the I/O unit can support four SBus slots and the "fourth SBus slot" is used by an on-board SCSI and buffered Ethernet interface.

The following Figure depicts a logical view of the Scorpion system board:



Scorpion System Board Logical Diagram

The Scorpion system board approximate layout is depicted by the following Figure.



Chapter 2
Memory Model

Sun-4D's memory model conforms to the reference memory model defined in Version 8 of the SPARC Architecture Manual. This chapter describes the memory model and provides additional details relevant to Sun-4D programmers. It begins with an overview, and then goes on to describe the **Total Store Ordering (TSO)**, and **Partial Store Ordering (PSO) models** supported by Sun-4D systems. Although the memory model treats I/O and processors as much alike as possible, there are still special considerations that apply to I/O; these are taken up last.

2.1 Overview

The memory model defines the semantics of memory operations such as load and store and specifies how the order in which these operations are issued by a processor is related to the order in which they are executed by memory. Except where indicated otherwise, the term processor refers to any entity that generates memory references, and the term memory includes device I/O registers. The model applies to all configurations of SunDragon and Scorpion whether they contain multiple processors or not. The standard SPARC memory **Total Store Ordering (TSO)** is supported, as is the higher performance version called **Partial Store Ordering (PSO)**. The PSO model is enabled via the PSO mode bit in the Viking's MMU control register.

The TSO and PSO models allow SunDragon and Scorpion to achieve higher performance than a machine that is restricted to using the well known **"Strong Consistency" model**. In this model, the loads, stores, and atomic load-stores of all processors are executed by memory serially in an order that conforms to the order in which instructions were issued by individual processors.

Programs written on Sun-4D systems using single-writer-multiple-readers locks will be portable across PSO, TSO, and Strong Consistency. Programs that use write-locks but read without locking will be portable across PSO, TSO, and Strong Consistency only if writes to shared data are separated by STBAR[1] instructions. If these STBAR's are omitted, then the code will be portable only across TSO and Strong Consistency. The guidelines for other programs are as follows: Programs written for PSO will work automatically on Sun-4D systems running in TSO mode or on a machine that implements Strong Consistency; programs written for TSO will work automatically on a machine that implements Strong Consistency; programs written for Strong Consistency may not work on Sun-4D systems regardless of the mode in which it is running; it is more likely that such a program will break

1. STBAR is defined in the SPARC Architecture Manual, Version 8. The encoding is identical to RDASR %r15,%g0.

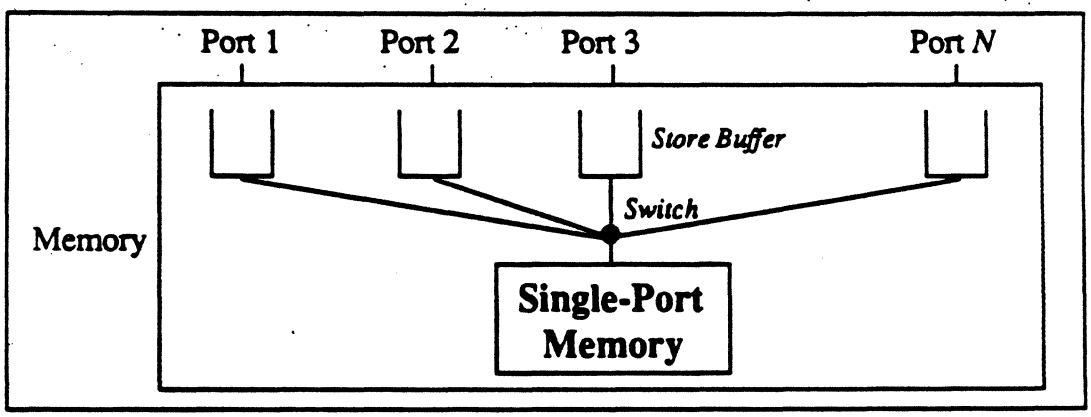
with PSO than with TSO. Finally, programs written for Sun-4D systems assuming TSO may not work when the machine is in PSO.

2.2 Basic Definitions

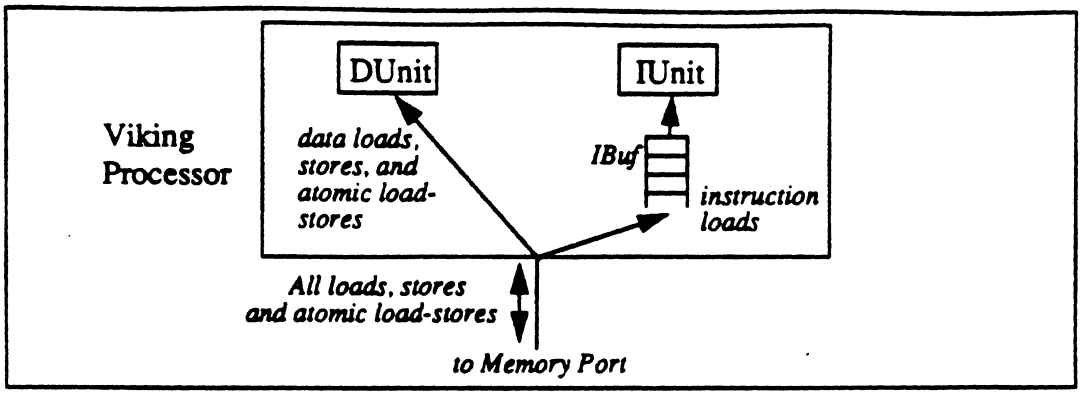
Memory is the collection of locations accessed by the SPARC load/store instructions; see the SPARC Architecture Manual for a list.These locations include real memory, I/O device registers, and registers accessible via alternate space. For the purpose of ordering, the memory model makes no distinction between the normal and alternate space versions of instructions. However, see the note in section 2.7 on page 20 for loads and stores to I/O devices.

Memory is byte addressed, with half-word accesses being aligned on a 2-byte boundary, word accesses being aligned on a 4-byte boundary, and double-word accesses being aligned on an 8-byte boundary. The largest datum that is atomically read or written by memory hardware is a double-word. Also, memory references to different bytes in a given double-word are treated for ordering purposes as references to the same location. Thus the unit of ordering for memory is a double-word.

Memory is modeled as an *N* port device where *N* is the number of processors. A processor initiates memory operations via its port in what is called the **issuing order**, or the **program order**. Each port contains a Store Buffer used to hold stores and atomic-load stores. A switch connects a single-port memory to one of the ports at a time for the duration of each memory operation. The order in which this switch is thrown from one port to another defines the **memory order** of operations. The Figure below shows this model.



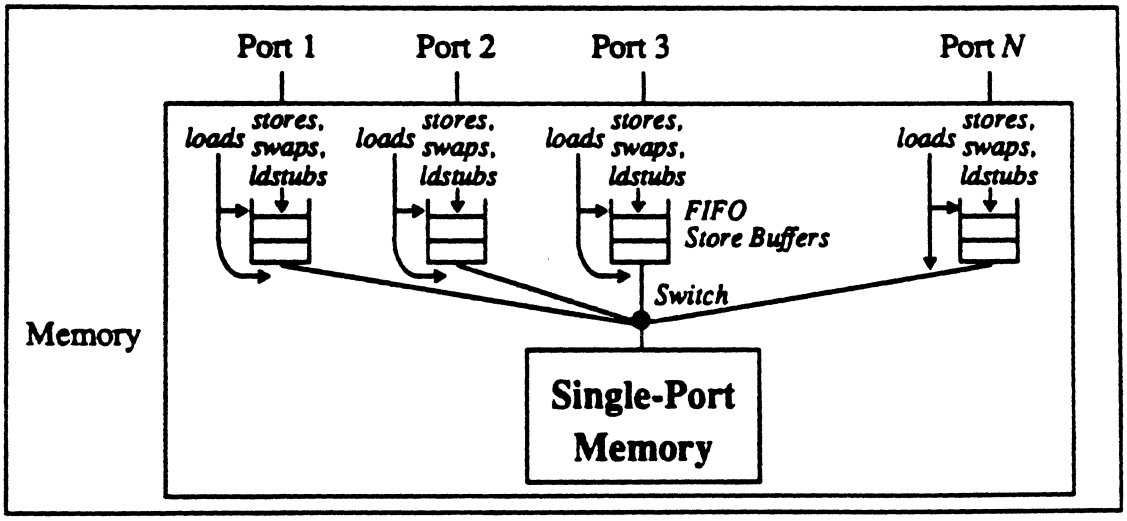
For purposes of the memory model, a Viking processor consists of three parts: a DUnit, an IUnit, and an IBuf.



The DUnit issues loads, stores and atomic load-stores to data; the IUnit issues loads to instructions; and the IBuf is a buffer where the value returned by such a load is placed when it arrives from memory (IBuf models Viking's internal instruction buffer). Notice that because of IBuf, a load to an instruction at address *A* may appear at the memory port *before* a data load or store issued by an earlier instruction to address *A*. The ordering of instruction loads relative to data loads and stores is enforced via the FLUSH instruction. When a Viking executes FLUSH *A*, the read data corresponding to location *A* is removed from IBuf if it is present.

2.3 Total Store Ordering (TSO)

The **Total Store Ordering** model guarantees that the store and atomic load-store instructions of all processors appear to be executed by memory serially in a single order called the memory order. Furthermore, the sequence of store and atomic load-store instructions in the memory order for a given processor is identical to the sequence in which they were issued by the processor. The Figure below shows the ordering constraints for this model graphically. A formal specification appears in V8 of the SPARC Architecture Manual.



Stores and atomic load-stores issued by a processor are placed in its dedicated Store Buffer, which is FIFO. Thus the order in which memory executes these operations for a given processor is the same as the order in which the processor issued them. The memory order of these operations corresponds to the order in which the switch is thrown from one port to another.

A load by a processor first checks its Store Buffer to see if it contains a store to the same location (atomic load-stores do not need to be checked for because they are blocking). If it does, then the load returns the value of the most recent such store, otherwise the load goes directly to memory. Since not all loads go to memory, loads in general *do not* appear in the memory order. A processor is blocked from issuing further memory operations until the load returns a value.

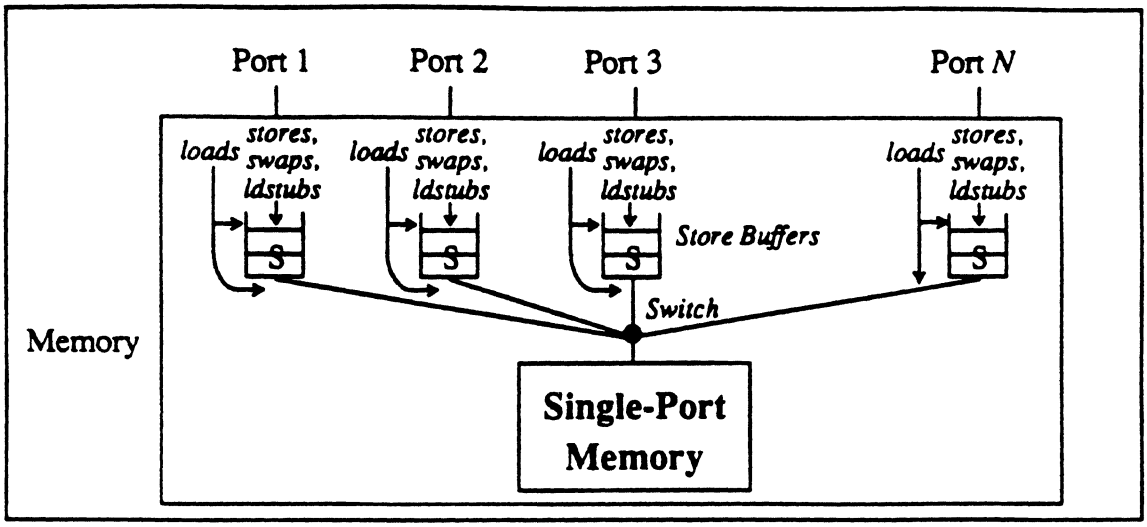
An atomic load-store (SWAP or LDSTUB) behaves both like a load and a store. It is placed in the Store Buffer like a store, and it blocks the processor like a load. A load therefore does not need to check for atomic load-stores in the Store Buffer because this situation cannot arise. When memory services an atomic load-store, it does so atomically: no other operation may intervene between the load and store parts of the load-store.

NOTE: In the definition of TSO, the term "processor" may be replaced everywhere by the term "process" or "thread" as long as the process or thread switch sequence is written properly. See the SPARC Architecture Manual V8 [1] for the correct process switch sequence.

2.4 Partial Store Ordering (PSO)

The **Partial Store Ordering** model guarantees that the store and atomic load-store instructions of all processors appear to be executed by memory serially in a single order called the memory order. However, the memory order of store and atomic load-store instructions for a given processor is in general *not* the same as the order in which they were issued by that processor. Conformance between issuing order and memory order is enforced by the STBAR instruction: if two of the above instructions are separated by an STBAR in the issuing order of a processor or if they reference the same location, then the memory order of the two instructions is the

same as the issuing order. The Figure below shows the ordering constraints for the model. A formal specification appears in V8 of the SPARC Architecture Manual.



Stores and atomic load-stores issued by a processor are placed in its dedicated Store Buffer. This buffer maintains the order of stores and atomic load-stores to the same location, but otherwise it is ordered only by STBAR instructions. These are shown in the figure as S's. Thus the order in which memory executes two stores or atomic load-stores separated by an STBAR for a given processor is the same as the order in which the processor issued them. The memory order of these operations corresponds to the order in which the switch is thrown from one port to another.

Load's first check the Store Buffer for the processor to see if it contains a store to the same location (atomic load-stores don't need to be checked for because they are blocking). If it does, then the load returns the value of the most recent such store, otherwise the load goes directly to memory. Since not all loads go to memory, loads in general *do not* appear in the memory order. A processor is blocked from issuing further memory operations until the load returns a value.

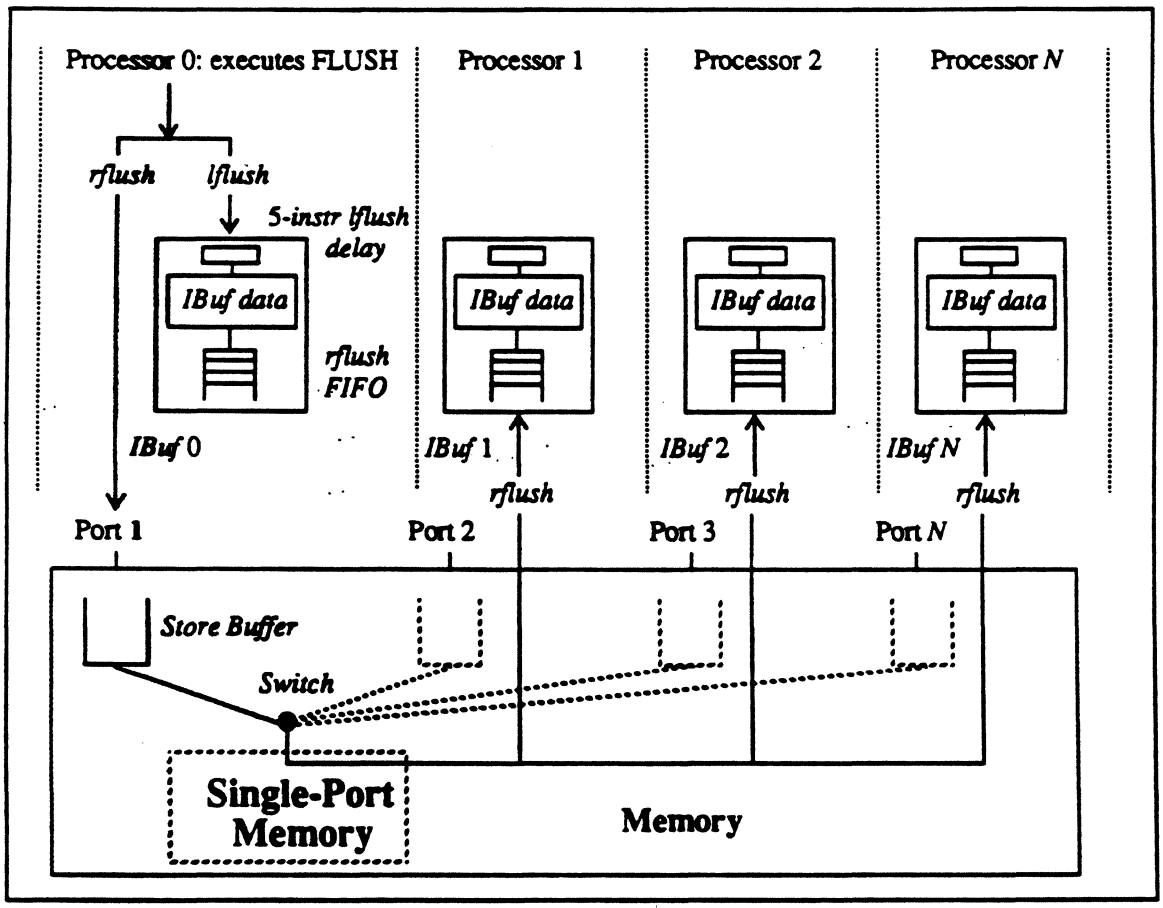
An atomic load-store (SWAP or LDSTUB) behaves both like a load and a store. It is placed in the Store Buffer like a store, and it blocks the processor like a load. A load therefore does not need to check for atomic load-stores because this situation cannot arise. When memory services an atomic load-store, it does so atomically: no other operation may intervene between the load and store parts of an atomic load-store.

NOTE: The advantage of PSO over TSO is that it allows an implementation to build a higher performance memory system. PSO therefore should be thought of as a performance optimization over TSO.

NOTE: In the definition of PSO, the term "processor" may be replaced everywhere by the term "process" or "thread" as long as the process switch sequence is written properly. See the SPARC Architecture Manual V8 [1] for the correct process switch sequence.

2.5 FLUSH instruction

The FLUSH instruction ensures that subsequent instruction fetches to the target of the FLUSH by the processor executing the FLUSH appear to execute after any loads, stores and atomic load-stores issued by that processor prior to the FLUSH. In a multiprocessor, FLUSH also ensures that stores and atomic load-stores to the target of the FLUSH issued by the processor executing the FLUSH prior to the FLUSH become visible to the instruction fetches of all other processors some time after the execution of the FLUSH. When a processor executes a sequence of store or atomic load-stores interspersed with appropriate FLUSH and STBAR instructions (the latter needed only for PSO), the changes appear to the instruction fetches of *all* processors to occur in the order in which they were made. The following figure shows the operation of FLUSH graphically, assuming Processor 0 is the one executing the FLUSH.



The IBuf of each processor consists of three elements: a 5-instruction local flush (*lflush*) delay that delays the execution of locally generated FLUSHes by at most 5 instructions; the IBuf data; and a remote flush (*rflush*) FIFO that contains flushes generated by remote processors. Processor 0 executes a FLUSH *A* by issuing a local flush (*lflush*) command to its IBuf and placing a remote flush (*rflush*) command in its store buffer and then proceeds to execute the next instruction. Processor 0's IBuf executes the lflush after at most a 5-instruction delay by invalidating the contents

of address *A* in IBuf. The rflush placed in the Store Buffer is treated exactly like a store for ordering purposes: it appears in the global memory order by going through the single port of memory and is then placed in the rflush FIFO's of processors other than the one executing the FLUSH (the rflush *does not* have any effect on the contents of Memory). A remote processor's IBuf invalidates the contents of address *A* when the rflush comes to the head of this processor's rflush FIFO.

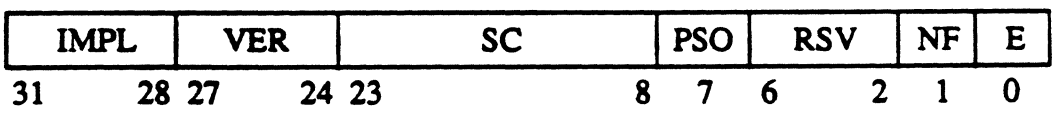
The lflush guarantees that an instruction fetch to address *A* issued 5 cycles or more after the FLUSH *A* by Processor 0 will miss the IBuf and turn into an instruction load that will appear in the issuing order defined at Processor 0's memory port. Given the guarantees provided by the memory model, this load will observe the value of a store done to *A* before the FLUSH. Note also, that the order of lflushes is preserved, so that if a processor executes the sequence ST *A*, FLUSH *A*, STBAR, ST *B*, FLUSH *B*, the instruction fetches of this processor will observe the two stores in the order in which they were issued.

The guarantee provided by rflush is weaker: copies of *A* in the IBuf's of remote processors will be invalidated some time after the FLUSH is executed by Processor 0. In particular, Processor 0 *may not* assume that the remote IBuf's have been invalidated at the time when it starts the execution of the instruction just after the FLUSH. Also note that since rflushes are ordered just like stores, the two stores in the sequence ST *A*, FLUSH *A*, STBAR, ST *B*, FLUSH *B* will be observed by the instruction fetches of remote processors in the issuing order. The STBAR is, of course, superfluous in TSO mode.

In Sun-4D systems, instruction caches are kept consistent in hardware. As a result, the FLUSH instruction really affects only the issuing processor. This information allows to implement more efficiently certain low-level kernel operations, but should not be used in user-level code for portability reasons.

2.6 TSO/PSO Mode Control

The memory model seen by each Viking processor is controlled by the PSO bit in its MMU control register. PSO=0 specifies Total Store Ordering, while PSO=1 specifies Partial Store Ordering. The figure below shows the location of this bit in the MMU control register. See Version 8 of SPARC Architecture manual for a definition of the other fields and the Viking User Documentation for a definition of the implementation specific fields.



On MMU reset, the MMU is disabled and the PSO bit is set to 0 by the hardware. Thus each Viking goes into TSO mode on reset.

The PSO mode bit should be kept along with process state that is saved and restored on process switches. Changes to this mode bit must be made in a carefully con-

trolled way since undisciplined access will result in bugs that are extremely hard to track down. It is recommended that this bit not be modified except during the process switch sequence.

Changing the mode from PSO to TSO is generally safe, because a program written for PSO will continue to work on TSO, albeit more slowly. Changing the mode from TSO to PSO must be done only when the program making the change, and any program it transfers control to has been written to work on PSO.

2.7 Special Considerations for I/O

Although the TSO and PSO models treat processor references to memory the same as processor references to I/O devices and I/O device references to memory, practical considerations force I/O to be treated somewhat differently in the implementations. This section describes what programmers need to know about the memory model as it relates to I/O.

The discussion of the memory model for I/O breaks down conveniently into three parts: processor initiated loads and stores that reference I/O devices; I/O device initiated loads and stores; and finally interrupts. The remainder of this section takes up each topic in turn.

2.7.1 Processor Initiated References to I/O Devices

A key difference between ordinary memory locations and I/O device registers is that the accesses to the latter have side-effects. In other words, while loads and stores to different double words in memory may be treated independently of one another, loads and stores to I/O registers in the same device cannot. For all practical purposes, the implication of this is that hardware must maintain the order of loads and stores to I/O devices specified by the program order (in principle, it is possible to violate the order between two I/O devices that do not have any way to communicate with one another except through main memory).

When a processor is in PSO mode, its external cache allows multiple outstanding writes as long as there is a guarantee that these writes will reach the I/O device in the order in which they were issued. The SunDragon and Scorpion hardware guarantees that the arrival order at an I/O device is the same as the issuing order as long as writes stay on the same Dynabus. Thus the Cache Controller keeps track of the Dynabus over which each I/O write is issued and any time the destination bus changes, it forces all previous I/O writes to complete before acknowledging the current I/O write to Viking. The Cache Controller also stalls writes when the physical page part of the address of a write changes, but this has to do with limitations in the Bus Watcher's implementation, *not* with enforcing ordering.

When a processor is in TSO mode, the Viking itself does not issue multiple outstanding I/O writes to the external cache. The Cache Controller itself has no knowledge of the TSO/PSO mode bit in Viking, so it continues to handle I/O writes just as before, i.e. only one I/O operation is outstanding at any given time.

2.7.2 I/O Device Initiated References to Memory

I/O devices access memory in one of two modes: *non-stream* and *stream*. In non-stream mode, each SBus slot is treated like a processor that is allowed to have exactly one operation outstanding at a time; an SBus slot reads and writes directly from consistent memory, and its accesses conform to the TSO model.

In stream mode, data written by an I/O device is buffered in the SBI and is written into memory either under explicit program control or when the buffer is needed by a concurrent stream access.

2.7.3 I/O Interrupts

Interrupts generated by devices on the SBus are treated by the SBI exactly like stores from an SBus slot. That is, the order in which interrupts and writes are issued on the XBus by an SBI on behalf of a given slot is the same as the order in which these events were presented to the SBI by the SBus slot. Interrupts are also completely serialized with respect to writes. That is, at most one operation of any kind (including an interrupt packet) may be outstanding from each slot at any time.

2.7.4 InterProcessor Interrupts

Interrupts issued by a processor (See section 4.4.5.4 on page 45) to another processor are handled by the hardware as a processor-initiated reference to an I/O device (i.e. a STORE operation to I/O space). As a result, processor-issued interrupts behave as STORE operations in the memory model and obey the same serialization rules.

Chapter 3
Address Spaces

In this chapter, a general description of the various address spaces is given. The following chapters describe in detail how individual functional units are accessed.

3.1 Virtual Addresses

3.1.1 SPARC Virtual Address Spaces

Virtual addresses are issued by all SPARC memory references and are the only addresses directly accessible to a programmer. Sun-4D systems use the virtual address space structure of the Viking processor, which follows the guidelines provided in Appendix I of the SPARC V8 architecture manual [1].

SPARC memory references (instruction fetches, loads, stores and atomic operations) produce a 32-bit virtual address (VA) and an 8-bit address space identifier (ASI). The ASI is either explicitly specified in the memory-reference instruction (`LDUBA`, `LDSBA`, `LDUHA`, `LDSHA`, `LDA`, `LDDA`, `STBA`, `STHA`, `STA`, `STDA`, `LDSTUBA` and `SWAPA`), or derived from the current operating mode for all other memory references (including instruction fetches) according to the following rules:

- ASI `0x08`: Instruction fetch in User mode
- ASI `0x09`: Instruction fetch in Supervisor mode
- ASI `0x0A`: Data access in User mode
- ASI `0x0B`: Data access in Supervisor mode

Virtual addresses are interpreted by the hardware in different ways depending on the ASI:

- ASIs `0x08` through `0x0B` are translated into 36-bit physical addresses using the Reference MMU [1], and the cacheability attribute of the memory reference is obtained from the corresponding PTE (Page table Entry)
- ASIs `0x20` through `0x2F`, known as the bypass ASIs, are translated into 36-bit physical addresses by prepending the low-order 4 bits of the ASI to the virtual address, and the cacheability attribute is obtained from the AC bit of the Viking MMU Control Register [7]
- ASI `0x02` is used to access internal registers of the processor's Cache Controller and External cache tags and data (see details in Chapter 4).

- ASI 0x03 is used to provide access to the Reference MMU mappings, including a facility to enforce multiprocessor MMU consistency (see section 4.2.3 on page 36 and [7])
- Other ASIs are used to address directly certain control registers and diagnostic features inside the Viking processor, and are not associated with physical addresses (see [7] for a complete map of the ASIs used in the Viking processor)

3.1.2 SBus DVMA Virtual Address Spaces

When an SBus Master initiates a DMA operation, it issues a 32-bit virtual address which is translated by the SBus MMU. Each SBus has a separate SBus MMU, which is described in detail in section 6.4.1 on page 131. Note that the translation mechanism for DVMA addresses is completely separate from the translation mechanism for CPU virtual addresses.

The result of the address translation is either the address of an SBus Slave device on the same SBus (in which case the access is handled locally to the SBus), or a 36-bit physical address which is always marked *cacheable*.

3.2 Physical Address Space

Physical addresses represent the effective addressing mechanism in a Sun-4D system. All memory locations and registers in the system, except for control registers internal to the processor unit, are accessed by physical addresses which are carried on Dynabus. Physical addresses are 36-bit long.

The Sun-4D architecture differs substantially from other SPARC systems because the structure of the physical address space is to a large extent programmable.

Although current Sun-4D implementations (SunDragon and Scorpion) are based on multiple boards, it must be noted that the board structure is invisible to the programmer except during POST operation.

POST (Power-On Self-Test firmware) is responsible for setting up the device registers which define the addresses for the various system components. OBP (Open Boot firmware) is responsible for creating a description of the physical address space, called the device tree, and pass it to the kernel (or other booted program). The kernel should rely exclusively on the information passed in the device tree.

3.2.1 Memory Space, I/O Space and Cacheability

Sun-4D systems use the cacheability attribute of a memory reference to distinguish references to main memory from references to I/O devices and control registers.

Memory Space designates the set of all locations which responds to *cacheable* memory references. I/O Space designates the set of all locations which responds to *non-cacheable* memory references. Sun-4D systems enforce consistency and ordering in Memory Space as described in Chapter 2 as long as caches are enabled. Sun-4D systems also enforce ordering in I/O Space as described in Chapter 2.

The object referenced by a memory operation depends *both* on the 36-bit physical address *and* the cacheability attribute of the reference. The object referenced by a memory operation does not depend upon the processor (or I/O device) which issued the reference, except for a part of I/O Space called Local Space, which is private to each processor unit (see section 3.2.8 on page 29).

NOTE: Current Sun-4D systems (i.e. SunDragon and Scorpion) assign Memory Space addresses in such a way that a given 36-bit physical address is either in Memory Space (PA[35]=0) or in I/O Space (PA[35]=1). This may change in future release of the architecture if main memory size exceeds 32 GB, or if new I/O devices requiring large chunks of address space, such as bus bridges, are added. To ensure software correctness with respect to future systems, it is recommended that programmers avoid assuming that a given physical address is either in Memory Space or in I/O Space.

NOTE: SBus DVMA devices cannot reference I/O Space, since all physical addresses generated by the SBus MMU are marked cacheable (DVMA to an SBus Slave on the same SBus does not use physical addresses).

The existence of the bypass ASIs (`0x20` to `0x2F`) and the AC bit in the Viking MMU Control Register (see [7]) provides direct access to arbitrary physical addresses with arbitrary cacheability from supervisor-mode programs using the alternate memory reference instructions (`LDUBA`, `LDSBA`, `LDUHA`, `LDSHA`, `LDA`, `LDDA`, `STBA`, `STHA`, `STA`, `STDA`, `LDSTUBA` and `SWAPA`).

The set of physical addresses which is implemented in a given system is determined by POST at system initialization time and is passed to the kernel (or other bootable program) by OBP in the *memory list* for Memory Space, and in the *device tree* for I/O Space. Software should rely exclusively on those two data structures.

NOTE: Memory Space is implemented by the Main Memory Unit, and the address ranges to which a given Main Memory Unit responds in Memory Space is controlled by the Address Decoding registers of the corresponding unit (see section 5.3.1 on page 87). Software may need to look at this information to localize memory errors to a specific component.

3.2.2 I/O Space Allocation

I/O Space is subdivided into the following spaces:

PA[35:28]	I/O Space
0x00-0x7F	Reserved
0x80-0xB7	SBus Space
0xB8-0xEF	Reserved
0xF0-0xFD	ECSR Space
0xFE	CSR Space
0xFF	Local Space

Address ranges marked as Reserved in the above table may be assigned in a later revision of the architecture.

The following sections describe each of the spaces listed in this table, and how their addresses are decoded.

In the descriptions of all functional units, all physical addresses refer to I/O space unless otherwise indicated.

3.2.3 Device Identifiers

Each Dynabus client in a Sun-4D system is assigned by POST a unique 8-bit identifier in the range 0x00 to 0xDF called the Device Identifier (DeviceID), used to decode I/O Space accesses to the Dynabus client.

NOTE: The assignment of DeviceIDs by POST uses geographical addressing. The 4 most significant bits of a DeviceID, noted bbbb, correspond to the backplane slot in which the corresponding board is plugged. On SunDragon, the board number is in the range 0 through 9, whereas in Scorpion, it is in the range 0 through 3. POST assigns DeviceIDs for functional units on a system board plugged in slot bbbb as follows:

Functional Unit	DeviceID	Dynabus Device
Processor Unit A	bbbb 0000	Bus Watcher(s)
I/O Unit	bbbb 0010	I/O Cache(s)
Processor Unit B	bbbb 1000	Bus Watcher(s)
Main Memory Unit	bbbb 0001	Memory Queue Handler

It must be noted that in a multiple Dynabus Sun-4D system POST assigns the same DeviceID to the Dynabus clients which are part of the same functional unit. For instance, the two Bus Watchers of a SunDragon processor unit have the same DeviceID value.
As mentioned earlier, the information in the former table should not be used by software outside of POST and OBP, which should rely instead on the device tree passed by OBP. Note that future extensions to the architecture may deviate from strict geographical addressing.

3.2.4 Unit Numbers

The Processor and I/O units in a Sun-4D system are assigned by POST a Unit Number (UnitNum) which is a 3-bit identifier. This identifier is used to decode some of the I/O Space accesses to the functional units.

NOTE: POST assigns UnitNum for functional units as follows:

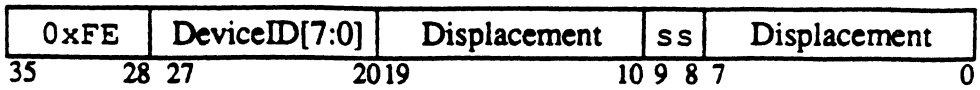
Functional Unit	UnitNum
Processor Unit A	000
I/O Unit	001
Processor Unit B	100
Memory Unit	N/A

It must noted that POST assigns Unit Numbers so that they are equal to DeviceID[3:1] for a given functional unit. In the various address tables of this document, the Unit Number is not mentioned because of this duality with DeviceID[3:1]. The information provided by this table should not be used by software outside of POST and OBP, which should rely instead on the device tree passed by OBP.

3.2.5 CSR Space

Each functional unit uses a section of Control Status and Register (CSR) Space for its Dynabus related control registers. The resources accessed using CSR Space are always associated to a specific Dynabus and correspond to the registers of devices which perform Dynabus interface functions: Bus Watchers, Memory Queue Handlers and I/O Caches.

An address in CSR Space is interpreted as:



In the description of each functional unit, "CSR Base" refers to a base address corresponding to this format.

NOTE: Since DeviceID is limited to the range `0x00` through `0xDF`, the portion of CSR Space from `0xFEE000000` through `0xFEFFFFFFF` is in effect reserved.

Devices accessible through CSR Space are physically replicated, one device per Dynabus. The **ss** field of the address indicates which bus must be used, according to the following table:

ss	4-Bus System	2-Bus System	1-Bus System
00	Bus 0	Bus 0	Bus 0
01	Bus 1	Bus 1	Bus 0
10	Bus 2	Bus 0	Bus 0
11	Bus 3	Bus 1	Bus 0

Each column indicates how many Dynabusses are configured in the system. The shaded area of this table should normally not be used, although it is guaranteed to operate as indicated in the table.

NOTE: SunDragon has two Dynabusses, and is configured by POST as a 2-Bus system or as a 1-Bus system (degraded mode). Scorpion has a single Dynabus, and is always configured as a 1-Bus system. Although the architecture allows for a 4-Bus system, no such system is currently implemented. Note that the bus numbers in the table are logical numbers, not physical numbers. For example, when SunDragon is configured as a 1-Bus system, software always perceives the operational bus as Bus 0, independently of which of the two physical busses is really enabled.

Software may need to be aware of the existence of multiple Dynabusses when accessing CSR Space. This information is conveyed by OBP in the device tree by providing multiple fragments of CSR space for a single functional unit.

The Displacement fields are interpreted by the addressed functional unit and are described in the corresponding chapters. In most descriptions, the **ss** field is assumed to be 00 for simplification, and the bottom 20 bits of the physical address are treated as a single 20-bit displacement.

All functional units provide three standard registers (per configured Dynabus) in CSR space, according to the following table:

Displacement[19:0]	Registers
0x00000	Component ID
0x00008	Dynabus Status and Control
0x00010	DynaData

The ComponentID register is a read-only word which identifies the hardware component. The Dynabus Status and Control register (DCSR) provides a standardized

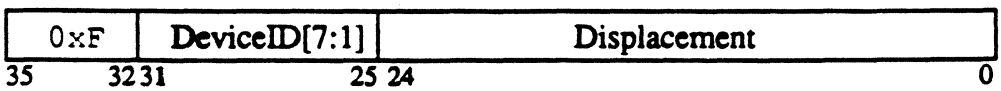
set of Dynabus parameters for the functional unit. The DynaData register (DDR) is used for error logging and bus monitoring.

NOTE: The DeviceID for a given functional unit is always accessible in the DeviceID field of the DCSR for that device. It is initialized by POST via JTAG.

3.2.6 ECSR Space

Certain functional units have resources which are not linked to the Dynabus configuration. Those resources are accessed in the Extended Control and Status Registers (ECSR) Space.

An address in ECSR Space is interpreted as:



In the description of each functional unit, "ECSR Base" refers to a base address corresponding to this format.

NOTE: Since DeviceID is limited to the range `0x00` through `0xDF`, there is no ambiguity between CSR, ECSR and Local spaces: CSR space uses what would be ECSR space for units with a DeviceID in the range `0xE0` through `0xEF`, whereas Local space uses what would be ECSR space for units with a DeviceID in the range `0xF0` through `0xFF`.

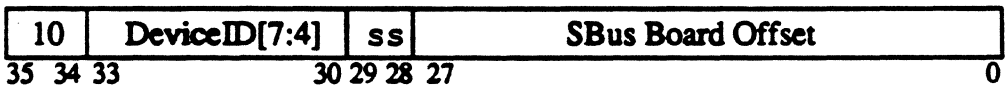
The 25-bit Displacement field is interpreted by the addressed functional unit and is described in the corresponding chapters. In the current implementations, Processor and I/O units use ECSR space, whereas Memory units do not.

NOTE: POST will always assign DeviceIDs in such a way that functional units which use ECSR Space have different values of DeviceID[7:1]. In the implementation of the functional units, the low order 3 bits of DeviceID[7:1] used in decoding ECSR Space accesses are actually not taken from the unit's DeviceID, but from the Unit Number (UnitNum). As already mentioned, POST must initialize this field to DeviceID[3:1].

3.2.7 SBus Space

Each SBus unit controls one SBus, which provides 4 SBus slots. Each SBus slot responds to a 28-bit physical space.

An address in SBus Space is interpreted as:



In the description of each functional unit, "SBus Base" refers to a base address corresponding to this format.

NOTE: Since a DeviceID may not exceed `0xDF`, this implies that physical addresses in the range `0xB80000000` to `0xBFFFFFFFF` are reserved.

ss is the SBus slot number and SBus Board Offset the offset within the addressed SBus board. The addressing structure within an SBus board depends on the specific SBus board which is plugged in and is beyond the scope of this manual.

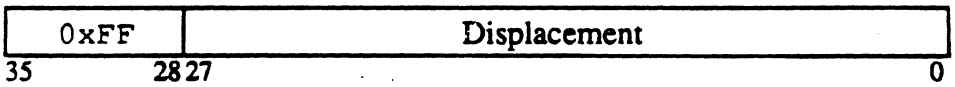
NOTE: In Scorpion, Slot 0 of each SBus is built-in and contains a combination SCSI and Ethernet interface. Software should not make any use of this knowledge, and should rely instead on the OBP device tree.

NOTE: POST will always assign DeviceIDs in such a way that SBus units have different values of DeviceID[7:4]. In the implementation of the SBus unit, the decoding of SBus space accesses is actually not derived from the unit's DeviceID, but from a separate 6-bit field located in the IOC Control register which is matched against the top 6 bits of the physical address. POST must initialize this field, to {0b10, DeviceID[7:4]}.

3.2.8 Local Space

A Processor unit has direct access to certain of its own resources (which are also accessible via CSR and ECSR space) using Local Space. The advantage of using Local Space is that the Processor unit which issues the access does not need to know its own DeviceID when using Local Space.

An address in Local Space is interpreted as:



The Displacement field is interpreted by the processor unit and is described in the corresponding chapter.

NOTE: Local space also allows a Processor unit to discover its own DeviceID since it allows to access the DCSR registers for the unit (which contain the DeviceID) without knowing the DeviceID.

3.3 System Board Physical Space

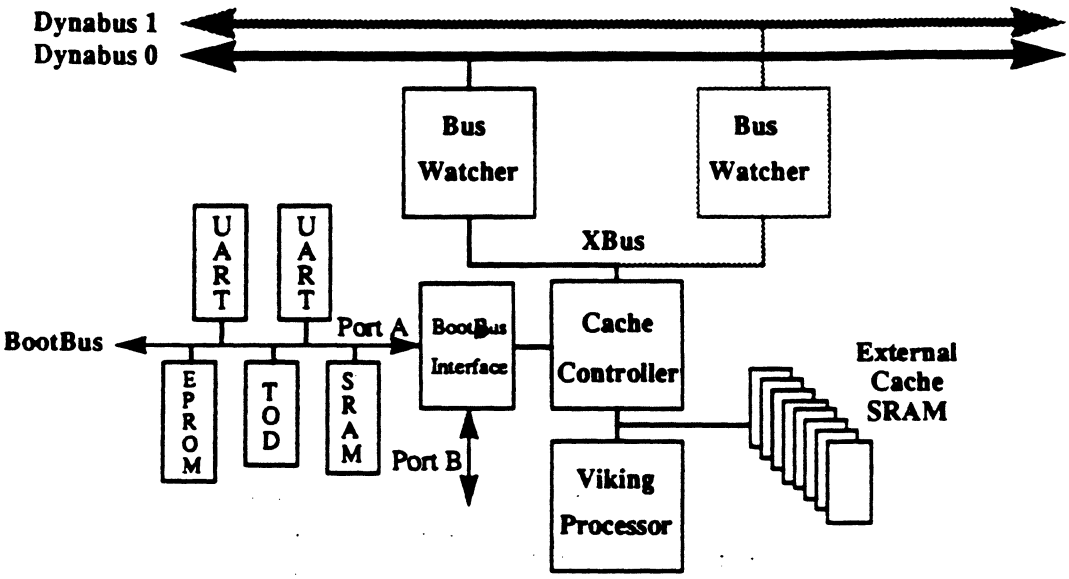
This section provides a global view of I/O Space within a system board in SunDragon and Scorpion. The information here is for information purposes only and should not be used directly since it embodies specific choices for DeviceID allocation which are subject to change. It is provided solely to help in debugging software and hardware problems.

Dots in the address field indicate bits which are decoded by the target device. Address configurations which are not covered in this table will cause time-out errors if referenced. The second columns gives, as an example, the base physical of each row for a system board plugged in slot 2.

PA[35:20]	Base PA in slot 2	Usage	Space
10bb bb00	0x880000000	SBus Slot 0 (built-in in Scorpion)	SBus
10bb bb01	0x890000000	SBus Slot 1	SBus
10bb bb10	0x8A0000000	SBus Slot 2	SBus
10bb bb11	0x8B0000000	SBus Slot 3	SBus
1111 bbbb 0000 0...	0xF20000000	BootBus (via CPU A)	ECSR
1111 bbbb 0000 100.	0xF20800000	External Cache Bus Tags (CPU A)	ECSR
1111 bbbb 0001 0...	0xF21000000	External Cache Data (CPU A)	ECSR
1111 bbbb 0001 10..	0xF21800000	External Cache Processor Tags (CPU A)	ECSR
1111 bbbb 0001 1111	0xF21F00000	Cache Controller Registers (CPU A)	ECSR
1111 bbbb 0010 0...	0xF22000000	SBus External Page Table	ECSR
1111 bbbb 0010 1...	0xF22800000	SBus Interface Registers	ECSR
1111 bbbb 1000 0...	0xF28000000	BootBus (via CPU B)	ECSR
1111 bbbb 1000 100.	0xF28800000	External Cache Bus Tags (CPU B)	ECSR
1111 bbbb 1001 0...	0xF29000000	External Cache Data (CPU B)	ECSR
1111 bbbb 1001 10..	0xF29800000	External Cache Processor Tags (CPU B)	ECSR
1111 bbbb 1001 1111	0xF29F00000	Cache Controller Registers (CPU B)	ECSR
1111 1110 bbbb 0000	0xFE2000000	Bus Watcher Registers (CPU A)	CSR
1111 1110 bbbb 0001	0xFE2100000	Memory Queue Handler Registers	CSR
1111 1110 bbbb 0010	0xFE2200000	I/O Cache Registers	CSR
1111 1110 bbbb 1000	0xFE2800000	Bus Watcher Registers (CPU B)	CSR
1111 1111 0000 0...	0xFF0000000	BootBus (Local Space)	Local
1111 1111 1111 000.	0xFFF000000	External Cache Bus Tags (Local Space)	Local
1111 1111 1111 1111	0xFFFF00000	Bus Watcher Registers (Local Space)	Local

Chapter 4
Processor Unit

A processor unit consists of a Viking processor, an external cache and system support devices connected to the BootBus. The figure below illustrates the main components of the processor unit and their interconnections.



Processor Unit

The External Cache includes a Cache Controller (CC), one or two Bus Watchers (BW or BWP) and 1 M-byte of parity-protected cache RAM. The BootBus, attached to the CC, provides a number of system-support peripherals. The BootBus is shared between two processor units in the SunDragon and Scorpion system boards.

4.1 Address Map

This section presents the address map for the processor unit. The processor unit uses CSR space, ECSR space and Local space. The External Cache processor tags, data and the CC registers are also accessible via ASI 0x2 from the attached processor.

The following table describes the CSR space address map for a processor unit:

CSR Space: PA[35:20] = 1111 1110 dddd dddd			
PA[19:0]	Size	Access	Description
0x00000	W	RO	BW Component ID Register
0x00008	D	R/W	BW Dynabus Control & Status Register
0x00010	D	R/W	BW DynaData Register
0x01000	W	R/W	BW Control Register
0x01040 to 0x01078	H	R/W	BW Interrupt Table
0x01080 to 0x010B8	H	WO	BW Interrupt Table Clear
0x010C0	H	R/W	BW Prescaler Register
0x02000	W D	R/W	BW Profiling Timer Limit Register BW User Timer Register
0x02008	W	R/W	BW Profiling Timer Non-destructive Limit
0x02010	W	R/W	BW Profiling Timer Counter
0x02018	W	R/W	BW Profiling Timer Control
0x03000	W	R/W	BW Tick Timer Limit Register
0x03008	W	R/W	BW Tick Timer Non-destructive Limit
0x03010	W	R/W	BW Tick Timer Counter

Note that bits [9:8] of the address are the bus selector in CSR Space, and that this address map should thus be interleaved 4 times with a step of 256 bytes.

The following table describes the ECSR space address map for a processor unit:

ECSR Space: PA[35:25] = 1111 dddd ddd			
PA[24:0]	Size	Access	Description
0x0000000 to 0x007FFFF	Any	RO	BB EPROM
0x0100000	B	RO	BB Status1 Register
0x0120000	B	RO	BB Status2 Register
0x0140000	B	RO	BB Status3 Register
0x0160000	B	WO	BB System Software Reset Register
0x0180000	B	RO	BB Version Register
0x01A0000	B	R/W	BB Semaphore0 Register
0x01A0004	B	R/W	BB Semaphore0 Status Register
0x01A0008	B	R/W	BB Semaphore1 Register
0x01A000C	B	R/W	BB Semaphore1 Status Register
0x01C0000 to 0x01C3FFF	Any	R/W	BB Shared SRAM

ECSR Space: PA[35:25] = 1111 dddd ddd			
PA[24:0]	Size	Access	Description
0x01E0000 to 0x01E1FFF	Any	R/W	BB Private SRAM
0x0200000	B	R/W	BB Serial Port B Control Register
0x0200002	B	R/W	BB Serial Port B Data Register
0x0200004	B	R/W	BB Serial Port A Control Register
0x0200006	B	R/W	BB Serial Port A Data Register
0x0240000	B	R/W	BB Mouse Control Register
0x0240002	B	R/W	BB Mouse Data Register
0x0240004	B	R/W	BB Keyboard Control Register
0x0240006	B	R/W	BB Keyboard Data Register
0x0280000 to 0x0281FF7	Any	R/W	BB NVRAM
0x0281FF8	B	R/W	BB TODC Control Register
0x0281FF9	B	R/W	BB TODC Seconds Register
0x0281FFA	B	R/W	3B TODC Minutes Register
0x0281FFB	B	R/W	BB TODC Hours Register
0x0281FFC	B	R/W	BB TODC Days Register
0x0281FFD	B	R/W	BB TODC Date Register
0x0281FFE	B	R/W	BB TODC Month Register
0x0281FFF	B	R/W	BB TODC Year Register
0x02C0000	B	R/W	BB Control Register
0x02E0000	B	R/W	BB LED Register
0x0300000	BHW	R/W	BB JTAG Master Register
0x0800000 to 0x09FFFFF	W	R/W	BW External Cache Bus Tags
0x1000000 to 0x11FFFFF	Any	R/W	CC External Cache Data
0x1800000 to 0x19FFFFF	DW	R/W	CC External Cache Processor Tags
0x1F00000 to 0x1F0003F	Any	R/W	CC Stream Data Register
0x1F00100	D	R/W	CC Stream Source Address Register
0x1F00200	D	R/W	CC Stream Destination Address Register
0x1F00300	D	R/W	CC Reference & Miss Count Register
0x1F00406	H	RO	CC Interrupt Pending Register
0x1F00506	H	R/W	CC Interrupt Mask Register
0x1F00606	H	WO	CC Interrupt Pending Clear Register

ECSR Space: PA[35:25] = 1111 dddd ddd			
PA[24:0]	Size	Access	Description
0x1F00704	W	WO	CC Int Generation Register (ASI 0x2 only)
0x1F00804	W	R/W	CC BIST Register (ASI 0x2 only)
0x1F00A04	W	R/W	CC Control Register
0x1F00B00	D	R/W	CC Status Register
0x1F00C04	W	R/W	CC Reset Register
0x1F00E00	D	R/W	CC Error Register
0x1F00F04	W	R/O	CC Component Identification Register

The following table describes the Local space address map for a processor unit:

Local Space: PA[35:28] = 1111 1111			
PA[27:0]	Size	Access	Description
0x0000000 to 0x007FFFF	Any	RO	BB EPROM
0x0100000	B	RO	BB Status1 Register
0x0120000	B	RO	BB Status2 Register
0x0140000	B	RO	BB Status3 Register
0x0160000	B	WO	BB System Software Reset Register
0x0180000	B	RO	BB Version Register
0x01A0000	B	R/W	BB Semaphore0 Register
0x01A0004	B	R/W	BB Semaphore0 Status Register
0x01A0008	B	R/W	BB Semaphore1 Register
0x01A000C	B	R/W	BB Semaphore1 Status Register
0x01C0000 to 0x01C3FFF	Any	R/W	BB Shared SRAM
0x01E0000 to 0x01E1FFF	Any	R/W	BB Private SRAM
0x0200000	B	R/W	BB Serial Port B Control Register
0x0200002	B	R/W	BB Serial Port B Data Register
0x0200004	B	R/W	BB Serial Port A Control Register
0x0200006	B	R/W	BB Serial Port A Data Register
0x0240000	B	R/W	BB Mouse Control Register
0x0240002	B	R/W	BB Mouse Data Register
0x0240004	B	R/W	BB Keyboard Control Register
0x0240006	B	R/W	BB Keyboard Data Register
0x0280000 to 0x0281FF7	Any	R/W	BB NVRAM
0x0281FF8	B	R/W	BB TODC Control Register
0x0281FF9	B	R/W	BB TODC Seconds Register
0x0281FFA	B	R/W	BB TODC Minutes Register

Local Space: PA[35:28] = 1111 1111			
PA[27:0]	Size	Access	Description
0x0281FFB	B	R/W	BB TODC Hours Register
0x0281FFC	B	R/W	BB TODC Days Register
0x0281FFD	B	R/W	BB TODC Date Register
0x0281FFE	B	R/W	BB TODC Month Register
0x0281FFF	B	R/W	BB TODC Year Register
0x02C0000	B	R/W	BB Control Register
0x02E0000	B	R/W	BB LED Register
0x0300000	BHW	R/W	BB JTAG Master Register
0xF000000 to 0xF1FFFFF	W	R/W	BW External Cache Bus Tags
0xFF00000	W	RO	BW Component ID Register
0xFF00008	D	R/W	BW Dynabus Control & Status Register
0xFF00010	D	R/W	BW DynaData Register
0xFF01000	W	R/W	BW Control Register
0xFF01040 to 0xFF01078	H	R/W	BW Interrupt Table
0xFF01080 to 0xFF010B8	H	WO	BW Interrupt Table Clear
0xFF010C0	H	R/W	BW Prescaler Register
0xFF02000	W D	R/W	BW Profiling Timer Limit Register BW User Timer Register
0xFF02008	W	R/W	BW Profiling Timer Non-destructive Limit
0xFF02010	W	R/W	BW Profiling Timer Counter
0xFF02018	W	R/W	BW Profiling Timer Control
0xFF03000	W	R/W	BW Tick Timer Limit Register
0xFF03008	W	R/W	BW Tick Timer Non-destructive Limit
0xFF03010	W	B/W	BW Tick Timer Counter

4.2 Viking processor

4.2.1 Control register

The Snoop Enable (SE) bit of the control register must be set to one, i.e. activity on the Viking bus may affect the internal cache state. If this bit is set to zero, the cache consistency algorithm will not operate properly.

In normal operation, the Tablewalk Cacheable (TC) bit should be set to one.

NOTE: When TC is zero, MMU table walking will occur in I/O space, which may cause the maintenance of the PTE R and M bits to fail since most I/O devices do not support SWAP (See section 4.2.2 on page 36). This problem can be avoided if all PTEs have the R and M bits set, which may be useful during firmware initialization.

4.2.2 PTE Referenced and Modified Bits

The Referenced (R) and Modified (M) bits of a Page Table Entry (PTE) are maintained by the Viking MMU in a manner which guarantees their consistency in a multiprocessor environment.

When the MMU wants to set the R bit of a PTE, it issues a SWAP operation between the MMU Page Descriptor Cache entry (which has R=1) and the PTE. If the cached entry had M=0 and the results of the SWAP operation indicate M=1, the MMU sets M=1 in the cached entry and issues a STORE of the cached entry to the PTE.

When the MMU wants to set the M bit of a PTE, it sets M=1 and R=1 in its cached entry and issues a STORE of the modified value to the PTE.

This algorithm provides consistency between PTE updates by multiple MMUs, but prohibits atomic accesses by programs to the page tables since MMU accesses are entirely transparent to the software. When software wants to modify a PTE, it must ensure that no MMU in the system has that PTE in its Page Descriptor Cache. The next section shows skeleton code to modify a PTE in software.

4.2.3 MMU Demapping

When an MMU flush operation is issued (using ASI 0x3), the operation affects *all* MMUs in the system. An MMU flush operation is also called a DeMap operation.

Only one DeMap operation can take place at a time. Software must use a system-wide critical section to protect all DeMap operations. Simultaneous DeMap operations are detected by BWs and cause a system watchdog reset (See section 4.5.3 on page 59).

When a DeMap completes, it is reasonable to expect that the following is true:

- All memory references concerning the physical space(s) mapped by the flushed PTE(s), that have been issued before the demap must have been completed.
- No MMU has a valid copy of the PTE(s).
- All memory references to the PTE(s) itself (themselves) that were issued before the demap have been completed.

However, in the case of Sun-4D systems, these properties are not true because a PTE can be revalidated whenever the R or M bit is updated. Then the PTE can be reloaded by any MMU. The hardware in SunDragon **does not guarantee** that an MMU flush is atomic system wide. To enforce the correctness of a demap, the pro-

cessor issuing the flush command **must** execute the following loop or any equivalent algorithm in order to change the value of a PTE:

```
        Lock                                /* Enter Critical Section */
        OldPte = 0
Again:  r=0
        Swap (Mem_PTE, r)                   /* Invalidate PTE */
        Flush MMU (Virtual_Address)         /* Flush MMUs */
        OldPte = OldPte | r                 /* Accumulation of R, M bits */
        If (Mem_PTE ≠ 0) goto Again         /* PTE has been revalidated */
        Mem_PTE = New_Pte                   /* Remapping */
        Unlock                              /* Exit Critical Section */
```

The loop is executed until all pending updates of the PTE have been completed. In the worst case, the loop is executed three times. The PTE might be reloaded by some MMUs in the system, but all accesses causing the revalidation will cease as soon as both the R and M bits are set to one [2].

4.3 The External Cache

4.3.1 Organization

The External Cache is a 1 M-byte direct-map physical address cache, handling both data and instructions. The block size is 256 bytes, composed of four 64-byte sub-blocks. A degraded 512 K-byte configuration is supported (for use when a single Dynabus is operational in a system which uses BW). The architecture also defines 2 M-byte and 4M-byte configurations, which are not supported by the current implementations.

A write-back, write-broadcast algorithm is used to maintain consistency between all External Caches and all I/O Caches in the system. The Dynabus specification [4] describes the consistency algorithm.

The Viking internal caches (instructions and data) are maintained consistent with the attached External Cache using a write-through, invalidate algorithm from the Viking side. The External Cache maintains the inclusion property with both Viking internal caches, i.e. any word present inside a Viking internal cache is also present in the attached External Cache.

NOTE: It is possible to violate the inclusion property by direct manipulation of the tags. Doing so may prevent the consistency algorithm from operating correctly.

There are two copies of the External Cache tags: one is used for accesses by the processor and is kept in the CC, the other copy is used to snoop transactions on Dynabus and is kept in the BWs. This arrangement improves performance by reducing tag contention between processor and bus to a strict minimum. Access to the External cache data and processor tags is described in section 4.4.1 on page 40 and sec-

tion 4.4.2 on page 41. Access to the External Cache bus tags is described in section 4.5.1 on page 54.

NOTE: Direct manipulation of the External Cache tags may prevent proper cache operation if the cache is not enabled, by making the two copies of the tags inconsistent one with another.

4.3.2 Cacheability

Cache behavior in Sun-4D systems is controlled by three items:

1. the programmer indicates whether an access is *cacheable* or not through the PTE C bit, as well as the TC and AC bits of the Viking Control register according to the following table:

Translation Mode	Cacheability
Boot Mode **BT=1, EN=X** **Instruction Fetch only**	**Non Cacheable**
Table Walk	**TC=1 ⇒ Cacheable** **TC=0 ⇒ Non-Cacheable**
MMU Disabled Mode **BT=0, EN=0**	**AC=1 ⇒ Cacheable** **AC=0 ⇒ Non-Cacheable**
MMU Enabled Mode **BT=0, EN=1**	**C=1 ⇒ Cacheable** **C=0 ⇒ Non-Cacheable**
MMU Transparent Mode **Data Access only**	**AC=1 ⇒ Cacheable** **AC=0 ⇒ Non-Cacheable**

NOTE: The BootMode case affects instruction fetches only. Cacheability of data accesses during BootMode can be determined from the other entries in this table.

2. the Viking internal Instruction Cache (respectively Data Cache) is enabled or disabled according to the state of the IE bit (respectively DE bit) of the Viking MMU Control register
3. the External Cache is enabled or disabled according to the state of the CE bit (bit[2]) of the CC Control register (See section 4.4.7 on page 47)

As mentioned in section 3.2.1 on page 24, all references to Memory Space must be marked as *cacheable* (according to the table above), whereas all references to I/O Space must be marked *non-cacheable*.

Non-cacheable references are never cached, independently of the state of the cache enable bits.

Cacheable references are cached only in the cache level(s) which are currently enabled. It is important to note that, if the External Cache is disabled, cacheable processor STORE operations (including LDSTUB and SWAP) will not be performed correctly, while LOAD operations will be performed normally. The external cache should be disabled *only* for diagnostics.

NOTE: To be more precise, stores and swaps do not update main memory. If the block where the store is taking place is present in other caches *which are enabled*, these copies are updated, but consistency is not guaranteed: future references to this block may return the modified value or the original value. Moreover, swaps do *not* return the old data value, but always

return the new data which is written.Writes to main memory performed through the block copy mechanism (See section 4.3.3 on page 32) are performed correctly when the cache is disabled.

The External Cache is disabled when the CC is reset (See section 4.4.12 on page 53). After a power-on reset, the contents of the External Cache are undefined. It is the responsibility of the software to initialize all tags, both in the CC and the BWs, to a non-Valid, non-Pending state.

4.4 Cache Controller

The External Cache processor tags, data and the CC registers are accessible via ASI 0x2 and via ECSR space.

When accessed via ECSR space, the 11 most significant bits of the physical address must be PA[35:25]=1111 dddd ddd_2, where dddd ddd_2 is DeviceID[7:1].

The address map for the Cache Controller is summarized below:

	ASI 0x2 (from Viking) VA[31:25] = 0000 000	ECSR Space PA[35:25] = 1111 dddd ddd
	VA[24:21]	PA[24:21]
ECache Tags	1100	1100
ECache Data	1000	1000
CC Registers	1111	1111

NOTE: Certain of the address bits in the above table may not be fully decoded in a given implementation. Accesses with addresses not described in this table may provide different results in different implementations.

Accesses to the External Cache processor tags, External Cache data and CC registers must be performed with the specified data size. A read with a smaller size will be performed normally. In the case of a read with a larger size, the nonexistent bits will be undefined. All writes with a size different from the specified one will have unpredictable effects. All SWAP or LDSTUB operations will result in a time-out error. Accesses to unspecified addresses within ASI 0x2 or the Cache Controller ECSR space will result in a time-out bus error, or will access some other valid location within the same space.

The address map for the CC registers is:

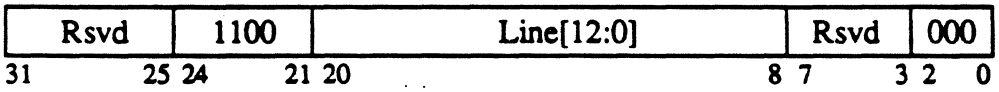
PA[35:25] = 1111 dddd ddd_2 (ECSR Space) VA[31:25] = 0000 000_2 (ASI 0x02)			
PA[24:0], VA[24:0]	Size	Access	Cache Controller Registers
0x1F00000 to 0x1F0003F	Any	R/W	Stream Data Register
0x1F00100	D	R/W	Stream Source Address Register
0x1F00200	D	R/W	Stream Destination Address Register
0x1F00300	D	R/W	Reference & Miss Count Register
0x1F00406	H	RO	Interrupt Pending Register

PA[35:25] = 1111 dddd ddd$_2$ (ECSR Space) VA[31:25] = 0000 000$_2$ (ASI 0x02)			
PA[24:0], VA[24:0]	Size	Access	Cache Controller Registers
0x1F00506	H	R/W	Interrupt Mask Register
0x1F00606	H	WO	Interrupt Pending Clear Register
0x1F00704	W	WO	Interrupt Generation Register (ASI 0x2 only)
0x1F00804	W	R/W	BIST Register (ASI 0x2 only)
0x1F00A04	W	R/W	Control Register
0x1F00B00	D	R/W	Status Register
0x1F00C04	W	R/W	Reset Register
0x1F00E00	D	R/W	Error Register
0x1F00F04	W	R/O	Component Identification Register

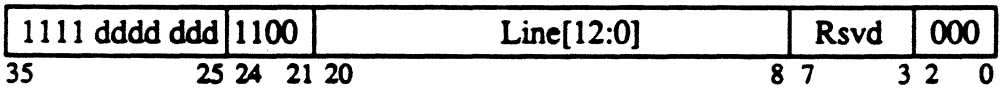
NOTE: Bits [19:12] and [7:3] of the address are not decoded in the current implementation.

4.4.1 External Cache Processor Tags

The External Cache processor tags may be read and written as double-words. The address format when the tags are accessed via ASI 0x2 is:



The physical address format when the Tags are accessed through ECSR space is:

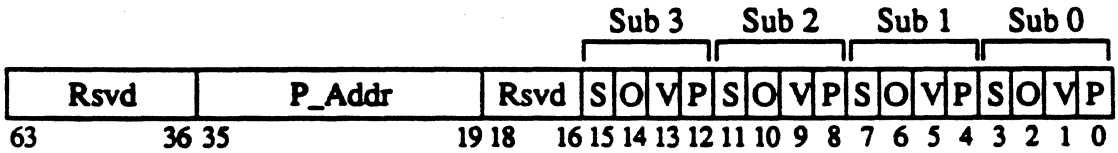


where:

Rsvd: Reserved. All reserved bit fields are ignored. They should be 0.

Line: Selects which tag entry is referenced. If the configured cache size is 1 M-byte (normal case in SunDragon and Scorpion), bit [20] (i.e. Line[12]) is ignored. If the configured cache size is 512 K-bytes, bits [20:19] (i.e. Line[12:11]) are ignored.

The format of a tag is:



The various bit fields have the following meanings:

P_Addr: Physical Address bits [35:19]. All bits of P_Addr are significant, independently of the cache size configured in the Control register (See section 4.4.7 on page 47).

S: Shared. When set to one, the corresponding subblock is potentially shared by other processors.

O: Owned. When set to one, the corresponding subblock is "owned" by this cache. This means the last modification to the subblock was done by the attached Viking processor.

V: Valid. When set to one, the corresponding subblock is valid.

P: Pending. Set to one by the hardware to indicate that an operation initiated by the attached processor is currently pending in the memory system for this subblock.

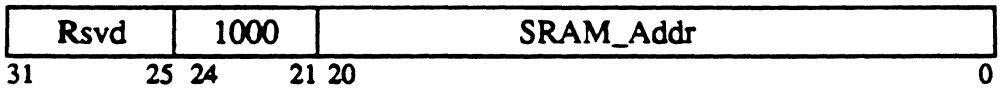
Rsvd: Reserved. Read as zero and ignored on a write.

Note that access to the cache tags is affected by the cache size information in the Control register (See section 4.4.7 on page 47).

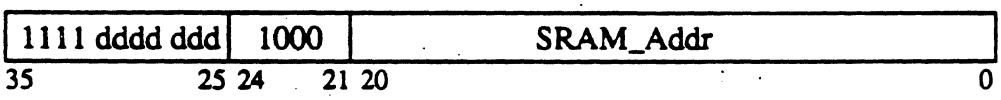
The tags are not modified by a reset. It is the programmer's responsibility to initialize the V and P bits of all tag entries to 0.

4.4.2 External Cache Data

The External Cache static RAMs may be read and written using all SPARC addressable quantities (byte, half-word, word and double-word). The address format for ASI 0x2 accesses is:



The address format for ECSR space accesses is:



where:

SRAM_Addr: Address referenced within the cache data array. Bit 20 is ignored if the configured cache size is 1 M-byte (which is the normal SunDragon and Scorpion configuration). Bits [20:19] are ignored if the configured cache size is 512 K-bytes.

Rsvd: Reserved. All reserved bits are ignored. They should be 0.

Note that access to the cache data is affected by the cache size information in the Control register (See section 4.4.7 on page 47).

4.4.3 Block Copy and Block Zero

4.4.3.1 Principles of Operation

The CC provides hardware support for block copy and block zero. The source and destination for data may either be in Memory Space or I/O Space.

The hardware consists of a 64-byte Stream Data register (SD) and two address registers, the Stream Source Address register (SSA) and the Stream Destination Address register (SDA). Each operation transfers 64 bytes at a time and all transfers must be aligned on a 64-byte boundary.

A Block_Load operation is initiated by a processor store into the SSA register. A Block_Load causes 64 bytes to be transferred from Memory or I/O Space (as determined by SSA) into the SD register. A Block_Store operation is initiated by a processor store into the SDA register. A Block_Store causes 64 bytes to be transferred from the SD register into Memory or I/O space (as determined by SDA).

The hardware provides interlock between stores to SSA and SDA *only when referenced via ASI 0x2* in such a way that Block_Load and Block_Store operations appear instantaneous, although they are pipelined to provide maximum throughput. This interlock allows a bcopy() loop to be simply a sequence of stores alternatively to the SSA and SDA registers in ASI 0x2, and a bzero() loop to be simply a sequence of stores to the SDA register.

The interlock does not extend to the Stream Data register. In consequence, when the programmer wants to inspect the contents of the Stream Data register after doing a Block_Load, he must poll the RDY bit of the SSA register for completion. In the same way, after initiating a Block_Store, the programmer must poll the RDY bit of the SDA register prior to modify manually the contents of the Stream Data register. The interlock does not apply either to references to the SSA and SDA register made using ECSR space, thus also requiring polling in this case.

NOTE: The block copy hardware support operates exclusively on physical addresses, and is completely independent of the address translation performed by the MMU. The programmer is responsible to emulate all required MMU functions, such as mapping virtual to real addresses, enforcing protection, locking pages in memory, and marking PTEs for the target of a block copy operation as referenced and modified

The Stream Data register **is not part of the shared memory image.** However, it is important to note that Block_Load and Block_Store operations for which C=1 take place from and to the Shared Memory Image.

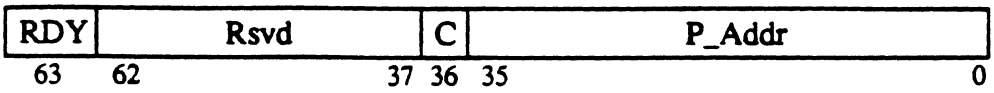
Block transfer operations to and from CSR space, ECSR space and Local space are not supported and will result in a bus time-out error code.

4.4.3.2 Stream Data Register

The Stream Data register is 64 bytes wide. It may be read and written as doublewords. Accesses to the Stream Data register are not interlocked with the operation of the Stream Address registers. Thus, the programmer must check that the Stream Address registers are not busy before accessing the Stream Data register. In normal operation, there is no need to access the Stream Data register (the bzero() function can initialize the Stream Data register through a Block_Load operation).

4.4.3.3 Stream Source Address Register

This register may be read and written as a double-word. A processor store to the Stream Source Address register triggers a Block_Load operation. The register format is:



where:

RDY: Ready. This bit is zero when a Block_Load operation is pending, one otherwise. This bit is read-only. It is ignored on a write.

Rsvd: Reserved. This field is ignored on a write and read as zero.

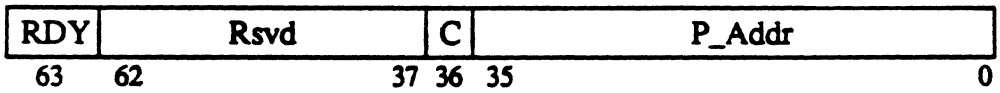
P_Addr: Physical Address for the Block_Load operation. Bits [5:0] are ignored, and all operations are carried out on 64-byte blocks aligned on a 64-byte boundary.

C: Cacheable/Non-Cacheable bit. If this bit is set to 1, the Block_Load operation is taking place in Memory Space. If it is set to zero, the operation is taking place in I/O Space.

Stores to the Stream Source Address register from ECSR space are *not* interlocked. If a store to the Stream Source Address register is done from ECSR space when RDY=0, the results are unpredictable.

4.4.3.4 Stream Destination Address Register

This register may be read and written as a double-word. A processor store to the Stream Destination Address register triggers a Block_Store operation. The register format is:



The various bit fields have the following meaning:

RDY: Ready. This bit is zero when a Block_Store operation is pending, one otherwise. This bit is read-only. It is ignored on a write.

Rsvd: Reserved. This field is ignored on a write and read as zero.

P_Addr: Physical Address for the Block_Load operation. Bits [5:0] are ignored, and all operations are carried out on 64-byte blocks aligned on a 64-byte boundary.

C: Cacheable/Non-Cacheable bit. If this bit is set to 1, the Block_Store operation is taking place in Memory Space. If it is set to zero, the operation is taking place in I/O Space.

Stores to the Stream Destination Address register from ECSR space are *not* interlocked. If a store to the Stream Destination Address register is done from ECSR space when RDY=0, the results are unpredictable.

4.4.3.5 Programming Notes

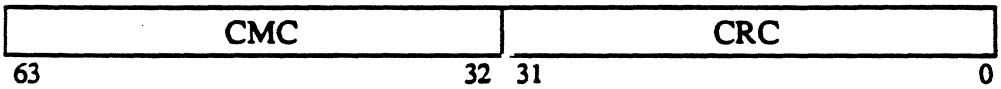
When an error occurs during a Block_Load or Block_Store operation, the error is logged in the Error register and a level 15 interrupt is issued to the processor.

Due to the pipelining of the Block_Store operation, an error occurring on a Block_Load operation (e.g. an ECC failure) may have written incorrect data on the corresponding Block_Store.

Block_Load and Block_Store operations do not follow the TSO/PSO memory model, and are not ordered with respect with previous or following load and store operations. Software is responsible to wait for RDY to become one in the stream address registers after completion of a block copy sequence to guarantee ordering with other operations. Note that this explicit busy loop is anyway necessary to guarantee that all errors resulting from the block copy operation are properly identified.

4.4.4 Reference/Miss Count Register

This register allows to measure the hit ratio for the External Cache. It may be read and written as a double-word. Its format is:



The two fields have the following meaning:

CMC: Cache Miss Count. This 32-bit counter is incremented whenever there is a miss in the External Cache.

CRC: Cache Reference Count. This 32-bit counter is incremented whenever the attached processor accesses the External Cache. When bit [31] becomes one, CMC and CRC are frozen until bit [31] is cleared in software.

When the RC bit of the CC Control register (See section 4.4.7 on page 47) is zero, all processor references are counted in CMC and CRC. If the RC bit is one, only read references and read misses are counted.

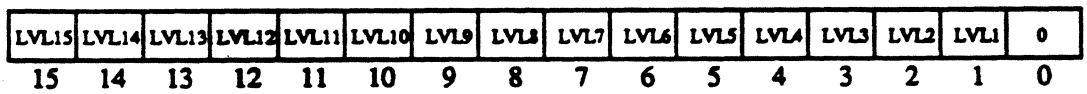
NOTE: CMC is not protected against overflow. If CRC and CMC are both initialized to 0, then CMC is always smaller than CRC and thus cannot overflow.

4.4.5 Interrupt Registers

This section describes the structure of the interrupt registers. The interrupt model for Sun-4D systems is discussed in Chapter 8.

4.4.5.1 Interrupt Pending Register

This register records the 15 pending interrupt levels defined by the SPARC architecture. It is accessible as a half-word and is read-only. Its format is:

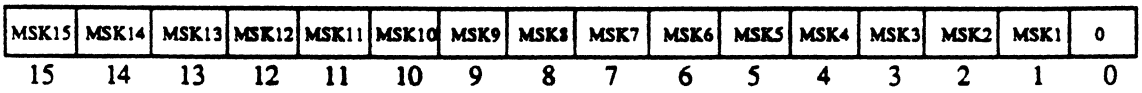


LVL*n* is set to one if an interrupt is pending at level *n*. Bit[0] is read as zero. The CC implements the prioritizing logic to encode the highest pending interrupt level on IRL [3:0] as specified by the SPARC architecture [1].

The Interrupt Pending register is cleared on reset.

4.4.5.2 Interrupt Mask Register

This register permits to mask out interrupts at certain levels. It may be read and written as a half-word. Its format is:



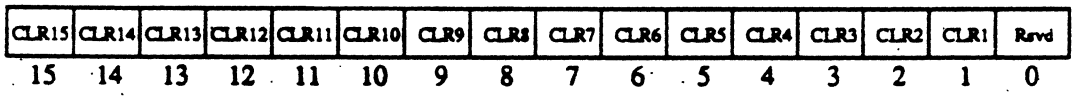
If MSK*n* is set to one, Level *n* interrupts are disabled. If MSK*n* is set to zero, Level *n* interrupts are enabled.

The Interrupt Mask is set to all ones (except bit 0) on reset in order to mask all interrupts.

NOTE: Level 15 interrupts can be disabled to prevent critical code from being interrupted.

4.4.5.3 Interrupt Pending Clear

This pseudo-register is used to clear bits of the Interrupt Pending register. It is accessible as a half-word and is write-only. The format is the same as the format of the Interrupt Pending register:

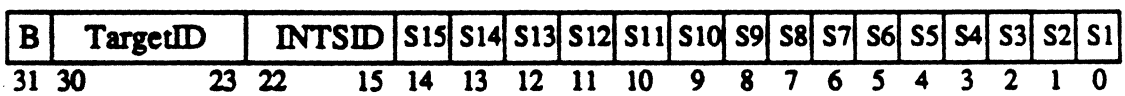


Writing a one in bit CLR*n* clears bit LVL*n* in the Interrupt Pending register. Writing a zero in bit CLR*n* has no effect. Bit[0] is ignored.

4.4.5.4 Interrupt Generation Register

This pseudo-register is used to issue interprocessor interrupts and to define the processor which receives SBus interrupts. It is accessible as a word and is write-only. The Interrupt Generation register is *not* accessible from ECSR space.

There are two formats for this register. The first format is used to issue interprocessor interrupts and is as follows:



where:

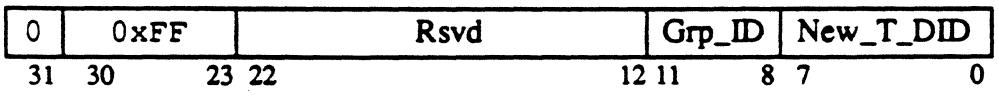
S1 to S15: Required interrupt level in decoded form. If S*n* is set to 1, the target processor will receive a Level *n* interrupt. Usually, a single bit in this group is set. If multiple bits are set, the target will receive interrupts on all the specified levels. If no bit is set, the target will not be interrupted.

INTSID: Interrupt Source Identifier. This field identifies the source of the interrupt. See section 8.2 on page 156 for the usage of this field.

TargetID: Device Identifier of target processor unit. This field is ignored if B is set. The 4 most significant bits of this field may not be equal to 0xF (this is not a restriction since DeviceIDs are restricted by the architecture to the range 0x00 to 0xDF).

B: Broadcast. If this bit is set to 1, the interrupt will be sent to all processors. If it is set to 0, the interrupt will be sent only to the processor specified by TargetID.

The second format is used to specify the processor which handles SBus interrupts. This format is as follows:



The bit fields have the following meaning:

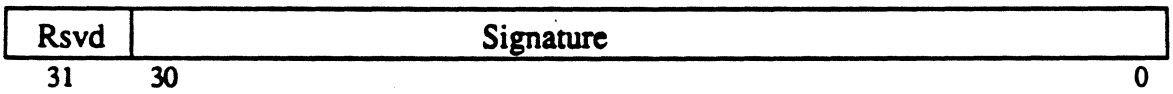
New_T_DID: New Target DeviceID. This field specifies the processor to which certain SBus interrupts will be directed.

Grp_ID: Group Identifier. This field identifies a subset of the SBusses in the system. The SBusses in this group will direct their interrupts to the processor attached to the BW with a DeviceID equal to New_T_DID.

See section 6.4.2.7 on page 140 for the use of this second format.

4.4.6 BIST Register

The BIST (built-in self-test) register is readable and writable. It is accessible only via ASI 0x2. On a write, the data is ignored and the Cache Controller initiates a built-in self-test sequence which provides a 31-bit polynomial signature. The self-test is initiated after all pending operations have completed, and lasts approximately 1 second. A read of the BIST register returns data in the following format:



The various bit fields have the following meaning:

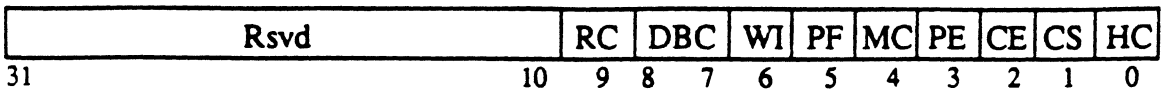
Rsvd: Reserved. This field is read as zero.

Signature: Signature obtained from the last built-in self-test performed. The state after a reset is unspecified. Following a BIST operation, the value of the signature for a chip which operates properly is a constant which depends exclusively on the revision of the chip as indicated in the MREV field of the Component Identification register (See section 4.4.11 on page 52). The correct value will be provided after chip manufacturing.

NOTE: During a BIST operation, the Cache Controller is logically disconnected from its busses and does not obey anymore the consistency rules. As a consequence, stores to the BIST register should be performed only when the External cache is disabled and no other processor tries to perform a DeMap operation. Moreover, after a BIST operation, the state of the cache controller is the same as after a system reset (See section 4.4.12 on page 53).

4.4.7 Control Register

This register may be read and written as a word. Its format is:



The various bit fields have the following meaning:

Rsvd: Reserved. This field is ignored on a write and read as zero.

RC: Read Reference Count. When this bit is one, only read references are counted in the Reference/Miss Count register (See section 4.4.4 on page 44). When this bit is zero, all references are counted.

DBC: Number of active Dynabusses in system. The encoding is (grayed out configurations are not supported in current implementations):

DBC	# of Dynabusses
00	1
01	2
1X	4

WI: Write Invalidation. When this bit is set to one, a write issued by the attached Viking processor to a shared subblock causes an invalidation of copies present in other External Caches. However, in most cases if an operation is pending on a subblock in another cache, the invalidation is not performed and the subblock is updated. This feature is provided to attempt controlling the artificial sharing which occurs when a process/thread migrates from processor to processor during its execution. This feature does not change the memory model.

PF: Prefetch Enable. If this bit is set to one, the CC prefetches subblocks: when the processor hits a subblock in the External Cache, the next sequential subblock *in the same line* is fetched from memory if it is not already cached.When PF is zero, subblocks are loaded in the External Cache on demand only. PF has no effect if the internal Viking caches are not enabled.

MC: Multiple Commands Enable. When this bit is set to one, the CC is allowed to issue multiple transactions without waiting for their completion.

NOTE: Setting MC=0 does not make stores synchronous in terms of error management (See section 7.1.1.6 on page 144), it just limits the number of outstanding stores to 1.

PE: Parity Enable. This bit controls parity generation and checking on the Viking data bus from CC's point of view. When it is set to one, even parity is checked on data transferred to CC (External cache reads for snooping, processor I/O writes/swaps) and even parity is generated on data transferred from CC (External cache refills, shared writes - inc. foreign writes, processor I/O reads). When this bit is cleared, parity is not checked on transfers to CC and odd parity is generated on transfers from CC. Note that parity of accesses by the processor to the External Cache are controlled by the PE bit of the MMU Control register in Viking.

NOTE: In normal operations, parity mode should be the same in CC and Viking, i.e. both enabled if the external cache has parity (normal Sun-4D systems case) or both disabled if the external cache does not have parity. If parity is disabled in CC and enabled in Viking, I/O space reads will result in a Viking parity error. It must be noted that, on a local software reset (See section 4.4.9 on page 49), Viking parity mode is disabled whereas CC parity mode is unchanged, which requires careful programming to avoid problems.

CE: Cache Enable. The External Cache is enabled when this bit is set to one, it is disabled otherwise.

CS, HC: Cache Size, Half Cache. Indicates the size of the external cache according to the following table (configurations not currently supported are grayed out):

CS	HC	External Cache Size
0	0	1 M-byte
0	1	512 K-byte
1	0	2 M-byte
1	1	Reserved

The Control register is cleared on reset.

4.4.8 Status Register

This register provides information on the internal state of the CC. It may be read and written as a double-word.

NOTE: Write access to the Status register is provided *exclusively* for diagnostics use. Writes into the Status register change internal CC information which is critical for proper operation and may generate hardware deadlocks (for example, setting SPC to a higher value than the current one will cause a deadlock in TSO mode).

The format of the Status register is:

The various bit fields have the following meaning:

Rsvd: Reserved. This field is ignored on a write and read as zero.

SXP: Store Exception Pending. This bit is equal to one when a store operation from Viking has failed, but Viking has not retried the store yet. This bit is cleared on a Local Software reset (See section 4.4.9 on page 49).

SM: Synchronous Mode.This bit is equal to one when CC operates on the same clock as Viking, it is 0 otherwise. This bit is read-only, as it reflects the state of the SYNC pin of the chip.

NCSID: Non-Cacheable Store Bus ID. This field records the index of the Dynabus on which the last I/O store (store marked as non-cacheable) was done. If the next I/O store is directed to the same Dynabus and lies within the same page (as indicated by NCSPA), the access is issued without waiting for the completion of the

previous I/O store. This means that multiple I/O stores can be pending as long as they are directed to the same Dynabus and within the same page. This feature speeds-up programmed I/O writes while maintaining store ordering.

NCSPA: Non-Cacheable Store Page Address. This field records the page address on which the last I/O store (store marked as non-cacheable) has been done (see explanation for NCSID).

NCSPC: Non-Cacheable Pending Store Count. This 4-bit counter keeps track of the number of pending I/O writes. It is incremented when an I/O store is issued by the CC on XBus, and is decremented when the reply is received from a Bus Watcher. This counter is used together with SPC to implement the STBAR primitive (See Chapter 2).

SPC: Store Pending Count. This 4-bit count keeps track of the number of pending stores to shared subblocks. It is incremented when a store to shared subblock is issued by the CC on XBus, and is decremented when the reply is received from a Bus Watcher.

BC: Boot Communication. Used for communication between Viking and the service processor. This bit is read-write and is accessible through a JTAG shadow loop (See section 10.3.3 on page 183).

NOTE: Writing into the BC bit should be attempted only when fields SXP, NCSPC, SPC, WP, RP and PP are zero. To ensure this, it is necessary to turn off prefetch from both Viking and CC, and wait for all those fields to become zero without issuing any store operation during the wait loop.

WP: Write Miss Pending. This bit is set to one when a processor store encounters an External Cache miss, and is set to zero when miss processing completes.

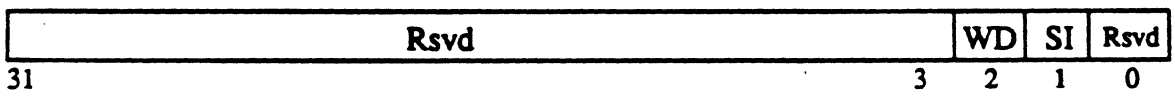
RP: Read Miss Pending. This bit is set to one when a processor load encounters an External Cache miss or accesses I/O space, and is set to zero when miss processing completes, or when the I/O space access completes.

PP: Prefetch Pending. This bit is set to one when a prefetch operation is initiated and is set to zero when it completes.

The Status register is cleared on reset.

4.4.9 Reset Register

This register may be read and written as a word. Its format is:



where:

WD: Watchdog Reset. This bit is set to one when the CC detects a Viking watchdog reset. Writing a one into this bit clears it. Writing a zero has no effect.

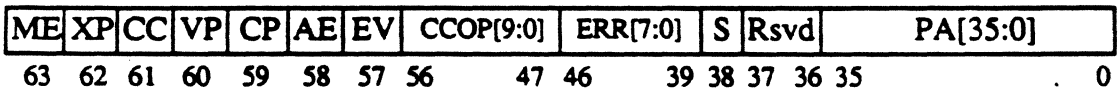
SI: Local Software Reset. When a one is written into this bit, the CC initiates a Local Software reset. When a zero is written into this bit, there is no side effect.
Before issuing a local software reset, the programmer should ensure that all pending writes have completed (fields NCSPC, SPC, WP, RP, PP of the Status register, section 4.4.8 on page 48, all zero). During a Local Software reset, CC prevents the processor from issuing additional memory operations, clears the SXP field of the Status register, clears the WD bit of the Reset register (SI remains one), and resets the Viking processor. No other CC state is modified.

Rsvd: Reserved. Reserved fields are ignored on a write and read as zero.

The Reset register is cleared on *System* reset, but only the WD bit is cleared on a local software reset.

4.4.10 Error Register

The Error register records information on errors detected by the CC (See Chapter 7 for a detailed explanation of error handling). It may be read and written as a double word. All bits in this register are write-one-to-clear, i.e. on a write to the Error register, bits into which a one is written are cleared whereas bits into which a zero is written are not modified. The format of the register is:



The various bit fields have the following meaning:

ME: Multiple Errors. This bit is set to one when, on occurrence of an error, the corresponding error bit (XP, CC, VP, CP, AE) is already set.

XP: XBus Parity Error. This bit is set to one when a parity error is detected on XBus. No error information is logged. This error causes a system watchdog reset.

CC: Cache Consistency Error. This bit is set to one when the CC detects an unexpected status for a subblock (See the CC documentation [19] for the error cases detected). If EV is not set, error information is logged in the CCOP, S and PA fields, and EV is set. This error causes a system watchdog reset.

VP: Viking Parity Error. This bit is set to one when a parity error is detected on the Viking bus during a Viking write operation. If EV is not set, error information is logged in the ERR field, and EV is set. This error generates a write buffer trap to the attached processor, with an error code UD (Undefined Error).

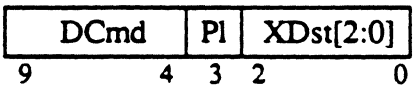
CP: Cache Parity Error. This bit is set to one if a parity error is detected when an owned subblock is read by CC. This can occur on a read from another processor or I/O device (foreign read) or when the block needs to be written back to memory (write-back). If EV is not set, error information is logged in the PA, CCOP and ERR fields and EV is set. This error issues a level 15 interrupt to the attached processor and returns a bus error with code UC to the requesting processor or I/O

device if the error was detected on a remote read. Note that the value of CCOP allows to find out whether the error occurred on a write-back or on a foreign read (See CCOP field on page 51).

AE: Asynchronous Error. This bit is set to one if an error is reported to the CC for an operation that has already been reported as complete to the processor (stream transfer or write to I/O Space). If EV is not set, error information is logged in the CCOP, ERR, S and PA fields, and EV is set. Note that the value of CCOP allows to find out whether the error occurred on a stream transfer or a write to I/O space (See CCOP field on page 51). This error issues a level-15 interrupt to the attached processor.

EV: Error valid. When EV is equal to one, the fields CCOP, ERR, S and PA contain detailed error information.

CCOP: Cache Controller Operation. This field is significant only for errors {CC, CP, AE}. It retains part of the XBus header cycle, in the following format:



where DCmd is the XBus data command, Pl the packet length and XDst the XBus destination ID. Refer to the XBus specification [12] for full details on the encoding of those bit fields. The "normal" cases are provided in the following two tables.

The following table indicates the possible values of DCmd when an asynchronous error is logged (AE=1):

DCmd (AE=1)	Error source
000101	Block_Load from Memory space
001111	Block_Store to Memory space
010011	Store to I/O space
010101	Block_Load from I/O space
010111	Block_Store to I/O space

The following table indicates the possible values of DCmd when a cache parity error is logged (CP=1):

DCmd (CP=1)	Error source
000100	Read by SBus stream device
000110	Write back by External cache to memory
001100	Consistent foreign read by other processor or SBus device
010000	Foreign read to cache data via ECSR space

ERR: Error. This field is significant for errors {VP, CP}, where it contains the parity bits, and for error AE. In the latter case, only bits [2:0] are used, with the following encoding:

ERR[2:0]	Indicates
000	Reserved
001	UC (Uncorrectable Error)
010	TO (Time-Out)
011	BE (Bus Error)
100	UD (Undefined Error)
101	Reserved
110	Reserved
111	Reserved

S: Supervisor. This field is significant for error AE. If is this bit is equal to one, the access in error was issued in supervisor mode, otherwise it was issued in user mode (**undefined** in Sun-4D systems).

Rsvd: Rsvd. Read as zero and ignored on a write.

PA: Physical Address. This field is significant for errors {CC, CP, AE}. If an AE error is reported with code TO in the ERR field, PA[11:0] may be incorrect (i.e. only the physical page number part of PA is guaranteed to be correct).

The Error Register is *not* modified on reset.

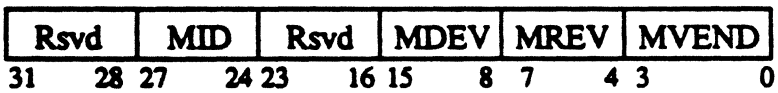
The Cache Controller will force a level 15 interrupt to the attached processor as long as the CP or AE bits are one. CP and AE must thus be cleared before bit LVL15 of the Interrupt Pending register (See section 4.4.5.1 on page 44) is cleared.

NOTE: It is possible to have both AE and CP set in the Error register. In that case, the value of the CCOP field may be used to determine for which of the two errors the log information was recorded (see DCMD subfield description in this section).

NOTE: When CC detects a fatal error (XP or CC) it sends an XBus request to the corresponding BW(s) which then request a system watchdog reset. If the BW(s) are frozen (FZN=1 in the Dynabus CSR) when the fatal error is detected no system watchdog reset takes place because BW ignores all external requests.

4.4.11 Component Identification Register

This register is accessible as a word and is read-only. Its format is:



The various bit fields have the following meaning:

Rsvd: Reserved. Reads as zero.

MID: Module ID. Not used in Sun-4D systems.

MDEV: MBus device number. Not used in Sun-4D systems.

MREV: Device revision number. This field contains the revision number of the chip, equal to 0x0. The same value can be accessed as the 4 most significant bits of the JTAG component identification register.

MVEND: MBus vendor number. Not used in Sun-4D systems.

The contents of fields MID, MDEV and MVEND are unspecified in Sun-4D systems.

4.4.12 Reset state of the Cache Controller

The Cache Controller state is reset *exclusively* by a system reset (See section 9.1.1 on page 169). The CC is *not* reset by a Viking watchdog reset (which only sets WD=1 in the CC Reset register) or a local software reset (which only sets WD=0, SI=1 in the CC Reset register).

On a system reset, the CC takes the following actions:

- Reset the Viking processor.
- Disable the External Cache.
- Reset all state-machines to idle.
- Reset all internal queues.
- Clear the Interrupt Pending register, Control register, Status register and Reset register, and set the Interrupt Mask register to all ones (except bit 0)

4.5 Bus Watcher

There are two versions of the Bus Watcher, which differ in the cache size they support. BW (first version) supports 512K-Bytes of External Cache, thus allowing for a 1 M-Byte External Cache in a two-Dynabus system. BWP (second version) supports 1 M-Byte of External Cache directly, thus allowing for a 1 M-Byte External Cache in a single-Dynabus system. BW and BWP present the same programming model, except for the difference in bus tags size and corresponding control bits. In the following, BW refers to both versions of the Bus Watcher unless specified otherwise.

NOTE: It is intended that only BWP will be used in FCS systems, but support for BW is required in the lab for bring-up and for alpha systems.

The Bus Watcher provides access to the External Cache bus tags via Local space and ECSR space, and to a set of internal registers via Local space and CSR space. Access to the tags is described in section 4.5.1 on page 54.

When the registers are accessed via Local Space, the 16 most significant bits of the physical address must be PA[35:20]=1111 1111 1111 1111_2. Using Local space allows BW resources like the Counter-Timers to be mapped at a fixed location.

When the registers are accessed via CSR space, the 16 most significant bits of the physical address must be PA[35:20]=1111 1110 dddd $dddd_2$, where

dddd dddd is the DeviceID of the referenced BW. All BWs in a given Processor Unit use the same DeviceID.

Bits [9, 8] of the physical address are used to select which BW is addressed (See section 3.2.5 on page 26).

All BW registers must be accessed with the specified data size. Reads are supported for all data sizes. Writes with a size different from the specified one return a time-out error (See section 7.2.3.1 on page 148). All atomic operations (SWAP, LD-STUB) cause a time-out error.

The address map for the BW registers is as follows:

CSR Space: PA[35:20] = 1111 1110 dddd dddd$_2$ Local Space: PA[35:20] = 1111 1111 1111 1111$_2$			
PA[19:0]	**Size**	**Access**	**Description**
0x00000	W	RO	Component ID Register
0x00008	D	R/W	Dynabus Control and Status Register
0x00010	D	R/W	DynaData Register
0x01000	W	R/W	Control Register
0x01040 to 0x01078	H	R/W	Interrupt Table
0x01080 to 0x010B8	H	WO	Interrupt Table Clear
0x010C0	H	R/W	Prescaler Register
0x02000	W D	R/W	Profiling Timer Limit Register User Timer Register
0x02008	W	R/W	Profiling Timer Non-destructive Limit
0x02010	W	R/W	Profiling Timer Counter
0x02018	W	R/W	Profiling Timer Control
0x03000	W	R/W	Tick Timer Limit Register
0x03008	W	R/W	Tick Timer Non-destructive Limit
0x03010	W	R/W	Tick Timer Counter

Note that bits [9:8] of the address are the bus selector in CSR Space, and that this address map should thus be interleaved 4 times with a step of 256 bytes.

4.5.1 External Cache Bus Tags

The External Cache Bus tags are direct-mapped. BW provides a 2 K-entry tag memory, with each entry corresponding to four External Cache sub-blocks of 64 bytes each, whereas BWP provides a 4K-entry tag memory.

BW and CC can support five different External Cache configurations:

- One M-byte External Cache with two Dynabusses: normal SunDragon configuration.

- 512 K-byte External Cache with one Dynabus: Scorpion configuration when BWP is not available, single-bus SunDragon configuration when BWP is not available (Single-bus SunDragon configuration is used when there is a permanent failure of the two Dynabusses).
- 512 K-byte External Cache with two Dynabusses: SunDragon degraded configuration when half the CC tags are unavailable (this configuration is used when there is a permanent failure in the External Cache).
- Two M-byte External Cache with four Dynabusses: not currently supported (this configuration is reserved for future implementations of high-end Sun-4D systems).
- One M-byte External Cache with four Dynabusses: not currently supported

BWP and CC can support the following additional configurations:

- One M-byte External Cache with one Dynabus: normal Scorpion configuration, single-bus SunDragon configuration.
- Two M-bytes External Cache with two Dynabusses: not currently supported.
- Four M-byte External Cache with four Dynabusses: not currently supported.

There must be a **one to one** mapping between a tag entry in a BW/BWP and a tag entry in the Cache Controller. Therefore, the number of enabled entries in the Cache Controller and the number of enabled entries in the BWs/BWPs must be identical. The Cache Controller contains 8 K-entries and can be configured to have one fourth, half or all entries enabled with bits CS and HC of the CC Control register (See section 4.4.7 on page 47). BW and BWP contain respectively 2 K-entries and 4 K-entries, and can be configured to have 1K to 4K entries enabled based on the value of the fields TgS and 4K of the BW Control register (See section 4.5.5 on page 66). The number of Dynabusses (which is equal to the number of BWs/BWPs in the processor unit) is specified by the field DNum of the Dynabus Control and Status register (See section 4.5.3 on page 59).

The table below shows the settings for the various configurations:

CS	HC	DNum	TgS	4K	External Cache Size	Total number of tag entries	Number of BWs	Number of BWPs
0	1	00	0	0	512 K-Byte	2 K	1	1
0	0	00	0	1	1 M-Byte	4 K	N/A	1
0	1	01	1	0	512 K-Byte	2 K	2	2
0	0	01	0	0	1 M-Byte	4 K	2	2
1	0	01	0	1	2 M-Byte	8 K	N/A	2
0	0	11	1	0	1 M-Byte	4 K	4	4
1	0	11	0	0	2 M-Byte	8 K	4	4
N/A	N/A	11	0	1	4 M-Byte	16 K	N/A	4

Configurations which are grayed out are impossible or not supported in current Sun-4D implementations.

NOTE: It is possible to build systems where multiple BWs are connected to the same Dynabus. For example it is possible to build a 1 M-byte External cache with two (or four) BWs connected to the same Dynabus. These configurations are not supported in current Sun-4D systems.

The External Cache Bus tags are not invalidated on a system reset. They must be initialized by software.

4.5.1.1 Address Format

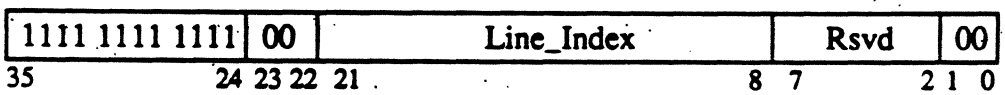
When the External Cache Bus tags are accessed in ECSR space, the 13 most significant bits of the physical address must be PA[35:23]=`1111 dddd ddd0` 1_2, where **dddd ddd** designates the 7 most significant bits of the DeviceID for the corresponding processor unit.

When they are accessed in Local space, the 13 most significant bits of the physical address must be PA[35:23]=`1111 1111 1111` 0_2.

The External Cache Bus tags should be read and written as words only. A read with a smaller size is performed normally. In the case of a double-word read, the non-existent word is undefined. Writes with a size different from a word cause a time-out error. All atomic operations (SWAP, LDSTUB) on the tags cause a time-out error.

The address format in ECSR space is:

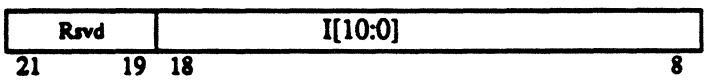
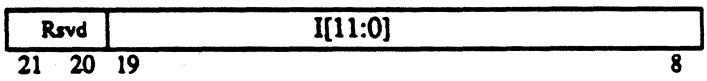
The address format in Local space is:

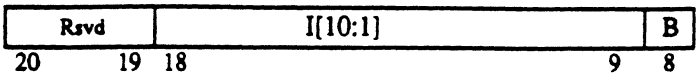
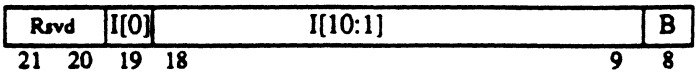
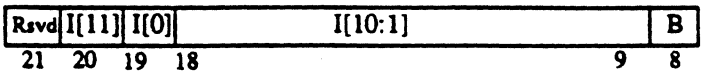
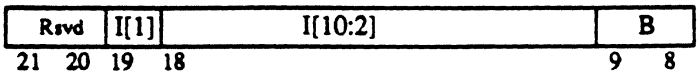
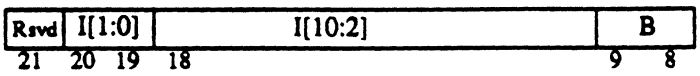
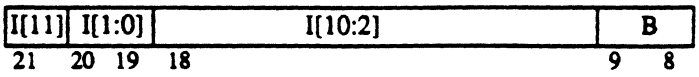


Line_Index designates the index of the required tag. Note that the above format provides 14 bits for the Line_Index field, which allows up to 16 K-entries of tags. For configurations with less than 16 K-entries, the high-order bits of Line_Index should be set to 0.

NOTE: For diagnostics purposes, it is useful to understand how Line_Index maps onto the index inside the BW or BWP Tag RAM, and how the Tag RAM index relates to the physical address which is cached.

The Tag RAM inside a BW/BWP is 2K or 4K entries, and the index is thus represented as a 12 bit number, with the most significant bit ignored in BW. The same mapping is used to transform addresses to Tag RAM indices for ECSR access to the tags and for normal cache operations. The mapping from Tag RAM index to Line_Index is provided by the following table:

DNum	TgS	4K	Configuration	Line_Index
00	0	0	512 KB, 1 BW/BWP	
00	0	1	1 MB, 1 BWP	

DNum	TgS	4K	Configuration	Line_Index
01	1	0	512 KB, 2 BW/BWP	
01	0	0	1 MB, 2 BWP	
01	0	1	2 MB, 2 BWP	
11	1	0	1 MB, 4 BW/BWP	
11	0	0	2 MB, 4 BW/BWP	
11	0	1	4 MB, 4 BWP	

The various bit fields have the following meaning:

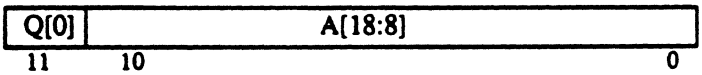
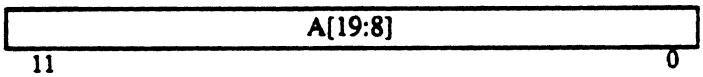
Rsvd: Reserved bits must be specified as 0.

B: Bus select. The BWs are interleaved on a 256-byte boundary. Bit [8] (two BWs) or bits[9:8] (four BWs) of the physical address must be set to select the BW.

I: Index for the referenced entry. For BW (2K entries), the index has 11 valid bits; for BWP (4K entries), 12 bits are valid. Index bits not shown above are obtained from the Control register (See section 4.5.5 on page 66) bits Q[1:0]. For example, in the 1 M-byte, 4 BW configuration, only bits I[10:1] are shown above. Bits I[11] and I[0] are obtained from the Control register. The usage of the Q bits is summarized in the table below:

Configuration	Usage of Q bits
512K-Byte, 1 BW/BWP 1 M-Byte, 2 BW/BWP 2 M-Byte, 4 BW/BWP	I[11] is Q[0] for BWP I[11] is undefined for BW
512K-Byte, 2 BW/BWP 1 M-Byte, 4 BW/BWP	I[0] is always Q[0] I[11] is Q[1] for BWP I[11] is undefined for BW
1 M-Byte, 1 BWP 2 M-Byte, 2 BWP 4 M-Byte, 4 BWP	Q[1:0] not used

The following table provides the inverse mapping, from address to Tag RAM index:

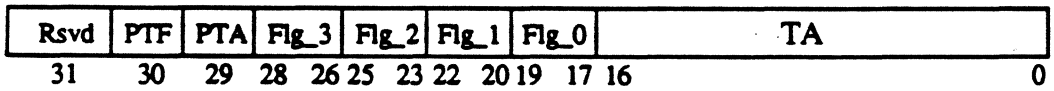
DNum	TgS	4K	Configuration	Tag RAM Index
00	0	0	512 KB, 1 BW/PWP	
00	0	1	1 MB, 1 BWP	

DNum	TgS	4K	Configuration	Tag RAM Index
01	1	0	512 KB, 2 BW/BWP	Q[1] A[18:9] Q[0] (11 10 1 0)
01	0	0	1 MB, 2 BW/BWP	Q[0] A[18:9] A[19] (11 10 1 0)
01	0	1	2 MB, 2 BWP	A[20] A[18:9] A[19] (11 10 1 0)
11	1	0	1 MB, 4 BW/BWP	Q[1] A[18:10] A[19] Q[0] (11 10 2 1 0)
11	0	0	2 MB, 4 BW/BWP	Q[0] A[18:10] A[20:19] (11 10 2 1 0)
11	0	1	4 MB, 4 BWP	A[21] A[18:10] A[20:19] (11 10 2 1 0)

Q bits from the Control register are used to provide index bits which are not dependent on the address for each configuration.

4.5.1.2 Tag Format

The format of a tag is:



where:

Rsvd: Reserved. This bit is read as zero and ignored on a write.

PTF: Flags Parity. Odd parity on the Flags fields for the entry.

PTA: Address Parity. Odd parity on the Tag Address field for the entry.

Flg_3: Flags for sub-block 3. See encoding in the table below.

Flg_2: Flags for sub-block 2. See encoding in the table below.

Flg_1: Flags for sub-block 1. See encoding in the table below.

Flg_0: Flags for sub-block 0. See encoding in the table below.

The encoding of the flags is indicated by the following table:

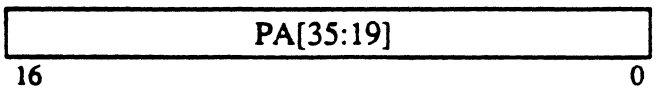
Flg_x	Sub-block State
0	Not Shared, Not Owner, Not Retry
1	Not Shared, Not Owner, Retry
2	Not Shared, Owner, Not Retry
3	Reserved
4	Shared, Not Owner, Not Retry

Flg_x	Sub-block State
5	Shared, Not Owner, Retry
6	Shared, Owner, Not Retry
7	Invalid

Not Shared means that there are no copies of the sub-block in other processor caches. Shared means that there may be copies in the caches of other processors. Owner means that the sub-block was modified last by the attached processor. Retry means that a Read Block Dynabus transaction issued by this BW must be retried. For more details on the cache consistency protocol, the reader may refer to [4].

The Flags must be initialized by software after a power-on reset. All tag entries must be written with the value `0x7FFE 0000`.

TA: Tag Address. This bit field has the following structure:

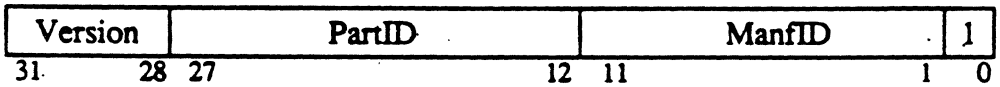


Note that all bits of TA are significant, independently of the configured External Cache size.

4.5.2 ComponentID Register

This register is read-only. It is accessible as a word.

The ComponentID register has the following structure:



The various bit fields have the following meaning:

Version: 4 bit version number.

PartID: Part ID. 16 bit part number.

ManfID: Manufacturer ID. 11 bit Sun JEDEC Identifier, set to **0x03E**.

The following PartIDs and versions are currently legal:

PartID	Version	ComponentID	Component
0xADB	1	0x10ADB07D	BW, 2K Tags
0xADB	2	0x20ADB07D	BW, 2K Tags
0xD39	1	0x10D3907D	BWP, 4K Tags

There is no functional difference between both versions of BW. The first version contains some logical bugs which are fixed in the second version. Only the second version will be shipped but the first version may be used for system bring-up.

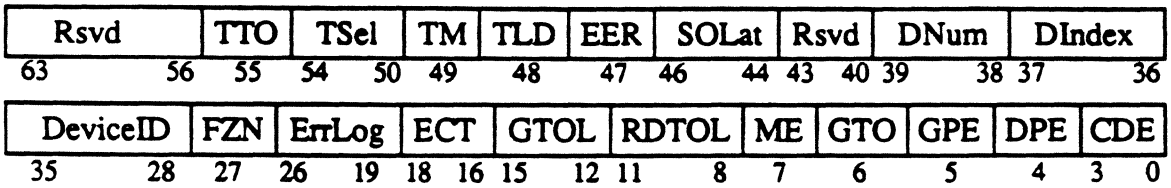
4.5.3 Dynabus Control and Status Register

This 64-bit register contains certain Dynabus parameters which must be loaded via JTAG after a power-on reset. It is also used to record information when a fatal er-

ror is detected by the Bus Watcher. A fatal error is an error that causes a system watchdog reset (See Chapter 7).

This register is accessible as a double word. It is also accessible via JTAG as part of shadow scan chain 0 (See section 10.3.4 on page 184).

The Dynabus Control and Status register has the following structure:



where:

Rsvd: Reserved. Reserved bits are ignored on a write and read as zero.

TTO: Test Time-Outs. When this bit is set to one, the time-out period indicated by the GTOL and RDTOL fields is divided by 1024. When this bit is set to zero, the time-out periods are computed normally. In normal operation, this bit *must* be set to *zero*. This bit is read-only.

TSel: Test Selector. This field indicates the test mode selected when TM is one. This field is read-only.

TM: Test Mode. When this bit is set to one, the chip verification test mode defined by TSel is enabled. When this bit is zero, the chip operates normally. This bit is read-only. It *must* be set to *zero* except for chip manufacturing test.

TLD: Top-Level Device. This bit must be set to zero if there is a second-level Dynabus cache above this BW in the Dynabus hierarchy. It must be set to one otherwise. In normal operation on current Sun-4D systems, this bit *must* be set to *one* through JTAG. This field is read-only. See [4] for details on the effect of TLD.

EER: Enable Error Reset. If this bit is set to one, any fatal error detected by the Bus Watcher causes a system watchdog reset. If this bit is set to zero when a fatal error is detected no system watchdog reset is issued. However, the error information is logged normally. This bit is read-only.

SOLat: Shared/Owner Latency. Latency in number of Dynabus cycles from the time an address appears at the input pins to the time the Shared/Owner signals are asserted at the output pins, minus 2 (See[4] for more informations on Dynabus). This field is read-only. In normal operation on current Sun-4D systems, this field must be initialized to 0x2 (corresponding to a 4 cycle Shared/Owner latency) through JTAG. Note that BW will not function properly if a value lower than 0x2 is used.

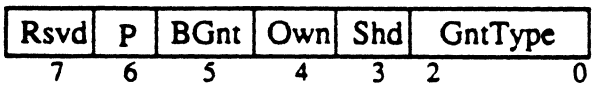
DNum: Dynabus(ses) Number - 1. Number of active Dynabus(ses) in the system minus one. This field is read-only. It is initialized through JTAG.

DIndex: Dynabus Index. This field indicates the number of the Dynabus to which this BW is connected. This field is read-only. It is initialized through JTAG.

DeviceID: Dynabus DeviceID. This field is read-only. It is initialized through JTAG. In current Sun-4D system the four most significant bits of this field must correspond to the board number.

FZN: Frozen. This bit behaves as if reserved on CSR space accesses (i.e. writes have no effect and reads return zero), but it is accessible via JTAG (See section 10.3.4 on page 184). FZN is set to one on a system reset. When FZN is one, BW remains in the reset state and can only be accessed using JTAG.

ErrLog: Error Log. This field is used to log the parity bits when a Dynabus parity error is detected. The 8 parity signals compute byte parity over the 64 Dynabus data signals. The position of the parity bit determines the byte it protects. The least significant bit protects the least significant byte of the data signals and so forth. The parity can be odd or even according to the type of Dynabus cycle (See [4] for more information). The ErrLog field is also used to log the value of the seven Board Arbiter lines when a parity error is detected on these lines. In this case, the structure of this field is the following:



The bit fields have the following meaning (see [9] for a detailed explanation of the arbiter signals):

GntType: Grant Type.
Shd: Shared. Logical OR of the Shared lines.
Own: Owner. Logical OR of the Owner lines.
BGnt: Board Grant.
P: Even Parity on the arbiter lines.
Rsvd: Reserved. Read as zero and ignored on a write.

This field cannot be written, even through JTAG, as long as fields GTO, GPE, DPE, CDE are nonzero.

NOTE: When a Dynabus parity error is detected during the very same cycle where FZN is cleared, the error is detected and recorded correctly (DPE is set to one) but the parity bits are not logged in the ErrLog field. The correct initialization sequence which guarantees that the parity bits are logged correctly is detailed in section 4.5.10 on page 71.

ECT: Enable Count. The value of this field specifies the type of events which are counted in the EC field of the DynaData register if there is no pending fatal error. It is readable and writable. The encoding is the following[1]:

ECT	Type of Events Counted
0	Counting Disabled
1	Nbr of cycles Pause has been asserted
2	Nbr of shared writes from this BW
3	Nbr of subblocks flushed by this BW
4	Nbr of cycles waiting for read miss reply

ECT	Type of Events Counted
5	Nbr of write-update cycles to CC (WS+WB)
6	Nbr of requests for owned data (RB) to CC
7	Nbr of writes to non-shared, non-owner subblock

GTOL: Grant Time-Out Log_Value. This field specifies the Grant Time-Out counter value in log base 2 of 1 K Dynabus clock cycles (the time-out period is divided by 1024 if bit TTO is one). This field is readable and writable. A Grant Time-Out is reported if the arbiter does not allocate the bus N Dynabus cycles after a bus request was issued. The Bus Watcher implementation guarantees that: $2^{(10+GTOL)} < N < 2*2^{(10+GTOL)}$.

RDTOL: Read Data Time-Out Log_Value. This field specifies the Read Data time-out counter maximum value in log base 2 of 1 K Dynabus cycles (the time-out period is divided by 1024 if bit TTO is one). This field is readable and writable.The BW supports three different time-out schemes (in the following, **RDTO** denotes a time equal to $2^{(10+RDTOL)}$):

- A time-out is reported if a demap (MMU flush) has not completed N Dynabus cycles after it was issued (See section 4.2.3 on page 36 for a description of demap in Sun-4D systems). The implementation guarantees that 8*RDTO < N < 16*RDTO.
- The second scheme is used for all Dynabus requests except WriteSingle and IO-Write transactions (See [4]). A time-out is reported if no reply packet of the same type is received N Dynabus cycles after a request was issued. The Bus Watcher implementation guarantees that RDTO < N < 2*RDTO.
- The third scheme guarantees that all issued WriteSingle and IOWrite requests get a reply. A WriteSingle request is issued when the processor writes a shared variable in Memory Space. An IOWrite request is issued for a write in I/O Space. The Sun-4D systems Memory Model (See Chapter 2) implies that multiple stores in Memory Space and I/O Space can be pending.

  In Total Store Ordering mode, a processor must wait for the completion of each write to a shared variable. In this case, a time-out is reported if no reply packet is received N Dynabus cycles after a WriteSingle was issued.
  In Partial Store Ordering mode a processor must wait for the completion of pending writes to shared variables only when a STBAR is executed (See section 2.4 on page 16). In this case, a time-out is reported if all WriteSingle reply packets have not been received N Dynabus cycles after the last WriteSingle request was issued.
  The Bus Watcher implementation guarantees that RDTO < N < 2*RDTO.

  Multiple I/O stores can be pending as long as they are directed to the same Dynabus and reference the same page. The processor must wait for the completion of the I/O store either when a STBAR is executed or when an I/O store is

1. Refer to the Dynabus specification [4] for more details on Dynabus transactions

directed to a different page or Dynabus. In this case, a time-out is reported if all IOWrite reply packets have not been received N Dynabus cycles after the last IOWrite was issued.

ME: Multiple Errors. This bit is set to one when multiple fatal errors have been detected. This bit is cleared by writing a one. In case of multiple errors, only the first one is logged. If multiple errors are detected in the same cycle, the following priority is used:

Priority	Error
1	Grant Parity Error
2	Dynabus Parity Error
3	Grant Time Out
4	Internal Errors

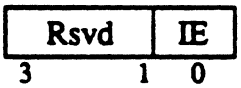
NOTE: If multiple internal errors occur within the same cycle, ME is not set to one, but multiple error bits may be set in the DynaData register (See section 4.5.4.2 on page 64).

DPE: Dynabus Parity Error. This bit is set to one when a parity error is detected on Dynabus for either a Data or a Header cycle. When a Dynabus parity error is detected, the faulting cycle is latched in the DynaData register and the parity signals are latched in the ErrLog field. This bit is cleared by writing a one.

GPE: Grant Parity Error. This bit is set to one when a parity error is detected on the arbitration signals. When such an error is detected, the value of the arbitration signals is latched in the ErrLog field. This bit is cleared by writing a one.

GTO: Grant Time-Out. This bit is set to one when the time-out counter for a Dynabus grant overflows. This bit is cleared by writing a one.

CDE: Client Device Errors. This field is used to report internal errors. The nature of the error is logged in the DynaData register. This field has the following structure:



The two bit fields have the following meaning:

Rsvd: Reserved. Read as zero and ignored on a write.

IE: Internal Error. This bit is set to one if one or multiple internal errors are detected. It is cleared by writing a one.

Notice that at most one of the fields DPE, GPE, GTO and CDE may be nonzero, and that usually, ME cannot be one if all of them are zero.

4.5.4 DynaData Register

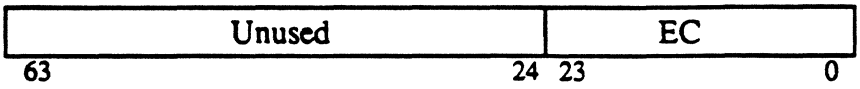
The DynaData register is accessible in read-write mode as a double-word. The DynaData register cannot be written, even through JTAG, as long as any error bit is set in the Dynabus Control and Status register (DPE, GPE, GTO, CDE).

This register serves three functions:

- Error logging: when a fatal error is indicated in the Dynabus Control and Status register (fields DPE, GPE, GTO, CDE non-zero), detailed error information is logged in the DynaData register
- Event counting: when no fatal error is logged in the Dynabus Control and Status register, and the ECT field is non-zero, the DynaData register is used for event counting
- JTAG communication area: the DynaData register can otherwise be used as a general communication area with the JTAG master.

4.5.4.1 Event counting

When the ECT field of the Dynabus Control and Status register is nonzero and all error bits of the Dynabus Control and Status register (DPE, GPE, GTO, CDE) are zero, the DynaData register is used to count various internal events for performance monitoring. In this mode, it has the following format:

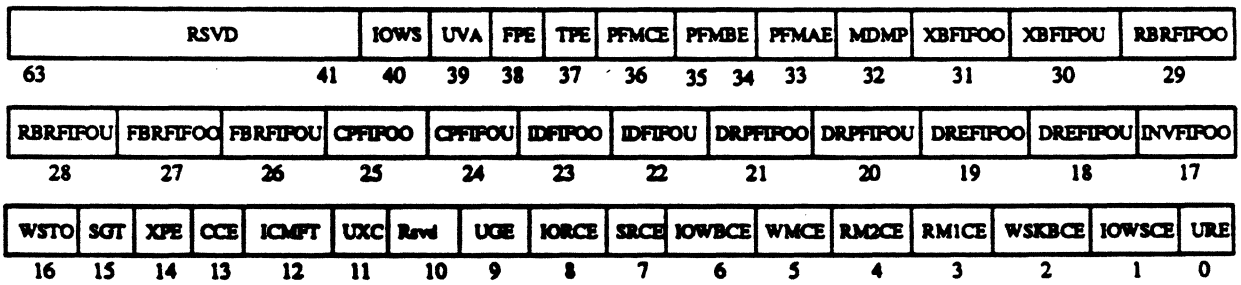


where EC is a 24-bit wraparound counter (see ECT field on page 61 for the various events which may be counted). The unused field is not modified by event counting.

4.5.4.2 Error logging

If DPE is one, the DynaData register contains the Dynabus data on which a parity error was detected. The contents of the DynaData register are unspecified when GPE or GTO is one. When IE is one, a bit of the DynaData register is dedicated to each possible internal error. A bit is set to one when the corresponding error is detected. If multiple internal errors are detected in the same cycle, multiple bits are set in the DynaData register.

The structure of the DynaData register for internal error logging is the following:



where:

URE: Undefined Reply Error. An undefined reply packet is received.

IOWSCE: IOWriteSingle Count Error. See definition of count errors below.

WSKBCE: WriteSingleInvalidate, WriteSingleUpdate, SwapSingleInvalidate, SwapSingleUpdate, WriteBlock or Interrupt Count Error. See definition of count errors below.

RM1CE: Read Miss 1 Count Error. See definition of count errors below.

RM2CE: Read Miss 2 Count Error. See definition of count errors below.

WMCE: Write Miss Count Error. See definition of count errors below.

IOWBCE: IOWriteBlock Count Error. See definition of count errors below.

SRCE: Stream Read Count Error. See definition of count errors below

IORCE: IOReadSingle or IOSwapSingle Count Error.

Count Errors occur when too many requests of a given type are issued by the Cache Controller or when more replies than outstanding requests are received.

UGE: Unexpected Grant error.

UXC: Undefined XBus command.

ICMFT: Incorrect Command Field in a MemFault cycle.

CCE: Cache Controller Error. When the Cache Controller detects a fatal error it sends a signal to the Bus Watcher which is then responsible to request a system watchdog reset. Information about the error is logged in the Cache Controller Error Register (See section 4.4.10 on page 50).

XPE: XBus Parity Error.

SGT: Spurious Grant. An unexpected bus grant is received.

WSTO: WriteSingleInvalidate, WriteSingleUpdate, SwapSingleInvalidate, SwapSingleUpdate, WriteBlock or Interrupt Time-out.

INVFIFOO: Invalidate FIFO Overflow. Control flow error.

DREFIFOU: Dynabus Request FIFO Underflow. Control flow error.

DREFIFOO: Dynabus Request FIFO Overflow. Control flow error.

DRPFIFOU: Dynabus Reply FIFO Underflow. Control flow error.

DRPFIFOO: Dynabus Reply FIFO Overflow. Control flow error.

IDFIFOU: ID FIFO Underflow. Control flow error.

IDFIFOO: ID FIFO Overflow. Control flow error.

CPFIFOU: Control Packets FIFO Underflow. Control flow error.

CPFIFOO: Control Packets FIFO Overflow. Control flow error.

FBRFIFOU: FlushBlock Requests FIFO Underflow. Control flow error.

FBRFIFOO: FlushBlock Requests FIFO Overflow. Control flow error.

RBRFIFOU: ReadBlock Requests FIFO Underflow. Control flow error.

RBRFIFOO: ReadBlock Requests FIFO Overflow. Control flow error.

XBFIFOU: XBus FIFO Underflow. Control flow error.

XBFIFOO: XBus FIFO Overflow. Control flow error.

MDMP: Multiple DeMap Error. Only one MMU DeMap can be pending at a given time. A processor must be in a critical section before issuing a DeMap operation. If a DeMap is issued while a previous one is still pending a fatal error is reported. See section 4.2.3 on page 36 for more information on MMU demapping.

PFMAE: Pending Flush Monitor Allocation Error.

PFMBE: Pending Flush Monitor Block Error (1 bit for each PFM line).

PFMCE: Pending Flush Monitor Contention Error.

TPE: Tags Parity Error. Parity error on the Tag Address field in an entry of the BW tags (See section 4.5.1.2 on page 58). Consistency error.

FPE: Flags Parity Error. Parity error on the flags fields in an entry of the BW tags (See section 4.5.1.2 on page 58). Consistency error.

UVA: Unspecified Victim Address. The displaced block is owned but no victim address is specified in the Dynabus request packet. Consistency error.

IOWS: Invalid Owned Write Single. A write single reply is received but the entry is invalid. Consistency error.

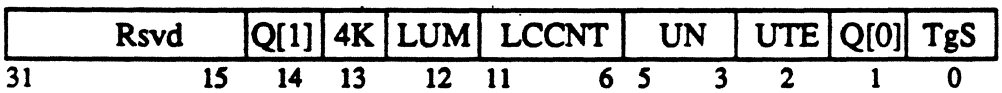
Rsvd: Reserved. Undefined.

For more informations on the BW internal errors the reader can refer to [15].

4.5.5 Control Register

This 32-bit register contains control bits for the Bus Watcher. It is accessible in read and write mode as a word.

The Control register has the following structure:



where:

TgS: Tags Size. TgS, 4K and DNum in the Dynabus Control and Status register define the tag configuration, see table at the end of this section.

Q[0]: See below for Q[1:0] definition.

UTE: User Timer Enable. The profile timer is configured in user timer mode if this bit is set to one (See section 4.5.8.1 on page 69).

UN: Unit Number. This field corresponds to the unit number for the Extended CSR space addressing (See section 3.2.6 on page 28). This value should be equal to DeviceID[3:1].

LCCNT: Limit of Competitive Caching Counter. This field specifies the limit of the competitive caching counter. The competitive caching counter is a 6-bit free running linear feedback shift register. When the value of this counter is strictly smaller than the limit, cache updates received from Dynabus (WriteSingle reply packets, see[4]) are transformed in cache sub-block invalidations. If the limit is set

to 0x00 the cache consistency protocol is a pure write-broadcast protocol. If the limit is set to 0x3F, *all updates* (except updates to pending blocks) are transformed into invalidations and the cache consistency protocol is pure write-invalidate. This feature allows control of the cache consistency protocol on multi-threaded applications.

LUM: LockUnlock Mode. When this bit is set to one, the LockUnlock Dynabus transactions are disabled [4]. This means that the BW will keep on retrying an access for as long as necessary without "locking" the subblock. When this bit is clear, the BW uses the Dynabus LockUnlock transactions when it judges it necessary. For more details the reader can refer to [15]. This bit is provided for debugging purposes only. It **must** be set to 0 under normal conditions.

4K: 4K mode. TgS, 4K and DNum in the Dynabus Control and Status register define the tag configuration, see table at the end of this section. In a BW, 4K is always read as zero and cannot be modified. In a BWP, 4K is readable and writable.

Q[1:0]: Quarter select mode. These two bits select which quarter or half of the entries are to be used when not all 2K or 4K tag entries are used. See the note on page 56 to see how the Q[1:0] bits affect the tag entry mapping. In a BW, Q[1] always reads as zero and cannot be modified. In a BWP, Q[1] is readable and writable.

Rsvd: Reserved. This field is ignored on a write and read as zero.

The table below gives the set of legal settings for TgS, 4K and DNum. Configurations which are not supported in current systems are grayed out:

DNum	TgS	4K	External Cache Size	Number of BWs	Number of BWPs
00	0	0	512 K-Byte	1	1
00	0	1	1 M-Byte	N/A	1
01	1	0	512 K-Byte	2	2
01	0	0	1 M-Byte	2	2
01	0	1	2 M-Byte	N/A	2
11	1	0	1 M-Byte	4	4
11	0	0	2 M-Byte	4	4
11	0	1	4 M-Byte	N/A	4

The Control register must be initialized before any access to the tags, since the address of the tags depends on the contents of the UN field. In a dual-processor system board, the Bus Watchers for both processors must have initialized their control register before the tags of either processor are accessed.

4.5.6 Interrupt Table

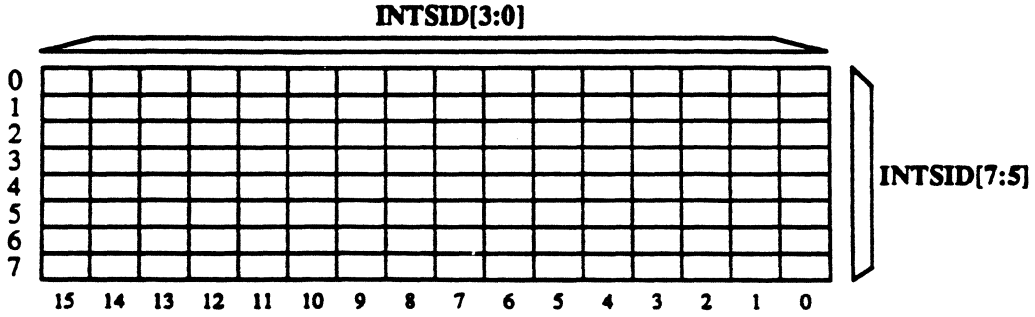
The Interrupt Table is used to identify the source of an interrupt and limit the number of devices polled before discovering which one generated an interrupt.

It consists of eight 16-bit registers that may be read and written as half-words.

The Interrupt Table half-words are aligned on a 64-bit boundary. A half-word of the Interrupt Table is selected by bits[5:3] of the physical address in the address range specified on the tables of page 32 and page 34.

A bit of the Interrupt Table is set by the hardware whenever an interrupt is received from Dynabus.When an interrupt is sent on Dynabus, an 8-bit Interrupt Source Identifier (INTSID), which is used to identify the interrupt source, is passed with the interrupt level.

When a BW receives an interrupt, it uses INTSID[7:5] to select one of the eight registers of the Interrupt Table and INTSID[3:0] to select a bit to be set in this register. INTSID[4] is reserved for a future extension of the Interrupt Table to eight 32-bit registers. This bit is ignored and should be zero.



The Interrupt Table is not cleared on a system reset. It must be cleared by issuing writes to the Interrupt Table Clear pseudo-registers before enabling interrupts.

4.5.7 Interrupt Table Clear

Interrupt Table Clear consists of eight 16-bit pseudo-registers that are used to clear bits of the Interrupt Table. These pseudo-registers are write-only and only accessible as half-words.

Interrupt Table Clear half-words are aligned on a 64-bit boundary just like Interrupt Table registers. A half-word is selected by bits[5:3] of the physical address in the address range specified on the tables of page 32 and page 34.

A write of the value one to any bit in these pseudo-registers clears the corresponding bit in the Interrupt Table register.

4.5.8 Counter-Timers

There are two Counter-Timers in each Bus Watcher.

The first Counter-Timer is used as **profile timer**. It supports a free running mode called the ***user timer mode***. This timer generates interrupts at level 14 (INTSID=0) to the attached processor. Software should use only the profile timer in the Bus Watcher attached to Dynabus # 0, to allow proper operation when a single Dynabus is enabled.

The second Counter-Timer is used as a **Unix tick timer**. It generates interrupts at level 10 (INTSID=1) to the attached processor. This Counter-Timer does not sup-

port the user timer mode. There **should** be only **one** such timer used in a Sun-4D systems system. As for the profile timer, the tick timer should be attached to Dynabus # 0. The processor whose tick timer is enabled will receive all the corresponding interrupts, and is in charge of re-dispatching them.

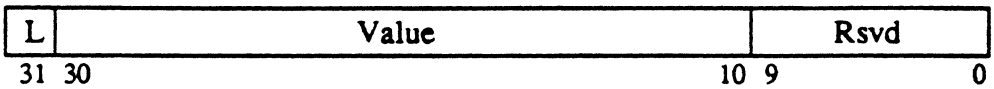
The Counter-Timers are cleared at reset.

4.5.8.1 Counter-Timers Structure

Both Counter-Timers have the same structure. They consist of a 32-bit Counter register, a 32-bit Limit register and a 32-bit Non-Destructive Limit register. The profile timer also includes a 32-bit Control register to enable the user timer mode and a 64-bit User Timer register. The structure of the registers is described below.

The Counter, Limit, Non-Destructive Limit and Control registers are accessed as words. The User Timer register is accessed as a word or double-word.

The Counter register has the following structure:



The various bit fields have the fcllowing meaning:

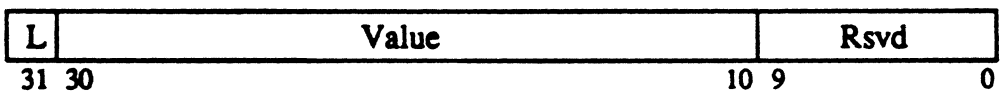
Rsvd: Reserved. This field is ignored on a write and read as zero

Value: The Counter value is incremented every microsecond. When the counter value becomes equal to the Limit value, the bit L is set to one, the Counter value is reset to one (and keeps being incremented) and an interrupt is issued to the attached processor if L was not already set. It must be noted that a specific bit is set in the Interrupt Table when a Counter-Timer interrupt is issued. The profile timer uses the INTSID value `0x00` and the tick timer the INTSID value `0x01`.

L: Limit bit. This bit is set to one when the Counter value is equal to the Limit value. This bit is cleared when the Limit register is read.

The Counter register is readable and writable. When it is read, the Counter value is frozen until the read completes but no increment is lost.

The Limit register has the following structure:



The various bit fields have the following meaning:

Rsvd: Reserved. This field is ignored on a write and read as zero

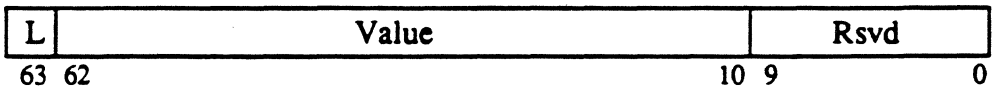
Value: Limit value. Setting the Limit value to zero configures the timer in a free-running mode where no interrupt is issued. Note that, when the timer is in free-running mode, the L bit is still set normally, although no interrupt is issued.

L: Limit bit. This bit is the same as the limit bit of the Counter register. It is set to one when the Counter value is equal to the Limit value. It is cleared when the Limit register is read.

The Limit register is readable and writable. When the Limit register is read the L bit is cleared. When the Limit register is written the Counter value is reset to one.

The Non-Destructive Limit register is the same physical register as the Limit register. A read of this register has the same effect as a read of the Limit register. When this pseudo-register is written the Counter value is not reset to one. This pseudo-register is provided to implement alarm-clock interrupts.

The following registers are defined **only for the profile timer**. The profile timer may be configured in user timer mode with bit UTE of the Bus Watcher Control register (See section 4.5.5 on page 66). When the user timer mode is enabled, the Counter and Limit registers are merged in a 64-bit counter register called the User Timer register. This register has the following structure:



The various bit fields have the following meaning:

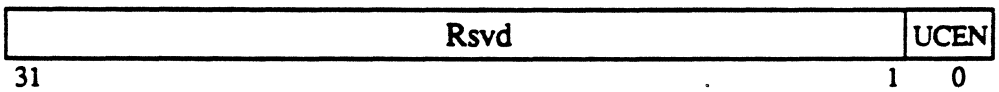
Rsvd: Reserved. This field is ignored on a write and read as zero

Value: Counter value. The Counter value is incremented every microsecond. When it overflows, the bit L is set to one but no interrupt is generated.

L: Limit bit. This bit is set to one when the Counter value overflows. It is cleared when the register is written.

The User Timer register should be accessed as a double word. When it is written, the bit L is cleared.

The Control register has the following structure:



The various bit fields have the following meaning:

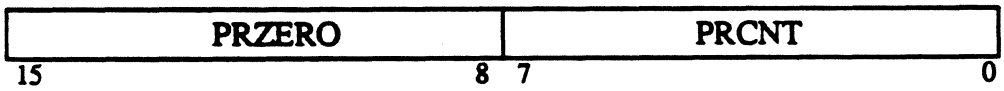
Rsvd: Reserved. This field is ignored on a write and read as zero

UCEN: Enable user timer counting when set to one. When it is zero the User Timer is frozen.

4.5.8.2 Prescaler Register

The Prescaler register is used to generate a 1 MHz clock. The Prescaler register is accessible as a half-word and is readable and writable.

The Prescaler register has the following structure:



The bit fields have the following meaning:

PRCNT: Prescaler Counter. The Prescaler Counter field is an 8-bit counter that is incremented every Dynabus clock cycle. When it overflows, the Counter-Timers are incremented and the value of the Prescaler Zero field is loaded into this field.

PRZERO: Prescaler Zero. This field is the initial value of the Prescaler Counter. The Prescaler Zero field must be loaded with a value that causes the Counter-Timers to be incremented every microsecond.

4.5.9 Reset state

The Bus Watchers are reset via a system reset. The various causes of system reset are detailed in Chapter 9.

On reset a Bus Watcher takes the following actions:

- Reset all state-machines to idle state.
- Reset all internal queues.
- Clear the Counter and Limit registers of the profile and the tick timer.
- Put the Bus Watcher in frozen mode i.e. set the FZN bit in the Dynabus Control and Status register (See section 4.5.3 on page 59).

4.5.10 Programming Note

After a power-on reset the Dynabus Control and Status register and the DynaData register must be initialized via JTAG. The following initialization sequence must be used:

- Load Dynabus CSR with EER set to 0 and FZN set to 0.
- Reload Dynabus CSR with ME, GTO, GPE and DPE set to 1 and CDE set to 0xF.
- Reload Dynabus CSR with EER set to 1.

This JTAG initialization sequence ensures that any fatal error present when the BW is "unfrozen" are properly logged.

4.6 Boot Bus

The BootBus is an 8-bit local bus controlled by the Cache Controller. It is used to access the following devices:

- A Time-Of-Day Clock/Non-Volatile RAM chip.
- 512 K-byte EPROM.
- A Z8530 Serial Channel Controller which provides two RS232 ports.
- A Z8530 Serial Channel Controller for a Keyboard/Mouse interface.
- 8 K-byte SRAM per processor for scratch pad purposes.
- The JTAG Master Interface.

- A LED register.
- Three Status registers.
- Two Semaphore registers.
- A Control register.
- A BootBus identification register

The BootBus is accessible in Local space with PA[35:24]=1111 1111 0000. The BootBus is also accessible from any processor via the system Extended CSR space with PA[35:24]=1111 dddd ddd0$_2$, where dddd ddd is DeviceID[7:1] of the corresponding processor unit.

The BootBus occupies 8 M-bytes of address space and is broken up as follows:

ECSR Space: PA[35:25] = 1111 dddd ddd0 Local Space: PA[35:25] = 1111 1111 0000				
PA[24:0]	Size	Access	Description	Mode
0x0000000 to 0x007FFFF	Any	RO	EPROM	Shr
0x0100000	B	RO	Status1 Register	Shr
0x0120000	B	RO	Status2 Register	Shr
0x0140000	B	RO	Status3 Register	Shr
0x0160000	B	WO	System Software Reset Register	Shr
0x0180000	B	RO	Version Register	Shr
0x01A0000	B	R/W	Semaphore0 Register	Shr
0x01A0004	B	R/W	Semaphore0 Status Register	Shr
0x01A0008	B	R/W	Semaphore1 Register	Shr
0x01A000C	B	R/W	Semaphore1 Status Register	Shr
0x01C0000 to 0x01C3FFF	Any	R/W	Shared SRAM	Shr
0x01E0000 to 0x01E1FFF	Any	R/W	Private SRAM	Shr
0x0200000	B	R/W	Serial Port B Control Register	Exc
0x0200002	B	R/W	Serial Port B Data Register	Exc
0x0200004	B	R/W	Serial Port A Control Register	Exc
0x0200006	B	R/W	Serial Port A Data Register	Exc
0x0240000	B	R/W	Mouse Control Register	Exc
0x0240002	B	R/W	Mouse Data Register	Exc
0x0240004	B	R/W	Keyboard Control Register	Exc
0x0240006	B	R/W	Keyboard Data Register	Exc
0x0280000 to 0x0281FF7	Any	R/W	NVRAM	Exc

ECSR Space: PA[35:25] = 1111 dddd ddd0 Local Space: PA[35:25] = 1111 1111 0000				
PA[24:0]	Size	Access	Description	Mode
0x0281FF8	B	R/W	TODC Control Register	Exc
0x0281FF9	B	R/W	TODC Seconds Register	Exc
0x0281FFA	B	R/W	TODC Minutes Register	Exc
0x0281FFB	B	R/W	TODC Hours Register	Exc
0x0281FFC	B	R/W	TODC Days Register	Exc
0x0281FFD	B	R/W	TODC Date Register	Exc
0x0281FFE	B	R/W	TODC Month Register	Exc
0x0281FFF	B	R/W	TODC Year Register	Exc
0x02C0000	B	R/W	Control Register	Exc
0x02E0000	B	R/W	LED Register	Exc
0x0300000	B/HW	R/W	JTAG Command Register	Exc
0x0300004	B	R/W	JTAG Control Register	Exc

Accesses to the BootBus **must** be marked as **non cacheable**. Cacheable BootBus accesses will result in a time-out bus error.

Both CPUs in a system board may access concurrently the BootBus address ranges which are marked as **Shr** (Shared) in the table above. Accesses to the **Exc** (Exclusive) address ranges must be serialized between the two CPUs using Semaphore 0 (See section 4.6.9.1 on page 81).

The BootBus interface supports all SPARC addressable data sizes. Physical accesses are done on bytes only. Accesses with larger data sizes are transformed into a burst of bytes accesses from the least significant to the most significant byte. Atomic accesses are not supported and will cause a time-out bus error. If the page tables are located in the BootBus space (i.e. in the SRAM) during the boot process, the Modify and Reference bits **must** be initialized to one to avoid faults.

4.6.1 EPROM

A one M-byte space is reserved on the BootBus for EPROM. Only 512 K-byte of EPROM are currently provided. Physical address bit [19] is not decoded and the EPROM is mirrored every 512 K-byte.

The EPROM can be accessed as bytes, half-words, words or double-words.

4.6.2 SRAM

A 256 K-byte space is reserved on the BootBus for Static RAM. 16K-byte of SRAM are actually implemented in a system board. The SRAM is accessible via two address ranges as bytes, half-words, words or double-words.

The address map for the SRAM is as follows:

PA[23:0]	CPU A	CPU B
0x1C0000 to 0x1C1FFF	Lower 8-K bytes	Lower 8-K bytes
0x1C20000 to 0x1C3FFFF	Upper 8-K bytes	Upper 8-K bytes
0x1E0000 to 0x1E1FFF	Lower 8-K bytes	Upper 8-K bytes

In the shared SRAM address range (0x1C0000 to 0x1DFFFF), all of the SRAM is accessible. Physical address bits [16:14] are not decoded and the SRAM is mirrored every 16-K byte.

In the private SRAM address range (0x1E0000 to 0x1FFFFF), only 8-K byte of SRAM are accessible: processor A can access only the lower 8-K bytes of the SRAM, while processor B can access only the upper 8-K bytes of the SRAM. Physical address bits [16:13] are not decoded and the SRAM is mirrored every 8-K byte.

4.6.3 Serial Ports

Two RS 232 serial ports, port A and port B, are provided with each BootBus in a Sun-4D system. One of these ports (port B) is reserved for the system console. The serial ports are implemented with an AMD Z8530 Serial Channel Controller (SCC). The reader can refer to the data sheets for further information [5].

The SCC registers **must** be accessed as **bytes only**. Accesses with larger data sizes will fail in non-predictable ways, without an error being reported to the programmer. Note that it is not necessary to provide software delays when accessing the SCC registers.

Serial ports generate interrupts at SPARC level 12. The serial port interrupts are directed to the processor which holds BootBus Semaphore 0, or to both processors if no processor holds Semaphore 0 (See section 4.6.9.1 on page 81).

4.6.4 KeyBoard/Mouse Interface

Another SCC provides the Keyboard/Mouse interface.

The SCC registers **must** be accessed as **bytes only**. Accesses with larger data sizes will fail in non-predictable ways, without an error being reported to the programmer. Note that it is not necessary to provide software delays when accessing the SCC registers.

Serial ports generate interrupts at SPARC level 12. The serial port interrupts are directed to the processor which holds BootBus Semaphore 0, or to both processors if no processor holds Semaphore 0 (See section 4.6.9.1 on page 81).

4.6.5 LED Register

Eight LEDs (Light Emitting Diodes) are provided with each board to encode information about the boards status. These 8 LEDs are physically on the board.

An 8-bit read-write LED register controls the LEDs. Writing a 0 in a bit of this register turns on the corresponding LED while writing a 1 turns it off. After a system reset, the LED register is cleared (i.e. all LEDs are turned on).

A 128 K-byte space is allocated to the LED register, but bits [16:0] of the physical address are ignored. The LED register is accessed with any byte address in this space. Accesses with larger quantities cause multiple accesses to the LED register.

4.6.6 TOD Clock and NVRAM

Sun-4D system boards use the Mostek MK 48T08(B) Zeropower/Timekeeper RAM chip for the calendar function. This chip consists of 8 K-byte of non-volatile RAM; the top 8 bytes are dedicated to the Time of Day Clock. The reader can refer to the data sheets for further information [6]. This chip provides its own battery to maintain the time and the RAM contents.

The TOD/NVRAM can be accessed in byte, half-word, word or double-word quantities.

Only one TOD clock should be used in a Sun-4D system although there is a TOD clock on each board. In general, the TOD clock on the board which is elected *master board* during the boot process (See Chapter 9) may be designated as the system's TOD clock.

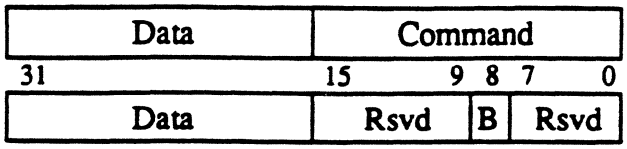
4.6.7 JTAG Master Interface

The JTAG Master interface provides control of the JTAG state machines in the various chips connected to the JTAG+ bus. These state machines control the shifting of data and instructions over the JTAG serial path (For a general description of the Sun-4D JTAG architecture see Chapter 10).

The JTAG Master interface consists of two registers that are described below.

4.6.7.1 JTAG Command Register

The format of the JTAG Command register is:



where:

Command: Indicate a JTAG command sequence to be issued. The Command field is write-only. It should be modified only when the Busy bit is zero[1].

B: Busy. The B bit is zero when there is no active JTAG command, one otherwise. The B bit is set when a command is issued and remains set until the command is performed. The B bit is read-only.

Data: Data to be shifted into or out of a JTAG scan ring. The data field is shifted to the right into the JTAG scan ring. Simultaneously, the JTAG scan ring data is right shifted into the data field most significant bit. The value of this field is valid on a read only when the Busy bit is zero.

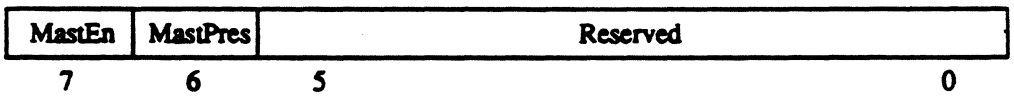
Rsvd: Reserved. The value of this bit field is unspecified on a read.

The command register supports byte, half-word and word access. When a write operation affects the least significant byte of the command field, a new JTAG command is initiated, which causes a sequence of JTAG state transitions. The list of commands can be found in section 10.2.2 on page 178.

4.6.7.2 JTAG Control Register

The JTAG Control register is read-write and must be accessed as a byte. Accesses with other sizes have unspecified results.

The JTAG Control register has the following structure:



where:

MastPres: Master Present. This bit is 0 when a system JTAG master is available, either because an external JTAG controller is connected (manufacturing only) or because one processor has set to one the MastEn bit in this register. This bit is one when no system JTAG master is available. Note that MastPres is zero after a system reset only if an external JTAG controller is connected. See section 10.2.1 on page 178 for a description of the JTAG control organization.

1. Stores to the Command field when Busy=1 have unpredictable effects.

MastEn: Master Enable. When MastEn is set to 1, the JTAG controller on this BootBus becomes the system JTAG controller (See section 10.2.1 on page 178). This bit is cleared on system reset, which means that the BootBus JTAG controller operates within the system board (or is disabled in the presence of an external JTAG controller).

4.6.8 Status and Control Registers

4.6.8.1 Control Register

A 128 K-byte space is allocated to the Control register but bits [16:0] of the physical address are ignored. The Control register is accessed with any address in this space.

The Control register is readable and writable as a byte. Accesses with other sizes have unspecified results.

The Control Register has the following structure:

BLGmB	EnFan	EnTmp	LTE_L	BGmL	SRGmL	SYelL	SLGmL
7	6	5	4	3	2	1	0

where:

SLGrnL: System Left Green LED. The left system green LED (on the front panel) is turned on by writing a zero in this bit. It is turned off if **all** processors set SLGrnL to a one. This bit is cleared on a system reset, which means the left system green LED is turned on by a system reset until turned off by all processors.

SYelL: System Yellow LED. The system yellow LED (on the front panel) is turned on by writing a one in this bit. It is turned off if **all** processors set SYelL to a zero. This bit is cleared on system reset.

SRGrnL: System Right Green LED. The right system green LED (on the front panel) is turned on by writing a one in this bit. It is turned off if **all** processors set SRGrnL to a zero. This bit is cleared on a system reset.

BGrnL: Board Green LED. The system board green LED is turned on by writing a one in this bit and turned off by writing a zero. This bit is cleared on a system reset. It should be used to reflect the status of processor A.

LTE_L: Link Test Enable. In Scorpion setting this bit to zero enables the Link Test feature of the Twisted Pair Ethernet T7213 DISC chip of the on-board SCSI/Ethernet interface. The result of the test is indicated by the LinkTest bit of the BootBus Status_2 register (See section 4.6.8.3 on page 78). In SunDragon, the LTE_L bit **must** always be zero. If a SunDragon board is plugged in a lab test-bed a level 15 interrupt will be permanently asserted if the LTE_L bit is nonzero. This bit is cleared on system reset.

EnTmp: Enable temperature warning interrupt. When this bit is set to zero, it disables the level 15 interrupt which is issued when the ambient air temperature exceeds a rated value. When this bit is set to one, an excessive temperature will issue a level 15 interrupt (See section 4.6.8.3 on page 78). This bit is cleared on system reset.

EnFan: Enable fan failure interrupt. When this bit is set to zero, it disables the level 15 interrupt which is issued when the fan fails. When this bit is set to one, a failure of the fan will issue a level 15 interrupt (See section 4.6.8.3 on page 78). This bit is cleared on system reset.

BLGrnB: Board Green LED B. The second system board green LED is turned on by writing a one in this bit and turned off by writing a zero. This bit is cleared on system reset. This LED should be used to indicate the status of processor B.

4.6.8.2 Status_1 Register

A 128 K-byte space is allocated to the Status_1 register, but bits [16:0] of the physical address are ignored. The Status_1 register is accessed with any byte address in this space.

This register is read-only, and must be accessed as a byte. Accesses with other sizes have unspecified results.

The Status_1 Register has the following structure:

Reserved	DiagMode	Reserved	Secure	Reserved	SRGrnLS	SYelLS	SLGnLS
7	6	5	4	3	2	1	0

where:

SLGrnLS: System Left Green LED Status. This bit is zero when the left system green LED is on, one when the left system green LED is off. The left green System LED is on if any processor has set SLGrnL to zero in its BootBus Control register (See section 4.6.8.1 on page 77).

SYelLS: System Yellow LED Status. This bit is zero when the system yellow LED is on, one when the system yellow LED is off. The yellow System LED is on if any processor has set SYelL to one in its BootBus Control register (See section 4.6.8.1 on page 77).

SRGrnLS: System Right Green LED Status. This bit is zero when the right system green LED is on, one when the right system green LED is off. The right green System LED is on if any system board has set SRGrnLS to one in its BootBus Control register (See section 4.6.8.1 on page 77).

Secure: System Secure. This bit is equal to one when the front panel key in the "System Secure" position, it is equal to zero otherwise. When this bit is one, the reset switch is disabled.

DiagMode: Diagnostic Mode. This bit is used as a switch for system diagnostics. If this bit is zero, extensive diagnostics are requested at power-on. If this bit is one, regular diagnostics are requested at power-on.

4.6.8.3 Status_2 Register

A 128 K-byte space is allocated to the Status_2 register, but bits [16:0] of the physical address are ignored. The Status_2 register is accessed with any byte address in this space.

This register is read-only, and must be accessed as a byte. Accesses with other sizes have unspecified results.

The Status_2 Register has the following structure:

Reserved		FanInt	TmpInt	LnkTst	ACInt	RstStat	
7	6	5	4	3	2	1	0

where:

RstStat: Reset Status. This bit field indicates the reason for the most recent system reset (See section 9.1 on page 169) according to the following table:

RstStat	Reset Source
00	Power-On, SVP reset or reset switch
01	System Watchdog Reset
10	System Software Reset
11	Unused

ACInt: AC Power Failure. This bit is set to one when the hardware detects an AC power fail condition. It remains set to one as long as the power supply reports an AC power loss. A level 15 interrupt (no INTSID) is issued whenever this bit is set to one. Because all system boards receive the ACInt signal, all processors receives a level 15 interrupt on an AC Power Failure. This interrupt cannot be masked. The interrupt is directed to the processor which holds BootBus Semaphore 0, or to both processors if no processor holds Semaphore 0 (See section 4.6.9.1 on page 81).

LnkTst: Link Test. This bit has two different usages in current Sun-4D systems. In Scorpion, this bit is set to one when the Link Test passes on the T7213 Twisted Pair Ethernet interface when it has been previously been enabled by setting the LTE_L bit to 0 in the BootBus Control register. In SunDragon, this bit is used to indicate if the system board is plugged in a regular backplane (LnkTst=0) or if the board in plugged in one of the lab test-beds (LnkTst=1). POST must read the value of this bit after a power-on reset to determine in what environment the board is plugged.

NOTE: If the board is plugged into a test-bed, the BARBs must remain in loopback mode after it has been tested by POST.

TmpInt: Temperature warning. This bit is one when the temperature on board exceeds a set value, it is zero otherwise. A level 15 interrupt (no INTSID) is issued when the temperature warning is active and the EnTmp bit is set (See section 4.6.8.1 on page 77). Since there is a temperature sensor on each system board, a temperature warning may be received on some system boards and not on others. The interrupt is directed to the processor which holds BootBus Semaphore 0, or to both processors if no processor holds Semaphore 0 (See section 4.6.9.1 on page 81).

FanInt: Fan Failure. This bit is one when the system fan is not operational, it is zero otherwise. A level 15 interrupt (no INTSID) is issued when the fan is not operational and the EnFan bit is set (See section 4.6.8.1 on page 77). Notice that a

fan failure is always notified to all system boards simultaneously. The interrupt is directed to the processor which holds BootBus Semaphore 0, or to both processors if no processor holds Semaphore 0 (See section 4.6.9.1 on page 81).

4.6.8.4 Status_3 Register

A 128 K-byte space is allocated to the Status_3 register, but bits [16:0] of the physical address are ignored. The Status_3 register is accessed with any byte address in this space.

This register is read-only, and must be accessed as a byte. Accesses with other sizes have unspecified results.

The Status_3 Register has the following structure:

BD_ID	P_ID	Reserved	Rst_B	Rst_A
7 4	3	2	1	0

where:

Rst_A: Processor A Forced Reset. This bit is zero if processor A is held in a forced Reset state by jumper A on the system board. This jumper is provided for manufacturing test only, and this bit should always be read as 1 in normal operation.

Rst_B: Processor B Forced Reset. This bit is zero if processor B is held in a forced Reset state by jumper B on the system board. This jumper is provided for manufacturing test only, and this bit should always be read as 1 in normal operation.

P_ID: Processor Identification. This bit is zero when read by processor A, one when read by processor B.

BD_ID: Board Identifier. These four bits indicate the physical location of the System Board on the backplane. These bits are used by POST during system initialization in order to allocate the Dynabus DeviceIDs (See section 3.2.3 on page 26).

4.6.8.5 System Software Reset Register

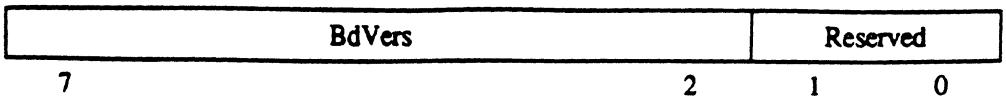
A 128 K-byte space is allocated to the System Software Reset pseudo-register. Any store in this space, whatever the quantity, causes a reset of the entire system (See section 9.1.1.3 on page 170). The data value is ignored. Reads to this register have no effect.

4.6.8.6 Board Version Register

A 128 K-byte space is allocated to the Board Version register, but bits [16:0] of the physical address are ignored. The Board Version register is accessed with any byte address in this space.

This register is read-only, and must be accessed as a byte. Accesses with other sizes have unspecified results.

The Board Version Register has the following structure:



where:

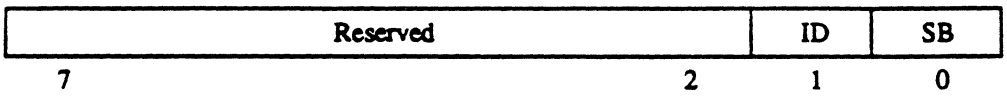
BdVers: Board Version. This 6-bit field indicates the type of system board, which in turn permits to identify the BootBus structure. A value of `0x01` indicates the BootBus implementation described in this manual. Other values are reserved. The value of this field is valid only immediately following a system reset and may change after the first access to the LED register.

Reserved: The value of a reserved bit is unspecified on a read.

4.6.9 Semaphore Registers

A 128 K-byte space is allocated to the two semaphore registers and their respective status registers. The status registers are read-only copies of the semaphore registers.

Semaphore and semaphore status registers have the following structure:



where:

SB: Semaphore Bit. This bit is zero no processor has taken the semaphore, one if a processor has taken the semaphore.

ID: Semaphore owner ID. This bit is significant only when SB is one. It is zero when processor A holds the semaphore, one when processor B holds the semaphore.

Reserved: The value of a reserved bit is unspecified on a read. Reserved bits are ignored on a write, but should be written as zero.

The semaphore registers are read-write and must be accessed as bytes only. The semaphore status registers are read-only and must be accessed as bytes only. Accesses with other sizes have unspecified results.

On a read to a semaphore register, the current value of ID and SB is returned. If SB was zero, SB is set to one and ID is set to reflect the identity of the requesting processor. If SB was one, the register is not modified. A read to a semaphore status register returns the current value of the semaphore register, but does not change the status of the semaphore. A write to a semaphore register modifies the ID and SB bits normally.

4.6.9.1 Semaphore 0

Semaphore 0 controls access to the exclusive portion of the BootBus and BootBus interrupt dispatching.

A processor may access exclusive BootBus devices only if it holds semaphore 0 (i.e. {ID, SB} = 01 for processor A accesses, 11 for processor B accesses). If a processor attempts to access exclusive BootBus devices when it does not hold semaphore 0, a read will return unspecified data and a write will be ignored. There is no error notification for exclusive BootBus accesses performed while Semaphore 0 is not held.

When exclusive BootBus devices are accessed via ECSR space, two addresses can be used: the address using the DeviceID of processor A or the address using the DeviceID of processor B. Semaphore 0 must be held by the processor corresponding to the DeviceID used in ECSR space address for the access to complete successfully. It must be noted that the processor which holds semaphore 0 is not necessarily the processor which initiated the ECSR access. This later maybe a processor from another system board.

When Semaphore 0 is held by a processor, BootBus interrupts (i.e. fan failure, temperature warning, AC failure, SCC interrupts) are dispatched to that processor exclusively. When semaphore 0 is not held by any processor (SB=0), BootBus interrupts are dispatched to both processors.

4.6.9.2 Semaphore 1

Semaphore 1 has no hardware side-effect and may be used for interlock between CPUs during firmware operation.

Chapter 5
Main Memory Unit

5.1 Main Memory Architecture

In Sun-4D systems the Main Memory unit consists of one, two or four memory banks according to the number of Dynabus(ses) supported by a given implementation. A **memory bank** is defined in Sun-4D as a memory array controlled by a specific ASIC called the **Memory Queue Handler** (MQH). A memory bank is the physical entity for address interleaving on the same Dynabus.

In a multiple Dynabus system the main memory is interleaved across Dynabusses on a 256 byte boundary. The main memory can also be interleaved on the same Dynabus on multiples of 64 bytes (See section 5.3 on page 87).

NOTE: The main memory interleaving is configured by POST after a power-on reset. It is not visible from the kernel which gets a memory map from POST. For all practical purposes the main memory interleaving is not visible to programmers.

In Sun-4D systems a memory bank consists of zero, one or two **groups** of 4 custom SIMMs (Single In-Line Memory Module). These are the **only** possible configurations, which means that the memory size increment in Sun-4D is the memory capacity of a group of 4 SIMMs.

There are two versions of the MQH. Only the second version of the MQH, called MQHP, will be shipped, but the first version may be used for system bring-up. From a software standpoint, the second version is upward compatible with the first one. The first version of the MQH can only supports two memory groups of 4 SIMMs. MQHP can support four memory groups of 4 SIMMs. MQH and MQHP present the same programming model except for the number of Group Type and Address Decoding registers. In this chapter MQH refer to both versions of the Memory Queue Handler unless specified otherwise.

The Memory Queue Handler supports read and write operations only on 64-byte subblocks. The MQH does not support writes on smaller quantities. Thus, the Memory Space **must be cached** and should not be accessed directly by a processor.

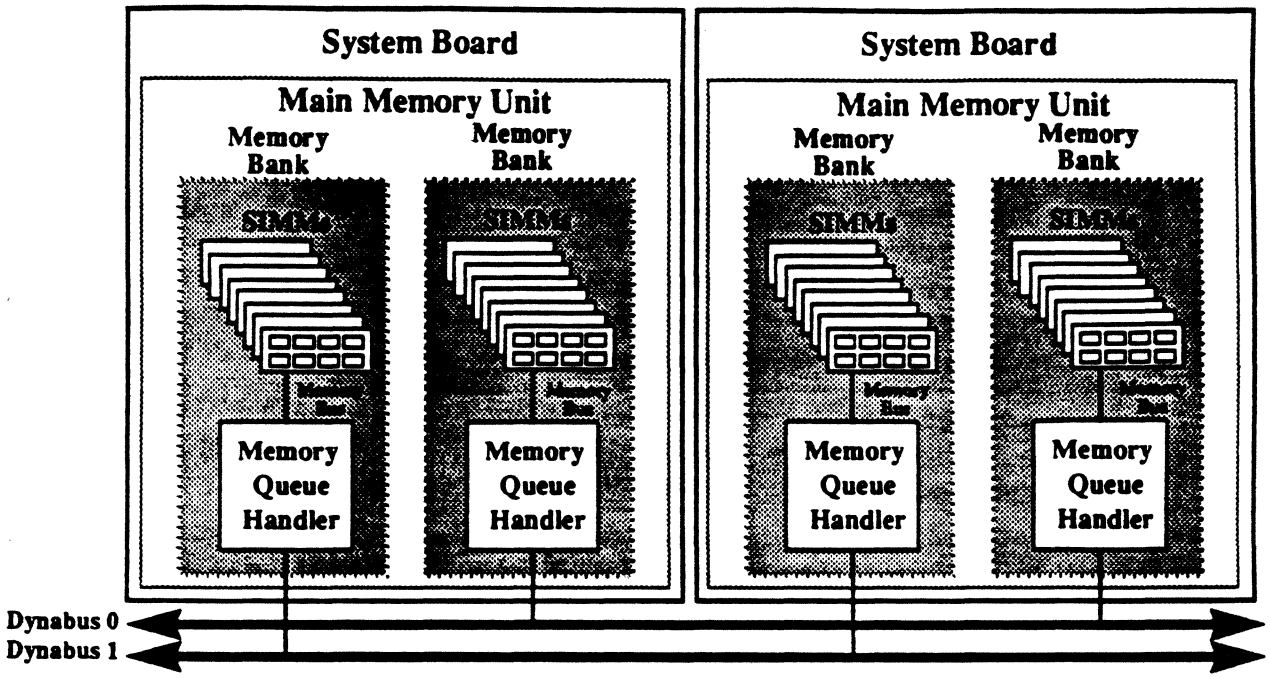
The main memory in Sun-4D systems supports an Error and Correction Code (ECC) which can detect and correct single-bit errors and detect double-bit errors. It can also detect triple-bit and quadruple-bit errors if the erroneous bits are in the same nibble.

5.1.1 SunDragon Memory Architecture

The main memory in SunDragon is interleaved across both Dynabusses on a 256-byte boundary.

Each SunDragon board supports a Main Memory unit. Each Main Memory Unit consists of two memory banks; one per Dynabus.

SunDragon Main Memory



In SunDragon a memory bank consists of zero, one or two **groups** of 4 custom SIMMs (Single In-Line Memory Module). These are the **only** possible configurations. Although MQHP can logically supports four groups of SIMMs, packaging constraints limit the SunDragon system boards to only **two** groups of SIMMs per memory bank.

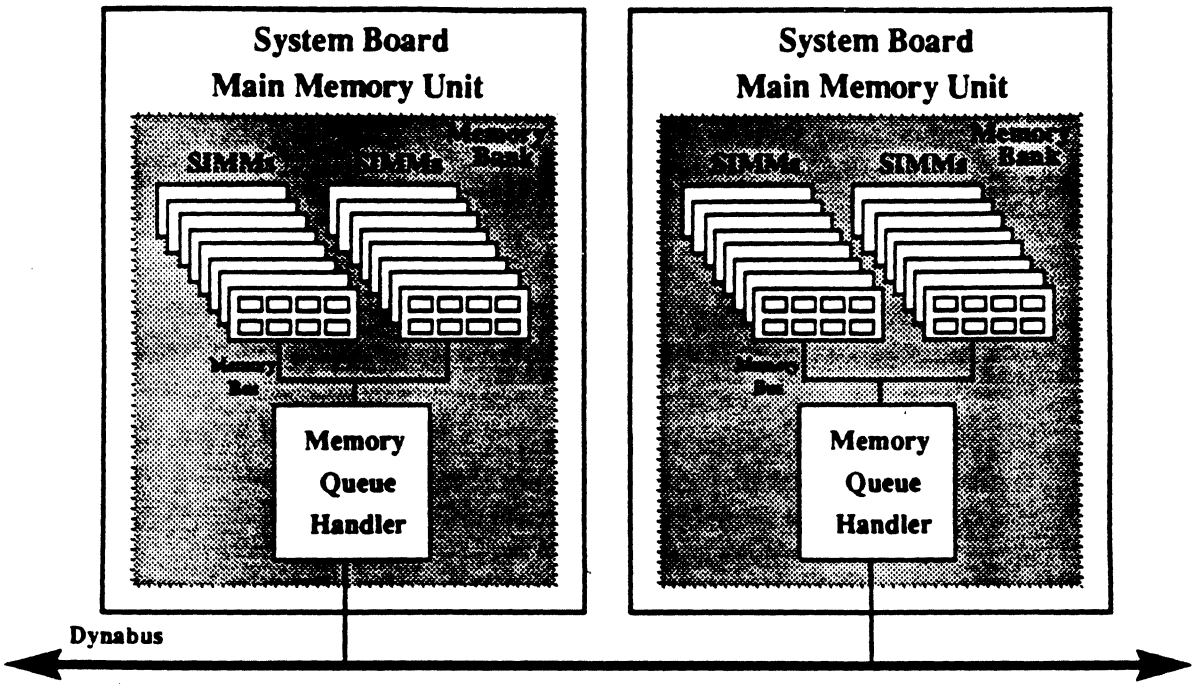
Due to bus interleaving, in SunDragon, the memory size increment is the memory capacity of two groups of 4 SIMMs. If only one of the two Dynabus is configured (due to a permanent failure on the other Dynabus) then, the memory size increment is the memory capacity of a group of 4 SIMMs.

5.1.2 Scorpion Memory Architecture

In Scorpion, the Main Memory unit consists of a single memory bank. A Scorpion memory bank consists of zero, one, two, three or four custom SIMMs. Because,

Scorpion has only one Dynabus, the main memory increment is a single group of 4 SIMMs.

Scorpion Main Memory



The first version of the MQH may also be used for Scorpion bring-up. In this case, the memory bank is limited to two groups of 4 SIMMs.

5.1.3 SIMM Organization

The Sun-4D memory architecture requires custom SIMMs using a Crossbar ASIC to multiplex data between the MQH and the memory chips. There are two types of SIMMs: the regular SIMMs using DRAM chips and the NV_SIMMs (Non-Volative SIMMs) which use SRAM chips along with a backup battery to ensure data integrity in case of power failure.

5.1.3.1 DRAM SIMM

There are two generation of SIMMs. The first generation of SIMMs are using 4 M-bit DRAM chips and the second generation 16 M-bit DRAM chips. They respectively have a capacity of 8 M-bytes and 32 M-bytes. Therefore, a memory group of 4 SIMMs has a capacity of 32 M-bytes or 128 M-bytes.

A SunDragon memory bank can have between 32 and 256 M-bytes of memory capacity. This implies that a two-Dynabus single-board SunDragon system has a minimal memory capacity of 64 M-bytes with 8 M-byte SIMMs and a total capacity of 512 M-bytes with 32 M-byte SIMMs. A fully populated backplane SunDragon system, with 10 System Boards, provides a maximum memory size of 5120 M-bytes with16 M-bit DRAM chips.

A Scorpion memory bank can have between 32 and 1024 M-bytes of memory. A single-board Scorpion system has a minimal capacity of 32 M-bytes with 8 M-byte SIMMs and a total capacity of 512 M-bytes with 32 M-bytes SIMMs. A fully populated Scorpion system has a maximum memory size of 1 G-byte.

MQH can also support 64 and 256 M-bit DRAM chips for future systems. The table in section 5.3.1 on page 88 defines the various SIMM types along with their sizes.

A memory group must always be composed of SIMMs having the same size. However, a single MQH can support groups with different types of SIMMs in each group.

5.1.3.2 NV_SIMM

The NV_SIMMs are using 1 M-bit SRAM chips organized in 128 K x 8. They have a capacity of 1 M-byte. Hence, a memory group of 4 NV_SIMMs has a capacity of 4 M-bytes.

Regular SIMMs cannot be mixed with NV_SIMMs inside the same memory group. A memory group must always be composed of SIMMs of the same type. However, it is possible for the same MQH to support one or multiple group(s) of NV_SIMMs along with one or multiple group(s) of regular DRAM SIMMs.

The Non Volatile memory also supports the same Error and Correction Code (ECC) as the regular main memory.

5.2 Memory Queue Handler Access

The MQH registers are accessible in the CSR space (See section 3.2.5 on page 26). This means that the 16 most significant bits of the physical address must be PA[35:20] = `1111 1110 dddd dddd`$_2$. **`dddd dddd`** is the DeviceID of the referenced Memory Queue Handler. The address map for the Memory Queue Handler registers is:

CSR Space: PA[35:24] = `1111 1110 dddd dddd`$_2$				
PA[19:0]	**Size**	**Access**	**MQH Registers**	**MQHP Registers**
0x00000	W	RO	Component ID Reg.	Component ID Reg.
0x00008	D	R/W	DynaBus Control & Status Reg.	DynaBus Control & Status Reg.
0x00010	D	R/W	DynaData Reg.	DynaData Reg.
0x01000	W	R/W	Group 0 Address Decoding Reg.	Group 0 Address Decoding Reg.
0x01008	W	R/W	Group 1 Address Decoding Reg.	Group 1 Address Decoding Reg.
0x01010	D	RO	Group 0 Type Reg.	Group 0 Type Reg.
0x01018	D	RO	Group 1 Type Reg.	Group 1 Type Reg.
0x01040	W	R/W	Reserved	Group 2 Address Decoding Reg.
0x01048	W	R/W	Reserved	Group 3 Address Decoding Reg.
0x01050	D	RO	Reserved	Group 2 Type Reg.
0x01058	D	RO	Reserved	Group 3 Type Reg.
0x01020	D	R/W	Memory Control and Status Reg.	Memory Control and Status Reg.
0x02000	D	R/O	Correctable Error Address Reg.	Correctable Error Address Reg.
0x02008	D	RO	Correctable Error Data Reg.	Correctable Error Data Reg.
0x02010	D	R/O	Uncorrectable Error Address Reg.	Uncorrectable Error Address Reg.

CSR Space: PA[35:24] = 1111 1110 dddd dddd$_2$				
PA[19:0]	Size	Access	MQH Registers	MQHP Registers
0x02018	D	RO	Uncorrectable Error Data Reg.	Uncorrectable Error Data Reg.
0x02020	D	R/W	ECC Diagnostic and Control Reg.	ECC Diagnostic and Control Reg.
0x03000 0x03098	D	R/W	Memory Timing Registers	Memory Timing Registers

Note that bits [9:8] of the address are the Dynabus selector in CSR Space, and that this address map should thus be interleaved 4 times with a step of 256 bytes.

All MQH registers must be accessed with the specified data size. A read with a smaller size will be performed normally. In the case of a read with a larger size, the nonexistent bits will be undefined. All writes with a size different from the specified one will be performed normally. The unspecified bits will be written to 0. All registers can be accessed with the atomic operations (SWAP, LDSTUB). As for write operations, the undefined bits are written to 0.

5.3 Address Decoding

When the MQH receives a Dynabus memory request, it must decide whether to handle the request. This function is done by address filtering logic, which is replicated for each group of 4 SIMMs. If the result of this test is positive, the MQH generates the memory group address that is used to access the DRAM chips of the selected group.

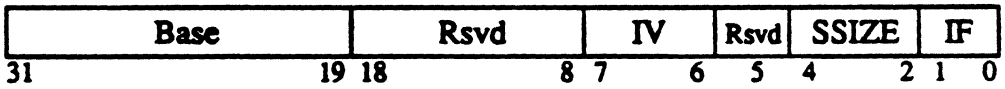
In the following, we present the structure of the Address Decoding registers. Then we explain how the information stored in these registers is used to filter Dynabus requests. Finally, the address generation process is detailed.

5.3.1 Address Decoding Registers

There is one 32-bit Address Decoding register associated with each group of SIMMs. These registers are used to decode the portion of the Memory Space where the group is mapped.

This register may be read and written as a word.

The Address Decoding register has the following structure:



The bit fields have the following meaning:

IF: Interleaving Factor. This field specifies the stride size on which the group is interleaved. The following table, describes the encoding of this field:

IF	Stride Size
00	64 bytes (No interleaving)
01	128 bytes (Two way interleaving)
10	256 bytes (Four way interleaving)
11	Reserved

The IF field is used to compute a 2-bit mask (noted IF_Mask). The computed mask is "anded" with bits [7:6] of the physical address (See section 5.3.2 on page 91). The value of this mask is given by the following table:

IF	IF_Mask
00	00
01	01
10	11
11	Reserved

NOTE: If IF is set to the reserved value 11_2, the MQH generates an IF_Mask equal to 00_2.

SSIZE: SIMM Size. This field specifies if the corresponding group is present and the amount of memory on the SIMMs (and therefore the size of the group). The encoding of this field is the following:

SSIZE	SIMM Size	Group Size
000	Group Not Present	Group Not Present
001	2 M-bytes (1 M-bit DRAM chips)	8 M-bytes
010	8 M-bytes (4 M-bit DRAM chips)	32 M-bytes
011	32 M-bytes (16 M-bit DRAM chips)	128 M-bytes
100	128 M-bytes (64 M-bit DRAM chips)	512 M-bytes
101	512 M-bytes (256 M-bit DRAM chips)	2 G-bytes
110 - 111	Reserved	Reserved

When a group is not present, the address decoding logic for the group is disabled. When SSIZE field is set to one of the reserved values, the MQH considers the corresponding group as absent even if SIMMs are effectively plugged. In this case, the address decoding logic for the group is disabled.

The SSIZE field must be set to 001 for a group of NV_SIMMs. Thus, an 8 M-byte space is allocated for a 4 M-bytes NV_SIMMs memory group. The most significant bit of the memory address is ignored and the NV_SIMM group is mirrored on the upper 4 M-bytes of the allocated 8 M-bytes space.

This field, the IF field and the DNum field of the Dynabus Control and Status register (See section 5.4.2 on page 98) are used to compute a 13-bit mask (noted SSIZE_Mask) which selects some high-order bits of the physical address. The result of the masking operation is matched against the Base field (See section 5.3.2 on page 91). The value of the mask is given by the following tables:

DNum	IF	SSIZE	SSIZE_Mask
00	00	001	1 1111 1111 1111
		010	1 1111 1111 1100
		011	1 1111 1111 0000
		100	1 1111 1100 0000
		101	1 1111 0000 0000
	01	001	1 1111 1111 1110
		010	1 1111 1111 1000
		011	1 1111 1110 0000
		100	1 1111 1000 0000
		101	1 1110 0000 0000
	10	001	1 1111 1111 1100
		010	1 1111 1111 0000
		011	1 1111 1100 0000
		100	1 1111 0000 0000
		101	1 1100 0000 0000

DNum	IF	SSIZE	SSIZE_Mask
01	00	001	1 1111 1111 1110
		010	1 1111 1111 1000
		011	1 1111 1110 0000
		100	1 1111 1000 0000
		101	1 1110 0000 0000
	01	001	1 1111 1111 1100
		010	1 1111 1111 0000
		011	1 1111 1100 0000
		100	1 1111 0000 0000
		101	1 1100 0000 0000
	10	001	1 1111 1111 1000
		010	1 1111 1110 0000
		011	1 1111 1000 0000
		100	1 1110 0000 0000
		101	1 1000 0000 0000

DNum	IF	SSIZE	SSIZE_Mask
11	00	001	1 1111 1111 1100
		010	1 1111 1111 0000
		011	1 1111 1100 0000
		100	1 1111 0000 0000
		101	1 1100 0000 0000
	01	001	1 1111 1111 1000
		010	1 1111 1110 0000
		011	1 1111 1000 0000
		100	1 1110 0000 0000
		101	1 1000 0000 0000
	10	001	1 1111 1111 0000
		010	1 1111 1100 0000
		011	1 1111 0000 0000
		100	1 1100 0000 0000
		101	1 0000 0000 0000

The SSIZE_Mask is obtained by shifting left a 13-bit word with all bits set to one. Vacated positions are filled with zeros. The shift count is: IF+ DNum+2*(SSIZE-1).

NOTE: If DNum is incorrectly set to 10_2 the MQH computes the SSIZE_Mask as if DNum is set to 11_2. If IF is incorrectly set to 11_2 the SSIZE_Mask generated is unpredictable and may cause unexpected positive decoding of Dynabus request packets.

NOTE: Writing SSIZE to 000_2 disables the refresh for the corresponding memory group. Clearing the SSIZE field causes the data contained in the memory group to be lost.

IV: Interleave Value. This field is compared with the results of the logical "and" of the IF_Mask with bits[7:6] of the physical address. If there is a match, the MQH will handle the request provided the filtering based on the high-order address bits is also positive.

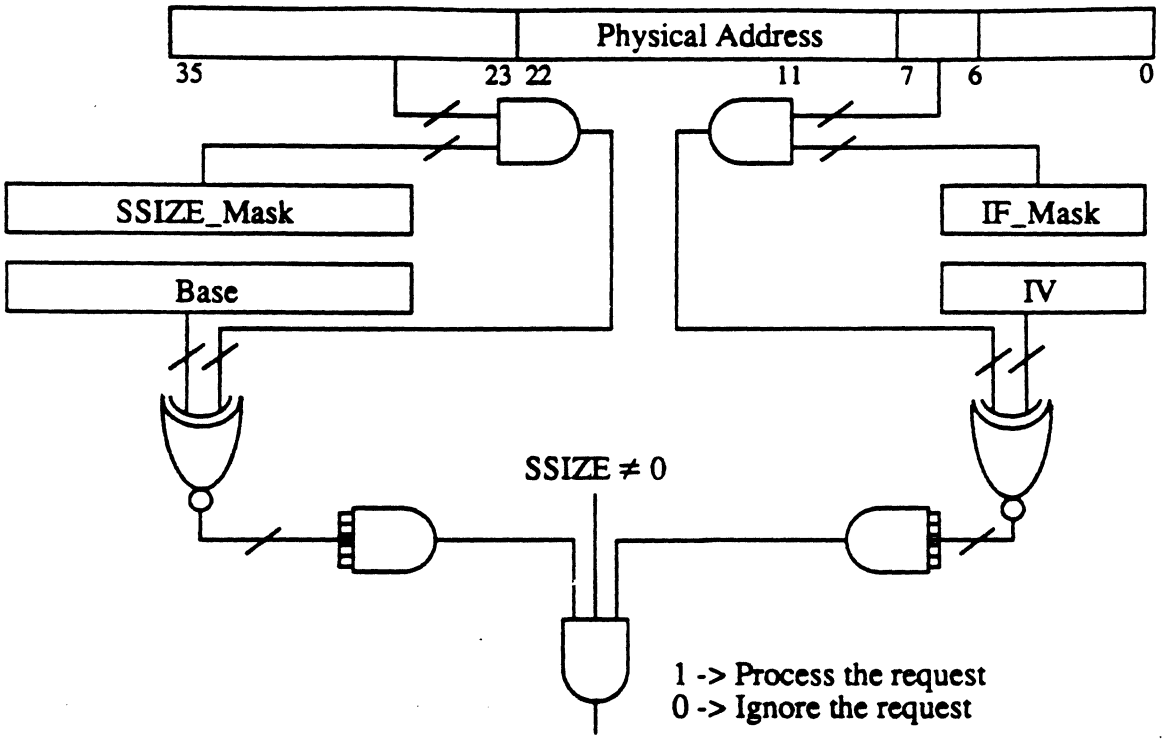
Base: This field defines the base address at which the group is mapped in the Memory Space. This field is compared with the result of the logical "and" of the SSIZE_Mask with bits[35:23] of the physical address. If there is a match, the MQH will handle the request provided the address interleaving filtering is also positive.

NOTE: In current Sun-4D systems PA[35]=0 for Memory Space addresses (See section 3.2.1 on page 24). The most significant bit of the Base must always be set to 0 in current systems, otherwise it may cause a fatal error due to multiple decoding.

Rsvd: Reserved. Reserved fields are ignored on a write and read as zero.They have no effect on the address decoding.

5.3.2 Address Filtering

The filtering of Dynabus addresses done for each memory group by the MQH can be summarized by the following diagram:



The filtering is done separately for each group. If, due to a programming error, the address filtering is positive for both groups with MQH, memory requests are always forwarded to group 0. With MQHP, when the address filtering is positive for two or more groups, all memory requests are ignored.

NOTE: It is possible to program the Address Decoding registers such that two or four memory groups attached to the same MQH are interleaved. It must be noted that it does not make sense because the requests for both groups are stored in the same queue. Therefore there will be no performance gain.

SunDragon systems can be configured with one or two Dynabus. In a multiple Dynabus system, the interleaving across the Dynabus is done on 256 bytes. This means that in a two Dynabus system all addresses received by a given MQH have the same value for bit[8]. Identically, in a four Dynabus Sun 4-D system all addresses received by a given MQH have the same value for bits[9:8].

5.3.3 Memory Group Address Generation

The address generated by the MQH to access the DRAM chips depends on the interleaving factor, the size of the group and the number of Dynabus in the system. The Memory Group address (noted MA, which addresses 64-byte subblocks) is extracted from the physical address (noted PA) as explained in the two following sub-

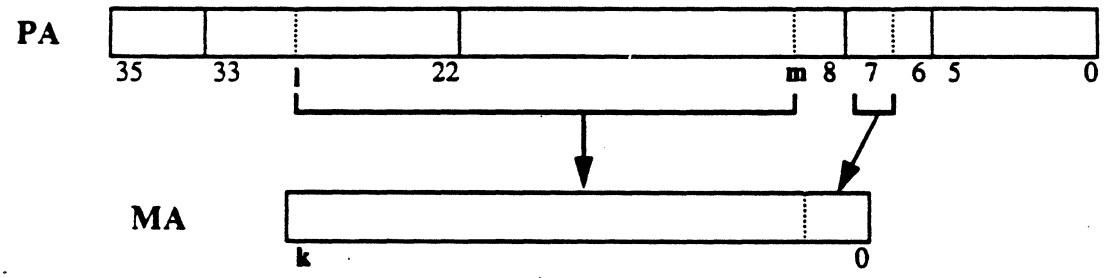
sections. Finally, the third subsection details how the Row Address and the Column Address which are issued to the DRAM chips are extracted from the Memory Group address.

It is important to note that to generate the Memory Group address the MQH must be aware of the number of Dynabus in the system. This parameter is specified by the DNum field of the Dynabus Control and Status register (See section 5.4.2 on page 98). In the usual SunDragon configuration, there are two Dynabus which implies that DNum *must* be set to 01_2. In Scorpion, there is only a single Dynabus and DNum *must* be set to 00_2.

5.3.3.1 No Interleaving or Two-way Interleaving

When there is no interleaving ($IF=00_2$) or two-way interleaving ($IF=01_2$) on the same Dynabus, the Memory Group address is generated by the concatenation of two bit fields of the physical address: MA[k:0] = {PA[l:m], PA[7:s]}.

k, l, m and s are computed as follows: s=6+IF, $m = 8+\log_2(DNum+1)$, $l = 22+\log_2(DNum+1)+IF+2*(SSIZE-1)$ and $k = 16+2*(SSIZE-1)$. This is summarized by the following diagram:

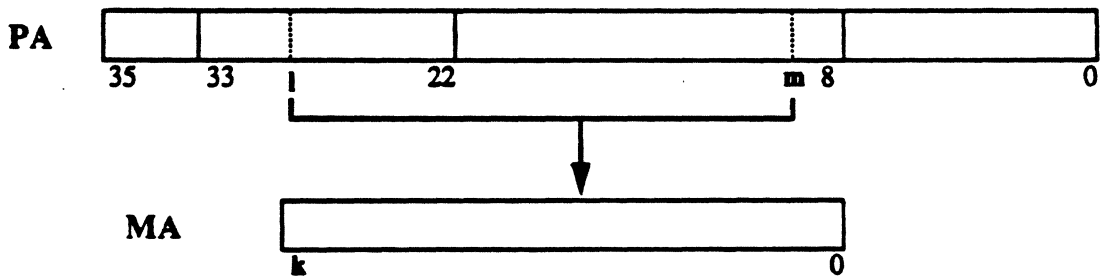


NOTE: When DNum is incorrectly set to 10_2 the MQH generates the memory address as if DNum=11_2.

5.3.3.2 Four-way Interleaving

With four-way interleaving ($IF=10_2$) on the same Dynabus, the Memory Group address is generated by extracting a contiguous field of the physical address MA[k:0] = PA[l:m].

k, l and m are computed as follows: $m = 8+\log_2(DNum+1)$, $l = 24+\log_2(DNum+1)+2*(SSIZE-1)$ and $k = 16+2*(SSIZE-1)$. This is summarized by the following diagram:



NOTE: When DNum is incorrectly set to 10_2 the MQH generates the memory address as if DNum=11_2.

5.3.4 Row and Column Address

The following table illustrate how the Row and Column Address are generated by MQH.

MQH		
SSIZE	Row Address[12:0]	Column Address[12:0]
001	{0000, MA[16:8]}	{0000, MA[7:0], b}
010	{000, MA[18:9]}	{000, MA[8:0], b}
011	{0, MA[9], MA[20:10]}	{00, MA[9:0], b}
100	{MA[10], MA[22:11]}	{00, MA[10:0], b}
101	MA[24:12]	{0, MA[11:0], b}

NOTE: For 16 M-bit DRAM (SSIZE=011) and 64 M-bit DRAM (SSIZE=100) the most significant bit of the Column Address (respectively MA[9] or MA[10]) is replicated as the most significant bit of the Row Address. This way the MQH can support DRAM chips in which the Row and Column Addresses are either symmetric or asymmetric.

The next table illustrates how the Row and Column Address are generated by MQHP:

MQHP		
SSIZE	Row Address[13:0]	Column Address[13:0]
001	{MA[3], MA[4], MA[5], MA[6], MA[7],MA[16:8]}	{00000, MA[7:0], b}
010	{MA[5], MA[6], MA[7], MA[8], MA[18:9]}	{0000, MA[8:0], b}
011	{MA[7], MA[8], MA[9], MA[20:10]}	{000, MA[9:0], b}
100	{MA[9], MA[10], MA[22:11]}	{000, MA[10:0], b}
101	{MA[11], MA[24:12]}	{00, MA[11:0], b}

NOTE: MQHP supports one more address line than MQH. The most significant bits of the Column Address are systematically replicated as the most significant bit of the Row Address. This way the MQH V.2 can support DRAM chips with symmetric or asymmetric Row and Column Addresses.

The DRAM chips are read or written in two consecutive accesses by using the fast page mode. The low order bit of the Column address, noted **b** in the previous tables, is toggled between the two accesses. On the first access b=PA[5] so that the part of the 64-byte subblock where the requested word resides is accessed first. This is done to reduce the memory access latency on a cache miss by delivering the data in cyclic order starting from the double word which caused the miss.

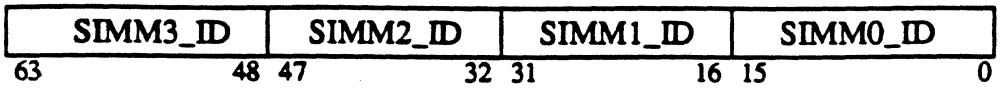
It must be noted that the Row and Column addresses described by the previous tables are the addresses generated by the MQH on the memory address bus. They are not necessarily the addresses presented on the DRAM input address pins. For packaging reason some address lines may be swapped on the SIMM board.

The NV_SIMMs are designed to be compatible with the memory bus DRAM interface provided by the MQH. The Row and Column addresses are demultiplexed to provide a single address to the SRAM chips. With the first version of the NV_SIMM, the SRAM address is: {MA[15:8], MA[7:0], b}.

5.3.5 Group Type Registers

There are two pseudo-registers that identify the type of SIMMs plugged in each group. These registers are read-only, and are accessible as double-words.

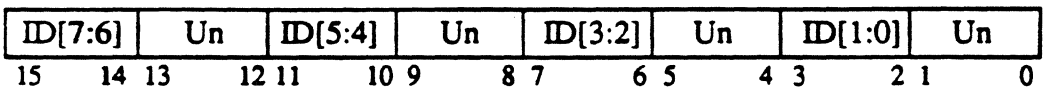
The structure of the Group Type registers is the following:



where:

SIMM*n*_ID: Identifier of SIMM *n*. An 8-bit identifier is hardwired on all SIMMs used in SunDragon to make them self identifying. During system initialization, the Group Type registers must be accessed to determine what type of SIMMs are plugged in each group. The identifiers can be used to retrieve the size and timing characteristics of the groups by accessing a table in EEPROM or NVRAM.

The format of the SIMM*n*_ID fields is the following:



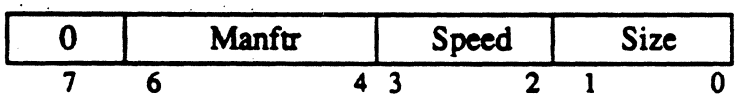
The bit fields have the following meaning:

Un: Undefined. These bits are undefined.

ID[7:0]: SIMM Identifier. There are two encoding of this field; one for regular DRAM SIMMs and one for NV_SIMMs.

DRAM SIMM:

With DRAM SIMMs the encoding of the identifier is the following:



For the two current suppliers of SIMMs for SunDragon (Mitsubishi Semiconductor and Texas Instruments) the bit fields have the following meanings:

Size: The encoding for the size is:

Size	DRAM Size
00	4 M-bit
01	16 M-bit
10	64 M-bit
11	256 M-bit

Speed: The speed is relative to the size and the manufacturer field. For the first SIMMs developed by Mitsubishi and Texas Instruments with 4 M-bit DRAM the encoding and the corresponding value for the Refresh Count and Request Delay

is:

Speed	Access Time	RCNT	RDLY
00	80 ns	001001_2	0010_2
01	100 ns	001001_2	0011_2
10	Reserved	Reserved	Reserved
11	Reserved	Reserved	Reserved

Manftr: SIMM Manufacturer code. The encoding is:

Manfr	Manufacturer
000	Reserved
001	Mitsubishi Semiconductor
010	Texas Instruments
011-111	Reserved

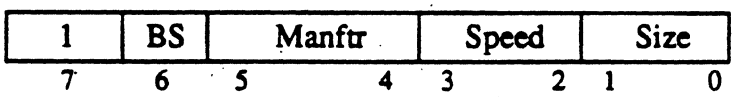
NOTE: An access to the Group Type register causes an extra refresh cycle of the group's DRAMs

NOTE: If no SIMMs are present in a given memory group the value 0xFFFF FFFF FFFF FFFF is returned on a read of the Group Type register. The memory group must be disabled by setting the SSIZE field to 000_2 in the corresponding Address Decoding register.

NOTE: It is possible to mix SIMMs of different manufacturers inside the same group provided they are of the **same size** and have the **same timing characteristics**. This implies the four half-word of the Group Type registers may have different values.

NV_SIMM:

With NV_SIMMs the encoding of the identifier is the following:



The bit fields have the following meanings:

BS: Battery Status. This bit is one is the battery is in good condition and zero if the battery failed.

Size: The encoding for the size is:

Size	SRAM Size
00	1 M-bit
01	4 M-bit
10	Reserved
11	Reserved

Speed: The speed is relative to the size and the manufacturer field. For the first SIMMs developed by Mitsubishi with 1 M-bit SRAM the encoding and the corresponding value for the Request Delay is:

Speed	Access Time	RDLY
00	70 ns	0011_2

Speed	Access Time	RDLY
01	85ns	0011_2
10	Reserved	Reserved
11	Reserved	Reserved

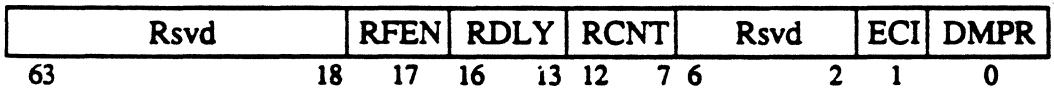
Manftr: SIMM Manufacturer code. The encoding is:

Manfr	Manufacturer
00	Reserved
01	Mitsubishi Semiconductor
10	Reserved
11	Reserved

5.3.6 Memory Control and Status Register

The Memory Control and Status Register is used to implement the address interleaving decoding. This register may be read and written as a double-word.

The Memory Control and Status Register has the following structure:



The various bit fields have the following meaning:

Rsvd: Reserved. Reserved fields are ignored on a write and read as zero.

RFEN: Refresh Enable. When this bit is zero, the DRAM chips are not refreshed. This bit is *not* affected by system resets, but is cleared when INIT is asserted (See description of INIT on page 98). See section 5.8.1 on page 109 for a description of the MQH reset sequence. This bit must be set to one if at least one of the active memory group consists of regular DRAM SIMMs. An active group is a group for which the SSIZE field in the Address Decoding register is not null. If all active groups are composed of NV_SIMMs, it is not necessary to enable refresh as they are composed of SRAM memory chips. An inactive group (group for which the SSIZE field is null) is not refreshed even it RFEN is set to one.

NOTE: DRAM chips must requires eight refresh cycles before being accessed. This condition can be satisfied through a software delay or by reading the Group Type register eight times (See section 5.8.1 on page 109 for a description of the MQH initialization).

RDLY: Request Delay. This field contains the number of Dynabus cycles the MQH must wait, from the beginning of the memory access, before issuing a bus request. It is used by the MQH to overlap the memory access with Dynabus arbitration when a 64-byte subblock is read. This parameter depends of the access time of the memory and the minimum request to grant delay of the Dynabus arbitration logic. See section 5.3.4 on page 93 for the predefined values of RDLY for the Mitsubishi and Texas Instruments SIMMs and for the Mitsubishi NV_SIMMs. If one or multiple NV_SIMM groups are mixed with one or more regular DRAM SIMMs

groups then, the NV_SIMMs RDLY value **must** be used. The SIMM Request Delay value must be used only if all memory groups are composed of regular DRAM SIMMs.

RCNT: Refresh Count. This field defines the refresh period minus 1 in multiple of 64 Dynabus clock cycles. The refresh period is: (RCNT+1)* 64*25ns. The minimum refresh period is 1.6 µs (RCNT=000000_2) and the maximum refresh period is 102.4 µs (RCNT=111111_2). The actual period at which DRAM chips receive refresh cycles depends of the number of active memory groups. An active group is a group for which the SSIZE field in the Address Decoding register is not null. If only one group is present it receives refresh cycles at the rate defined by the RCNT value. If N memory groups are presents (N=2,3 or 4) then the DRAM receives refresh cycles at 1/Nth of the refresh period defined by RCNT. The following table defines what value of RCNT should be used for the current SIMMs with a 40 Mhz Dynabus clock:

	One Group MQH/MQHP	Two Groups MQH/MQHP	Three Groups MQHP	Four Groups MQHP
RCNT Mitsubishi	001001_2	000100_2	000011_2	000010_2
RCNT TI	001001_2	000100_2	000011_2	000010_2

NOTE: The value of RCNT depends on the number of active memory groups only and not of the type of the memory groups. If NV_SIMMs groups and regular DRAM SIMMs groups are connected to the same MQH, then it is the total number of groups which determines the value of RCNT. For instance, if an MQHP supports two regular SIMM groups along with an NV_SIMM group, then RCNT must be set to 000011_2.

NOTE: If the system is running with a Dynabus clock frequency which is smaller than 40 Mhz, the RCNT value must be scaled accordingly.

The value of the Refresh Count is *not* affected by system resets, but it is cleared when INIT is asserted (See description of INIT on page 98).

ECI: Enable Correctable Error Interrupt. When this bit is set to one, a level 15 interrupt is broadcast when a correctable error is detected and no previous error is pending in the Correctable Error Address register (i.e. the ME and SBE bits are clear in the Correctable Error Address register). If this bit is zero no interrupt is broadcast but the error is still logged in the Correctable Error Address and Data registers (See section 5.5.3 on page 106).

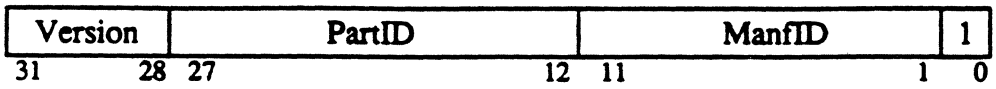
DMPR: Demap Reply Enable. When this bit is set to one, the MQH issues a DemapReply packet on Dynabus whenever a DemapRequest is received [4]. When this bit is clear, DemapRequest packets are ignored. On a given Dynabus, one and only one MQH must be configured with this bit set to one for the Demap mechanism to work properly. The address filtering logic is bypassed for Demap packets.

5.4 Dynabus Registers

5.4.1 ComponentID Register

This register is read-only. It is accessible as a word.

The ComponentID Register has the following structure:



The various bit fields have the following meaning:

Version: 4 bit version number. Set to `0x1`.

PartID: Part ID. 16 bit part number.

ManfID: Manufacturer ID. 11 bits Sun JEDEC Identifier. Set to `0x03E`.

The following PartIDs and versions are currently legal:

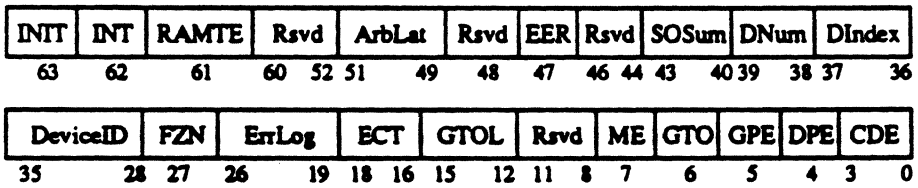
PartID	Version	ComponentID	Component
`0x0ADC`	`1`	`0x10ADC07D`	MQH
`0x0D86`	`1`	`0x10D8607D`	MQHP

5.4.2 Dynabus Control and Status Register

This 64-bit register contains certain Dynabus parameters which must be loaded via JTAG after a power-on reset. It is also used to record when a **fatal error** is detected by the MQH. A fatal error is an error which causes a system watchdog reset (See Chapter 7).

This register is accessible as a double word.

The Dynabus Control and Status Register has the following:



The various bit fields have the following meaning:

INIT: Initialize. This bit behaves as if reserved on CSR space accesses (i.e. writes have no effect and reads return zero), but it is accessible via JTAG (See section 10.3.5 on page 185). When INIT is set to one, it initializes the memory finite state machines in MQH which are not initialized by a normal reset in order to keep refreshing memory. It also sets RFEN and RCNT to 0 in the Memory Control and Status register (See section 5.3.6 on page 96). When INIT is cleared, the MQH operates normally. See section 5.7 on page 109 for a description of MQH initialization and the use of INIT.

INT: Interrupt. This bit behaves as if reserved on CSR space accesses (i.e. writes have no effect and reads return zero), but it is accessible via JTAG (See section 10.3.5 on page 185). When this bit is set to one, the MQH sends an interrupt packet on the Dynabus with the broadcast bit (B) set to 0, the TargetID field set to 0xEF, the INTSID field set to 0x00 and the Levels fields are all set to 0 (See section 8.3 on page 162 for a description of Dynabus Interrupt packets). Because all Level fields are null, no processor will actually receive an interrupt. This feature is provided to put the BIC out of loopback mode on a memory only board (See section 9.3 on page 171 for a description of the loopback mode). It is not used in the current Sun-4D implementations.

RAMTE: RAM Test Enable. This bit behaves as if reserved on CSR space accesses (i.e. writes have no effect and reads return zero), but it is accessible via JTAG (See section 10.3.5 on page 185). This bit is reserved for testing the internal RAM structures of the MQH. It *must* be set to 0 for normal operations.

ArbLat: Arbitration Latency. Minimum latency, in number of Dynabus cycles, from the time an arbitration request is driven by the MQH to the time the corresponding grant is driven by the arbiter, minus 3. (See [4] for more information on Dynabus). This field is read-only. In SunDragon, this field *must* be initialized to 010_2 (corresponding to a 5-cycle minimum arbitration latency) through JTAG.

Rsvd: Reserved. Reserved bits are ignored on a write and read as zero.

EER: Enable Error Reset. If this bit is set to one, any fatal error detected by the MQH causes a system watchdog reset. If this bit is set to zero when a fatal error is detected no system watchdog reset is issued. However, the error information is logged as if EER was one. This bit can be read and written.

SOSum: Shared/Owner Sum. Sum, in number of Dynabus cycles, of the Shared/Owner latency and the pipeline depth required to logically or the Shared/Owner signals of all Dynabus clients, minus 2 (See [4] for more informations on Dynabus). This field is read-only. In current Sun-4D systems, this field *must* be initialized to 0111_2 (corresponding to a 9-cycle latency) through JTAG.

DNum: Dynabus(ses) Number -1. Number of Dynabus(ses) in the system minus one. In SunDragon, this field *must* be set to 01_2 for normal operation. In Scorpion it must be set to 00_2. This field is read-only. It must be initialized through JTAG.

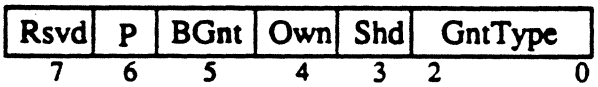
DIndex: Dynabus Index. This field indicates the number of the Dynabus to which this MQH is connected. This field is read-only. It is initialized through JTAG.

DeviceID: Dynabus DeviceID. This field is read-only. It is initialized through JTAG. In current Sun-4D systems, the four most significant bits of this field must correspond to the board number.

FZN: Frozen. This bit behaves as if reserved on CSR space accesses (i.e. writes have no effect and reads return zero), but it is accessible via JTAG (See section 10.3.5 on page 185). FZN is set to one on a system reset. When FZN is one,

MQH remains in the reset state and can only be accessed using JTAG. However, the memory continue to be refreshed when the FZN bit is set to one as long as the RFEN bit in the Memory Control and Status register is set to one (See section 5.3.6 on page 96) and the INIT bit remains set to zero.

ErrLog: Error Log. This field is used to log the parity bits when a Dynabus parity error is detected. The 8 parity bits compute byte parity over the 64 Dynabus data signals. The position of the bit determines the byte it protects. The least significant bit protects the least significant byte of the data signals and so forth. The parity can be odd or even according to the type of Dynabus cycle (See [4] for more information). The ErrLog field is also used to log the value of the seven Board Arbiter lines when a parity error is detected on these lines. In this case, the structure of this field is the following:



where:

GntType: Grant Type.
Shd: Shared. Logical OR of the Shared lines.
Own: Owner. Logical OR of the Owner lines.
BGnt: Board Grant.
P: Even Parity on the arbiter line.
Rsvd: Reserved. Read as zero and ignored on a write.

For a detailed explanation of the arbiter signals see [9].

When a parity error is detected in the very same cycle in both the arbiter signals and the Dynabus signals, the ErrLog field logs the value of the Board Arbiter lines.

This field cannot be written, even through JTAG, as long as fields GTO, GPE, DPE, CDE are nonzero.

ECT: Enable Count. The value of this field specifies the type of events which are counted in the DynaData register if there is no pending fatal error. It is readable and writable. The encoding is as follows (EC1 and EC2 designate the two counter fields of the DynaData register, See section 5.4.3 on page 102):

Type of Events Counted		
ECT	**EC1 Counter**	**EC2 Counter**
0	Counting Disabled	Counting Disabled
1	Nbr of RBRqst received by this MQH	Nbr of RBRply sent by this MQH
2	Nbr of cycle MQH asserts Hold	Nbr of FBRqst and WBRqst received
3	Nbr of 2-cycle Rqst & Rply packets (total)	Nbr of 9-cycle Rqst & Rply packets (total)
4	Nbr of Lock Request (total)	Nbr of UnLock Request (total)
5	Nbr of ReadBlockRqst (total)	Nbr of NCReadBlockRqst (total)
6	Nbr of FlushBlockRqst (total)	Nbr of WriteBlockRqst (total)
7	Nbr of Write & Swap Single Up(total)	Nbr of Write & Swap Single Inv (total)

NOTE: In the previous table "total" means that all the corresponding packets issued on the Dynabus to which the MQH is connected are counted even if these packets are not processed by the MQH.

NOTE: When ECT=7, the EC1 counter records the number of WriteSingleInvalidate and WriteSingleUpdate requests. EC2 records WriteSingleInvalidate and SwapSingleUpdate requests.

GTOL: Grant Time-Out Log_Value. This field specifies the Grant time-out counter maximum value in log base 2 of 1 K Dynabus clock cycles. This field is readable and writable. A Grant Time-Out is reported if the arbiter does not allocate the bus N Dynabus cycles after a bus request was issued. The implementation guarantees that $N = 2^{(10+GTOL)}$.

ME: Multiple Errors. This bit is set to one when multiple fatal errors have been detected. This bit is cleared by writing a one. In case of multiple successive errors only the first one is logged.

If multiple errors are detected in the very same cycle, the ME bit is not set to one but the error bits are set correctly. In most cases of simultaneous errors there is no conflicts on any bit fields of the Dynabus CSR or the DynaData to log the error information. The only exceptions are either simultaneous MQH specific errors or, a Grant Parity error along with a Dynabus Parity Error. The MQH specific errors are prioritized and the contents of the CDE field is defined by the table on page 101. In the case of a simultaneous Grant Parity error and Dynabus Parity error both the DPE and GPE bits are set. The ErrLog field logs the Arbiter lines and the DynaData register logs the Dynabus data.

DPE: Dynabus Parity Error. This bit is set to one when a parity error is detected on Dynabus for either a Data or a Header cycle. When a Dynabus parity error is detected, the faulting cycle is latched in the DynaData register and the parity signals are latched in the ErrLog field. This bit is cleared by writing a one.

GPE: Grant Parity Error. This bit is set to one when a parity error is detected on the arbitration signals. When such an error is detected, the value of the arbitration signals is latched in the ErrLog field. This bit is cleared by writing a one.

GTO: Grant Time-Out. This bit is set to one when the time-out counter for a Dynabus grant reaches the value specified by the GTOL field. This bit is cleared by writing a one.

CDE: Client Device Errors. This field is used to report MQH specific fatal errors. It is cleared by writing all bits to one. These errors are encoded according to the following table:

CDE	Error
0x0	No internal error
0x1	Reserved
0x2	OQFIFO finite state machine error (illegal state reached)
0x3	CMDQUE finite state machine error (illegal state reached)
0x4	MEM_MSTR finite state machine error (illegal state reached)
0x5	MEM_SLV finite state machine error (illegal state reached)
0x6	MCRAM_CTR finite state machine error (illegal state reached)

CDE	Error
0x7	Reserved
0x8	CMDQUE Overflow error
0x9	DynaBus packet length error
0xA	Unexpected Arbiter Grant
0xB-0xF	Reserved

This table also defines the priority when multiple MQH specific errors are detected during the same cycle.

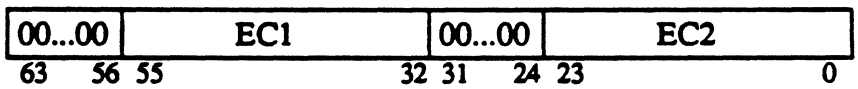
Notice that at most one of the fields DPE, GPE, GTO and CDE may be nonzero, and that ME cannot be one if all of them are zero.

5.4.3 DynaData Register

The DynaData register may be read and written as a double-word. It is also accessible via JTAG as part of shadow scan chain 0 (See section 10.3.5 on page 185). It cannot be written, even through JTAG, as long as any error bit is set in the DynaBus Control and Status register (fields DPE, GPE, GTO, CDE).

This 64-bit register serves three functions. These functions by order of priority are:

- Error logging: when a Dynabus parity error is detected, DPE is set to one and the DynaData register contains the DynaBus data. The contents of the DynaData register are unspecified when another type of fatal error is detected (GPE or GTO set to one or CDE non zero).
- Event counting: when the ECT field of the DynaBus Control and Status register is nonzero and the DynaData register is not being used for error logging, it is used to count various internal events for performance monitoring. In this mode, it has the following format:



  where EC1 and EC2 are 24-bit wraparound counters (see ECT field on page 100 for the various events which may be counted). The most significant byte of each half-word is forced to 0 when the event counters are enabled.
- JTAG communication area: the DynaData register can otherwise be used as a general communication area with the JTAG master.

5.5 ECC Memory Architecture

5.5.1 ECC Memory Operations

The MQH implements a SEC-DED-S4ED Error Correcting code on 64-bit memory words [8]. This code is able to detect and correct all single-bit errors, to detect all double-bit errors and to detect triple- and quadruple-bit errors if all erroneous bits are in the same nibble of the memory word.

The physical implementation of the memory uses DRAM chips organized in 4-bit words. However, all bits of a 64-bit memory word are coming from distinct DRAM

chips. This implies that a chip failure will be detected by a succession of correctable errors. This will give enough time to reconfigure the main memory. Because of the particular organization of the SIMM used in Sun-4D systems, the ECC also provides detection of a Crossbar Switch chip failure (See [16] and [17] for more information on the memory bank physical organization).

The MQH does not scrub memory during a refresh. Therefore, the ECC code is checked only on a memory read. ECC checking can be enabled/disabled with bit DECC in the ECC Diagnostic and Control register (See section 5.5.3.5 on page 107).

The MQH automatically corrects a single-bit error and writes-back the corrected word in memory. Single-bit errors are not reported to the requestor but they are logged in the Correctable Error Address register and the Correctable Error Data register. The Correctable Error Address register and the Correctable Error Data register record the **first occurrence** of a single-bit error (See section 5.5.3 on page 106). This means that a single-bit error is not logged if the error bit (noted SBE) in the Correctable Error Address register is already set to one. In this case, a bit (noted ME) of the Correctable Error Address register is set to indicate that multiple errors occurred. If the ECI bit of the Memory Control and Status register is set to one (See section 5.3.6 on page 96), a level 15 interrupt is broadcast whenever a single-bit error is logged in the Correctable Error Address and Data registers. Interrupts are issued so that single-bit errors can be logged by software without having to periodically poll these registers in all MQH. The level 15 interrupt is issued with INTSID[7:0]=0x02.

Uncorrectable errors detected by the ECC are reported to the Dynabus requestor (I/O Cache or Bus Watcher). The **first occurrence** of an uncorrectable error is logged in the Uncorrectable Error Address Register and the Uncorrectable Error Data Register (See section 5.5.3 on page 106). This means that another uncorrectable error will not be logged as long as the error bit (noted UE) has not been cleared in the Uncorrectable Address register but a bit (noted ME) will be set to indicate that multiple errors occurred. A level 15 interrupt is broadcast for logging purpose whenever an uncorrectable error is recorded in the Uncorrectable Error Address and Data registers. The interrupt is issued with INTSID[7:0]= 0x03. An uncorrectable error that occurs on a processor access will be reported as a trap. An uncorrectable error that occurs on a DVMA access is reported to the corresponding SBus board which is then responsible to report or not the error to a processor (See section 7.1.1 on page 143).

Because the memory is only accessed on 64-byte subblocks, multiple errors may be detected on a single access. When multiple single-bit errors occur on the same access, a single Level 15 interrupt with INTSID=0x02 is broadcast because only the first error is logged. If multiple uncorrectable errors occur on the same access, a single Level 15 interrupt with INTSID=0x03 is broadcast because only the first error is logged. If one or multiple single-bit error(s) **and** one or multiple uncorrectable error are detected during the same access **and** if at least one of these errors is correctly logged a Level 15 interrupt with INTSID=0x04 is broadcast.

Note that, when software handles an error interrupt from MQH, it needs to poll the Error Address registers of all MQHs attached to the same Dynabus to identify the source of the error (See section 8.2.3.2 on page 160).

5.5.2 Syndromes

The Correctable and Uncorrectable Error Address registers provide the syndrome associated with the error. The syndrome provides information about the error pattern and also gives the position of the erroneous bit for single-bit errors. The table on page 104 gives indications for the various syndromes.

A null syndrome means that no error is detected, therefore a null syndrome cannot be found in the Correctable or Uncorrectable Error Address registers immediately after an error.

All single-bit errors are corrected by the hardware and the syndrome is logged in the Correctable Error Address register. The erroneous bit number is indicated on the syndrome table. Bits are numbered from the least significant to most significant.

Double-bit errors (D) are reported as uncorrectable errors. The syndrome is logged in the Uncorrectable Error Address register.

Triple-bit (T) and quadruple-bit (Q) errors where all erroneous bits belong to the same nibble are reported as uncorrectable errors and the syndrome is logged in the Uncorrectable Error Address register. Syndromes for quadruple-bit and double-bit errors overlap. Syndrome values marked as Q on the following table indicate that either a double-bit or a quadruple-bit error occurred

Other errors that cannot be precisely identified are also detected. These errors can be multiple-bit errors including triple and quadruple-bit errors where the erroneous bits are not in the same nibble. In this case, the syndrome (MU) is logged in the Uncorrectable Error register.

Syndrome bits S3 S2 S1 S0	S7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	S6	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	S5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	S4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0 0 0 0		*	C4	C5	D	C6	D	D	T	C7	D	D	T	D	T	T	Q
0 0 0 1		C0	D	D	0	D	25	MU	D	D	5	17	D	8	D	D	12
0 0 1 0		C1	D	D	1	D	29	36	D	D	MU	21	D	13	D	D	9
0 0 1 1		D	32	33	D	42	D	D	MU	47	D	D	MU	D	T	T	D
0 1 0 0		C2	D	D	10	D	27	7	D	D	MU	19	D	2	D	D	14
0 1 0 1		D	57	61	D	59	Q	D	MU	63	D	Q	MU	D	MU	MU	D
0 1 1 0		D	MU	4	D	39	D	D	22	MU	D	D	30	D	16	24	D
0 1 1 1		T	D	D	MU	D	MU	54	D	D	50	MU	D	T	D	D	MU
1 0 0 0		C3	D	D	15	D	31	MU	D	D	38	23	D	3	D	D	11
1 0 0 1		D	37	MU	D	MU	D	D	18	6	D	D	26	D	20	28	D
1 0 1 0		D	49	53	D	51	Q	D	MU	55	D	Q	MU	D	MU	MU	D
1 0 1 1		T	D	D	MU	D	MU	62	D	D	58	MU	D	T	D	D	MU
1 1 0 0		D	40	45	D	34	D	D	T	35	D	D	T	D	MU	MU	D
1 1 0 1		T	D	D	T	D	MU	48	D	D	52	MU	D	MU	D	D	MU
1 1 1 0		T	D	D	T	D	MU	56	D	D	60	MU	D	MU	D	D	MU
1 1 1 1		Q	44	41	D	46	D	D	MU	43	D	D	MU	D	MU	MU	Q

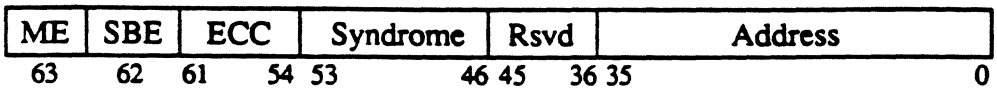
*** : No error detected**
CX: Single-bit Error in bit X of the ECC code
X: Single-bit Error in bit X of the 64-bit memory word.
D: Double-bit Error
T: Triple-bit Error in the same nibble
Q: Quadruple-bit Error in the same nibble or Double-bit error
MU: Multiple-bit Error unidentified

5.5.3 ECC Registers

5.5.3.1 Correctable Error Address Register

This register captures the first occurrence of a single-bit error. This register is accessed as a double-word.

The Correctable Error Address register has the following structure:



The various bit fields have the following meaning:

ME: Multiple Error. If a single-bit error is detected while SBE=1, this bit is set to one but the rest of the register is left unchanged. Note that no interrupt is issued when this bit is set to one. The ME bit is cleared on any write to the register.

SBE: Single Bit Error. This bit is set to one when the ECC logic detects and corrects a single-bit error. The Correctable Error Address register is updated when a single-bit error occurs only if this bit is clear at the time the error is detected. If the ECI bit is set to one in the Memory Control and Status register a level 15 broadcast interrupt is also issued when the SBE bit switches from 0 to 1. The interrupt is issued with INTSID=0x02 when one or multiple single-bit errors are detected during the same access. The interrupt is issued with INTSID=0x04 if one or multiple single-bit errors are detected along with one or multiple uncorrectable errors during the same access. The SBE bit is cleared by any write to the register.

ECC: ECC code associated with the 64-bit memory word. This field is read-only.

Syndrome: Syndrome. The value of this field identifies the erroneous bit (See table on page 104). This field is read only.

Rsvd: Reserved. This field is ignored on a write and read as zero.

Address: Physical address of the erroneous 64-bit memory word. This field is read only.

5.5.3.2 Correctable Error Data Register

This register records the uncorrected 64-bit memory word when a single-bit error is captured. The contents of this register is valid only if bit SBE of the Correctable Error Address Register is set to one.

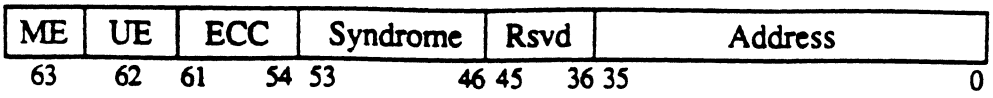
The Correctable Error Data Register is read-only.

NOTE: This register may be written but there is no guarantee that it will retain the value written. Any memory read will overwrite the current contents of this register. This register can be written for testing purpose by POST when no main memory access is allowed.

5.5.3.3 Uncorrectable Error Address Register

This register captures the first occurrence of a double-bit, T, Q or MU error. This register is accessed as a double word.

The Uncorrectable Error Address register has the following structure:



The various bit fields have the following meaning:

ME: Multiple Error. If an uncorrectable error (D, T, Q or MU error) is detected while UE=1, this bit is set to one but the rest of the register is left unchanged. The ME bit is cleared on any write to the register.

UE: Uncorrectable Error. This bit is set to one when the ECC logic detects an uncorrectable error (D, T, Q or MU error). The Uncorrectable Error Address register is updated when an uncorrectable error occurs only if this bit is clear at the time the error is detected. A level 15 broadcast interrupt with INTSID=0x03 is issued when the UE bit switches from 0 to 1. The interrupt is issued with INTSID=0x04 if one or multiple uncorrectable errors are detected along with one or multiple single-bit errors during the same access. The UE bit is cleared on any write to the register.

ECC: ECC code associated with the 64-bit memory word. This field is read-only.

Syndrome: Syndrome. This field is read only. The syndrome value may identify the nature of the error (See table on page 104).

Rsvd: Reserved. This field is ignored on a write and read as zero.

Address: Physical address of the erroneous 64-bit memory word. This field is read only.

5.5.3.4 Uncorrectable Error Data Register

This register records the uncorrected 64-bit memory word when a D, T, Q or MU error is captured. The contents of this register is valid only if bit UE of the Uncorrectable Error Address Register is set to one.

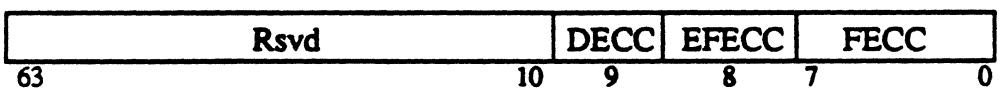
The Uncorrectable Error Data Register is read-only.

NOTE: **This register may be written but there is no guarantee that it will retain the value written. Any memory read will overwrite the current contents of this register. This register can be written for testing purpose by POST when no main memory access is allowed.**

5.5.3.5 ECC Diagnostic and Control Register

This register is provided for diagnostic purposes. It enables to force the ECC code of any memory word. This register is accessible in read and write mode as a double-word.

The Memory Diagnostic Register has the following structure:



The various bit fields have the following meaning:

Rsvd: Reserved. This field is ignored on a write and read as zero.

DECC: Disable ECC. When this bit is set to zero, MQH checks ECC on memory reads. When this bit is set to one, MQH does not check ECC on reads.

EFECC: Enable Forced ECC Code. When this bit is set to one, the FECC field is written into memory ECC bits on writes. When this bit is set to zero, MQH computes the ECC bits based on the data which is written to memory.

FECC: Forced ECC Code. If EFECC is set to one, this field is substituted for the ECC code for every 64-bit word written in the DRAM array controlled by the MQH. This allows any error to be generated on demand for diagnostic purposes (See section 5.8.4 on page 110 for an explanation on the ECC testing).

5.6 Memory Timing Registers

The MQH contains twenty 64-bit Memory Timing registers to accommodate the timing characteristics of different DRAM chips. Only the 11 low-order bits of each register are significant. The53 most significant bits are reserved; they are read as zero and are ignored on a write.

A register is selected by bits [7:3] of the physical address in the address range specified on the table of page 86.

The following table provides the Memory Timing register values for the first generation of SIMMs manufactured by Mitsubishi Semiconductor and Texas Instrument with 4 M-bit and 16 M-bit DRAM. It also provides the value for the first generation of NV_SIMMs manufactured by Mitsubishi.

PA[19:0]	Regular SIMM	NV_SIMM (Default Values)
0x03000	0x0000000000000141	0x0000000000000120
0x03008	0x0000000000000021	0x0000000000000065
0x03010	0x000000000000022d	0x0000000000000025
0x03018	0x00000000000004af	0x00000000000000a7
0x03020	0x0000000000000147	0x000000000000062f
0x03038	0x0000000000000117	0x000000000000007b
0x03030	0x000000000000021b	0x0000000000000013
0x03038	0x00000000000000ca	0x0000000000000113
0x03040	0x0000000000000002	0x000000000000038b
0x03048	0x0000000000000012	0x000000000000004b
0x03050	0x0000000000000090	0x0000000000000012
0x03058	0x0000000000000000	0x0000000000000012
0x03060	0x0000000000000000	0x0000000000000092
0x03068 - 0x03098	0x0000000000000000	0x0000000000000000

There is one set of Memory Timing register for all groups of SIMMs. If SIMMs with different timing characteristics are used in each group, conservative timing parameters must be used. The current conservative parameters are those defined for the NV_SIMMs. These default values must be used whenever there is at least one

memory group composed of NV_SIMM. The values for the regular SIMMs must be used only if all active groups consist of DRAM.

NOTE: The NV_SIMM timing parameters are compatible with the current DRAM SIMMs. However, they should not be used when all memory groups are composed of DRAM as it causes a performance loss on memory access.

There will be a set of conservative parameters that will accommodate all types of DRAM chips qualified for Sun-4D systems.

For more information on the Memory Timing registers, the reader should refer to [11].

5.7 Reset

The MQH chips are reset through a system reset. The various causes for a system reset are detailed in Chapter 12.

On a reset the MQH takes the following actions:

- Reset most state-machines to idle.
- Reset all internal queues.
- Put the MQH in frozen mode i.e. set the FZN bit in the Dynabus Control and Status register (See section 5.4.2 on page 98).

The logic controlling the refresh of the DRAM chips **is not affected** by a system reset. The DRAM chips will continue to be refreshed periodically as long as the INIT bit in the Dynabus Control and Status register is not set to one through a JTAG access. It must also be noted that the MQH completes any access which is taking place on the memory bus at the time the reset line is asserted. This means that no data is lost except the data associated with write transactions that were sitting in the MQH input queues.

Note also that no visible register is cleared by a system reset. This must be done explicitly by software.

5.8 Programming notes

5.8.1 MQH Initialization

A system reset does not fully reset MQH in order to let the current memory cycle terminate and permit refresh to continue while FZN is asserted. On a power-on reset, additional steps must be performed by the POST code to reset MQH. The correct power-on sequence is as follows:

1. Set INIT and FZN to one via JTAG (FZN is already one, since a system reset occurred). This in turn sets RFEN to 0.
2. Load the DynaBus CSR via JTAG, with INIT and FZN both zero.
3. Load the memory timing registers with the default values (NV_SIMM values).
4. Read the Group Type registers (See section 5.3.5 on page 94).

5. Load the memory timing registers with the values required for the type of SIMMs used, based on the group type.
6. Set RFEN to one and load the RCNT and RDLY values in the Memory Control and Status register.
7. Setup the address decoding registers for the present memory groups.
8. Read the Group Type registers of the DRAM groups 8 times.
9. Initialize & test memory.

Loading default values into the Memory Timing registers is crucial, since access to the Group Type registers uses a refresh cycle, which is controlled through the Memory Timing registers. Notice also that the Memory Timing registers should never be updated while RFEN is one.

On other resets, the firmware must not set INIT to one, and must not modify RFEN. It should nevertheless verify that the timing registers contents agree with the type of SIMMs used.

The DRAM chips requires eight refresh cycles as initialization sequence. Reading a Group Type register causes a refresh cycle on the corresponding memory group.Therefore, reading the Group Type register of DRAM memory groups eight time before actually accessing the memory guarantees that the DRAM chips are properly initialized.

5.8.2 Main memory test

Memory tests should be performed using the Cache Controller Block Copy hardware support (See section 4.4.3 on page 41). This allows all memory tests to be performed independently of the state of the processor External Cache, and also improves memory testing speed.

5.8.3 ECC Errors Clearing

The SBE bit in the Correctable Error Address register and the UE bit in the Uncorrectable Address register must be cleared though a SWAP access on the high-order 32 bits of the corresponding register. The SWAP returns the value of ME and SBE (for the Correctable Error Address register) or ME and UE (for the Uncorrectable Error Address register) at the time these bits are cleared. This guarantees that if any error(s) occurs between the time the Correctable/Uncorrectable Address register is read and the time the SBE/UE bit is cleared the software will be informed.

5.8.4 ECC Testing

When EFECC is set to one in the ECC Diagnostic and Control register, all double-words written into memory are written with the ECC code specified in the FECC field. Because the MQH only supports read and write operations on 64 bytes sub-blocks, it must be noted that the eight double-words composing a subblock are all written with the same ECC value.

To test error cases on 64-bit data words it is necessary to do the following steps. A subblock with seven identical double-word (noted Data_0) and one double word in which an error has been introduced (noted Data_1) must be written with EFECC set to one. The FECC field must contain the ECC corresponding to the Data_0. Then the block must be read with DECC set to 0.

To test error cases on the ECC stored with a data word (noted Data_0) it is necessary to do the following. The FECC must contain the erroneous ECC value which needs to be tested. By inverting the ECC code, a data word (noted Data_1) corresponding to this ECC must be computed. A subblock with seven double-words equal to Data_1 and one double-word equal to Data_0 must be written with EFECC set to one. Then the block must be read with DECC set to 0.

5.8.5 Scrubbing

No hardware scrubbing is done by the MQH. Main memory scrubbing can be done in software by a background task to improve the uncorrectable error failure rate.

The scrubbing **must** be done by using Block_Load operations (See section 4.4.3 on page 41), not through regular memory reference. If a single-bit error is detected, it is automatically corrected by the MQH, therefore there is no need to write-back the data (in fact, Block_Store operations should never be issued while scrubbing since the Stream registers are not part of the Shared Memory Image).

5.8.6 On-line spare memory blocks

The memory unit allows software to provide hot spares for memory blocks. For large memory configurations, software should reserve a few blocks of memory (spares) which will not be used in normal operation. When a permanent single-bit ECC failure is detected during normal system operations, the kernel can reconfigure a spare memory block, copy the data from the failing block to a spare (ECC correction guarantees no data is lost unless a soft DRAM error occurs during the copying), disable physical address decoding for the failing block and setup physical address decoding for the spare block at the same physical address as the initial failing block. This feature allows to improve system availability significantly with minimal kernel impact (only the error handler for correctable ECC errors is affected).

5.8.7 Non Volatile Memory

5.8.7.1 Introduction

Sun-4D systems provide non volatile memory (also called NVRAM) by using NV-SIMMs. The main purpose of using non volatile memory is to accelerate synchronous disk writes. Write accesses to the disk can take place in two phases. First the data is copied by the disk driver into non volatile memory and the write access is acknowledged. Later, in a second phase the actual write to disk takes place when a scheduling algorithm determines it is time to do so.

Disk writes are done much faster because accessing the NVRAM is much faster than accessing current technology disks. In current Sun-4D systems, the NVRAM bandwidth is very comparable to main memory bandwidth. NVRAM provides the

same functionality as disks as data integrity is guaranteed by the non-volatility property. Namely, data is safe after the write has been acknowledged even in case of power loss. The contents of the NVRAM can still be flushed onto disks after power is reestablished.

The NVRAM can be seen as a software cache for disk writes and many different update policies are possible. The driver can schedule the actual write to disk whenever it wants. For instance, the disk can be updated by a write-through approach. In this case, the NVRAM acts as a sophisticated write-buffer. Disks can also be updated through a write-back policy. In this case, some disk writes can be cancelled as only the latest writes to the same data need to take place.

5.8.7.2 NVRAM Access

The NVRAM is part of the memory space. However, the NVRAM must always be accessed as non-cacheable memory otherwise it defeats the purpose of guaranteeing data integrity on power loss. In Sun-4D systems the memory space is always considered cacheable as the main memory is not actually updated until a cache block is flushed (See section 3.2.1 on page 24). Therefore, the NVRAM cannot be written directly from the processor through regular STORE operations because in this case the NVRAM is not actually written until the corresponding cache block is displaced. The NVRAM **must** be updated through the Block Copy/ Block Zero hardware provided by the Cache Controller (See section 4.4.3 on page 41). The NVRAM driver must include a layer to handle BCopy transfers.

It should be noted that it is possible to read the NVRAM contents through regular LOAD operations.

5.8.7.3 Address Mapping

On a power-on reset, POST accesses all MQH to determine the physical memory configuration. By reading the Group Type registers of all memory groups, POST can detect the presence of NV_SIMMs. POST should not attempt to test the NVRAM as it destroys the contents.

In the memory list passed by POST to the kernel, the NVRAM must be flagged as such.

The NV_SIMM should only be tested when they are installed or on specific request when it known to be safe. The NVRAM driver is the only piece of software which writes the NVRAM under regular conditions.

5.8.7.4 Battery Status Checking

The NVRAM driver can check the battery status which is indicated by bit BS of the SIMM_ID field (See section 5.3.5 on page 94). By polling the BS bits regularly, the driver can provide graceful degradation in case of battery failure. When a battery failure is detected the driver can force an immediate update of the disks. Then the disk driver should stop using the NVRAM until the defective NV_SIMM is replaced.

It must be noted that in Sun-4D systems the batteries of a memory group NV_SIMM are connected in parallel to provide redundancy. If one of the battery fails, the other batteries of the group can provide enough current to guarantee that no data is lost.

5.8.7.5 Multiple Bus Configuration

In multiple Dynabus Sun-4D systems, like SunDragon, there need to be the same number of active NV_SIMMs groups on all busses. Therefore, in SunDragon the minimum NV_RAM configuration is two groups of 4 NV_SIMMs, i.e. 8 M-bytes.

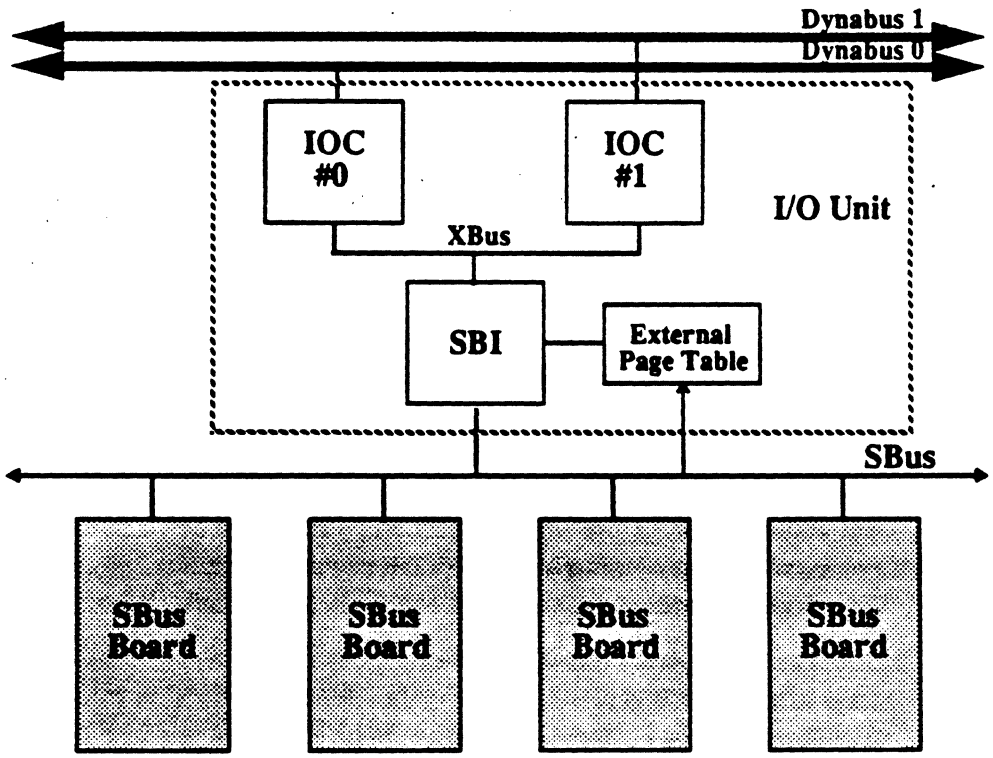
Having the same number of NV_SIMM groups on each Dynabus masks the Dynabus interleaving on DVMA operations. This way a disk update can be done through a single DVMA transfer.

There is no restriction on the number of NV_SIMM memory group in single Dynabus Sun-4D systems. In Scorpion the minimum configuration is a single group, i.e. 4 M-bytes.

Chapter 6
I/O Unit

Each of the current Sun-4D system board has an I/O unit that interfaces four SBus slots to the two Dynabusses. The I/O unit consists of one (Scorpion) or two (Sun-DRagon) I/O Cache chips (IOC) and an SBus Interface chip (SBI), using an external I/O page table (XPT). In Scorpion, there are only three SBus connectors due to packaging constraints but the fourth SBus slot is used by an on-board SC-SI/Buffered Ethernet interface.

The IOCs and the SBI are connected by an XBus. The SBI implements SBus master and slave ports. It supports stream DVMA read/write buffers for high-throughput I/O devices and provides DVMA address translation (See section 6.4.1 on page 131). The IOC contains a small fully consistent cache for low-throughput DVMA accesses (See section 6.1.2.1 on page 116). It also has dedicated queues to support non-consistent stream DVMA accesses (See section 6.1.2.2 on page 117).



6.1 I/O Model

6.1.1 Programmed I/O

Programmed I/O consist of read and write accesses from a processor to the physical space supported by an SBus board. These accesses are not cached in the External or Internal caches of a processor. They take place in the SBus Space and have no effect on the contents of the Shared Memory Image.

The SBus interface supports all SPARC addressable quantities (byte, half-word, word, double-word). It will issue multiple back-to-back SBus transfers if the target SBus device does not support the required size (automatic resizing).

The 64-byte Block_Load and Block_Store (See section 4.4.3.1 on page 41) are also supported if the target SBus device supports burst accesses. The interface will issue back-to-back bursts to perform a Block operation. Note that Block operations must be aligned on a 64-byte boundary. Unaligned transfers will cause an error which will be reported by the Cache Controller through a level-15 interrupt (See Chapter 7).

6.1.2 DVMA I/O

DVMA accesses issued by an SBus board can take place in two modes: *consistent mode* and *stream mode*. Stream mode is intended for most bulk transfers. Consistent mode is intended for random accesses and for access to control blocks shared between the driver and the DVMA device.

The mode is selected by a bit of the page table entry used to translate an SBus address into a Memory Space address.

6.1.2.1 Consistent Mode DVMA

Consistent mode DVMA accesses use the IOC cache. The IOC caches are maintained consistent in hardware. Consistent mode DVMA moves data from (or to) Memory space directly into (or from) SBus space.

All SBus transfer sizes are supported in consistent mode by the I/O unit; namely 1, 2, 4, 8, 16, 32 and 64 bytes.

In consistent mode, all store accesses issued by the same SBus board are guaranteed to be observed in the issuing order by all processors. This means that consistent DVMA accesses conform to the TSO Memory Model as defined in Chapter 2.

In consistent mode, only one transaction between an SBus board and the memory system is allowed at a given time. A read or write transaction must be completed before the next one is issued. This policy is necessary to enforce the Memory Model, but limits throughput.

NOTE: Consistent transfers using 64-byte bursts have significantly better performance than other consistent transfers.

6.1.2.2 Stream Mode DVMA

Stream DVMA accesses are buffered inside the SBI. Each SBus board has its own set of read and write buffers. This means there are 4 sets of Read and Write buffers in the SBI. These buffers *are not part* of the Shared Memory Image.

All SBus transfer sizes are supported in stream mode by the I/O unit, namely 1, 2, 4, 8, 16, 32 and 64 bytes.

Stream Mode DVMA has the following restrictions:

1. Consistency is maintained by software, which must invalidate the read buffers at the beginning of a stream read DVMA and flush the write buffers at the end of a stream write DVMA operation.
2. A DVMA device should issue consecutive increasing addresses on reads. If it does not, performance will be reduced due to unneeded prefetches, but the operation will still be performed correctly.
3. A DVMA device should issue consecutive increasing addresses on writes. If it does not, performance will be greatly reduced due to partial sub-block updates, but the operation will still be performed correctly: *only* the bytes written by the DVMA device will be modified in main memory.

Those conditions imply that memory buffers which are used for stream DVMA should not be used by other processors or I/O devices while a transfer is in progress, since they do not obey the normal TSO/PSO memory model. Communications between the driver and an intelligent SBus card should use consistent mode DVMA.

6.1.2.3 Stream Read Buffers

There are two 64-byte read buffers per SBus board in the SBI. Each buffer has a physical address (or tag) and a valid bit associated with it. On each read access in stream mode, one of the two read buffers is selected by comparing its tag with the physical address delivered by the I/O MMU and checking the valid bit.

If there is a valid read buffer with a tag match, the appropriate part of the read buffer is returned immediately to the SBus device. If there is no valid read buffer with a tag match, a rerun acknowledge is issued to the SBus device while the requested block is fetched from memory.

Accesses to a read buffer also initiate a prefetch for the next sequential block, except if this would cross a page boundary. If the DVMA operation does not use consecutive increasing addresses, the transfer will still occur correctly, though very inefficiently.

Although the read buffers are not part of the shared memory image, it is important to note that 64-byte blocks are loaded in the read buffer from the shared memory image: the block read was consistent with cached copies at the time of the read.

A device driver using stream mode *must* invalidate the read buffers when initiating a new stream DVMA read transfer to ensure that the data provided to the DVMA device is up-to-date (See section 6.4.2.5 on page 138). The invalidation could be

performed, for example, as part of the DVMA demapping process. Note that it is possible to invalidate the read buffers even while a following stream DVMA read is in progress, the only effect would be to force a reload of the stream buffers from memory.

6.1.2.4 Stream Write Buffers

There are two 64-byte write buffers per SBus board. Each buffer has a physical address (or tag) and 64 modified bits (one per byte) associated with it[1]. On each write access in stream mode, one of the two write buffers is selected by comparing its tag with the physical address delivered by the I/O XPT.

If there is a write buffer with a tag match, it is immediately updated with the SBus data and the modified bits corresponding to the bytes written are set. If there is no write buffer with a tag match, one of the write buffers is selected, flushed if it was marked modified, its tag set to the new value, the SBus data written into it and the modified bits corresponding to the bytes written set.

When a write buffer is flushed, all bytes marked as modified in the buffer are updated in the shared memory image and the buffer is marked as not modified.

If the DVMA operation does not use consecutive increasing addresses, performance may be reduced drastically.

When the last access of a stream DVMA write transfer has completed, the write buffers for the slot *must* be flushed by the driver (See section 6.4.2.5 on page 138) to make the data visible in the shared memory image. Note that it is possible to flush the write buffers even while a following stream DVMA write is in progress, the only effect being to slow down the current DVMA operation.

6.2 Address Map

This section presents the address map for the I/O unit. The I/O unit uses CSR space and ECSR space.

The following table describes the CSR space for an I/O unit:

CSR Space: PA[35:20] = 1111 1110 dddd dddd$_2$			
PA[19:0]	**Size**	**Access**	**Description**
0x00000	W	RO	IOC Component ID Register
0x00010	D	R/W	IOC Dynabus Control & Status Register
0x00018	D	R/W	IOC DynaData Register
0x01000	W	R/W	IOC Control Register
0x80000 0x80F00	W	R/W	IOC Dynabus Tags

1. This is a conceptual model. The current implementation is slightly different, but behaves exactly as if there was a modified bit per byte.

CSR Space: PA[35:20] = 1111 1110 dddd dddd$_2$			
PA[19:0]	Size	Access	Description
0xA0000 0xA0F00	W	R/W	IOC SBus Tags
0xC0000 0xC0F00	D	R/W	IOC Cache State Bits
0xE0000 0xE0F3F	H,W,D	R/W	IOC Cache Data

Note that bits [9:8] of the address are the bus selector in CSR Space and that this address map should thus be interleaved 4 times with a step of 256 bytes.

The following table describes the ECSR space address map for a processor unit:

ECSR Space: PA[35:25] = 1111 dddd ddd$_2$			
PA[24:0]	Size	Access	Description
0x0000000 to 0x000FFFC	W	R/W	SBI External Page Table
0x0800000	W	RO	SBI Component ID Register
0x0800004	W	R/W	SBI Control Register
0x0800008	W	R/W	SBI Status Register
0x0800010	W	R/W	SBus Slot 0 Configuration Register
0x0800014	W	R/W	SBus Slot 1 Configuration Register
0x0800018	W	R/W	SBus Slot 2 Configuration Register
0x080001C	W	R/W	SBus Slot 3 Configuration Register
0x0800020	W	R/W	SBus Slot 0 Stream Buffers Control Registers
0x0800024	W	R/W	SBus Slot1 Stream Buffers Control Registers
0x0800028	W	R/W	SBus Slot2 Stream Buffers Control Registers
0x080002C	W	R/W	SBus Slot3 Stream Buffers Control Registers
0x0800030	W	R/W	SBI Interrupt State Register
0x0800034	W	R/W	SBI Interrupt TargetID Register
0x0800038	W	WO	SBI Interrupt Diagnostic Register

6.3 I/O Cache Chip

The I/O Cache chip, or IOC, supports consistent mode DVMA transfers by implementing a small internal cache. It also acts as a buffered interface between SBI and Dynabus for stream DVMA transfers and programmed I/O.

6.3.1 Cache Organization

The IOC Cache is a 256-byte, write-back, fully-associative cache. It is organized in four 64-byte lines.

The IOC Cache is maintained consistent with other caches in the system (processor External caches and IOC caches). It is physically addressed both from the SBus and Dynabus ports.

The replacement algorithm dedicates a line for each SBus board (line *n* is reserved for SBus slot number *n*). When a miss is handled the victim line is chosen based on the SBus board which caused the miss. In SunDragon, there are two IOCs per SBus which means that each SBus board in fact has two 64-byte cache lines. In Scorpion, the I/O unit is composed of a single IOC and therefore there is a single line per SBus board.

The IOC Cache is *non-blocking*. It can still handle requests from SBus when a miss is processed provided the new requests are from other SBus cards than the one for which the miss occurred. The IOC Cache can handle one miss at a time. If another miss occurs while the first miss is pending no other requests coming from SBus are handled. Although the IOC Cache cannot handle multiple misses, the implementation guarantees that the activity of an SBus board has minimal detrimental effect on the activity of others.

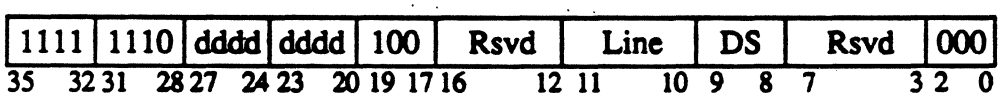
6.3.2 Access to the cache

The IOC Cache is accessible via CSR Space for diagnostic purposes. Accesses to the IOC Cache while DVMA is in progress may have unpredictable side effects.

Accesses to the IOC Cache must be performed with the specified data size. A read with a smaller size will be performed normally. In the case of a read with a larger size, the non-existent bits will be undefined. All writes with a size different from the specified one will result in a time-out bus error. SWAP and LDSTUB operations are supported if the corresponding write is supported.

6.3.2.1 Dynabus Tags

The Dynabus directory is used to snoop Dynabus transactions. It may be read and written as words. The address format is as follows:



where:

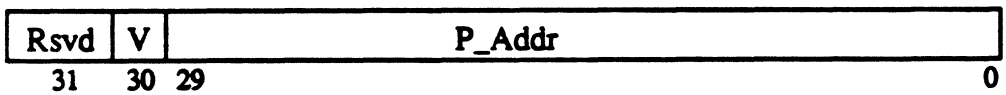
DS: Dynabus Select. In a multiple Dynabus system, the Dynabusses are interleaved on a 256-byte boundary. This field selects the Dynabus to which the referenced IOC is connected (See section 3.2.5 on page 26). In SunDragon only bit[8] is used. In Scorpion, this field is ignored.

Line: Line Index. Selects the referenced entry in the directory.

dddd dddd: DeviceID of the referenced IOC.

Rsvd: Reserved. All reserved bits are ignored, they must be zero.

The format of a Dynabus tag is:



The various bit fields have the following meaning:

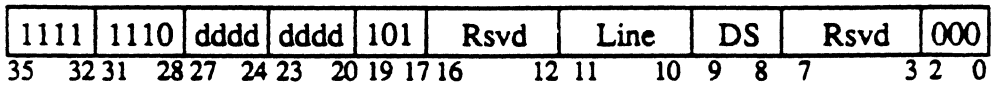
P_Addr: Physical Address bits [35:6].

V: Valid bit. This bit indicates if the physical address and only the physical address is valid. After a power-on reset the valid bits are undefined and must be initialized by software.

Rsvd: Reserved. Read as zero and ignored on a write.

6.3.2.2 SBus Tags

The SBus directory is used for SBus accesses. It may be read and written as words. The address format is as follows:



where:

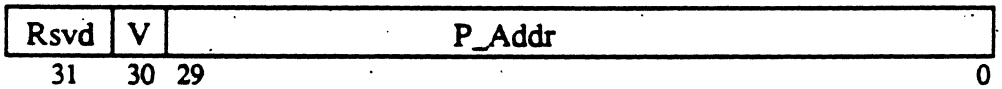
DS: Dynabus Select. In a multiple Dynabus system, the Dynabusses are interleaved on a 256-byte boundary. This field selects the Dynabus to which the referenced IOC is connected (See section 3.2.5 on page 26). In SunDragon only bit[8] is used. In Scorpion, this field is ignored.

Line: Line Index. Selects the referenced entry in the directory.

dddd dddd: DeviceID of the referenced IOC.

Rsvd: Reserved. All reserved bits are ignored.

The format of an SBus tag is:



The various bit fields have the following meaning:

P_Addr: Physical Address bits [35:6].

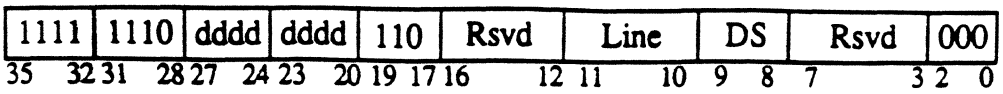
V: Valid bit. This bit indicates if the physical address is valid. After a power-on reset the valid bits are undefined and must be initialized by software.

Rsvd: Reserved. Read as zero and ignored on a write.

6.3.2.3 Cache State Bits

The Cache State bits associated with each line of the IOC Cache are shared by Dynabus and SBus directories. They may be read and written as words. The address

format is as follows:



where:

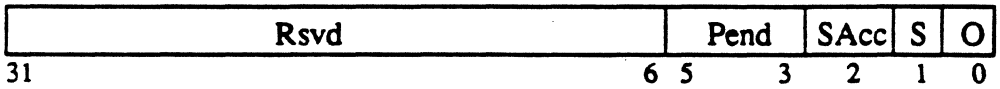
DS: Dynabus Select. In a multiple Dynabus system, the Dynabusses are interleaved on a 256-byte boundary. This field selects the Dynabus to which the referenced IOC is connected (See section 3.2.5 on page 26). In SunDragon only bit[8] is used. In Scorpion, this field is ignored.

Line: Line Index. Selects the state bits associated with the corresponding line.

dddd dddd: DeviceID of the referenced IOC.

Rsvd: Reserved. All reserved bits are ignored.

The format of the state bits associated with each line of the IOC Cache is:



where:

O: Owned. When set to one, the line is "owned" by this IOC Cache. This means that, of all cached copies, this line has been modified most recently.

S: Shared. When set to one, the line is potentially shared by other IOC or processor caches.

SAcc: Shared Accumulator. This bit is used to compute the state of the S bit for a line being fetched from the memory system. For more details on the Dynabus cache consistency protocol, the reader may refer to [4].

Pend: Pending. This field indicates which operations are pending on this cache line according to the following encoding (See [20] for details):

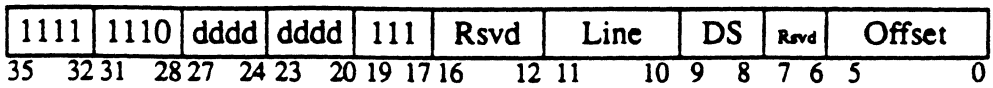
Pend (Binary)	Meaning
000	No pending Dynabus operation
001	ReadBlock pending
010	Reserved
011	FlushBlock pending
100	1 WriteSingle/SwapSingle pending
101	2 WriteSingle/SwapSingle pending
110	3 WriteSingle/SwapSingle pending
111	4 WriteSingle/SwapSingle pending

NOTE: The Pend field does not contain all the state associated with a pending operation. Modifying the Pend field should be done exclusively to test the state RAM, while all SBus cards are disabled.

Rsvd: Reserved. Read as zero and ignored on a write.

6.3.2.4 Cache Data

The Cache data may be read and written as bytes, half-words, words or double-words. The address format is as follows:



where:

Offset: Selects the addressed item within the selected line.

DS: Dynabus Select. In a multiple Dynabus system, the Dynabusses are interleaved on a 256-byte boundary. This field selects the Dynabus to which the referenced IOC is connected (See section 3.2.5 on page 26). In SunDragon only bit[8] is used. In Scorpion, this field is ignored.

Line: Line Index. Selects the referenced line.

bbbb dddd: DeviceID of the referenced IOC.

Rsvd: Reserved. All reserved bits are ignored.

6.3.3 Registers

The IOC registers are accessible via CSR space. The 16 most significant bits of the physical address must be PA[35:20]=`1111 1110 dddd dddd`$_2$, where **dddd dddd** is the DeviceID of the referenced IOC.

The address map for the IOC registers is as follows:

PA[19:0]	Size	Access	IOC Registers
0x00000	W	RO	Component ID Register
0x00008	D	R/W	Dynabus Control & Status Register
0x00010	D	R/W	DynaData Register
0x01000	W	R/W	Control Register

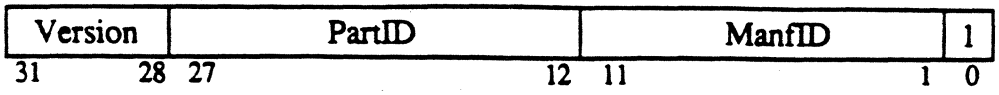
Note that bits [9, 8] of the physical address are used to select which IOC is addressed as explained in section 3.2.5 on page 26.

Accesses to IOC registers must be performed with the specified data size. A read with a smaller size will be performed normally. In the case of a read with a larger size, the non-existent bits will be undefined. All writes with a size different from the specified one will result in a time-out bus error. SWAP and LDSTUB are allowed if a store of the corresponding size (respectively word and byte) is allowed. Accesses to unspecified addresses within the IOC CSR space will result in a time-out bus error.

6.3.3.1 ComponentID Register

This register is read only. It is accessible as a word.

The ComponentID register has the following format:



The various bit fields have the following meaning:

Version: 4 bit version number, set to `0x1`.

PartID: Part ID. 16 bit part number, set to `0x0ADD` (decimal 2781).

ManfID: Manufacturing ID. 11 bits Sun JEDEC Identifier, set to `0x03E`.

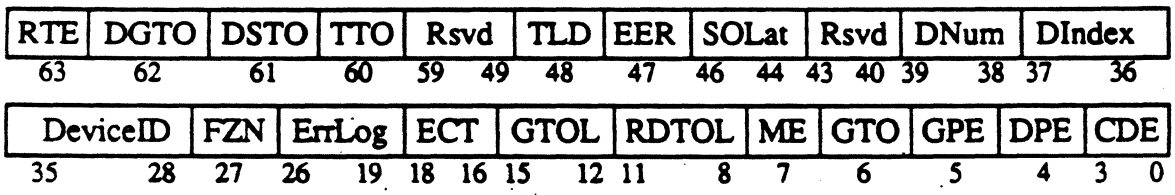
The ComponentID value for the IOC is: `0x10ADD07D`.

6.3.3.2 Dynabus Control and Status Register

This 64-bit register contains certain Dynabus parameters which must be loaded via JTAG after a power-on reset. It is also used to record some information when a **fatal error** is detected by the IOC. A fatal error is an error that causes a system watchdog reset (See Chapter 7).

This register is accessible as a double word. It is also accessible via JTAG as part of shadow scan chain 0 (See section 10.3.6 on page 185).

The Dynabus Control and Status register has the following structure:



where:

RTE: RAM Test Enable. When this bit is set to one, the IOC is placed in a special test mode for on-chip RAMs. When this bit is set to zero, IOC operates normally. In current Sun-4D systems, this bit *must* be set to *zero*. This bit is read-only.

DGTO: Disable Grant Time-out. When this bit is set to one, Grant time-outs are disabled (see GTOL field for a description of Grant time-outs). When this bit is set to zero, Grant time-outs are enabled. This bit should be set to zero in normal operation. This bit is read-only.

DSTO: Disable Store Time-out. When this bit is set to one, store time-outs are disabled (see RDTOL field for a description of store time-outs). When this bit is set to zero, store time-outs are enabled. This bit should be set to zero in normal operation. This bit is read-only.

TTO: Test Time-Outs. When this bit is set to one, the time-out periods indicated by the GTOL and RDTOL fields are divided by 1024. When this bit is set to zero, the time-out periods are computed normally. In current Sun-4D systems, this bit *must* be set to *zero*. This bit is read-only.

Rsvd: Reserved. Reserved bits are ignored on a write and read as zero.

TLD: Top-Level Device. This bit must be set to zero if there is a second-level Dynabus cache above this IOC in the Dynabus hierarchy. It must be set to one otherwise. In current Sun-4D systems, this bit *must* be set to *one*. This field is read-only. See [4] for details on the effect of TLD.

EER: Enable Error Reset. If this bit is set to one, any fatal error detected by the IOC causes a system watchdog reset. If this bit is set to zero when a fatal error is detected no system watchdog reset is issued. However, the error information is logged normally. This bit can be read and written.

SOLat: Shared/Owner Latency. Latency in number of Dynabus cycles from the time an address appears at the input pins to the time the Shared/Owner signals are asserted at the output pins, minus 2 (See[4] for more informations on Dynabus). This field is read-only. In current Sun-4D systems, this field must be initialized to `0x2` (corresponding to a 4 cycle Shared/Owner latency) through JTAG. Note that the IOC will not function properly if a value lower than `0x2` is used.

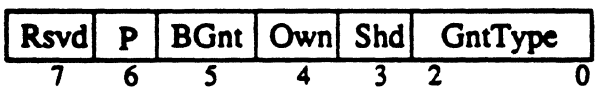
DNum: Dynabus(ses) Number - 1. Number of Dynabus(ses) in the system minus one. In SunDragon, this field must be set to 01_2. In Scorpion, this field must be set to 00_2.This field is read-only. It is initialized through JTAG.

DIndex: Dynabus Index. This field indicates the number of the Dynabus to which this IOC is connected. This field is read-only. It is initialized through JTAG.

DeviceID: Dynabus DeviceID. This field is read-only. It is initialized through JTAG. In current Sun-4D systems, the four most significant bits of this field must correspond to the board number.

FZN: Frozen. This bit behaves as if reserved on CSR space accesses (i.e. writes have no effect and reads return zero), but it is accessible via JTAG (See section 10.3.6 on page 185). FZN is set to one on a system reset. When FZN is one, IOC remains in the reset state and can only be accessed using JTAG.

ErrLog: Error Log. This field is used to log the Dynabus parity bits when a Dynabus parity error is detected. The 8 parity signals compute the byte parity over the 64 Dynabus signals. The position of the parity bit determines the byte it protects. The least significant bit protects the least significant byte of the data signals and so forth. The parity can be odd or even according to the type of Dynabus cycle (See [4] for more information). The ErrLog field is also used to log the value of the seven Board Arbiter lines when a parity error is detected on these lines. In this case, the structure of this field is the following:



where:

GntType: Grant Type.
Shd: Shared. Logical OR of the Shared lines.
Own: Owner. Logical OR of the Owner lines.
BGnt: Board Grant.

P: Even parity on the arbiter lines.

Rsvd: Reserved. Read as zero and ignored on a write.

For a detailed explanation of the arbiter signals see [9].

This field cannot be written, even through JTAG, as long as fields GTO, GPE, DPE, CDE are nonzero.

ECT: Enable Count. The value of this field specifies the type of events which are counted in the DynaData register if there is no pending fatal error. It is readable and writable. The encoding is as follows (EC1 and EC2 designate the two counter fields of the DynaData register, See section 6.3.3.3 on page 128):

Type of Events Counted		
ECT	**EC1 Counter**	**EC2 Counter**
0	Counting Disabled	Counting Disabled
1	Nbr of cycles IOC asserts Pause	Nbr of cycles IOC asserts Hold
2	Nbr of stream reads from SBI	Nbr of stream writes from SBI
3	Nbr of consistent reads from SBI	Nbr of consistent writes from SBI
4	Nbr of IOC cache misses	Nbr of IOC cache references
5	Nbr of prog. IO reads to SBI/SBus	Nbr of prog. IO writes to SBI/SBus
6	Unused - counting disabled	Unused - counting disabled
7	Unused - counting disabled	Unused - counting disabled

GTOL: Grant Time-Out Log_Value. This field specifies the Grant Time-Out master counter maximum value in log base 2 of 1 K Dynabus clock cycles. This field is readable and writable. A Grant Time-Out is reported if the arbiter does not allocate the bus N Dynabus cycles after a bus request was issued. The IOC implementation guarantees that $2^{(10+GTOL)} < N < 4*2^{(10+GTOL)}$. The time-out period is divided by 1024 if TTO is one, and grant time-outs are disabled if DGTO is one.

RDTOL: Read Data Time-Out Log_Value. This field specifies the Read Data time-out counter maximum value in log base 2 of 1 K Dynabus cycles. This field is readable and writable.The IOC supports two different time-out schemes (in the following, **RDTO** denotes a time equal to $2^{(10+RDTOL)}$):

- A time-out is reported if no reply packet for a cache miss refill (ReadBlock) is received N Dynabus cycles after a request was issued. The IOC implementation guarantees that RDTO< N < 2*RDTO. When this time-out occurs (for example as a consequence of an incorrect mapping during a DVMA operation), the SBus device which initiated the cache access will receive an error acknowledge on SBus and is responsible for signalling the error back to the driver. No information is logged in IOC.
- A store time-out occurs if a cache store operation (DVMA write or swap to a shared location in consistent mode) or an interrupt transaction does not complete in N Dynabus cycles. The IOC implementation guarantees that RDTO< N < 2*RDTO. When this error occurs, IOC sets bit STO of field CDE and initiates a system watchdog reset (fatal error). No additional information is logged[1]. Store time-outs are disabled if DSTO is one.

Note that, in both cases, the time-out duration is divided by 1024 if TTO is one.

ME: Multiple Errors. This bit is set when multiple fatal errors have been detected. This bit is cleared by writing a one. In case of multiple errors, only the first one is logged. If multiple errors are detected in the same cycle, the following priority is used:

Priority	Error
1	Grant Parity Error
2	Dynabus Parity Error
3	Grant Time Out
4	Internal Errors

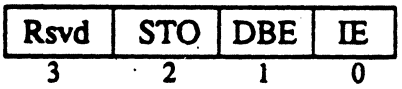
The priority between internal errors is not specified.

DPE: Dynabus Parity Error. This bit is set to one when a parity error is detected on Dynabus for either a Data or a Header cycle. When a Dynabus parity error is detected, the faulting cycle is latched in the DynaData register and the parity signals are latched in the ErrLog field. This bit is cleared by writing a one.

GPE: Grant Parity Error. This bit is set to one when a parity error is detected on the arbitration signals. When such an error is detected, the value of the arbitration signals is latched in the ErrLog field. This bit is cleared by writing a one.

GTO: Grant Time-Out. This bit is set to one when the time-out counter for a Dynabus grant overflows. This bit is cleared by writing a one.

CDE: Client Device Errors. This field is used to report internal IOC fatal errors. This field has the following structure:



where:

Rsvd: Reserved for future use. Reads as zero. Writes to this bit are ignored.

STO: Store time-out. A Dynabus time-out occurred on a cache shared write operation (WriteSingle or SwapSingle) or during transmission of an interrupt (See section 7.2.4.6 on page 153). This bit is cleared by writing a one into it.

DBE: Dynabus error. An illegal transaction was received from the Dynabus. The header (or MemFault cycle) in error is logged in the DynaData register. This bit is cleared by writing a one into it.

IE: Internal error. The detailed error reason is logged in the DynaData register (See following section). This bit is cleared by writing a one into it.

Notice that at most one of the fields DPE, GPE, GTO and CDE may be non-zero, that ME cannot be one if all of them are zero, and that at most one bit may be set in the CDE field.

---

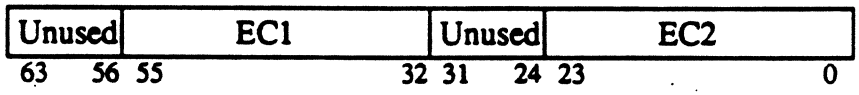
1. Note that a consistent mode store to an undefined location will be reported as a Read time-out, since the cache refill operation will time-out. The store time-out case correspond to a major hardware failure.

6.3.3.3 DynaData Register

The DynaData register may be read and written as a double-word. It is also accessible via JTAG as part of shadow scan chain 0 (See section 10.3.6 on page 185). It cannot be written, even through JTAG, as long as any error bit is set in the Dynabus Control and Status register (fields DPE, GPE, GTO, CDE).

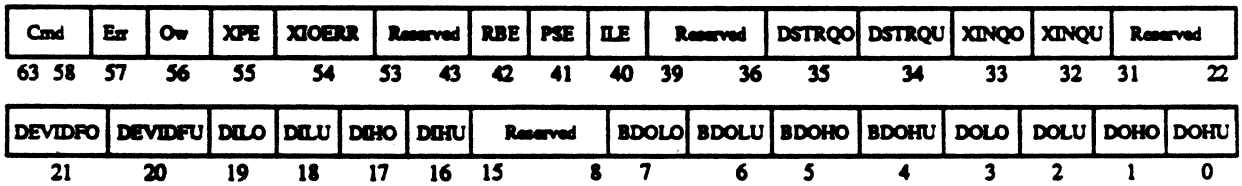
This 64-bit register serves three functions, depending on the state of the Dynabus Control and Status register. In decreasing priority order, these functions are:

- Error logging: when the Dynabus Status and Control register indicates an error (fields DPE, GPE, GTO, CDE nonzero), the DynaData register provides error logging information and *becomes read-only*. If DPE is one, the DynaData register contains the Dynabus data on which a parity error was detected. The contents of the DynaData register are unspecified when GPE or GTO is one. If the DBE bit of the CDE field of the Dynabus Control and Status register is set, the DynaData register contains the Dynabus packet header in error. If the IE bit of the CDE field of the Dynabus Control and Status register is set, the DynaData register contains detailed error information which is described at the end of this section.
- Event counting: when the ECT field of the Dynabus Control and Status register is non-zero and the DynaData register is not being used for error logging, it is used to count various internal events for performance monitoring. In this mode, it has the following format:



  where EC1 and EC2 are 24-bit wraparound counters (see ECT field on page 126 for the various events which may be counted). The two unused fields are not modified by event counting.
- JTAG communication area: the DynaData register can otherwise be used as a general communication area with the JTAG master.

The detailed error logging format is used when an internal error has been detected (bit IE of the CDE field of the Dynabus Control and Status register set to one) and is as follows:



where:

DOHU: Underflow of the Dynabus reply output queue.

DOHO: Overflow of the Dynabus reply output queue.

DOLU: Underflow of the Dynabus request output queue.

DOLO: Overflow of the Dynabus request output queue.

BDOHU: Underflow of the Dynabus reply output buffer.

BDOHO: Overflow of the Dynabus reply output buffer.

BDOLU: Underflow of the Dynabus request output buffer.

BDOLO: Overflow of the Dynabus reply output buffer.

DIHU: Underflow of the Dynabus reply input queue.

DIHO: Overflow of the Dynabus reply input queue.

DILU: Underflow of the Dynabus request input queue.

DILO: Overflow of the Dynabus request input queue.

DEVIDFU: Underflow of the DeviceID register file.

DEVIDFO: Overflow of the DeviceID register file.

Reserved: Reads as zero.

XINQU: Underflow of the XBus input cache queue.

XINQO: Overflow of the XBus input cache queue.

DSTRQU: Underflow of the DataStore queue.

DSTRQO: Overflow of the DataStore queue.

ILE: Invalid Line Error. A cache line was not marked valid when an IOC reply is received from Dynabus. Additional header information is logged in the Cmd, Err and Ow fields.

PSE: Pending State Error. A cache line was not marked as pending when an IOC reply is received from Dynabus. Additional header information is logged in the Cmd, Err and Ow fields.

RBE: ReadBlock Error. A cache line was marked owned when an IOC ReadBlock reply is received from Dynabus. Additional header information is logged in the Cmd, Err and Ow fields.

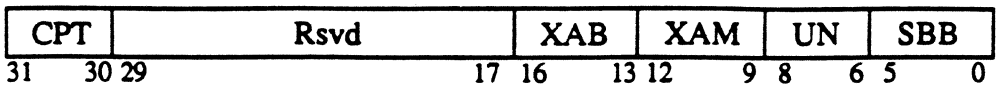
XIOERR: SBI detected a fatal error. Additional information is logged in the SBI Status register (See section 6.4.2.3 on page 135).

XPE: An XBus parity error occurred.

Ow, Err, Cmd: These fields are valid only if ILE, PSE or RBE is set. They then contain respectively bits [54], [56] and [63:58] of the header of the Dynabus packet which raised the error.

6.3.3.4 Control Register

The Control register contains configuration bits for the IOC. It may be read and written as a word. Its format is as follows:



where:

CPT: Cache Partition. This field controls the cache replacement algorithm. The encoding is:

CPT	LRU Size
00	4-way split
01	2-way split
10	Global
11	Reserved

Four-way split is the normal operation mode (See section 6.3.1 on page 119) where a miss from slot *n* victimizes line *n*. In two-way split, a miss from slots 0 or 1 will use lines 0 or 1. In global mode, a miss from any slot may use any line. Only four-way split should be used in current Sun-4D systems. CPT is set to 00_2 on reset.

Rsvd: Reserved. Read as zero and ignored on a write.

XAB: Extended Address Base. This field is provided for future extensions and *must* be initialized to 1111_2 in current Sun-4D systems. XAB is set to 1111_2 on reset.

XAM: Extended Address Mask. This field is provided for future extensions and *must* be initialized to 0000_2 in current Sun-4D systems. XAB and XAM are used to increase the range of I/O space addresses forwarded by IOC to XBus: if the four most-significant bits of the address "*anded*" with XAM are equal to XAB, the Dynabus access is forwarded to the XBus. The required values of XAB and XAM ensure this never occurs in current Sun-4D systems. XAM is set to 0000_2 on reset.

UN: Unit Number. This field corresponds to the unit number of the I/O unit used for the Extended CSR space addressing (See section 3.2.4 on page 26). The UN field should be initialized to 001_2 in current Sun-4D systems. UN is set to 000_2 on reset.

SBB: SBus Space Base. This field is used to decode SBus space addresses. It is compared with the 6 most significant bits of physical addresses to determine if an access must be forwarded to the SBI. This field must be initialized to $10\ bbbb_2$ in current Sun-4D systems, where bbbb is the board number where the IOC resides (See section 3.2.7 on page 28). SBB is set to 0000002 on reset.

6.3.4 Reset

The IOCs are reset through a system reset. The various causes of system reset are detailed in Chapter 12.

On a reset the IOC takes the following actions:

- Reset all state-machines to idle.
- Reset all internal queues.
- Put the IOC in frozen mode i.e. set the FZN bit in the Dynabus Control and Status register (See section 6.3.3.3 on page 128).

Note that no other software-visible register is modified. It is the responsibility of software to initialize all visible registers to a proper state. The proper initialization sequence is to setup the Dynabus Control and Status register via JTAG (which also clears FZN), and *immediately* afterwards load the Control register via CSR space. No other reference to I/O space should be done before the Control register is initialized.[1]

6.4 SBus Interface Chip

The SBus Interface chip, or SBI, implements the functions of SBus controller, interrupt manager and gateway between SBus and XBus. As SBus controller, it handles SBus arbitration and DVMA mapping through an I/O MMU implemented using an external RAM array called the External Page Table (XPT). As interrupt manager, it transforms SBus interrupts into Dynabus interrupts and provides facilities to identify interrupt sources. Finally, as a gateway between SBus and XBus, it provides the read and write buffers for stream mode DVMA (See section 6.1.2.2 on page 117), forwards consistent mode DVMA to the IOCs and supports programmed I/O from processors to SBus devices.

The SBus implementation provided by SBI conforms to the SBus Specification, Rev.B0 [13], including parity support (which can be enabled on a per-slot basis) and all burst sizes. It does not support the 64-bit transfer protocols. In addition, the Reset and Interrupt lines are not shared between slots, which provides more flexibility.

SBus resources are accessible via the ECSR space with PA[35:24] = 1111 dddd ddd0$_2$, where dddd ddd is DeviceID[7:1] for the I/O unit.

NOTE: In the decoding of ECSR Space, the 3 least significant bits of DeviceID[7:1] are not taken from the IOC's Dynabus CSR DeviceID field but from the IOC's Control Register UN field (See section 6.3.3.4 on page 130). POST initializes this field to DeviceID [3:1].

6.4.1 I/O MMU

The I/O MMU consists of an External Page Table (XPT) containing the Page Table Entries (PTE) and control logic implemented in the SBI. The XPT contains 16 K 32-bit wide entries. Each entry can hold a PTE mapping a 4 K-byte page, allowing for 64 M-bytes of DVMA address space.

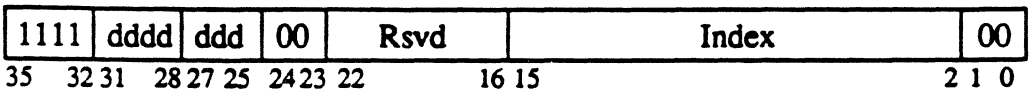
---

1. The key point is that the XAB, XAM and SBB fields must be initialized immediately to prevent incorrect decoding. The IOC implementation guarantees that accesses to the IOC CSR space will operate correctly even if the Control register contains random values.

The XPT implements a **single- level page table**. SBus address bits [25:12] are used to index the XPT in order to provide a PTE to SBI. The XPT is managed entirely in software. If there is a miss, an error acknowledge is returned to the SBus board which issued the access. It is the responsibility of device drivers to load all necessary PTEs in the XPT before a DVMA transfer is initiated.

The XPT is accessible in ECSR Space (See section 3.2.6 on page 28). The entries are readable and writable as words only. Other data sizes generate a BE bus error. SWAP and LDSTUB operations also result in a time-out bus error.

The address format is as follows:



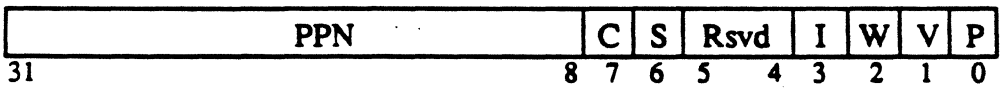
where:

Index: Selects which of 16384 entries is referenced.

Rsvd: Reserved. All reserved bits are ignored and should be set to 0.

dddd ddd: DeviceID[7:1] for this I/O unit.

The PTE format is:



The various bit fields have the following meaning:

PPN: Physical Page Number. High-order 24 bits of the 36-bit physical address.

C: Cacheable. If this bit is set to one, the access is directed to Memory Space (Shared Memory Image). If this bit is set to zero, the access is directed to I/O space (*the SBI ignores the C bit and behaves as if C=1*).

S: Stream Enable. When this bit is set to one, the access is done in Stream Mode. When this bit is set to zero, the access is done in Consistent Mode.

Rsvd: Reserved. This field is not interpreted by the hardware. It can be read and written. The software can use this field at its discretion.

I: Intra SBus. If this bit is set to one, the access takes place directly on the same SBus without going over Dynabus. If this bit is set to zero, the access takes place in the SunDragon memory space.

W: Write Enable. This bit must be set to one for the page to be writable by an SBus device. If an SBus device attempts to write the page while this bit is clear, the access is acknowledged with an error.

V: Valid. This bit must be set to one to indicate a valid entry. If this bit is clear, all SBus accesses to this page are acknowledged with an error.

P: Parity. The parity of a PTE must be odd. If the DXP bit in the Control register (See section 6.4.2.2 on page 134) is zero, this bit is ignored on writes and the P bit for the PTE is generated by the SBI to provide the parity specified by the FXP bit (0: odd, 1: even). If the DXP bit is one, this bit is written in the PTE as specified.

6.4.2 Registers

The SBI registers are accessible via ECSR space. The 13 most significant bits of the physical address must be PA[35:23]=`1111 dddd ddd0 1` (See section 3.2.6 on page 28), where `dddd ddd` is DeviceID[7:1] for this I/O unit.

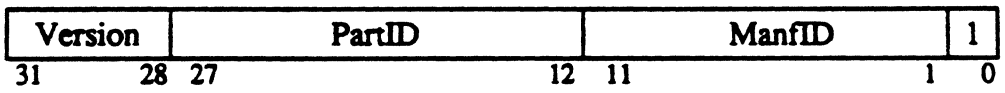
The address map for the SBI registers is as follows:

ECSR Space: PA[35:23] = `1111 dddd ddd0` 1_2			
PA[22:0]	**Size**	**Access**	**SBI Register**
`0x00000`	W	RO	Component ID Register
`0x00004`	W	R/W	Control Register
`0x00008`	W	R/W	Status Register
`0x00010`	W	R/W	Slot 0 Configuration Register
`0x00014`	W	R/W	Slot 1 Configuration Register
`0x00018`	W	R/W	Slot 2 Configuration Register
`0x0001C`	W	R/W	Slot 3 Configuration Register
`0x00020`	W	R/W	Slot 0 Stream Buffers Control Registers
`0x00024`	W	R/W	Slot1 Stream Buffers Control Registers
`0x00028`	W	R/W	Slot2 Stream Buffers Control Registers
`0x0002C`	W	R/W	Slot3 Stream Buffers Control Registers
`0x00030`	W	R/W	Interrupt State Register
`0x00034`	W	R/W	Interrupt TargetID Register
`0x00038`	W	WO	Interrupt Diagnostic Register

Accesses to the SBI registers must be performed with the specified data sizes. Any access with a size different from the specified ones will result in a BE bus error. SWAP is supported only to the Interrupt State register. LDSTUB or SWAP operations to other registers will result in a BE bus error. Accesses to unspecified addresses within the SBI ECSR space will result in a TO bus error.

6.4.2.1 ComponentID Register

This register is read only. It is accessible as a word. Its format is:



The various bit fields have the following meaning:

Version: 4 bit version number, set to `0x1` or `0x2`.

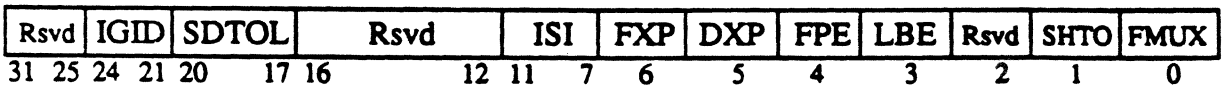
PartID: Part ID. 16 bit part number, set to `0x0ADE` (decimal 2782).

ManfID: Manufacturing ID. 11 bits Sun JEDEC Identifier, set to `0x03E`.

The ComponentID values for the SBI are: `0x10ADE07D` and `0x20ADE07D`. There is no functional difference between both versions of the SBI. The first version contains some logical bugs which are fixed in the second version. Only the second version will be shipped but the first version may be used for system bring-up.

6.4.2.2 Control Register

This register provides control, configuration and diagnostic features for the SBI. It may be read and written as a word. The format of the Control register is as follows:



where:

Rsvd: Reserved. Read as zero and ignored on a write.

IGID: Interrupt Group Identifier. This field specifies the interrupt dispatch group to which this SBI belongs. See section 6.4.2.7 on page 140 for the use of this field.

SDTOL: Slave Data Time-Out Log_Value. This field specifies the Slave Data time-out counter maximum value in log base 2 of 1K Dynabus cycles (or of 32 Dynabus cycles if SHTO is one). This field is readable and writable.The time-out is applied to DVMA operations[1]. If an operation does not complete in the required time, it behaves as if a time-out bus error had been received. The exact time-out value is between $2^{10+SDTOL}$ and $2*2^{10+SDTOL}$ Dynabus clock cycles.

ISI: Interrupt Source ID. This field contains the 5 least significant bits of the Interrupt Source ID (INTSID) for all interrupts issued by the SBI. This includes both SBus interrupts and level-15 interrupts used to signal errors. This field *must* be initialized to $0dddd_2$, where $dddd_2$ are the four most significant bits of the I/O unit DeviceID. Because, of the geographical addressing used in current Sun-4D systems, these four bits are also equal to the backplane slot number.

FXP: Force External Page Table Parity Errors. This bit is active only if DXP is zero. If this bit is zero, odd parity is generated on PTE writes. If it is one, even parity is generated on PTE writes. This bit must be 0 in current Sun-4D systems under normal conditions.

DXP: Disable External Page Table Parity. When this bit is zero, the SBI checks PTE odd parity on DVMA address translations and forces parity on writes to PTEs based on the value of FXP. When this bit is one, parity checking is disabled and PTE parity is modified on writes as required by the programmer. This bit should be 0 in current Sun-4D systems under normal conditions.

---

1. Responsibility for time-out implementation is shared between IOC and SBI. In normal operation, the RDTOL field of the IOC Dynabus Control and Status register and the SDTOL field of the SBI Control register should be set to the same value.

FPE: Force Parity Error. Setting this bit to one causes an SBus parity error on all subsequent SBus master transfers issued by the SBI and SBus DVMA reads. This feature can be used in conjunction with the loopback mode to test the parity checking hardware. This bit should be 0 in current Sun-4D systems under normal conditions.

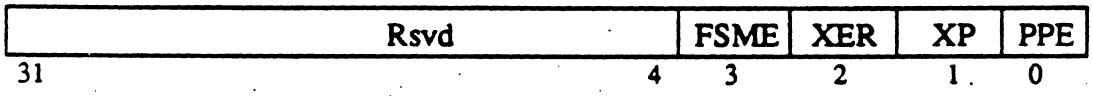
LBE: LoopBack Enable. Setting this bit to one puts the SBI in DVMA loopback mode. This feature is provided for diagnostic purposes and must be used withciption of loopback mode. This bit is cleared on a system reset.

SHTO: Short time-out. Setting this bit to one changes the time-out prescaling for SDTOL from 1024 to 32 Dynabus clocks. This bit is provided for chip testing and should never be set to one for normal operation.

FMUX: Flush Multiplexor. Setting this bit to one disables interrupt levels 6 and 7 for all four slots and reassigns the corresponding inputs to force respectively the FWB bit (level 6) or the IRB bit (level 7) of the corresponding Slot Stream Buffer Control register (See section 6.4.2.5 on page 138), thus initiating a stream write buffer flush (level 6) or a stream read buffer invalidate (level 7). This bit is provided for chip testing and should never be set to one for normal operation.

6.4.2.3 Status Register

This register records information on errors detected by the SBI which are not associated to a particular SBus slot. It may be read and written as a word. The format of the Status register is as follows:



where:

Rsvd: Reserved. Read as zero and ignored on a write.

FSME: FSM Error. This bit is set to one when a finite state machine inside SBI reaches an illegal state. This is a fatal error, which will result in a system watchdog reset. (See Chapter 10). This bit is cleared by writing a one.

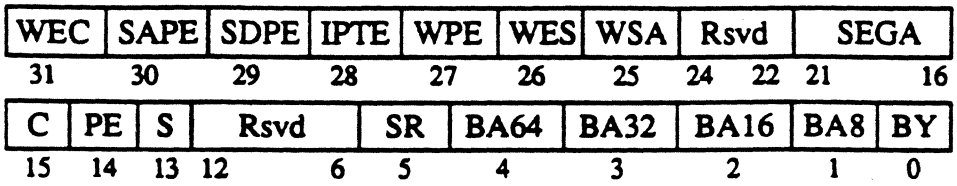
XER: XBus Protocol Error. This bit is set to one when a protocol error is detected on XBus. This is a fatal error, which will result in a system watchdog reset. (See Chapter 10). This bit is cleared by writing a one.

XP: XBus Parity Error. This bit is set to one if a parity error is detected on XBus. This is a fatal error, which will result in a system watchdog reset (See Chapter 10). This bit is cleared by writing a one.

PPE: PTE Parity Error. Set to one when a parity error is detected on a PTE access from the External Page Table to perform DVMA address translation. A level-15 interrupt (INTSID=0x05) is broadcast to inform the processors and the SBus access is acknowledged with an error. This bit is cleared by writing a one.

6.4.2.4 Slot Configuration Registers

There is a Slot Configuration register per SBus slot. These registers are used to configure the SBus slot and to record errors reported to the SBus device. They may be read and written as words. The format of a Slot Configuration register is:



where:

WEC: Write Error Consistent. This bit is set to one when an error is reported on DVMA write in consistent mode. Since DVMA writes are acknowledged by the SBI before being performed, the device that issued the write is not informed. It is the responsibility of software to check the value of this bit after a DVMA input transfer to ensure that there was no error. This bit is cleared by writing a one.

SAPE: SBus Address Parity Error. This bit is set to one when a parity error is detected on an SBus address. The transfer is acknowledged with an error. It is the responsibility of the device to inform the driver. This bit is cleared by writing a one.

SDPE: SBus Data Parity Error. This bit is set to one when a parity error is detected on a data cycle during a DVMA write transfer. The transfer is acknowledged with an error or late error. It is the responsibility of the device to inform the driver. This bit is cleared by writing a one.

IPTE: Invalid PTE. This bit is set to one if the indexed PTE is invalid in the XPT during a DVMA access. The transfer is acknowledged with an error. It is the responsibility of the device to inform the driver. This bit is cleared by writing a one.

WPE: Write Protection Error. This bit is set to one if the indexed PTE is marked read-only (W=0, see section 6.4.1 on page 131) when a device tries to do a DVMA write. The transfer is acknowledged with an error. It is the responsibility of the device to inform the driver. This bit is cleared by writing a one.

WES: Write Error Stream. This bit is set to one when an error is reported on DVMA write in stream mode. Since DVMA writes are acknowledged by the SBI before being performed, the device that issued the write is not informed. It is the responsibility of software to check the value of this bit after a DVMA input transfer to ensure that there was no error. This bit is cleared by writing a one.

WSA: Write Segment Address. This bit is write-only. When a write to a slot configuration register has a one in the WSA position, the SEGA, C and S fields (as well as all the other fields of the register) are updated. When a write to a slot configuration register has a zero in the WSA position, the SEGA, C and S fields are not modified, while all other fields in the register are updated. This feature allows to share the slot configuration register safely with the SBus device. This bit always reads as zero.

Rsvd: Reserved. Read as zero and ignored on a write.

SEGA: Segment Address. This six bit field is used when bypass mode is enabled (BY=1) and SA[31:30]=00_2. SA designates an SBus address issued on a DVMA transfer. In this case, the physical address is not obtained through the XPT, but as follows: PA[35:30]=SEGA; PA[29:0]=SA[29:0]. This field is updated on a write operation only if the WSA bit is set to one in the written data, it is not affected by a write operation otherwise. The SEGA field is also accessible by SBus devices when bypass mode is enabled (see BY field on page 138).

C: Cacheable. This bit is provided for compatibility with the Sun-4M architecture, and has no effect in current Sun-4D systems. It is updated on a write operation only if the WSA bit is set to one in the written data, it is not affected by a write operation otherwise. The C bit is also accessible by SBus devices when bypass mode is enabled (see BY field on page 138).

PE: Parity Enable. This bit must be set to one to enable SBus parity checking for the slot. Parity checking is disabled otherwise.

S: Stream mode. When this bit is one, SBus DVMA in physical address bypass mode issued by this slot operates in stream mode. When it is zero, SBus DVMA in physical address bypass mode issued by this slot operates in consistent mode. It is updated on a write operation only if the WSA bit is set to one in the written data, it is not affected by a write operation otherwise. The S bit is also accessible by SBus devices when bypass mode is enabled (see BY field on page 138).

SR: Slot Reset. Setting this bit to one resets the SBus device attached to this slot. This bit must be set to one for 512 SBus cycles for the reset to be effective, as defined by the SBus specification. It is the responsibility of the software to assert this bit long enough. This bit is also set on a system reset and when a re-run time-out occurs. In both cases, this bit must be cleared by software. Programmed I/O issued while SR is set to one will result in time-out bus errors.

BA64, BA32, BA16: 64/32/16-Byte Burst Master Enable. At least one of these bits must be set to one to enable Block operations from Dynabus to the SBus device on this slot. SBI will use the largest size specified, i.e. a single 64-byte burst if BA64 is one, two 32-byte bursts if BA64 is zero and BA32 is one, four 16-byte bursts if BA64 and BA32 are zero and BA16 is one. If all three bits are zero, SBI will not issue burst cycles to this slot and Block operations will result in an error.

BA8: 8-Byte Burst Master Enable. This mode is not supported by the SBI. This field is reserved. It is read as zero and ignored on a write. Double-word transfers are always performed by the SBI as two back-to-back word transfers.

BY: Bypass Enable. When this bit is set to one, MMU bypass is enabled for DVMA accesses issued by this slot. The behavior depends on the value of bits [31:30] of the DVMA address, SA, as shown in the following table:

SA[31:30]	Function
00	Physical Address Mode
01	Control register Access
1X	Normal Mode

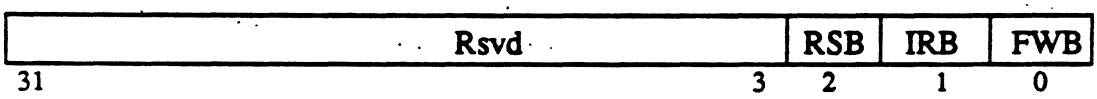
In physical address mode, the I/O MMU is bypassed and the physical address is obtained as described above: PA[35:30]=SEGA; PA[29:0]=SA[29:0]. Physical address DVMA is always directed to Memory space and uses stream mode if the S bit of the Slot configuration register is one, consistent mode if the S bit of the Slot Configuration register is zero.

When Controller register access is selected, the DVMA address is used to access registers in the SBus controller. Burst transfers are not supported and will be rejected with an error indication. If an unspecified register is accessed, the access will be acknowledged with an error. The only register currently available is the Slot Configuration register, specified using SA[29:24]=0. On DVMA writes, only the SEGA field is updated, all other fields are read-only.

In normal mode, address translation is performed as if the BY bit was not set to one.

6.4.2.5 Slot Stream Buffer Control Registers

There is one Stream Buffer Control register per slot. These registers are used to invalidate the Read Buffers and flush the Write Buffers. They may be read and written as words. The format of a Stream Buffers Control register is as follows:



The bit fields have the following meaning:

FWB: Flush Write Buffers. When a one is written into this bit, the Write Buffers associated with the slot are flushed. If a buffer is modified, its contents are written in memory and then it is marked as invalid. When the flush has been completed for both buffers, the FWB bit is cleared by the hardware. It is the responsibility of the software to check the value of this bit to make sure that a flush has completed. Writing a zero into this bit has no effect.

IRB: Invalidate Read Buffers. When a one is written into this bit, the Read Buffers associated with the slot are invalidated. When the invalidation has been completed for both buffers, the IRB bit is cleared by the hardware. It is the responsibility of the software to check the value of this bit to make sure the invalidation has completed. Writing a zero into this bit has no effect.

RSB: Reset Stream Buffers. This bit is used to initialize the stream buffers after a system reset. This operation must be done in 3 steps:

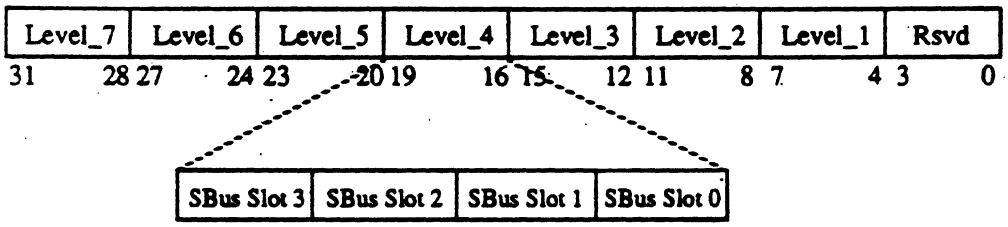
1. Write 0x7 into the Stream Buffer Control register (RSB, IRB & FWB set to one).
2. Wait until IRB=FWB=0.
3. Write 0x0 into the Stream Buffer Control register (RSB, IRB & FWB clear).

Rsvd: Reserved. This field is read as zero and ignored on a write.

NOTE: The slot streams buffers *must* be initialized following a system reset, and *only* then. Asserting RSB while DVMA operations are in progress may result in a system reset. The slot stream buffers must be initialized even if no SBus card on this SBus uses stream DVMA.

6.4.2.6 Interrupt State Register

The Interrupt State register records interrupts issued by SBus devices and provides a mechanism to allow multiple CPUs to service interrupts concurrently. It may be accessed as a word, and supports read, write and swap operations. Its format is as follows:

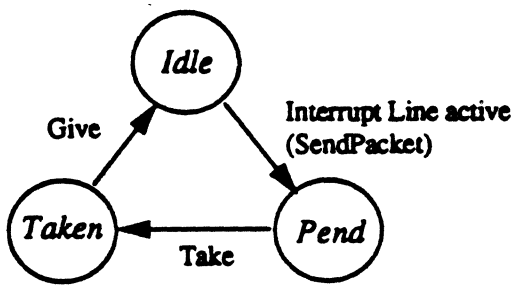


The bit fields have the following meaning:

Rsvd: Reserved. Read as zero and ignored on a write or swap.

Level_n: Four bits are dedicated to each SBus interrupt level. For each interrupt level, one bit is dedicated to each SBus slot.

Each of the 28 interrupts may be in one of three states as described in the following diagram:



On a read of the Interrupt State register, a one is returned if the corresponding interrupt is in the *Pend* state, and a zero otherwise.

A swap to the Interrupt State register performs the *Take* transition: the returned data is first computed as for a normal read, then each interrupt bit into which a one is written enters the *Taken* state if it was in the *Pend* state.

A write to the Interrupt State register performs the *Give* transition: each interrupt bit into which a one is written will enter the *Idle* state if it was in the *Taken* state.

See section 8.4 on page 163 for a description on how to use the Interrupt State register in software.

The *Idle⇒Pend* transition is performed by hardware whenever the SBus interrupt line is active. In addition, an interrupt is sent over Dynabus #0 to the processor specified in the Interrupt TargetID register. The INTSID field is computed as:

- INTSID[7:5] ← SBus Interrupt Level
- INTSID[4:0] ← ISI field of the Control register

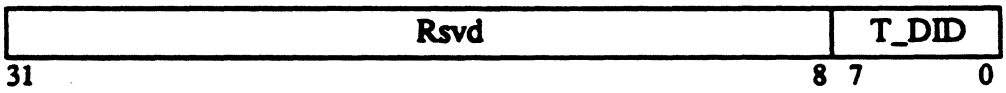
The interrupt level is derived from the SBus interrupt level according to the following table:

SBus Level	SPARC Level
1	2
2	3
3	5
4	7
5	9
6	11
7	13

On a system reset, all interrupts are set to the *Idle* state.

6.4.2.7 Interrupt TargetID Register

This register holds the Dynabus DeviceID of the Processor Unit to which SBus interrupts are forwarded. It may be read and written as a word. The format of the Interrupt TargetID register is as follows:

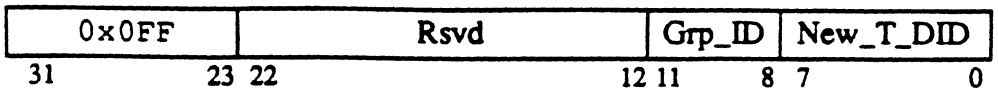


The bit fields have the following meaning:

T_DID: Target Dynabus DeviceID. DeviceID of the Processor Unit chosen to receive SBus interrupts.

Rsvd: Reserved. Read as zero and ignored on a write.

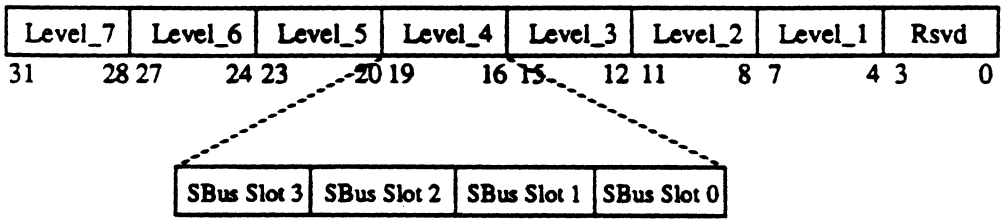
The Interrupt TargetID register is also updated when a processor issues a store to the CC Interrupt Issuing register using the second format (See section 4.4.5.4 on page 45). This format is:



If Grp_ID matches the IGID field of the Control register, the Interrupt TargetID register is updated with the value of New_T_DID. Note that this is the normal way to update the Interrupt TargetID register and that access via CSR space should be used only for diagnostic purposes.

6.4.2.8 Interrupt Diagnostic Register

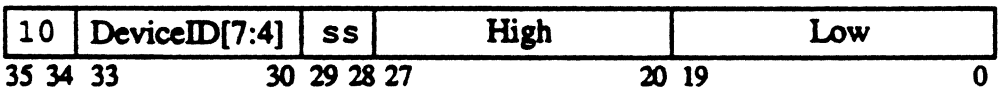
The Interrupt Diagnostic register provides a way to force SBus interrupts for diagnostic purposes. It is write-only and must be accessed as a word. The format is the same as for the Interrupt State register:



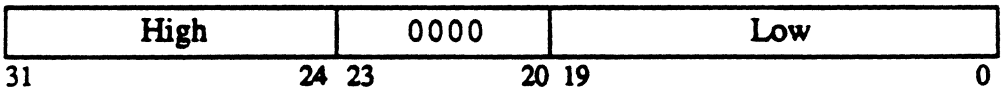
When a write to the Interrupt Diagnostic register is issued, SBI behaves as if the interrupt lines corresponding to ones in the written data word were asserted for a short period of time: if the corresponding state machine in the Interrupt State register (See section 6.4.2.6 on page 139) was Idle, it enters the Pend state and an Interrupt transaction is issued over Dynabus. If the state machine is in any other state, or if the written bit is a zero, no state change occurs.

6.4.3 Loopback mode

Loopback mode for SBI is controlled by the LBE bit of the Control register (See section 6.4.2.2 on page 134). It should be used exclusively for diagnostics. When the SBI is in loopback mode, programmed I/O accesses to the SBus space controlled by the SBI are used to simulate DVMA accesses. If a physical address of:



is issued, SBI will behave as if the device connected to SBus slot **ss** had issued a DVMA cycle of the same size (1, 2 or 4 bytes operation, or four 16-byte burst operations) as the Dynabus operation and with a DVMA address of:



Note that this address mapping allows all of the bypass cases to be properly exercised, but does not provide access to the full XPT range. XPT connectivity can be fully checked by reading and writing the XPT directly, since this uses the same path as used in DVMA mapping.

NOTE: Loopback mode should be used exclusively when all SBus devices are in the reset state, which is achieved by setting the SR bit in all four Slot Configuration registers (See section 6.4.2.4 on page 136). Unpredictable errors may occur if loopback mode is used while SBus cards are not in the reset state.

6.4.4 Reset

The SBIs are reset through a system reset. The various causes of system reset are detailed in Chapter 12.

On a reset the SBI takes the following actions:

- Reset all state-machines to idle state.
- Reset all internal queues.
- Reset all SBus slots. The bit SR in all Slot Configuration registers is set to one and must be cleared in software to enable the slot.
- Reset all interrupts to the *Idle* state.

Note that no programmer-visible register is modified, except for the Slot Configuration registers and the Interrupt State register.

Chapter 7
Error Management

This chapter describes the errors that can occur in Sun-4D systems and how they are handled. Errors which are strictly internal to the Viking processor are not described in detail here. The reader should refer to the SPARC Architecture Manual [1] for details.

7.1 Error reporting mechanisms

This section describes how the different errors are reported to the programmer by the Viking processor.

7.1.1 Bus errors

Bus errors are issued to the processor when the processor does a reference to virtual space or physical space which cannot be satisfied for hardware reasons (this excludes MMU protection violations, for example). This section describes how bus errors are handled by the processor and DVMA devices.

7.1.1.1 Types of bus errors

In Sun-4D systems, there are 4 codes available for bus errors:

TO: Time Out; this code is returned when the addressed location does not return any answer after some fixed amount of time.

BE: Bus Error; this code is returned when the addressed location rejects the required action because it is illegal.

UC: Uncorrectable Error; this code is returned when the addressed location rejects the required action because of an internal failure.

UD: Undefined Error[1]; this code is for certain hardware errors.

7.1.1.2 Bus errors on prefetch operations

If a bus error is notified on a prefetch operation, it is completely ignored by the processor.

A prefetch is a bus operation which is not explicitly required by the executing program. The cases for prefetch by the processor unit in Sun-4D systems are:

- prefetch by Viking into the internal instruction or data caches.
- prefetch by the CC into the external cache.

1. This is a misnomer, inherited from previous systems

- instruction fetches performed before it is known whether the instruction will be executed (an instruction which has been fetched may be discarded due to an intervening branch or trap).

7.1.1.3 Bus errors on instruction fetches

An instruction fetch accesses an instruction which will be executed. If a bus error is notified on an instruction fetch, an `instruction_access_exception` trap will be taken by the CPU when that instruction reaches the execution stage.

The Viking Fault Status register will contain the bus error code in the (UD, UC, TO, BE) bits, FT=5, AT=2/3, FAV=0. The virtual address of the error is the error PC, which is saved in register 11 by normal trap processing. The Viking Fault Address register is not updated.

7.1.1.4 Bus errors on data loads

A data load is issued by any load instruction when it is executed. If a bus error is notified during a data load, a `data_access_exception` trap will be taken by the CPU.

The Viking Fault Status register will contain the bus error code in the (UD, UC, TO, BE) bits, FT=5, AT=0-3, FAV=1. If the access was through an ASI other than the MMU-related ASIs (`0x08-0x0E`, `0x20-0x2F`), the CS bit in the Viking Fault Status register will be set. The virtual address of the error is saved in the Viking Fault Address register.

7.1.1.5 Bus errors on synchronous data stores

A synchronous data store is issued by the `ldstub`, `ldstuba`, `swap` and `swapa` instructions to any ASI, or `sta` instructions to ASIs others than `0x08-0x0B` and `0x20-0x2F` (for ASIs `0x08-0x0B` and `0x20-0x2F`, see section 7.1.1.6 on page 144). If a bus error is notified during a synchronous data store, a `data_access_exception` trap will be taken by the CPU.

The Viking Fault Status register will contain the bus error code in the (UD, UC, TO, BE) bits, FT=5, AT=4-7, FAV=1. If the access was through an ASI other than the MMU-related ASIs (`0x08-0x0B`, `0x20-0x2F`), the CS bit in the Viking Fault Status register will be set. The virtual address of the error is saved in the Viking Fault Address register.

7.1.1.6 Bus errors on asynchronous data stores

An asynchronous data store is issued by any store instruction except `ldstub`, `ldstuba`, `swap` and `swapa` to any ASI, and `sta` to an ASI other than `0x08-0x0B` and `0x20-0x2F` (stores to those ASIs use the MMU to access physical space and are handled like normal `st` instruction). The effect of bus errors on asynchronous data stores depends on whether this is an early or late error, and on the state of the Viking store buffer.

An early error is an error which is notified to Viking before the bus operation is effectively launched by the CC. There are two cases of early store errors:

- Memory store which misses in the external cache and for which the cache receives a bus error indication from memory when it tries to load the missing cache block.
- Viking bus parity error (See section 7.2.3.5 on page 150).

A late error is an error which is notified to Viking after the operation has been launched (and acknowledged to Viking by the CC). Late errors occur only on stores to I/O space.

If the Viking store buffer is disabled and an early error is notified, it is handled as for a synchronous data store.

If the Viking store buffer is enabled and an early error is notified, the CPU will take a `data_store_error` trap. The SF bit of the Viking Fault Status register will be set, but the bus error code is **not logged**. The Viking Fault Address register is not updated.

If a late error is notified (independently of the store buffer state), a level-15 interrupt will be issued to the CPU. The CC Error register (See section 4.4.10 on page 50) will contain the error information, with the AE bit set to one and the DCmd subfield of CCOP set to 010011_2.

Note that the value of the MC bit of the CC Control register (See section 4.4.7 on page 47) does not affect error handling of store operations.

7.1.1.7 Bus errors on block copy operations

Block copy operations are initiated by using the CC Stream registers (See section 4.4.3 on page 41). If a bus error is notified during a block copy operation, a level-15 interrupt will be issued to the CPU. The CC Error register (See section 4.4.10 on page 50) will contain the error information, with the AE bit set.

Note that block copy errors may be differentiated from late asynchronous store errors by the value logged in subfield DCmd of field CCOP in the CC Error register, which is 000101_2 or 010101_2 for Block_Load errors, 001111_2 or 010111_2 for Block_Store errors, and 010011_2 for asynchronous store errors.

7.1.1.8 Bus errors on MMU TLB operations

An MMU TLB operation is a memory reference issued by the Viking MMU to load entries into its TLB or to set the Modified or Referenced bits of a PTE. If a bus error is notified during an MMU TLB operation, the CPU will take either an `instruction_access_exception` trap or a `data_access_exception` trap depending on whether the operation was initiated using ASIs `0x08-0x09` (including instruction fetch) or any other ASI (including normal load and store operations).

If an `instruction_access_exception` trap is taken, the Viking Fault Status register will contain the bus error code in the (UD, UC, TO, BE) bits, FT=4,

FAV=0. The virtual address of the error is the error PC, which is saved in register 11 by normal trap processing. The MMU FAR is not updated.

If a `data_access_exception` trap is taken, the Viking Fault Status register will contain the bus error code in the (UD, UC, TO, BE) bits, FT=4, FAV=1. If the access was through an ASI other than the translation ASIs (0x8-0xB, 0x20-0x2F), the CS bit in the Viking Fault Status register will be set. The virtual address of the error is saved in the MMU FAR.

7.1.1.9 Bus errors on SBus DVMA read operations

When a bus error is signaled in response to a DVMA read operation (i.e. transfer from main memory to the I/O device), the SBus device receives an SBus ErrAck indication (See [13] for details on SBus). The behavior is the same in consistent or stream mode.

The device should then terminate the DVMA operation and indicate an error condition to the driver. This behavior is device-dependent and is not specified by the Sun-4D architecture.

7.1.1.10 Bus errors on SBus DVMA write operations

When a bus error is signaled in response to a DVMA write operation (i.e. transfer from the I/O device to main memory), the SBus device does not receive an SBus error indication as the write operation is acknowledged before being performed. Instead, the WEC or WES bit in the Slot Configuration register (See section 6.4.2.4 on page 136) for the corresponding SBus device will be set. WEC is used if the error occurred during a consistent mode access, WES is used if the error occurred during a stream mode access. It is the driver's responsibility to check the status of this bit after a DVMA transfer (typically when the buffer is demapped from the I/O page tables).

7.1.2 Interrupts

Interrupts are issued to the processor to notify error conditions which are asynchronous with its normal operations.

7.1.2.1 Local level-15 interrupt

A local level-15 interrupt is issued to the processor when an asynchronous error is detected by CC. This may correspond to an error which was initiated by an action from the CPU (for example, a late error in an asynchronous data store, see section 7.1.1.6 on page 144), or to an error which was initiated by an action from another processor or I/O device, but which was detected locally by CC (for example, a cache parity error on data owned by this external cache, see section 7.2.3.4 on page 149).

When a local level-15 interrupt is issued, the error information is logged in the CC Error register.

7.1.2.2 Broadcast level-15 interrupt

A broadcast level-15 interrupt can be issued by any device to signal of an error condition which needs to be logged. Error information is stored in the corresponding's device error register.

7.1.3 Resets

See Chapter 9 for details on reset management.

7.1.3.1 CPU watchdog reset

A CPU watchdog reset is initiated when a trap condition occurs while traps are disabled and the MMU Control register NF bit is not set. The CPU branches to the instruction at physical address 0xF F000 0000. The WD bit in the CC Reset register is set to 1 on a watchdog reset.

A watchdog reset affects only one CPU and has no other effect on the system.

7.1.3.2 System watchdog reset

When a fatal error is detected, a system watchdog reset is initiated. A system watchdog reset affects all CPUs and I/O devices. Writes in progress may be lost, but the state of main memory is not altered (main memory continues to be refreshed after a system watchdog reset).

7.2 Types of errors

7.2.1 Software errors

Errors which do not originate in a hardware malfunction are classified as software errors. All such errors are detected by the Viking CPU and are reported only to that CPU. See the Viking specification [7] for details regarding error handling in Viking.

7.2.2 Hardware-corrected errors

Hardware-corrected errors are always signaled by a broadcast level-15 interrupt for error logging purposes. No recovery action is normally necessary.

7.2.2.1 Memory correctable ECC error

When the memory subsystem detects a single-bit error on a read, the MQH rewrites the corrected data into main memory and delivers the corrected word to the requestor. The MQH also issues a level-15 broadcast interrupt with INTSID=0x02 if it is the **first occurrence** of a single-bit error, i.e. if the error is logged in the Correctable Error Address and Data registers (See section 5.5.1 on page 102). This means that an interrupt is issued if the SBE bit is set to one because of the error and if the ECI bit of the MQH Control and Status register is set (see section 5.3.6 on page 96). When MQH detects the first occurrence of a single-bit error it keeps the address, data, ECC and syndrome for the double-word which was corrected.

If multiple single-bit errors occur during the same sub-block access only one interrupt is issued because only the first error is logged.

If one or multiple single-bit error(s) and one or multiple uncorrectable ECC error(s) occur in the same 64-byte sub-block, the INTSID is instead set to 0x04.

A correctable memory ECC error can be generated for diagnostics purposes by forcing incorrect generation through the MQH ECC Diagnostic and Control register (See section 5.5.3.5 on page 107).

7.2.3 Recoverable errors

Recoverable errors which have a hardware cause are usually signaled by a bus error indication to the requesting device and a level-15 interrupt (which may be broadcast). Error recovery is normally handled by the trap routines, while error logging is done by the level-15 interrupt handler.

Recoverable errors which do not have a hardware cause are signaled only by a bus error indication.

7.2.3.1 Dynabus time-out

A recoverable Dynabus time-out occurs in the following cases:

1. Read or Write access to a non-existent I/O space location from a processor (DVMA devices cannot access I/O space)
2. Read access to a non-existent memory space location from a processor or a DVMA device
3. Write access to a non-existent memory location from a processor or a DVMA device *if the external cache (respectively the IOC) is enabled* (in which case it is actually the read miss which will cause the bus error)
4. DeMap time-out (Processor STA to ASI 0x03, See section 7.2.3.6 on page 150)

The effect of a recoverable Dynabus time-out is to return a bus error indication **TO** to the requesting processor or DVMA device. This error may be forced by referencing an address known to be inexistent.

NOTE: Dynabus time-outs are handled by BW in the processor unit and by SBI and IOC in the I/O unit. SBI handles time-outs which correspond to operations which do not affect the IOC cache, whereas IOC is responsible for time-outs involving the IOC cache. This is not visible to programmers except when time-out values are specified during system initialization.

Note that accessing an inexistent SBus device will result in a Dynabus time-out error if the addressed system board is not present, and in an SBus time-out error if the addressed system board exists, but the SBus slot is not populated. The only programmer-visible difference is the time it takes to return the time-out.

7.2.3.2 Dynabus rejection

If an access to a control register cannot be performed, a bus error indication **BE** or **TO** will be issued to the requesting processor. Typical error cases may be accessing with the wrong size specification, or issuing a swap operation on a register for

which it is not supported. This error may be forced by performing an illegal operation.

7.2.3.3 Memory uncorrectable ECC error

When the memory subsystem detects an uncorrectable ECC error, the MQH notifies a bus error indication UC to the requesting CPU or DMA device. The MQH also issues a level-15 broadcast interrupt with INSTID=0x03 if the error is logged in the Uncorrectable Error Address and Data registers (See section 5.5.1 on page 102). This means that an interrupt is issued if the error causes the UE bit to be set in the Uncorrectable Error Address register. In this case, MQH keeps the address, data, ECC and syndrome for the double-word in error.

If multiple uncorrectable errors are detected in the same sub-block access only a single interrupt is issued because only the first error is logged.

If one or multiple correctable error(s) and one or multiple uncorrectable ECC error(s) occur in the same 64-byte block, the interrupting INTSID is instead set to 0x04.

A uncorrectable memory ECC error can be generated for diagnostics purposes by forcing incorrect generation through the MQH ECC Diagnostic and Control register (See section 5.5.3.5 on page 107).

7.2.3.4 External cache parity error

A parity error in the external cache may be discovered in three cases, which are handled differently.

1. If a processor or DVMA device reads data from another processor's external cache because it is the current owner of this data and a parity error is detected, the reader will receive a bus error indication UC and the Viking attached to the owner cache will receive a local level-15 interrupt. The error information is logged in the Error register of the owner CC, with CP=1 (See section 4.4.10 on page 50).
2. If CC detects a parity error when trying to write back a line to main memory due to the replacement algorithm, it will write incorrect data back to memory and raise a local level-15 interrupt to its processor. The error information is logged in the CC Error register, with CP=1 (See section 4.4.10 on page 50).
3. If a processor detects an external cache parity error during a read from its own external cache, the error condition will be handled as if a bus error indication UC had been received by the processor for the read operation. In addition, the Viking Fault Status register P bit will be set.

The first 2 cases may be distinguished by the value logged in subfield DCmd of field CCOP in the CC Error register: DCmd is 000110_2 in case 2 only (See section 4.4.10 on page 50).

A cache parity error can be generated by forcing incorrect parity generation through the CC Control register (See section 4.4.7 on page 47) or the Viking Control register.

7.2.3.5 Viking write parity error

If a parity error is detected by CC during a write from Viking, an early bus error indication **UD** will be issued to the processor (See section 7.1.1.6 on page 144 for an explanation on how Viking handles early errors on writes). In addition, the VP bit in the CC Error register will be set (See section 4.4.10 on page 50).

A Viking write parity error can be generated by forcing Viking to issue incorrect parity on writes using the Viking Control register.

7.2.3.6 DeMap time-out

A DeMap operation issued by a processor (broadcast TLB flush, performed by a store alternate to ASI 0x03) may receive a time-out bus error, resulting in a trap (See section 7.1.1.5 on page 144). This error will occur if the store buffer of another (or "target") processor in the system becomes disabled after the DeMap operation is initiated and before the target processor has finished performing it, or certain hardware failure modes of a target processor.

Since this error reflects a temporary condition and does not necessarily indicate an hardware error, software should retry the DeMap operation on a DeMap time-out a few times before considering an unrecoverable error occurred.

7.2.3.7 SBus time-outs

If SBI detects a time-out on an SBus master access, it will issue a bus error indication **TO** to the requesting processor. An SBus time-out error can be forced by accessing an SBus slot known to be unpopulated on an existing system board (this may not be possible in all configurations).

7.2.3.8 SBus PTE Errors

If a DVMA operation accesses an invalid PTE, SBI returns ErrAck to the device and logs the error in the IPTE bit of the Slot Configuration register for that device (See section 6.4.2.4 on page 136). The device *must* terminate the DVMA operation, log the error internally and issue an interrupt (the details are device-dependent).

If a DVMA write operation accesses a valid PTE which does not indicate write permission, SBI returns ErrAck to the device and logs the error in the WPE bit of the Slot Configuration register for that device (See section 6.4.2.4 on page 136). The device *must* terminate the DVMA operation, log the error internally and issue an interrupt (the details are device-dependent).

This error can be forced by writing adequate values in a PTE.

7.2.3.9 SBus rejection

If the SBI detects an ErrAck on an SBus master access, it will issue a bus error indication **BE** to the requesting processor. There is currently no way to force this error (except with adequate knowledge of the behavior of a specific SBus card).

If the SBI detects a LateError indication on an SBus master access, it will issue a bus error indication **UC** to the requesting processor. This error may be forced through the SBI Control register (See section 6.4.2.2 on page 134).

7.2.3.10 SBus parity errors

There are 5 cases of SBus parity errors:

1. Master read: if parity is enabled for the slot and SBI detects an error, SBI issues a bus error indication UC to the requesting processor. No information is logged in SBI.
2. Master write: if the SBus device checks parity and detects an error, the device must issue a LateError indication to SBI, which will be transformed by SBI into a bus error indication with code UC to the requesting processor. The device may also log the error internally and issue an interrupt (this is device-dependent). No information is logged in SBI. This error may be forced through the FPE bit of the SBI Control register (See section 6.4.2.2 on page 134).
3. DVMA read: if the SBus device checks parity and detects an error, the device *must* terminate the DVMA operation, log the error internally and issue an interrupt (the details are device-dependent). SBI does not log any information. This error may be forced though the FPE bit of the SBI Control register (See section 6.4.2.2 on page 134).
4. DVMA write: if parity is enabled for the slot and SBI detects an error, SBI will issue a LateError signal to the device and log the error in the SDPE bit of the Slot Configuration register (See section 6.4.2.4 on page 136). The device *must* terminate the DVMA operation, log the error internally and issue an interrupt (the details are device-dependent). There is currently no way to force this error.
5. DVMA address: if parity is enabled for the slot and SBI detects an error during the address phase, SBI will issue ErrAck to the device and log the error in the SAPE bit of the Slot Configuration register (See section 6.4.2.4 on page 136). The device *must* terminate the DVMA operation, log the error internally and issue an interrupt (the details are device-dependent). There is currently no way to force this error.

7.2.3.11 SBus external page table parity error

If an external page table entry used during a DVMA operation has incorrect parity, SBI will return ErrAck to the device, logs the error in the PPE bit of the Status register (See section 6.4.2.3 on page 135) and issues a broadcast level-15 interrupt with INTSID=0x05. This error can be forced through the FXP bit of the SBI Control register (See section 6.4.2.2 on page 134).

7.2.4 Fatal errors

All fatal errors initiate a system watchdog reset if no Service Processor is attached to the system (See section 9.1.1.4 on page 170). When a Service Processor is attached to the system fatal errors cause the Dynabus traffic to be suspended. Fatal errors correspond to hardware errors in which proper system operation cannot be guaranteed.

NOTE: In current Sun-4D systems (SunDragon and Scorpion) there is no plan to support a Service Processor and all fatal errors are handled through a system watchdog reset.

The reason for a fatal error is logged in the Dynabus CSR of the device which reported the error (BW, MQH or IOC). See section 4.5.3 on page 59 for the BW, section 5.4.2 on page 98 for the MQH, and section 6.3.3.2 on page 124 for the IOC. The reason may be an error detected by an associated XBus device (CC for BW, SBI for IOC), in which case the corresponding error register must also be checked.

Fatal errors reporting can be disabled on a per-chip basis through the EER bit of the Dynabus Control and Status registers of BW, MQH and IOC. This feature is intended for system bringup and diagnostics and should not be used during normal operation.

7.2.4.1 Dynabus parity error

Dynabus parity errors may be detected by BW, MQH and IOC. The DPE bit of the Dynabus CSR for the detecting device is set, and the data and parity bits are recorded. As a side effect, the parity logging in BICs is stopped, which allows error analysis software to locate the source of the parity error and the byte on which it occurred.

This error can be forced using JTAG boundary scanning.

7.2.4.2 Dynabus arbitration parity error

Dynabus arbitration parity errors may be detected by BW, MQH, IOC, BARB and CARB.

If the error is detected by BW, MQH or IOC, the GPE bit of the Dynabus CSR for the detecting device is set, and the arbitration signals are recorded.

If the error is detected by BARB or CARB, a bit corresponding to the arbitration port in error will be set in the corresponding error register (accessible only through JTAG, see section 10.3.8 on page 186 for CARB and section 10.3.9 on page 187 for BARB).

This error can be forced using JTAG boundary scanning.

7.2.4.3 Dynabus arbitration time-out

Dynabus arbitration time-outs may be detected by BW, MQH and IOC. The GTO bit of the Dynabus CSR for the detecting device is set. The time-out duration is parameterized through the GTOL field of the corresponding Dynabus CSR.

This error can be forced by using the TTO bit in the BW Dynabus CSR (See section 4.5.3 on page 59).

7.2.4.4 XBus parity error

XBus parity errors may be detected by BW, CC, IOC and SBI.

If the error is detected by BW or IOC, the CDE field of the corresponding Dynabus CSR will be set to a chip-specific value. No additional data is logged.

If the error is detected by CC, the CDE field of the Dynabus CSR in the corresponding BW will be set to a device-dependent value to indicate a CC error and the Error

register of CC will indicate an XBus parity error (See section 4.4.10 on page 50 and section 4.5.3 on page 59).

If the error is detected by SBI, the CDE field of the Dynabus CSR in the corresponding IOC will be set to a device-dependent value to indicate an SBI error and the error register of SBI will indicate an XBus parity error (See section 6.3.3.2 on page 124 and section 6.4.2.3 on page 135)

Those errors may be forced by forcing XBus signals through JTAG boundary scanning.

7.2.4.5 Cache consistency errors

Cache consistency errors may be detected by BW, CC or IOC.

If the error is detected by BW or IOC, the IE bit of the CDE field of the Dynabus CSR is set to one and detailed error information is logged in the BW DynaData register (See section 4.5.4.2 on page 64 for BW and section 6.3.3.3 on page 128 for IOC).

If the error is detected by CC, the IE bit of the CDE field of the Dynabus CSR in the corresponding BW is set to one and the CCE bit of the BW's DynaData register is set to one to indicate a CC error. The CC Error register will indicate a cache consistency error (See section 4.4.10 on page 50, section 4.5.3 on page 59 and section 4.5.4.2 on page 64).

Those errors may be forced by modifying directly the tags in CC, BW or IOC in inconsistent ways.

7.2.4.6 WriteSingle time-out

This error is generated when no reply is received after a Dynabus memory write operation. If the external cache is enabled, the only case where this can happen is if an MQH accepted a BlockRead (cache line fill) for an address and later ignored a write to that address. Note that, if the external cache is disabled, *any* write to memory space will issue this error.

The error is logged in the BW, IE is set in the CDE field of the Dynabus CSR and the WSTO bit set in the DynaData register. The time-out value is controlled by the RDTOL field of the BW Dynabus CSR (See section 4.5.3 on page 59).

The same error can occur in the IOC for consistent mode stores to shared data. In that case, the error is logged in the IOC as bit STO of the CDE field of the Dynabus CSR. The time-out value is controlled by the RDTOL field of the IOC Dynabus CSR (See section 6.3.3.2 on page 124)

7.2.4.7 Multiple DeMaps

This error is generated when two or more DeMap operations are active at the same time. It is detected by the BWs (See section 4.5.4 on page 63). The IE bit of the CDE field of the Dynabus CSR is set to one and the MDMP bit is set in the DynaData register (See section 4.5.4.2 on page 64).

7.2.4.8 Device-specific errors

Those errors can be issued by any device in the system. They correspond to a failure of internal consistency checks.

Refer to the individual unit description for a list of device-specific errors.

7.2.5 Critical errors

Critical errors require immediate system shutdown and power-off. They are notified through level-15 broadcast interrupts. Note that, since those interrupts are issued via the BootBus, they do not provide an INTSID and are dispatched only to the processor which currently holds the BootBus semaphore 0, or to both processors if none of them holds the semaphore (See section 4.6.9.1 on page 81).

7.2.5.1 AC Failure

If an AC power failure is detected (loss of line power), a level-15 interrupt is issued to all processors and the ACInt bit of the BootBus Status_2 register (See section 4.6.8.3 on page 78) is set in all CPUs. This interrupt cannot be masked individually. When the interrupt is issued, the maximum guaranteed system operation time is 5 ms.

7.2.5.2 Temperature Warning

If the temperature raises above normal operational level on a given system board, a level-15 interrupt is issued to the processors on that board and the TmpInt bit of the BootBus Status_2 register (See section 4.6.8.3 on page 78) is set in the corresponding board. This interrupt can be masked for by using the EnTmp bit of the BootBus Control register (See section 4.6.8.1 on page 77). The system should be shutdown in a timely manner when a temperature warning is notified.

7.2.5.3 Fan Failure

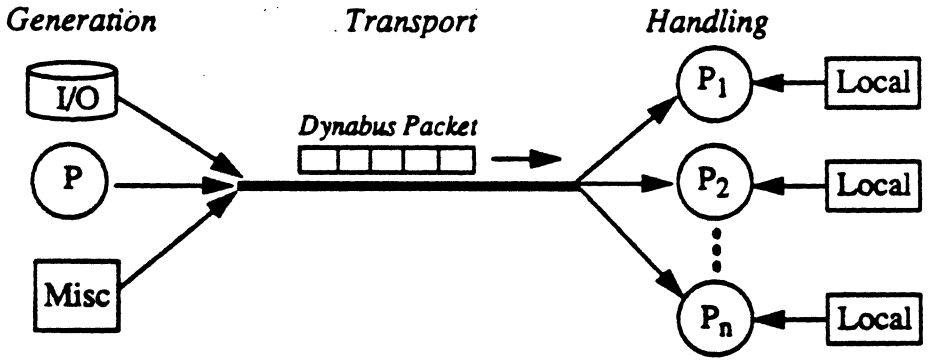
If a fan failure is detected, a level-15 interrupt is issued to all processors and the FanInt bit of the BootBus Status_2 register (See section 4.6.8.3 on page 78) is set in all CPUs. This interrupt can be masked for each BootBus by using the EnFan bit of the BootBus Control register (See section 4.6.8.1 on page 77). The system should be shutdown in a timely manner when a fan failure is notified.

Chapter 8
Interrupts

This chapter describes the interrupt model for the Sun-4D architecture. It begins with an overview of the model, followed by a detailed description of each of three phases an interrupt goes through during its lifetime: *generation*, *transport*, and *handling*. The section on generation describes the various sources for interrupts, how they produce interrupts, and what type of interrupts are generated by each source; the section on transport describes how interrupts are transported from a source to a processor; and the section on handling focuses on what an interrupted processor must do to handle an interrupt properly. The chapter closes with a map of interrupt usage for current Sun-4D systems.

8.1 Overview of Interrupt Model

The figure below shows the basic interrupt model. Interrupts are generated by four types of sources: *I/O*, *Processors*, *Miscellaneous*, and *Local*. I/O interrupts are generated by devices connected to a standard I/O bus like SBus (this is the only I/O bus in the current Sun-4D systems). Processor interrupts are generated directly by Vikings. The Miscellaneous category includes all other sources that use the Dynabus to transport interrupts.



When one of these non-Local sources generates an interrupt, the interrupt is transported over the Dynabus to the cache of one or more target processors. At the target, the cache records information about the source, sets appropriate bits in the Interrupt Pending register, and interrupts its processor at the highest unmasked level. An interrupt may be either *directed*, in which case the specific processor named by the interrupt is targeted, or *broadcast*, in which case all processors in the system are targeted. The source of an interrupt determines whether it is directed or broadcast.

Local sources do not use the Dynabus transport mechanism at all, but cause bits to be set directly in the local cache's Interrupt Pending register.

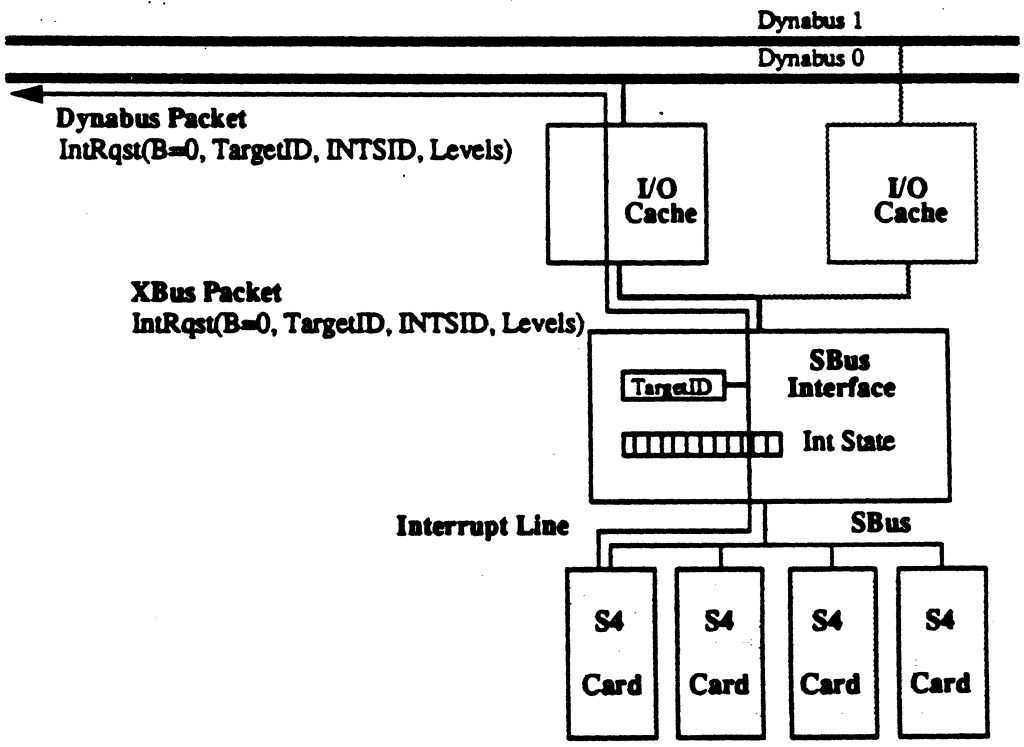
When a processor is interrupted, it examines state kept in its cache to determine the source of the interrupt and then invokes the appropriate handler. The overall handling of interrupts is structured into three levels: a *Top-Level Handler* that is independent of the interrupt source, various source specific *Low-Level Handlers*, and *Device Drivers*. This structuring facilitates understanding and also makes it easier to incorporate new interrupt sources and drivers in the future.

8.2 Interrupt Generation

Interrupt generation refers to the first phase of an interrupt. For non-Local interrupts, generation starts when a device raises its interrupt line and ends when an interrupt packet has been placed on the Dynabus. For Local interrupts, generation consists of actions taken prior to setting bits in the Interrupt Pending register. The generation mechanism is different for each type of interrupt source, so each will be taken up separately.

8.2.1 I/O Interrupts

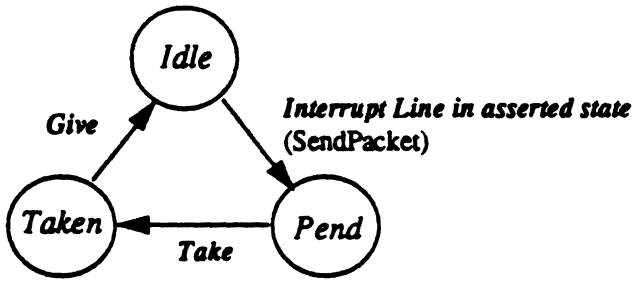
In the current Sun-4D implementations (SunDragon and Scorpion), all I/O interrupts originate at SBus devices. Future implementations may have other buses that serve as a source of I/O interrupts. Devices on a given SBus communicate interrupts to the SBI for that SBus via 28 interrupt lines. The SBI converts interrupt levels on these lines into XBus interrupt packets destined for Dynabus 0 in a multiple Dynabus system. In a single Dynabus system, by default the Dynabus is logically Dynabus 0. When the IOC connected to Dynabus 0 receives an interrupt packet over the XBus, it forwards the packet on to its Dynabus. The figure below shows this sequence of events.



When the SBI generates an XBus interrupt packet for an SBus interrupt, it sets the broadcast bit (B) to 0, the TargetID to the contents of the Interrupt TargetID register (See section 6.4.2.7 on page 140), INTSID[7:5] to the SBus interrupt level number, INTSID[4:0] to the contents of the ISI field in the SBI Control register (See section 6.4.2.2 on page 134), and the Levels field to the SPARC interrupt level corresponding to the SBus level (See table on page 140 for the mapping between SBus levels and SPARC levels). The ISI field should be set up at system initialization to {0, DeviceID[7:4]} by POST. POST assigns the 4 most significant bits of I/O units DeviceID to **bbbb** which is the number of the Dynabus slot into which the system board containing the I/O unit is plugged.

The Interrupt TargetID register, on the other hand, is intended to be modified at run time so that the processor receiving interrupts may be changed dynamically to balance the interrupt load on processors. This register can be read and written by the software individually for each SBI, but it may also be updated in broadcast mode for any selected subset of SBI's (See section 4.4.5.4 on page 45 and section 6.4.2.7 on page 140). Broadcast update capability is provided to keep the run time overhead of changing the target processor low.

For each interrupt line, the SBI keeps internal state that allows it to convert a level on that line into a packet, and to ensure that exactly one processor attempts to service the interrupt (two processors may attempt to service the same interrupt if the Interrupt TargetID register changes at an inopportune time: consider the case where two SBus cards on the same SBus send interrupts at the same level to two different processors). This state is accessed via the Interrupt State register in SBI, with different bit positions of this register corresponding to the 28 different interrupt lines (See section 6.4.2.6 on page 139 for a description). The state and possible state transitions for a given interrupt are shown in the diagram below.



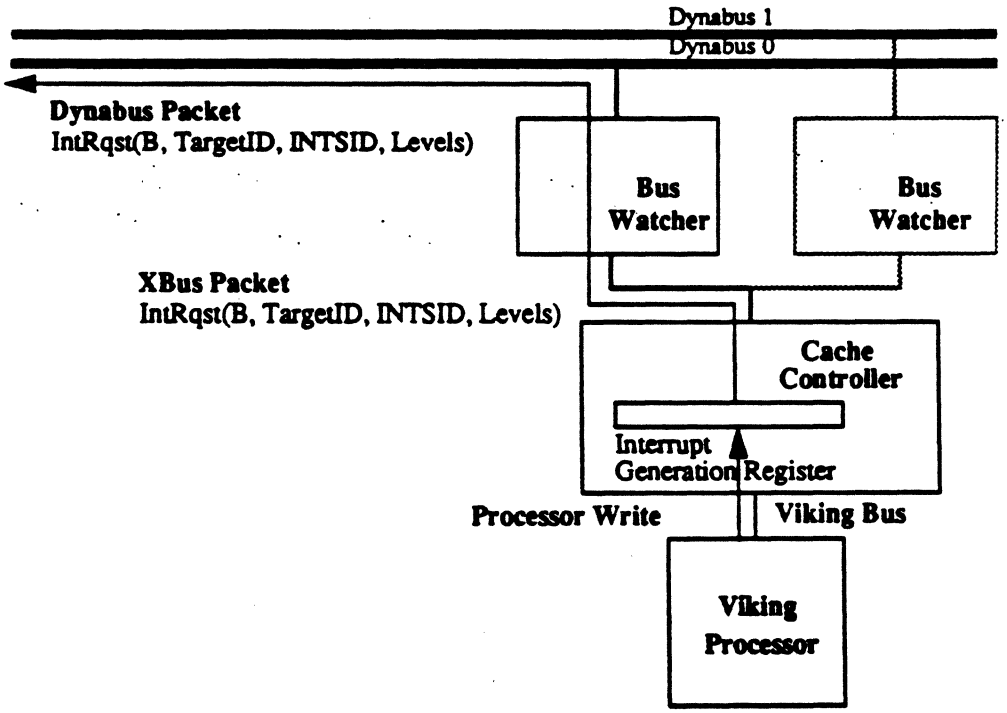
The state *Idle* means that no interrupt is pending and there is no interrupt handler currently servicing an interrupt from that line. The state *Pend* means that an interrupt is pending, but no handler has taken responsibility for servicing it. This state is entered whenever the current state is *Idle* and the interrupt line is in the asserted state; note that the interrupt line is level sensitive, *not* edge sensitive, since it is sampled continuously while the state machine is in *Idle*. The SBI sends an interrupt packet whenever there is a transition from *Idle* to *Pend*. The state *Taken* means that a handler has requested and obtained responsibility for handling this interrupt. This state is entered when the current state is *Pend* and a handler *Take*s an interrupt by doing a Swap to the Interrupt State register in SBI with a value of 1 for the bit that

corresponds to the line in question. A Swap is used because multiple handlers may be trying to ***Take*** the interrupt. The value returned by the Swap for the bit corresponding to this line tells the handler whether it succeeded (returns a 1) or not (returns a 0). Finally, a handler returns the state to *Idle* via a ***Give*** when it has completed servicing the interrupt. A ***Give*** is executed by issuing a Store to the Interrupt State register with a value of 1 for the bit that corresponds to the line.

Note that it is legal for a handler to ***Take*** multiple interrupts with a single Swap by having the appropriate bits set in the data value, and ***Give*** multiple interrupts via a single Store the same way. For correct operation, a handler may ***Give*** only those interrupts that it has ***Taken*** earlier. Note also that a Read to the Interrupt State register allows a handler to check which lines are in the state *Pend* without causing any state transitions in the process.

8.2.2 Processor Interrupts

Processor generated interrupts provide the mechanism for one processor to interrupt another. A processor generates an interrupt by writing into its Interrupt Generation register, which is located in the CC chip. On receiving a write to this register, the CC generates an XBus interrupt packet destined for Dynabus 0. The various fields in the interrupt packet are set directly from the data value of the write. The figure below shows this sequence of events.



The cache controller sets the broadcast bit (B), the TargetID, INTSID, and Levels in the XBus interrupt packet directly from the corresponding fields of the 32 bit value written to the Interrupt Generation Register. This means that a processor can "fake" any non-Local interrupt in the system.

When the broadcast bit (B) is zero, the interrupt is directed to the processor specified by TargetID. That is, the processor whose Bus Watcher has the Dynabus DeviceID equal to TargetID will be interrupted. When the broadcast bit (B) is one, TargetID is ignored and all processors will be interrupted.

The INTSID field allows the target(s) of the interrupt to determine who generated the interrupt. Note that since processors can generate interrupts with INTSID set to any of the 256 possible values, it is possible to confuse the software that is trying to determine the interrupt source just by looking at the INTSID. It is recommended that except for diagnostics, processors use INTSID in a way that allows the source to be determined without excessive polling.

The levels field contains one bit for each SPARC interrupt level. A 1 in the position for level *i* specifies that the target processor will be interrupted at level *i*. A single interrupt packet may specify that a target be interrupted at multiple levels.

8.2.3 Miscellaneous Interrupts

Miscellaneous sources are all sources other than I/O and Processors that use the Dynabus interrupt transport mechanism.

8.2.3.1 Counter-Timers

Each Bus Watcher chip has two counter-timers capable of generating interrupts to the local processor. One of these is the *profiling timer*, and the other the *tick timer*.

The *profiling* counter-timer issues interrupts at SPARC level 14 when it is configured as a kernel timer (UTE=0 in the BW Control register) and the limit value is non zero. Whenever the counter value is equal to the limit value, the Bus Watcher sends an interrupt packet out on its Dynabus. The various fields of the interrupt packet are set as follows. The broadcast bit is set to 0; the TargetID is set to the Dynabus DeviceID of the issuing Bus Watcher; the INTSID is set to `0x00`; and the Levels field is set to a 1 for the bit corresponding to level 14 and zeros for all other levels. Note that setting the TargetID equal to the Dynabus DeviceID of the issuing Bus Watcher causes the interrupt packet to "loop back" to the issuing processor and interrupt it.

Although the profiling timers in any of the Bus Watchers of a multiple Dynabus Processor unit may be used, it is recommended that only those in Bus Watcher 0 (the one connected to Dynabus 0) be used. This makes the interrupt handling code portable across all current Sun-4D systems configurations because *each* configuration is required to have a Dynabus 0.

The *tick* counter-timer issues interrupts at SPARC level 10 when the limit value is non zero. Whenever the counter value is equal to the limit value, the Bus Watcher sends an interrupt packet out on its Dynabus. The various fields of the interrupt packet are set as follows. The broadcast bit is set to 0; the TargetID is set to the DeviceID of the issuing Bus Watcher; the INTSID is set to `0x01`; and the Levels field is set to a 1 for the bit corresponding to level 10 and zeros for all other levels. Note that setting the TargetID equal to the Dynabus DeviceID of the issuing Bus Watcher causes the interrupt packet to "loop back" to the issuing processor and interrupt it.

The tick counter-timers in the system must be configured so that there is exactly one tick timer enabled in the whole system. If other processors need to be notified, this is done by having the target processor redispatch the interrupt explicitly. It is recommended for reasons of code portability that this timer be in one of the Bus Watchers connected to Dynabus 0.

8.2.3.2 Memory Queue Handler Chips

A Memory Queue Handler generates a level 15 broadcast interrupt when it records the first occurrence of a correctable or uncorrectable memory error in respectively the Correctable Error Address and Data registers or the Uncorrectable Error Address and Data registers. If an error is correctable, an interrupt packet is issued only if the ECI bit within the MQH's Control and Status register is set to one (See section 5.3.6 on page 96). An interrupt packet is always issued when an uncorrectable error is logged.

The MQH sets the various fields of an interrupt packet as follows: the broadcast (B) bit is set to 1; the TargetID is set to an unspecified value, since it is ignored when the broadcast bit is set; INTSID is set to `0x02` for correctable errors, `0x03` for uncorrectable errors, `0x04` when one or multiple correctable errors and one or multiple uncorrectable errors occur within the same block; and Levels is set to a 1 for the bit corresponding to level 15 and zeros for all other bits.

An MQH issues an interrupt packet on the Dynabus to which it is connected. This means that MQH interrupts, unlike I/O and processor generated interrupts, may arrive over any of the Dynabusses to which a processor is connected in a multiple Dynabus system. An interrupt handler can determine which bus the interrupt arrived on by examining state kept in the Interrupt Table of the appropriate Bus Watcher. See section 4.5.6 on page 67 for a detailed explanation of Interrupt Tables.

8.2.3.3 SBus Interface Chips

An SBus Interface Chip generates a broadcast level 15 interrupt when it detects a parity error on accessing its External Page Table. After setting the PPE bit in the SBI Status register (See section 6.4.2.3 on page 135), the SBI sends an XBus interrupt packet directed to Dynabus 0.

The SBI sets the various fields of the interrupt packet as follows: the broadcast bit (B) is set to 1; the TargetID is set to an unspecified value, since it is ignored when the broadcast bit is set; INTSID is set to `0x05`; and Levels is set to a 1 for the bit corresponding to level 15 and zeros for all other bits.

8.2.4 Local Interrupts

Local sources are those that set bits directly in the local Cache Controller's Interrupt Pending register. These sources do not use the Dynabus to transport interrupts, so no state is kept in Bus Watcher Interrupt Tables for interrupts from these sources.

8.2.4.1 Cache Controller

When a Cache Controller encounters an asynchronous error (See section 4.4.10 on page 50) it logs information about the error in its Error register and generates a level

15 interrupt for its processor by setting the appropriate bit in the Interrupt Pending register.

NOTE: The Cache Controller will force a Level 15 interrupt as long as its Error register indicates an error is pending (See section 4.4.10 on page 50). Software must clear the Error register before clearing bit LVL15 in the Cache Controller Interrupt Pending register.

8.2.4.2 Serial Ports

Each BootBus contains four serial ports. Two of the ports are dedicated to the Keyboard/Mouse interface (See section 4.6.4 on page 74) while the other two are intended for the system console (See section 4.6.3 on page 74).

Serial port interrupts set bits directly in the Interrupt Pending register of the local Cache Controller. All four ports issue interrupts at SPARC level 12. When a handler is invoked for this level, it must examine the I/O registers for each of the four local serial ports to determine which ones need service.

Note that, like all other BootBus interrupts, the serial port interrupts are level-sensitive: LVL12 will be forced in the CC Interrupt Pending register as long as at least one of the serial port asserts its interrupt line.

The serial port interrupts are directed to the processor which holds BootBus Semaphore 0, or to both processors if no processor holds Semaphore 0 (See section 4.6.9.1 on page 81).

8.2.4.3 Environment Interrupts

System logic monitors continually environmental conditions and issues a level 15 interrupt when an environmental failure occurs. Environment interrupts are dispatched to processors via the attached BootBus. The environment interrupts are directed to the processor which holds BootBus Semaphore 0, or to both processors if no processor holds Semaphore 0 (See section 4.6.9.1 on page 81).

Note that, like all other BootBus interrupts, the environment interrupts are level-sensitive: LVL15 will be forced in the CC Interrupt Pending register as long as the corresponding interrupt source is active.

The status for each environment condition is provided by a bit in the BootBus Status_2 register (See section 4.6.8.3 on page 78). Certain of the interrupts may in addition be masked, on a per-BootBus basis, using a bit in the BootBus Control register (See section 4.6.8.1 on page 77).

There are three environment interrupts:

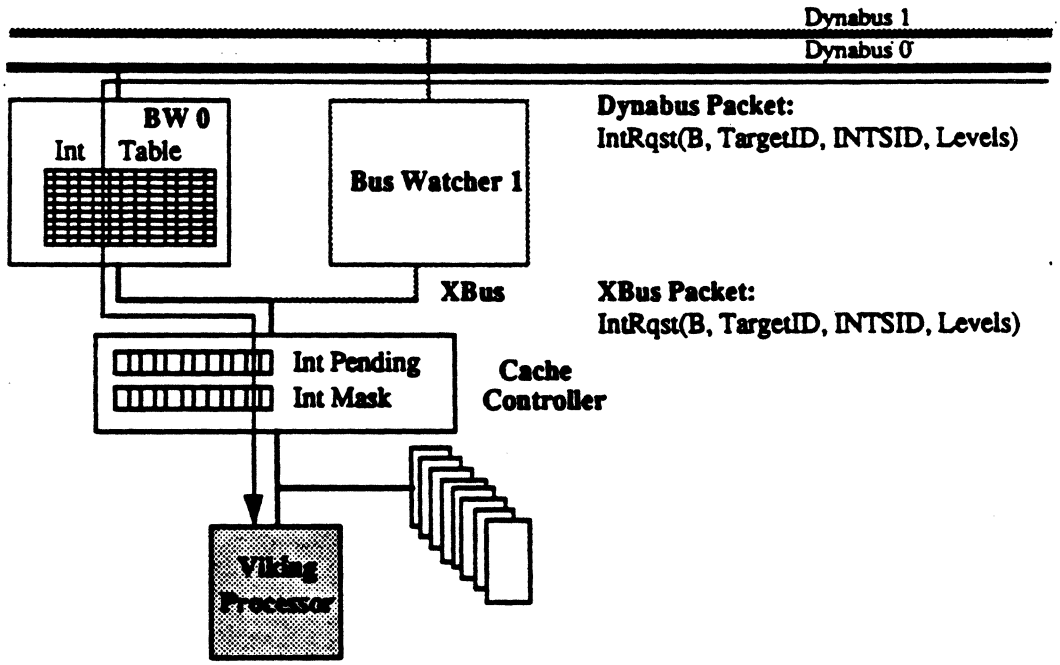
- Fan failure: the condition is asserted when the system fan stops drawing power. The interrupt is enabled by the EnFan bit of the BootBus Control register. The status is available in the FanInt bit of the BootBus Status_2 register. A fan failure is reported to all BootBusses simultaneously.

- Temperature warning: the condition is asserted when the ambient temperature exceeds normal operating conditions. The interrupt is enabled by the EnTmp bit of the BootBus Control register. The status is available in the TmpInt bit of the BootBus Status_2 register. Temperature failure is detected on a per-board basis. Note that a temperature warning may occur as a consequence of a fan failure.
- Line power failure: the condition is asserted when the AC supply drops below normal operating conditions. The interrupt is always enabled. The status is available in the ACInt bit of the BootBus Status_2 register. A line power failure is reported to all BootBusses simultaneously.

8.3 Interrupt Transport

Interrupt transport refers to the phase that begins when an interrupt packet is placed on the Dynabus and ends when the appropriate bits have been set in the Interrupt Pending register in the target processors' Cache Controllers. The transport phase for Local interrupts is not program visible so it will not be described.

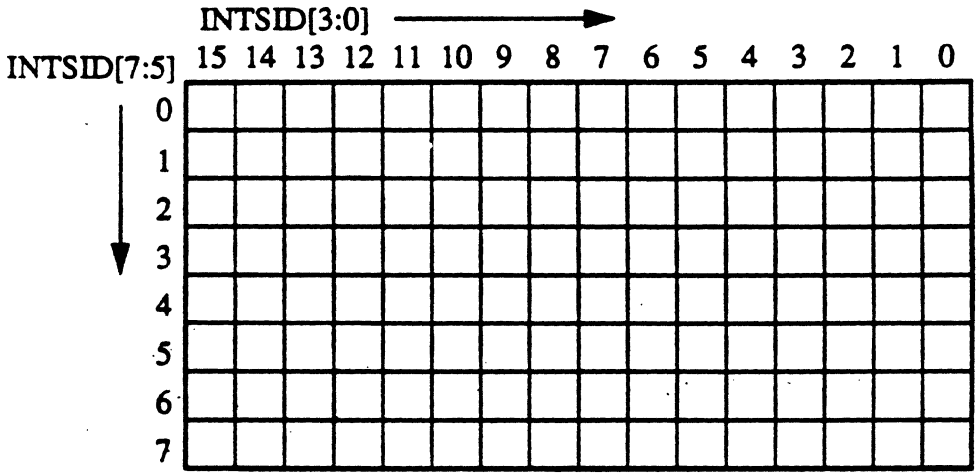
When an interrupt packet is placed on a Dynabus, it is received by all Bus Watchers on that Dynabus. If a Bus Watcher's processor is the target of the interrupt packet, the Bus Watcher updates its Interrupt Table to record information about the interrupt source and sends an XBus interrupt packet down to its Cache Controller. On receiving this packet, the Cache Controller sets appropriate bits in its Interrupt Pending register and interrupts its processor if the interrupt is not masked. The figure below shows this process.



A Dynabus interrupt packet contains four fields: a broadcast bit (B), a TargetID, an INTSID, and a Levels field. Bus Watchers use the B bit and the TargetID to determine if their processor is targeted. A Bus Watcher's processor is the target if either

the broadcast bit is set or if the TargetID field equals the Bus Watcher's Dynabus DeviceID.

The INTSID field identifies the interrupt source and helps to eliminate polling to determine who generated the interrupt. Bus Watchers maintain a 128 bit Interrupt Table that records INTSID information. When a Bus Watcher's processor is the target, the Bus Watcher sets the Interrupt Table bit addressed by the concatenation of INTSID[7:5] and INTSID[3:0] to record that an interrupt has been received from this INTSID. (The INTSID field is 8 bits, but the current implementation of the Bus Watchers cannot accommodate a 256 bit table; software must therefore be careful not to use INTSID[4] for discrimination when allocating INTSID's.) Interrupt Table bits are set by the hardware and cleared by software during interrupt handling.The figure below shows the structure of the Interrupt Table. Information about how the table is allocated amongst the various interrupt sources is consolidated in section 8.5 on page 167.

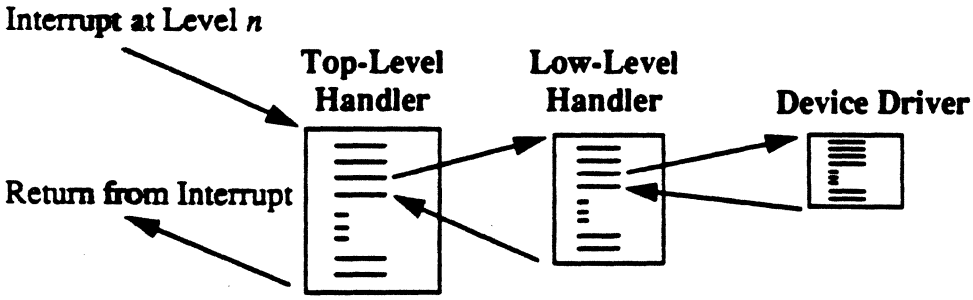


The Levels field in an interrupt packet contains one bit for each SPARC interrupt level at which an interrupt is to be raised. Typically, however, exactly one of the bits in the Levels field is set. When a Cache Controller receives an XBus interrupt packet, it ignores all other fields except for the Levels field. It OR's in the contents of the Levels field into the Interrupt Pending register. The Cache Controller also contains an Interrupt Mask register, which contains one bit for each interrupt level. Level *i* interrupts are masked if the bit for level *i* is a one. The Cache Controller combinatorially drives the processor IRL[3:0] lines to the number of the highest pending level that is not masked.

8.4 Interrupt Handling

Interrupt Handling refers to the phase that begins when one or more processors take an interrupt and ends when any interrupt processing has been completed, and all hardware state associated with the interrupt has been cleared. All of the work done in this phase is initiated by interrupt handling software.

For purposes of discussion, it is useful to structure the interrupt handling software into three levels: a *Top-Level Handler*, several *Low-Level Handlers*, and *Device Drivers*. The figure below shows this three-tier structure.



As the name implies, the Top-Level Handler is invoked immediately after a processor takes an interrupt. It is responsible for determining which Low-Level Handler to call and then invoking this handler. There is one Low-Level Handler for each type of interrupt source. For example, each of I/O, Processor, Miscellaneous, and Local interrupts may have their own Low-Level Handler. The lowest level is implemented by the Device Driver, which actually services the interrupting device and clears the reason for the interrupt at the device. Although this three-level structure is useful for discussion, it is also a good model for the actual implementation since it helps ensure a clean separation between device servicing, source specific servicing, and top-level servicing. With this structuring, adding a new device driver or a new type of interrupt source should be relatively straightforward.

The remainder of this section describes key actions taken by each of the three levels. Needless to say, the order of actions at each of the three levels is crucial to the correct overall functioning of the interrupt handling software.

8.4.1 Top-Level Handler

The figure below shows the code skeleton for the Top-Level Handler for a given interrupt level *lvl*.

```
Top-LevelHandler[lvl]={
   <save process state>
   While TRUE do {
      For each bit i set in Int Table for lvl do {
         Int Table[lvl,i]=0;
         Call Low-Level Handler[lvl,i]
         }
      Clear Interrupt Pending[lvl] in CC
      If Int Table[lvl,*]==0 Then EXITLOOP
      }
   <restore process state>
   }
```

Note that each Interrupt Table bit that is set is cleared *before* the Low-Level Handler is called. This is crucial because the interrupt reason is cleared by the Device Driver which in turn is called by the Low-Level Handler. If the order of the call to the Low-

Level Handler and the clearing of the bit in the Interrupt Table was reversed, interrupts could be lost because of a race condition between a new interrupt packet to this level and the code. If the control flow stops for arbitrarily long just before clearing the bit in the Interrupt Table, any interrupts that arrive before the clearing happens will be lost.

Note also that the Interrupt Table must be checked again just after clearing the Interrupt Pending register in the Cache Controller. This is to prevent interrupts that arrive after the Interrupt Table table has been examined in the For loop but before the pending bit is cleared from being lost. Interrupts that arrive after the Pending Bit has been cleared will not be lost anyway because they will interrupt the processor.

Note that this Top-Level Handler skeleton does not apply to Local interrupts which do not cause any bit to be posted in the Interrupt Table. The serial port interrupts (Level 12) must be handled like regular level-sensitive interrupts. A skeleton for the Level 12 Handler is given below:

```
Top-LevelHandler[12]={
  <save process state>
  For each serial port do {
    If serial port requests interrupt Then {
      Call Driver
      }
    }
  Clear Interrupt Pending[lvl] in CC
  <restore process state>
  }
```

The handling of Level 15 interrupt is much more complex and require a very specific Top-Level Handler. Some Level 15 interrupts are broadcast while other are local which makes the identification of the issuer very delicate. A skeleton will be provided in the future.

8.4.2 Low-Level Handler

The figure below shows a skeleton of the Low-Level Handler for I/O interrupts.

```
IOHandler[lvl,i]={
  IntStateAdr= I/O Address of Int State for source i
  For each slot s at level lvl do {
    mask=1<<(4*lvl+s)
    If mask & Swap[IntStateAdr,mask] Then {
      Call Driver
      Store[IntStateAdr,mask]
      }
    }
  }
```

The Swap statement attempts to ***Take*** the interrupt indivisibly (section 6.4.2.6 on page 139). The statement mask = 1<<(4**lvl*+*s*) simply generates the bit pattern containing a 1 in the position corresponding to level *lvl* and slot *s* and zeros in other

positions. If the Swap returns a 1 in the appropriate position then the ***Take*** is successful and the Handler calls the driver and then does a ***Give*** by executing a Store.

The key point to note about this code is that the call to the Driver is bracketed by a ***Take-Give*** pair, ensuring that two processors do not attempt to service the same device at the same time. The code is implemented so as to ***Take*** each slot in turn. It could also have been written to ***Take*** all four slots at once and service all of the ones that were acquired. However, if this is done, then the ***Give*** must be executed only for those bits that the ***Take*** returned as ones. The code below shows this method.

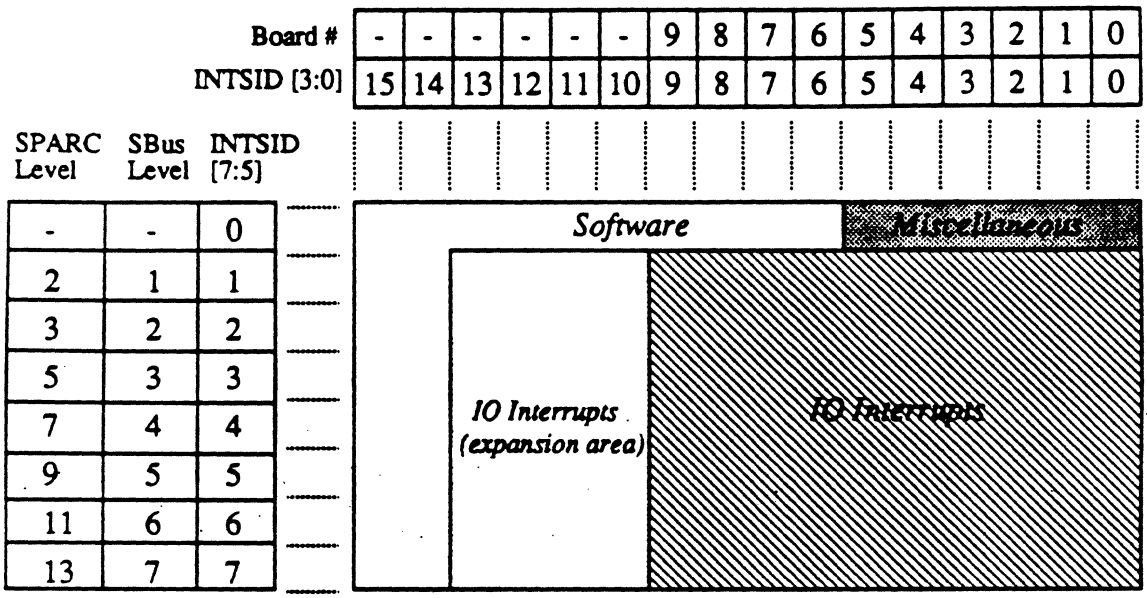
```
IOHandler[lvl,i]={
  IntStateAdr= I/O Address of Int State for source i
  mask=0xF<<(4*lvl)
  gotSlots=mask & Swap[IntStateAdr,mask]
  For each slot s that has bit set in gotSlots do {
     Call Driver
     }
  Store[IntStateAdr,gotSlots]
  }
```

8.4.3 Device Driver

The only detail that is relevant for the Device Driver code is that it must clear the reason for the interrupt at the device after it has serviced the device but before returning control to the Low-Level Handler.

8.5 Map of Interrupt Usage

The figure below consolidates all of the information about interrupt usage in current Sun-4D systems. The table on the left shows the correspondence between SPARC levels, SBus levels, and the value of INTSID[7:5]. The table on the top shows the correspondence between Board number and INTSID[3:0]. The map in the center shows how the Interrupt Table is allocated to different sources. Recall that the interrupt hardware in the Bus Watchers indexes into the Interrupt Table using INTSID[7:5] as the vertical index and INTSID[3:0] as the horizontal index; this is shown by the dotted horizontal and vertical lines, respectively. Finally, the text for *Miscellaneous* Interrupts at the bottom of the figure shows the detailed allocation of INTSID to the three sources in this category.



Interrupt Table Allocation

Miscellaneous Interrupts:

0 - Profiling Timer @ SPARC level 14
1 - Tick Timer @ SPARC level 10
2 - MQH correctable error @ SPARC level 15
3 - MQH uncorrectable error @ SPARC level 15
4 - MQH correctable & uncorrectable errors in the same block @ SPARC level 15
5 - SBI XPT parity error @ SPARC level 15

Chapter 9
Reset

This chapter describes the reset model for Sun-4D systems. It begins by outlining the different types of reset and describing how each is initiated. Next, it provides a detailed description of "loopback mode" in which each system board is cut off from the others, allowing all of them to be tested in parallel without interfering with one another. The Chapter closes with the sequence of actions needed to initialize the ASICs and boot the machine.

9.1 Reset Types

There are two types of reset in Sun-4D systems, *System* and *Local*.

- A *System* reset causes the entire system to be reset. The hardware is careful to initialize only the minimum amount of state needed to boot so as to provide as much information as possible for post-error analysis. Following a system reset, each system board automatically goes into loopback mode (loopback mode is described in Section 9.3). The detailed effect of system reset on the Processor, Memory, and I/O units was described in earlier chapters devoted to these units.
- A *Local* reset causes a single Viking processor, and perhaps its Cache Controller to be reset.

9.1.1 System Reset

There are four possible sources of system reset in Sun-4D systems. Each is described below in turn.

9.1.1.1 Power-On Reset

Power-On reset is generated when power is turned on or when the POR switch (located inside the cabinet) is activated. The switch is provided mainly as a convenience to avoid power cycling the machine frequently during the machine's bring up phase in the lab. This switch is disabled when the front panel key is is in the "System Secure" position. There is no way to distinguish whether a Power-On reset was generated by a real power-on or by the switch.

A Power-On reset sets the RstStat field of the BootBus Status_2 registers of all Processor Units to 00_2 (See section 4.6.8.3 on page 78), and sets the MastPres bit in the BootBus of all processor units to one (See section 4.6.7 on page 75).

9.1.1.2 Service Processor Reset

Sun-4D systems provides for an optional Service Processor which is connected to the machine via the system control board. A system reset initiated by the Service Processor is called a Service Processor Reset.

A Service Processor reset sets the RstStat field of the BootBus Status_2 registers of all Processor Units to 00_2 (See section 4.6.8.3 on page 78), and sets the MastPres bit in the BootBus of all processor units to zero (See section 4.6.7 on page 75).

A Service Processor reset cannot be distinguished from a Power-On reset. This is not a problem because the Service Processor automatically assumes responsibility for initializing and booting the system whenever it is present.

NOTE: Although the Sun-4D architecture provides hardware support for a service processor, the current version of POST does not provide the necessary hooks to attach a service processor to a Sun-4D system, except for JTAG full scan during manufacturing, in which case POST stops as soon as it detects a Service Processor Reset.

9.1.1.3 System Software Reset

A System Software Reset can be generated by any processor by storing into the System Software Reset register located on its BootBus interface (See section 4.6.8.5 on page 80).

A System Software reset sets the RstStat field of the BootBus Status_2 registers of all Processor Units to 10_2 (See section 4.6.8.3 on page 78), and sets the MastPres bit in the BootBus of all processor units to one (See section 4.6.7 on page 75).

9.1.1.4 System Watchdog Reset

If no Service Processor is present in the system, all errors considered fatal generate a system reset called System Watchdog reset (See section 7.2.4 on page 151).

A System Watchdog reset sets the RstStat field of the BootBus Status_2 registers of all Processor Units to 01_2 (See section 4.6.8.3 on page 78), and sets the MastPres bit in the BootBus of all processor units to one (See section 4.6.7 on page 75).

If a Service Processor is present, fatal errors simply stop Dynabus traffic but *do not* initiate a system reset. The Service Processor must monitor the Central Arbiters to detect whether a Dynabus Stop condition exists, and on detecting this condition decide whether to reset the system or not.

NOTE: Fatal errors detected by CC and SBI are reported by the BW(s) and IOC(s) to which they are respectively attached. If all BWs attached to a CC are frozen, a fatal error detected by CC is not reported, except for the error bits in CC. If all IOCs attached to an SBI are frozen, a fatal error detected by SBI does not cause a System Watchdog Reset. This is of interest only during POST.

9.1.2 Local Reset

There are two types of Local Reset, one explicitly initiated by software and the other resulting from a software error.

9.1.2.1 Local Software Reset

A Viking processor can reset itself by storing a one into the SI bit of the CC Reset register (See section 4.4.9 on page 49). The Reset register is also accessible from other processors via ECSR space so that a processor can generate a Local Software reset for any processor in the system. Local Software reset should not be confused with System Software reset.

Before issuing a local software reset, the programmer should ensure that all pending writes have completed (fields NCSPC, SPC, WP, RP, PP of the Status register, section 4.4.8 on page 48, all zero). During a Local Software reset, CC prevents the processor from issuing additional memory operations, clears the SXP field of the Status register, clears the WD bit of the Reset register (SI remains one), and resets the Viking processor. No other CC state is modified.

NOTE: Local software resets are not supported by POST, OBP or the operating system, and should not be used.

9.1.2.2 Watchdog Reset

If a synchronous trap occurs while traps are disabled, the Viking processor does an internal reset referred to as a Watchdog reset [7]. Watchdog resets are defined by the SPARC architecture [1].

On a Watchdog reset, Viking takes a reset trap and its internal MMU enters Boot mode (the BT bit of the Viking Control register is set to zero). None of the Viking internal state machines are reset, and all pending operations are allowed to complete normally. Additionally, the WD bit of the CC Reset register is set to one (See section 4.4.9 on page 49). There is no other effect on any other component.

9.2 Reset Source Identification

When a Viking starts executing instructions right after a reset, it first needs to identify the type of reset that occurred. It can do this by examining state bits in its CC and in the Boot Status registers on the BootBus.

The various resets must be checked for in the following order: Watchdog reset; Local Software reset; System Software reset, System Watchdog reset, Service Processor reset, and Power-On reset.

The table below summarizes the values of the relevant state bits immediately after a reset for the various types of resets:

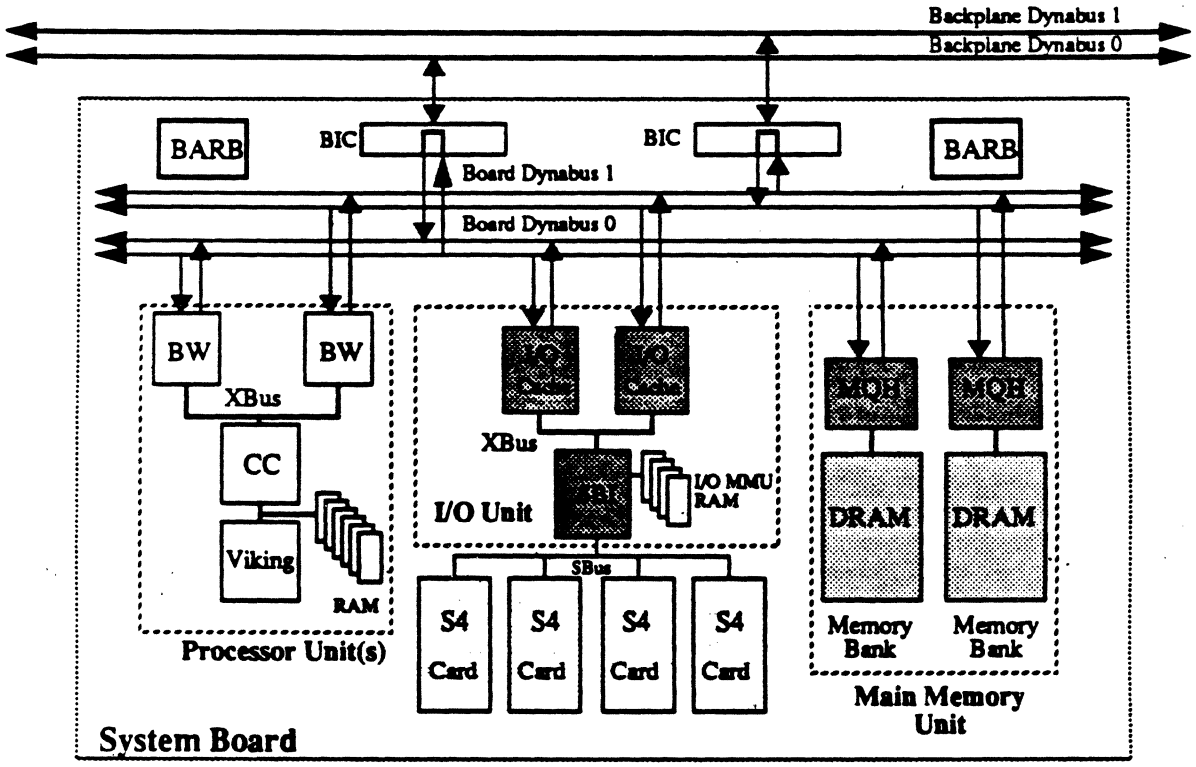
WD	SI	RstStat	MastPres	Reset Source
1	X	XX	X	Watchdog Reset
0	1	XX	X	Local Software Reset
0	0	01	X	System Watchdog Reset
0	0	10	X	System Software Reset
0	0	00	0	Service Processor or Power-On Reset
0	0	00	1	Power-on Reset

9.3 Loopback Mode

Loopback mode provides a way to isolate a given system board from the backplane such that other system boards on the same backplane cannot affect this board and this board cannot affect them. The prime motivation for loopback mode is to allow boards to be tested in parallel during booting without having boards interfere with each other. A system board in loopback mode forms a complete system, containing

its own processor (s), memory and I/O. Dynabus arbitration among these units is handled by the on board BARBs.

In describing loopback mode, it is helpful to refer to the figure below. The figure shows that each of the two DynaBusses on a SunDragon is divided into a backplane segment, and two on-board segments (in/out) per system board (Scorpion follows the same physical structure, except that there is only one DynaBus instead of two). The various segments are connected by single stage pipeline registers implemented by BIC chips. When a board is put in loopback mode, the BIC's on that board disconnect the board from the backplane segment, and connect the output segment of the board to the input segment through a single pipeline stage.



LoopBack Mode (SunDragon case)

Loopback mode is controlled via the LBT bit in the BIC and the LB bit in BARB. These bits are accessible only via JTAG. On a system reset, the LBT bit for all BICs and the LB bits for all BARBs in the system are set to one, putting all system boards into loopback mode (See section 10.3.10 on page 187).

Loopback mode may be exited by clearing the LBT bits *and* issuing a packet on the Dynabus for which loopback is being removed. The first packet issued on the on-board Dynabus after LBT has been cleared actually takes a BIC out of loopback mode.

NOTE: If the LBT bit is cleared but no packet is issued onto the corresponding Dynabus, packets on the backplane segment will not be placed on the local segment.

If there is no Service Processor in the system (MastPres=1), the JTAG Master interface allows Viking to access the on-board JTAG bus during loopback. However,

it is not possible to access scan rings on other System Boards or on the System Control Board.

9.4 System Bootstrap

The specification for the Sun-4D Power-On Self-Test (POST) firmware [21] provides a detailed overview of how to handle the hardware reset sequence in the absence of a service processor.

In the current implementation, the presence of a service processor precludes normal system operation and is allowed only for hardware diagnostics during system bring-up and manufacturing. Future releases of POST may provide more generic service processor support.

Chapter 10
JTAG

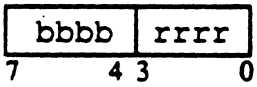
The scan methodology in Sun-4D is based on the proposed IEEE P1149.1 (or JTAG) specification [10].

This chapter **requires** familiarity with the JTAG specification as well as with the hardware organization of SunDragon.

JTAG is used in SunDragon for system initialization and diagnostics. The JTAG interface may be controlled by any SunDragon processor as well as by an external service processor (for manufacturing or field service).

10.1 Scan Rings

Current Sun-4D system provide multiple scan rings which are identified by a scan ring number, which has the following format:



Each scan ring consists of a number of chips chained in TDI->TDO order. All chips in a ring share a common TMS signal. The board slot number **bbbb** may have the following values:

bbbb	Designates
0x0	System slot #0
0x1	System slot #1
0x2	System slot #2
0x3	System slot #3
0x4	System slot #4
0x5	System slot #5
0x6	System slot #6
0x7	System slot #7
0x8	System slot #8
0x9	System slot #9
0xA-0xE	Reserved
0xF	System Control Board

NOTE: SunDragon can have up to 10 system board while Scorpion is limited to 4 system boards.

Access to a ring number which is reserved will have unspecified results (either another ring will be used instead, or TDO will always be 1). Access to a ring which is defined, but absent from the current system configuration (e.g. not all system slots are populated) will always result in TDO=1, which allows software to identify the current configuration.

10.1.1 System Board

The allocation of rings `rrrr` on a SunDragon system board (`bbbb` = 0 to 9) is:

rrrr	TDI -> TDO Chain order
0x0	74BCT8244, 29F816
0x1	BIC0#3, BIC 0#2, BIC0#1, BIC0#0, BARB0, BIC1#3, BIC1#2, BIC1#1, BIC1#0, BARB1
0x2	BW0, Viking, CC, BW1 (processor A)
0x3	MQH0, MQH1
0x4	IOC0, SBI, IOC1
0x5	BW0, Viking, CC, BW1 (processor B)
0x6-0xF	Reserved

The allocation of rings `rrrr` on a Scorpion system board (`bbbb` = 0 to 3) is:

rrrr	TDI -> TDO Chain order
0x0	74BCT8244, 29F816
0x1	BIC0#3, BIC 0#2, BIC0#1, BIC0#0, BARB0,
0x2	BW0, Viking, CC (processor A)
0x3	MQH0
0x4	IOC0, SBI
0x5	BW0, Viking, CC (processor B)
0x6-0xF	Reserved

where:

74BCT8244: Scan octal buffer

29F816: 16K-bit JTAG scan EEPROM

BIC*n*#*p*: Bus Interface Chip; *n* specifies the DynaBus number (0 or 1) and *p* the half-word buffered by the chip (0 to 3, 0 is least significant half-word, 3 is most significant half-word)

BW*n*: Bus Watcher; *n* specifies the DynaBus number

BARB*n*: Board Arbiter; *n* specifies the DynaBus number

Viking: Viking CPU

CC: Cache Controller chip

MQH*n*: Memory Queue Handler chip; *n* specifies the DynaBus number

IOC*n*: I/O Cache chip; *n* specifies the DynaBus number

SBI: SBus Interface

NOTE: Viking and CC form a field-replaceable unit (FRU), the Viking module, which may be absent. For example, a system board may have 0, 1 or 2 Vikings. When a Viking module is not present for a given processor unit, the corresponding scan ring will appear to fail since there is no TDI to TDO continuity. This is a normal case and not an error. The corresponding BWs cannot be tested, but should remain in the frozen state after a system reset and be totally invisible to system operation.

The 74BCT8244 outputs on the system board are used to drive TCK on the on-board scan rings as follows:

2Y1: Driver for TCK on ring 1 (BICs & BARBs)

2Y2: Driver for TCK on ring 2 (Viking, CC & BWs for processor A)

2Y3: Driver for TCK on ring 3 (MQHs)

2Y4: Driver for TCK on ring 4 (IOCs & SBI)

1Y4: Driver for TCK on ring 5 (Viking, CC & BWs for processor B)

Boundary scan on the 74BCT8244 can be used to disable TCK for on-board scan rings (except scan ring 0). When the 74BCT8244 is not in boundary scan mode, TCK is driven normally on the other scan rings.

The board type is indicated in the 29F816 (format TBD).

10.1.2 System Control Board

The allocation of rings on the SunDragon system control board (**bbbb** = 0xF) is as follows:

rrrr	TDI->TDO chain order
0x0	29F816
0x1	CARB0, CARB1
0x2	74BCT8244
0x1-0xf	Reserved

The allocation of rings on the Scorpion system control board (**bbbb** = 0xF) is as follows:

rrrr	TDI->TDO chain order
0x0	29F816
0x1	CARB0
0x2	74BCT8244
0x1-0xf	Reserved

where:

74BCT8244: Scan octal buffer

29F816: 16K-bit JTAG scannable EEPROM

CARB*n*: Central Arbiter; *n* specifies the DynaBus number

The 29F816 provides 16K bits of EEPROM, e.g. for system identification and Ethernet address.

The 74BCT8244 outputs on the system control board are connected as follows (the default state is provided from the corresponding inputs, which are driven to the outputs when a boundary external test is not forced via JTAG):

1Y1: Forces fan failure when set to 1, defaults to 0

1Y2: Forces DC not OK failure when set to 1, defaults to 0

1Y3: Forces AC failure when set to 1, defaults to 0

2Y4: Kick AC breaker (i.e. system power off) when set to 1, defaults to 0

Extreme care should be exercised when accessing the system control board ring 0x2 since it allows to remove power from the system.

10.2 Software Interface to JTAG

10.2.1 Active JTAG Controller

Each board (system board or system control board) contains a Board Access Controller which drives all on-board scan rings. The BAC can be controlled from three sources:

1. Service processor: when a service processor is present, all BACs are controlled by it. In this mode, a processor has no control over JTAG activity.
2. System master: when there is no service processor and a processor has set the MastEn bit to one (See section 4.6.8.1 on page 77 and section 4.6.7 on page 75), the JTAG Master controller on the corresponding BootBus controls all BACs.
3. Local master: if there is no service processor and no processor has MastEn set, the JTAG Master controller on each board controls the board's BAC, but it does not have access to the other board's BAC. This is normal state after a reset (unless a service processor is present), which allows each board to perform its initialization sequence independently from other system boards.

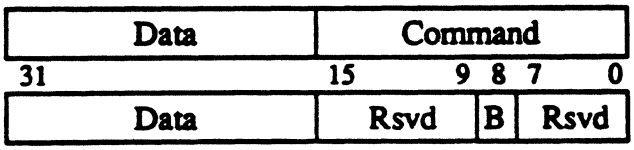
The presence of a service processor is detected by the POST software when MastPres is set (See section 4.6.8.2 on page 78 and section 4.6.7 on page 75) just following a system reset (MastPres is set in cases 1 and 2, and MastEn is cleared by a system reset).

MastEn should never be asserted if a service processor is present, and should be asserted by at most one system board if there is no service processor. This must be enforced by the POST software and is not checked in hardware.

10.2.2 Programming the JTAG Controller

Each system board contains, on the BootBus (See section 4.6.7 on page 75), a JTAG Command register which allows to issue JTAG commands. Depending on the current configuration (see section 10.2.1), the JTAG Command register may control only the on-board JTAG or give access to all boards.

The format of the JTAG Command register is:



where:

Command: Indicate a JTAG command sequence to be issued. The Command field is write-only. It should be modified only when the Busy bit is zero.

B: Busy. The B bit is zero when there is no active JTAG command, one otherwise. The B bit is set when a command is issued and remains set until the command is performed. The B bit is read-only.

Data: Data to be shifted into or out of a JTAG scan ring. The data field is shifted to the right into the JTAG scan ring. Simultaneously, the JTAG scan ring data is right shifted into the data field most significant bit. The value of this field is valid on a read only when the Busy bit is zero.

Rsvd: Reserved. The value of this bit field is unspecified on a read.

The command register supports byte, half-word and word access. When a write operation affects the least significant byte of the command field, a new JTAG command is initiated, which causes a sequence of JTAG state transitions.

When the Busy bit is zero, the JTAG state machine is always in one of the states Test logic reset, Run-test/Idle, Pause-DR or Pause-IR. The following commands are available:

Code	Command	Initial state	Final state
0x6000	SelRing	Run-test/Idle	Run-test/Idle
		Test logic reset	Test logic reset
0x5050	SelDataReg	Run-test/Idle	Pause-DR
		Test logic reset	Pause-DR
0x5068	SelInstReg	Run-test/Idle	Pause-IR
		Test logic reset	Pause-IR
0x00A0 to 0xF0A0	Shift 1 bit to Shift 16 bits	Pause-DR Pause-IR	Pause-DR Pause-IR
0x50C0	RunIdle	Pause-DR	Run-test/Idle
		Pause-IR	Run-test/Idle
0x50E8	IRtoDR	Pause-IR	Pause-DR
0x50F4	DRtoIR	Pause-DR	Pause-IR
0x50FF	Reset	Any	Test logic reset

Using any other command code will produce unpredictable results. If a command is issued and the initial state is not specified in this table, it will produce unpredictable results.

NOTE: Bits [11:8] of the Command code are actually not decoded. They are reserved for use by the SBus JTAG master card.

10.2.2.1 Reset

The Reset command places the currently selected ring in the Test logic reset state. The value of the Data field when the command completes is unspecified.

10.2.2.2 SelRing

The SelRing command allows to change the currently selected ring. The value of Data[7:0] defines the new selected ring, using the `bbbbrrrr` format. If the newly selected ring is not defined, the results are unspecified. SelRing must be issued from the Run-test/Idle or Test logic reset states. If the command is issued from another state, the results are unspecified. The command leaves the current ring in its current state and selects the new ring without modifying its state.

The value of the Data field when the command completes is unspecified.

10.2.2.3 SelDataReg

The SelDataReg command must be initiated from the Run-test/Idle or Test logic reset states. If the command is initiated from another state, the results are unspecified. The command causes the TAP to go to the Pause-DR state via the Capture-DR state.

The value of the Data field when the command completes is unspecified.

10.2.2.4 SelInstReg

The SelInstReg command must be initiated from the Run-test/Idle or Test logic reset states. If the command is initiated from another state, the results are unspecified. The command causes the TAP to go to the Pause-IR state via the Capture-IR state.

The value of the Data field when the command completes is unspecified.

10.2.2.5 Shift

The Shift command must be issued from the Pause-DR or Pause-IR states. If the command is issued from another state, the results are unpredictable. The command shifts 1 to 16 bits from the Data field onto TDI, starting with the least significant bit of Data, and copies TDO into the Data field, starting with the most significant bit. The number of bits shifted in indicated in the four low-order bits of the command, with `0x0` indicating 1 bit and `0xF` indicating 16 bits. When the command completes, the TAP is in the same state as when the command was initiated.

10.2.2.6 RunIdle

The RunIdle command may be issued from the Pause-DR or Pause-IR states. If the command is issued from another state, the results are unpredictable. The command causes the TAP to go to the Run-test/Idle state via the Update-DR state if the initial state was Pause-DR, or via the Update-IR state if the initial state was Pause-IR.

The value of the Data field when the command completes is unspecified.

10.2.2.7 IRtoDR

The IRtoDR command must be issued from the Pause-IR state. If the command is issued from another state, the results are unpredictable. The command causes the TAP to go to the Pause-DR state via the Update-IR and Capture-DR states.

The value of the Data field when the command completes is unspecified.

10.2.2.8 DRtoIR

The DRtoIR command must be issued from the Pause-DR state. If the command is issued from another state, the results are unpredictable. The command causes the TAP to go to the Pause-IR state via the Update-DR and Capture-IR states.

The value of the Data field when the command completes is unspecified.

10.2.3 Service Processor Interface

The software interface between a SunDragon CPU and the service processor is provided by the BC bit of the Cache Controller status register (See section 4.4.8 on page 48), which is used as a semaphore, and the DynaData register of the Bus Watchers (See section 4.5.4 on page 63), which is used as a mailbox.

10.3 On-chip JTAG Instructions

10.3.1 Common Instruction Set

All SunDragon components, except Viking and CC, support a common set of JTAG instructions and registers. As required by the JTAG specification, scanning is LSB first (i.e. shift right).

Note that current Sun-4D systems components do not use the JTAG reset signal.

10.3.1.1 Instruction Register

The instruction register size is 6 bits long. It is loaded with `000001` during the Capture-IR state if the clock is connected to the PLL output, it is loaded with 0001010 if the chip is in direct clock mode. When the JTAG controller enters the Reset state, the Instruction register is initialized to `111110` (IDCODE).

The following instructions are supported:

Binary Code	Instruction
0 0 B 0 T 0	Boundary
0 1 B C 1 0	FullScan
1 0 0 0 T U	Shadow0
1 0 0 1 T U	Shadow1
1 0 1 0 T U	Shadow2
1 0 1 1 T U	Shadow3
1 1 1 1 0 0	Clr Clk Bypass
1 1 1 1 0 1	Set Clk Bypass
1 1 1 1 1 0	IDCODE
1 1 1 1 1 1	BYPASS

where:

B: Boundary Disable. If this bit is one, boundary cells do not interfere with normal operation. If it is zero, boundary cells replace the input/output value with the value in their holding register. Instructions for which the B bit is not specified act as if B was one (i.e. do not interfere with normal operation).

T: Test Clock Enable. If this bit is one, the system clock is replaced with a gated version of the JTAG test clock which is active in the Shift-DR state and, if the C bit is specified and is one, in the Capture-DR state. If it is zero, the system clock is used normally inside the chip. Instructions for which the T bit is not specified behave as if T was zero, except for the FullScan instruction which behaves as if T was one. If the bit is ever set to one, it cannot be safely reset to 0 without resetting the chip because the on-chip clock PLLs may have lost synchronization.

C: Capture Enable. If this bit is one, the full scan chain loads data from its parallel inputs on the rising edge of the JTAG test clock following entry in the Capture-DR state. If it is zero, no such parallel load is performed.

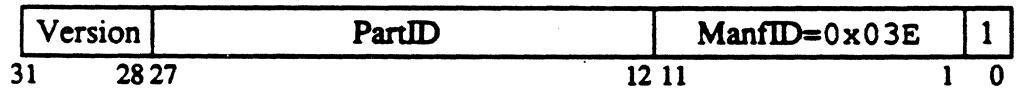
U: Update Enable. If this bit is one, the shadowed register will be updated in the Update-DR phase. If this bit is zero, the shadowed register is not modified in the Update-DR phase.

The Clr Clk Bypass /Set Clk Bypass instructions allow to define the on-chip clock as being either the output of the PLL on the system clock, or the (ungated) test clock. Both instructions select the bypass register. On a power-on, the on-chip clock is connected to the PLL output.

All instructions not specified in this table have the same effect as BYPASS.

10.3.1.2 ID Register

The ID register has the following format:



where

Version: 4-bit chip version number (0x1=1, ..., 0xF=15, 0x0=16).

PartID: 16 bit part number (to be allocated by manufacturing for each chip).

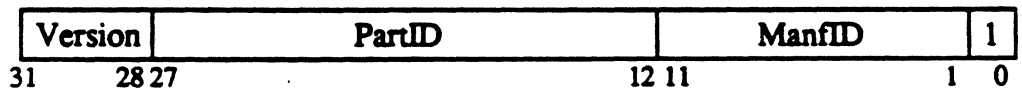
ManfID: Sun's JEDEC identifier, 0x3E.

10.3.2 Viking

Viking does not use the same JTAG instructions as most SunDragon components.

The instruction register size is 5 bits. It is loaded with 00001 during the Capture-IR state. When the JTAG controller enters the Reset state, the Instruction Register is initialized to 10000 (IDCODE).

The IDCODE register has the following format:



where

Version: 4-bit chip version number (0x0)

PartID: 16 bit part number (0x0004)

ManfID: Manufacturer's JEDEC identifier, `0x017`.

The encoded ID is thus `0x0000402F`.

The BYPASS instruction is encoded as `11111`.

See the Viking specification [7] for all other JTAG instructions in Viking.

10.3.3 CC

CC does not use the same JTAG instructions as most SunDragon components.

The instruction register size is 4 bits. It is loaded with `0001` during the Capture-IR state. When the JTAG controller enters the Reset state, the Instruction register is initialized to `0110` (IDCODE).

The following instructions are supported:

Binary Code	Scan chain
0 0 0 0	EXTEST
0 0 0 1	SAMPLE
0 0 1 0	INTEST
0 0 1 1	SS
0 1 0 0	INTSCAN
0 1 0 1	DATASCAN
0 1 1 0	IDCODE
0 1 1 1	SHADOWSCAN
1000 - 1110	Reserved
1 1 1 1	BYPASS

where:

EXTEST, SAMPLE, INTEST, IDCODE, BYPASS: Standard JTAG public instructions

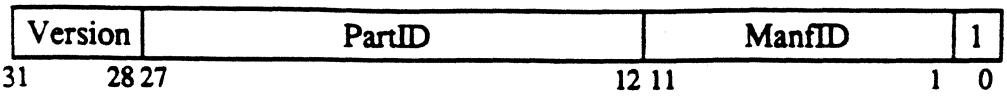
SS: Single Step. Behaves as INTEST, except that the JTAG CK signal is used to clock all on-chip flip-flops, instead of the normal clock signals PCLK and BCLK.

INTSCAN: Internal Scan. Selects the full scan internal chain.

DATASCAN: Data Scan. Selects an internal register shadow scan chain for the following registers: Control register, Reset register, Error register and Status register. For this instruction, the Update-DR phase has no effect, i.e. the internal registers can only be read, not written, using this command

SHADOWSCAN: Shadow Scan. Selects an internal shadow scan chain which contains only the BC bit of the Status register. The Update-DR phase modifies the value of BC.

The IDCODE register has the following format:



where

Version: 4-bit chip version number (`0x0`)

PartID: 16 bit part number (`0x0003`)

ManfID: Manufacturer's JEDEC identifier, `0x017`

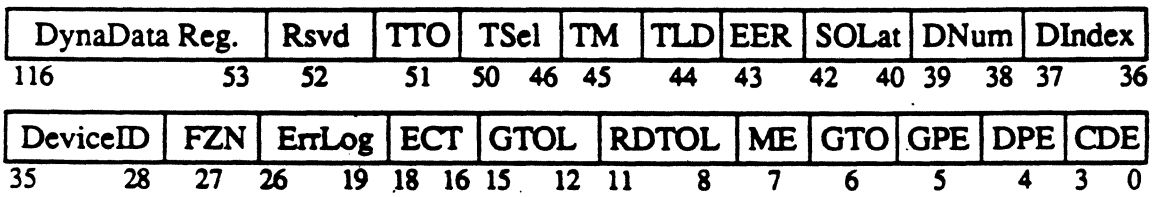
The encoded ID is thus `0x0000302F`.

10.3.4 BW

BW uses the standard SunDragon JTAG instructions with the following PartIDs:.

PartID	Version	ComponentID	Component
`0xADB`	1	`0x10ADB07D`	BW, 2K Tags
`0xADB`	2	`0x20ADB07D`	BW, 2K Tags
`0xD39`	1	`0x10D3907D`	BWP, 4K Tags

BW supports a 117-bit Shadow0 scan chain which has the following structure:



See section 4.5.3 on page 59 for a description of the BW Control and Status register and section 4.5.4 on page 63 for a description of the BW DynaData register.

All bits of the scan chain are read on a Capture-DR operation. On an Update-DR operation, the following limitations apply:

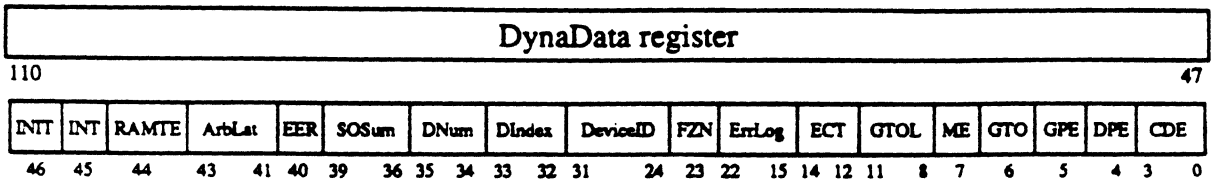
- ME, GTO, GPE, DPE, CDE are write-one-to-clear: an Update-DR with a value of zero does not modify the current value, whereas an Update-DR with a value of one clears the corresponding bits. This is the same behavior as when these fields are accessed via CSR space
- DynaData register and ErrLog are write protected when the current value of fields GTO, GPE, DPE, CDE is not all zeroes: an Update-DR to those two fields has no effect in this case.
- All other bits are normally updated on Update-DR, even if they are specified as read-only when accessed via CSR or Local space

10.3.5 MQH

MQH uses the standard SunDragon JTAG instructions with the following PartIDs:

PartID	Version	ComponentID	Component
0x0ADC	1	0x10ADC07D	MQH
0x0D86	1	0x10D8607D	MQHP

MQH supports a 111-bit Shadow0 scan chain which has the following structure:



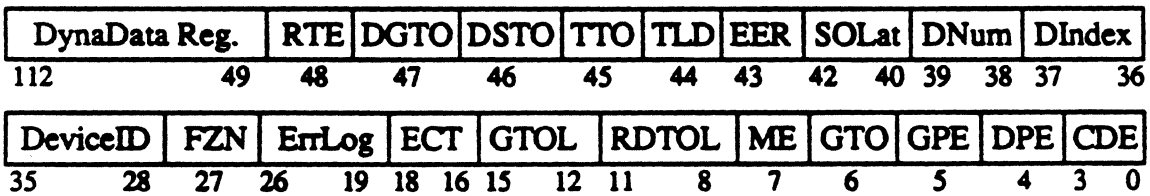
See section 5.4.2 on page 98 for a description of the MQH DynaBus Control and Status register and section 5.4.3 on page 102 for a description of the MQH DynaData register.

All bits of the scan chain are read on a Capture-DR operation. On an Update-DR operation, the following limitations apply:

- ME, GTO, GPE, DPE, CDE are write-one-to-clear: an Update-DR with a value of zero does not modify the current value, whereas an Update-DR with a value of one clears the corresponding bits. This is the same behavior as when these fields are accessed via CSR space
- DynaData register and ErrLog are write protected when the current value of fields GTO, GPE, DPE, CDE is not all zeroes: an Update-DR to those two fields has no effect in this case.
- All other bits are normally updated on Update-DR, even if they are specified as read-only when accessed via CSR space

10.3.6 IOC

IOC uses the standard SunDragon JTAG instructions with a PartID=0x0ADD (decimal 2781), Version=0x1, which results in an encoded ID of 0x10ADD07D. IOC supports a 113-bit Shadow0 scan chain which has the following structure:



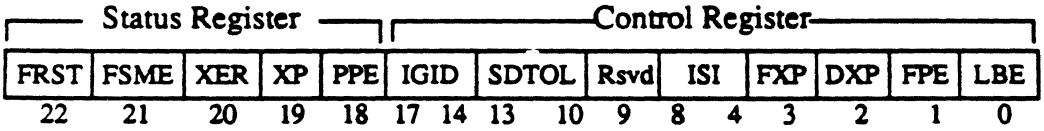
The IOC DynaData register is described in section 6.3.3.2 on page 124. The other fields of this scan chain are the non-reserved subfields of the IOC DynaBus Control and Status register, which is described in section 6.3.3.3 on page 128.

All bits of the scan chain are read on a Capture-DR operation. On an Update-DR operation, the following limitations apply:

- ME, GTO, GPE, DPE, CDE are write-one-to-clear: an Update-DR with a value of zero does not modify the current value, whereas an Update-DR with a value of one clears the corresponding bits. This is the same behavior as when these fields are accessed via CSR space.
- DynaData register and ErrLog are write protected when the current value of fields GTO, GPE, DPE, CDE is not all zeroes: an Update-DR to those two fields has no effect in this case.
- All other bits are normally updated on Update-DR, even if they are specified as read-only when accessed via CSR space.

10.3.7 SBI

SBI uses the standard SunDragon JTAG instructions with a PartID=0x0ADE (decimal 2782), Version=0x1, which results in an encoded ID of 0x10ADE07D. SBI supports a 22-bit Shadow0 scan chain which covers the following registers:



where:

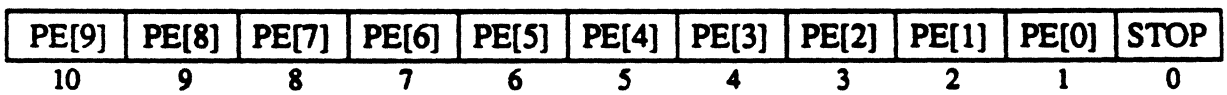
FRST: Forced Reset. This bit is cleared on a system reset. When set to one, it forces an internal reset of SBI. When it is zero, SBI operates normally.

All other bits are described in the SBI Control register (See section 6.4.2.2 on page 134) or in the SBI Status register (See section 6.4.2.3 on page 135).

All bits of the scan chain are read on a Capture-DR operation. All the bits in the Shadow0 scan chain are updated on an Update-DR operation.

10.3.8 CARB

CARB uses the standard SunDragon JTAG instructions with a PartID=0x0AE0 (decimal 2784), Version=0x1, which results in an encoded ID of 0x10AE007D. CARB supports an 11-bit Shadow0 scan chain which covers the following bits:



where:

STOP: Stopped. This bit is set to one when CARB receives a STOP code on one of its input ports. When STOP is one, CARB issues the STOP code on all output ports. This bit is *cleared* on a system reset, and is intended to provide a simple way for a service processor to poll the system for error conditions.

PE[i]: Parity Error. PE[*i*] is set to one when CARB encountered a parity error on input port *i*. When PE[*i*] is one, CARB issues the STOP code on all output ports.

All the bits may be read and written.

10.3.9 BARB

BARB uses the standard SunDragon JTAG instructions with a PartID=0x0AD9 (decimal 2777), Version=0x1, which results in an encoded ID of 0x10AD907D. BARB supports an 11-bit Shadow0 scan chain which covers the following bits:

LB	PE[3]	PE[2]	PE[1]	PE[0]	GntPE	QOF[3]	QOF[2]	QOF[1]	QOF[0]	GTE
10	9	8	7	6	5	4	3	2	1	0

where:

LB: Loopback Mode. If this bit is one, BARB arbitrates all requests locally within the board and does not communicate with CARB. If this bit is zero, arbitration operates normally.

PE[i]: Parity Error. PE[*i*] is set to one when a parity error is detected on client input port *i*. When PE[*i*] is equal to one, BARB issues the STOP code on the CARB output port (if LB=1, BARB issue STOP to all clients and BICs).

GntPE: Grant Parity Error. This bit is set to one when a parity error is detected on the inputs from CARB. When this bit is one, BARB issues the STOP code on the CARB and client output ports, and to the BICs (if LB=1, the GntPE bit is set but no other action is taken).

QOF[i]: Queue Overflow. This bit is set to one when an arbitration request queue associated to port *i* overflows. When this bit is one, BARB issues the STOP code on the CARB output port (if LB=1, BARB issue STOP to all clients and BICs).

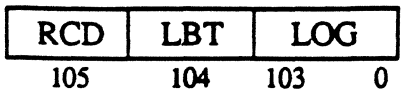
GTE: Grant Type Error. This bit is set to one when the grant type provided by CARB does not match the grant type precomputed by BARB. When this bit is one, BARB issues the STOP code on the CARB output port (if LB=1 and this error occurs, the GntPE bit is set but no other action is taken).

All the bits may be read and written. All bits must first be initialized to 0 by POST except LB. LB is cleared when the system board has been tested and is put out of loopback mode by POST.

10.3.10 BIC

BIC uses the standard SunDragon JTAG instructions with a PartID=0x0ADA (decimal 2778), Version=0x1, which results in an encoded ID of 0x10ADA07D. BIC

supports a 106-bit Shadow0 scan chain which covers the following bits:



where:

RCD: Record Data. If this bit is one, the parity logging history buffer is enabled. If this bit is zero, it is disabled. When the associated BARB receives the STOP code from CARB, RCD is cleared. RCD may be read and written.

LBT: Loopback Mode for Transmission. If this bit is one, data received from the on-board bus is not transmitted to the backplane, but is instead transmitted back onto the on-board bus. If this bit is zero, data received from the on-board bus is normally transmitted on the backplane bus. There is a second loopback mode bit, LBR (Loopback mode for reception) which is not accessible via the Shadow0 scan chain. LBR is set to one whenever LBT is set to one. LBR is set to zero when LBT is zero and BIC receives non-idle data from the board (i.e. LBR is cleared on the first header received from the board after LBT is cleared). When the associated BARB received the STOP code from CARB, LBT is set to one. LBT may be read and written.

LOG: Parity Log. The parity log is divided in 4 sections:

LOG[103:78]: byte A received from board
LOG[77:52]: byte B received from board
LOG[51:26]: byte A received from backplane
LOG[25:0]: byte B received from backplane

Within each section, the least significant bit is the oldest recorded cycle. Each bit indicates the parity syndrome (0=even, 1=odd) for a cycle. The Update-DR phase never modifies the LOG bits.

10.3.11 JTAG EEPROM

A Texas Instruments 16K-bit serial JTAG flash EEPROM, TMS29F816, is used on the system board and on the system control board. Refer to the TMS29F816 data sheet [14] for a description of the JTAG interface to this part.

The JTAG EEPROMs are configured so that the first 128 bytes of data cannot be erased or reprogrammed. This area is intended for manufacturing information.

The JTAG EEPROM on the system control board contains, information that was in the IDPROM of previous systems (e.g. Ethernet address, unique system ID for software licensing). The contents of this virtual IDPROM is copied to the BootBus NVRAM during system initialization.

10.3.12 74BCT8244

A Texas Instruments octal buffer with JTAG support (74BCT8244), is used on the system board and on the system control board for diagnostics purposes. Refer to the SN74BCT8244 data sheet [18] for a description of the JTAG interface to this part.

References

[1] The SPARC Architecture Manual, Version 8, Sun Microsystems, November 1989.

[2] The SPARC Reference MMU Architecture, Rev. 1.3, Sun Microsystems, October 1988.

[3] Demap: Definition and Implementation, Pradeep Sindhu, March 1989, Revised August 1989.

[4] Dynabus Logical Specification, Rev 2.3, Pradeep Sindhu, July 1989, Revised Nov 1991

[5] AmZ8030/AmZ8530 Serial Communications Controller, AMD, Part # 7513A, May 1986.

[6] MK48T02/03/12/13 2Kx8 Zeropower/Timekeeper RAM, Thomson Components Mostek Memory Data Book, Publication #4430219, 1987.

[7] The Viking Microprocessor (T.I. TMS390Z50) User Documentation, Release 2.0, Sun Microsystems, November 1990.

[8] A Class of Odd-Weight-Column SEC-DED-S*b*ED Codes for Memory System Applications, Shigeo Kaneda, IEEE Transactions on Computer, Vol. C-33, No. 8, August 1984.

[9] Board Arbiter (BARB), Data Sheet, Rev 2.0, Tom Holman and Jeff Hoel, May 1991.

[10] IEEE Standard Test Access Port and Boundary-Scan Architecture, P1149.1

[11] Memory Queue Handler, Data Sheet, Fred Cerauskis, Gil Chesley and Michel Cekleov, June 1989, Revised September 1991.

[12] XBus Specification, Rev. 2.7, Pradeep Sindhu, May 1989, Revised April 1991.

[13] SBus specification, Rev. B0, Sun Microsystems, 1990

[14] TMS29F816: 16,384-bit serial JTAG 5-V Flash EEPROM, Texas Instruments, 1989

[15] Bus Watcher Chip, Data Sheet, Bjorn Liencres, January 1992.

[16] Single In-line Memory Module Product Specification, 16 Meg DRAMs (4MegX4), Rev. 1.1, March 1991, Revised June 1992.

[17] Single In-line Memory Module Product Specification, Rev. 2.1, February 1990, Revised March 1992.

[18] SN74BCT8244: Scan test device with octal buffer, Texas Instruments, 1990

[19] Viking Cache Controller Specification, J.H. Chang, 1990

[20] I/O Cache, Data Sheet, Dave Basset and Bill Brougher, September 1989, Revised January 1992.

[21] SunDragon FCS Post Specification, Rev. 1.6, October 1991, Revised February 1992.