Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
FAX 415 969-9131

For U.S. Sales Office locations, call:
800 821-4643
In California:
800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Bagshot Manor, Green Lane
Bagshot, Surrey GU19 5NL
England
0276 51440
FAX 0276 51287

Australia: (02) 413 2666
Belgium: 32-2-759 5925
Canada: 416 477-6745
Finland: 358-0-5022700
France: (1) 30 67 50 00
Germany: (0)89-46 00 8-0
Hong Kong: 852 5-8651688
Italy: 039 60551
Japan: (03) 3221-7021
Korea: 2-563-8700
The Netherlands: 033 501234
New Zealand: (04) 499 2344
Nordic Countries: +46 (0)8 705 30 00
PRC: 1-8315568
Singapore: 224 3388
Spain: (91) 5551648
Switzerland: (1) 825 71 11
Taiwan: 2-7213257
UK: 0276 20444

Europe, Middle East, and Africa,
call European Headquarters:
0276 51440

Elsewhere in the world,
call Corporate Headquarters:
415 960-1300
Intercontinental Sales

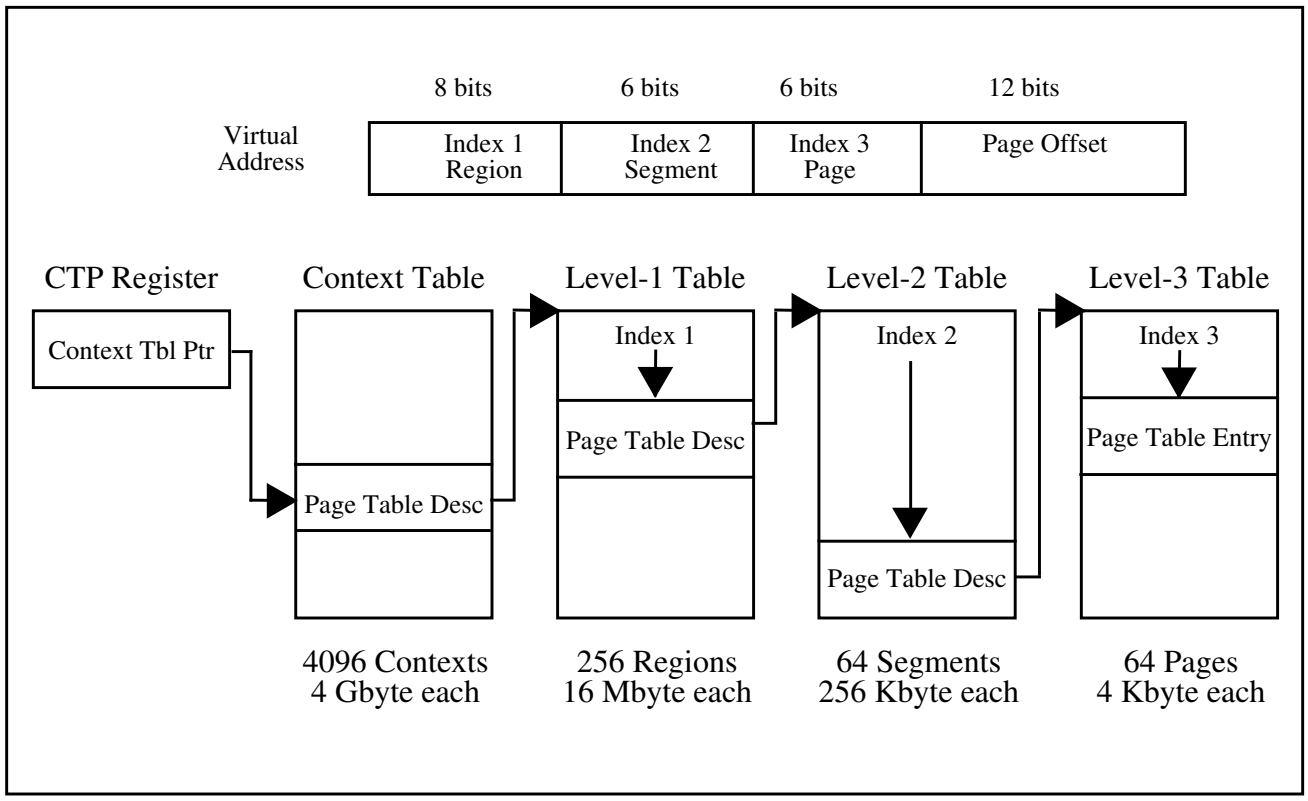

Figure B-1 SPARC Reference MMU Table Structure

SPARC Reference Memory Management Unit

SRMMU has four levels of tables, corresponding to *context*, *region*, *segment*, and *page* in the traditional Sun MMU design (see Figure B-1). Entries in these tables contain either a page table pointer (PTP) or a page table entry (PTE). A PTP points to the next level of table; a PTE contains a physical address, protection information, and other page status. A PTE can be found at any level in the translation process; if found at the *page* level it maps a 4KB page. A PTE found at the *segment*, *region*, or *context* level maps a page of 256KB, 16MB, or 4GB respectively. When servicing a TLB miss the MMU controller does a *table walk* to find the translation; if no valid translation is found then a page fault occurs.

Currently only 4 Kbyte pages are used by SunOS system software, but as the kernel is tuned it will be able to take advantage of larger page groupings (as segments or regions) to minimize TLB misses. Each processor operates in its own context, but the set of contexts is shared among all processors in the system.

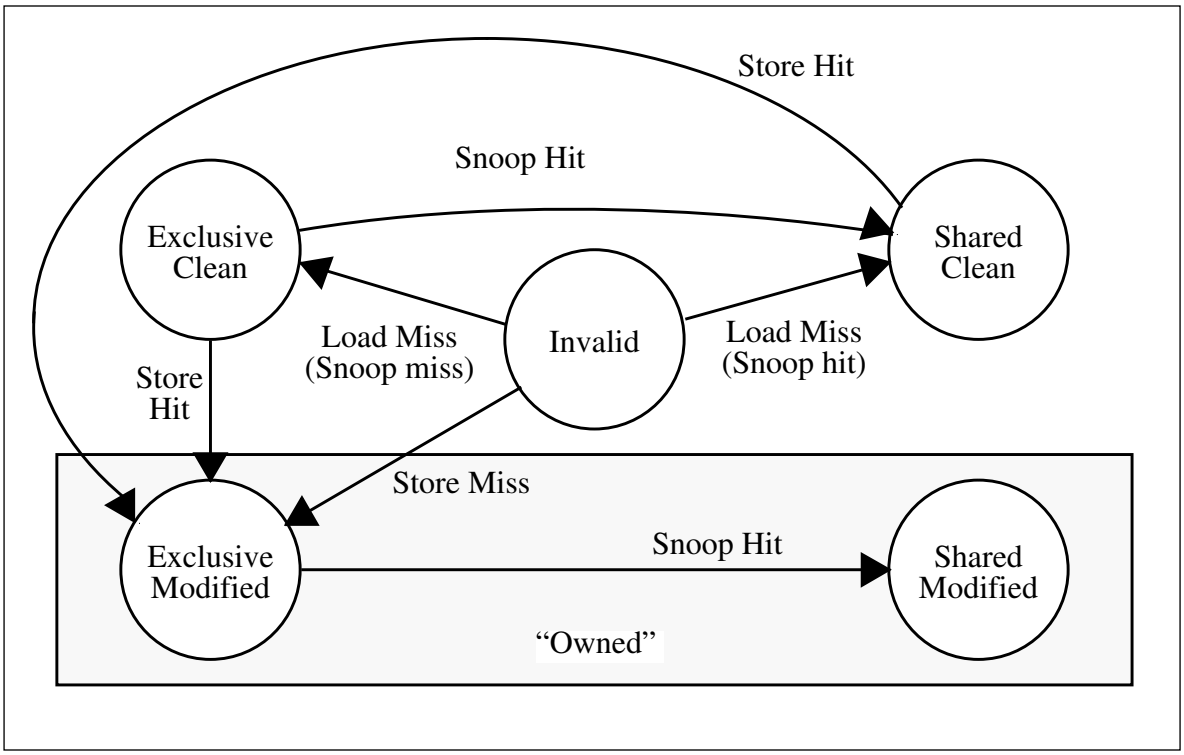

Figure A-1 Cache Coherency State Diagram

Simultaneous stores to the same block are sorted out by bus arbitration; a block may not be updated in a cache until that cache has exclusive ownership of that block. An atomic load-store sequence is treated in the same way as a store. All of this coherence protocol is transparent to software.

If more than one cache has a copy of the same block, that block is considered to be *shared*. If only one cache has a copy then the block is *exclusive*. In order to maintain a coherent memory image the various copies of a *shared* block in all of the caches must match at all times. In the general case of a multiple cache environment, when one processor/cache modifies a block it can either alert all other caches to update their copy (*write broadcast*) or to discard their stale copy (*write invalidate*). MBus implements a *write invalidate* protocol. Any time a write is done to a *shared* line the cache will issue a *coherent invalidate* transaction on the MBus. The other caches that have a copy of that block will discard it.

A line will be marked as *shared* in all caches that have a copy if it was provided by cache intervention. It will also be marked as shared if a snooping cache sees that it has a copy but is not the owner when another processor/cache allocates this block. In both cases the status is signaled to the receiving cache by all caches that have a copy.

Based on the above it can be seen that each cache line has 5 states to it; invalid, shared clean, exclusive clean, shared modified, and exclusive modified (see Figure A-1). The latter two are the states that indicate that this cache *owns* this block. Only one cache at any time will have ownership of a particular block. When a processor/cache wants to store to a line that is *invalid*, it will issue a *coherent read and invalidate* operation; this gets the most recent copy of this block from whoever owns it (either memory or some cache) and instructs all caches (including the supplier of the data) to discard their copy of the block. A write to a *shared* line will simply generate a *coherent invalidate* on the MBus. Obtaining a copy of a block and assigning it to a cache line is called *allocation* of the line. If the line was already in use for a different block then that block gets *displaced*.

The *owner* of a block is responsible for writing the valid information back to memory. This will occur either when a *modified* block is displaced (that is, when the cache allocates that line to a new block) or when the operating system explicitly flushes that cache line. This *copy-back* operation is done with a simple *write* on the MBus. A *coherent write and invalidate* is used only for 32-byte writes to cacheable space (i.e. from DVMA) or for coherent write-through caches, which Sun does not use.

Cache Coherency

MBus specifies two levels of operation. Level-1 MBus is a simple protocol that allows for the operations of *read* and *write* of sizes ranging from 1-128 bytes. Level-2 MBus adds four other transactions; *coherent read*, *coherent read and invalidate*, *coherent write and invalidate*, and *coherent invalidate*. With this simple set of transactions it is possible to implement caches that maintain a consistent (or *coherent*) image of memory at all times.

The basic notion that allows for coherence among multiple caches is that of cache block ownership. Copy-back caches are used in order to minimize bus traffic, so memory is allowed to contain *stale*, or out-of-date, data. If a line has been written to, the processor/cache that has the most recent copy is considered to be the *owner* of that block. This means that if any other processor/cache (or DVMA) wants to read that data, it must be supplied by the owner rather than by memory. A read of cacheable data is done with a *coherent read* operation. Whenever any coherent operation goes by on the bus, each participating cache monitors the transaction type and address. This is known as *snooping*. If a cache recognizes that it is the owner of the requested block (a *snoop hit*), it will abort the memory access and provide the up-to-date copy of the block instead. The term for this is *cache intervention*.

One complication in this scheme is having a virtual cache rather than a physical cache, since two different virtual addresses could map to the same physical address. In this processor implementation, all aliases map to the same cache line to ensure that the consistency scheme works properly.

Summary

The SPARCsystem 600MP series introduces new technologies for higher application performance while protecting customer investments in existing hardware and software. Multiprocessing, and easy upgrades to future SPARC CPU technology, provide scalability and growth for Sun's mid-range and high-end servers. Hardware features like the SPARC Reference MMU, the separate I/O MMU, Sun's I/O cache, and store buffers increase application throughput in a wide variety of server environments. With both SBus and VMEbus for I/O expansion, Sun customers benefit from new low-cost, high-performance SBus options while maintaining compatibility with previous investments in VME technology. And Sun's MP software strategy, with binary compatibility in SunOS 4.1.2 and sophisticated multithreading in SunOS 5.0, provides the right flexibility for protecting existing investments and taking advantage of innovation.

The new technology in SPARCsystem 600MP series products provides the flexibility, scalability, and growth to employ client-server solutions throughout the enterprise.

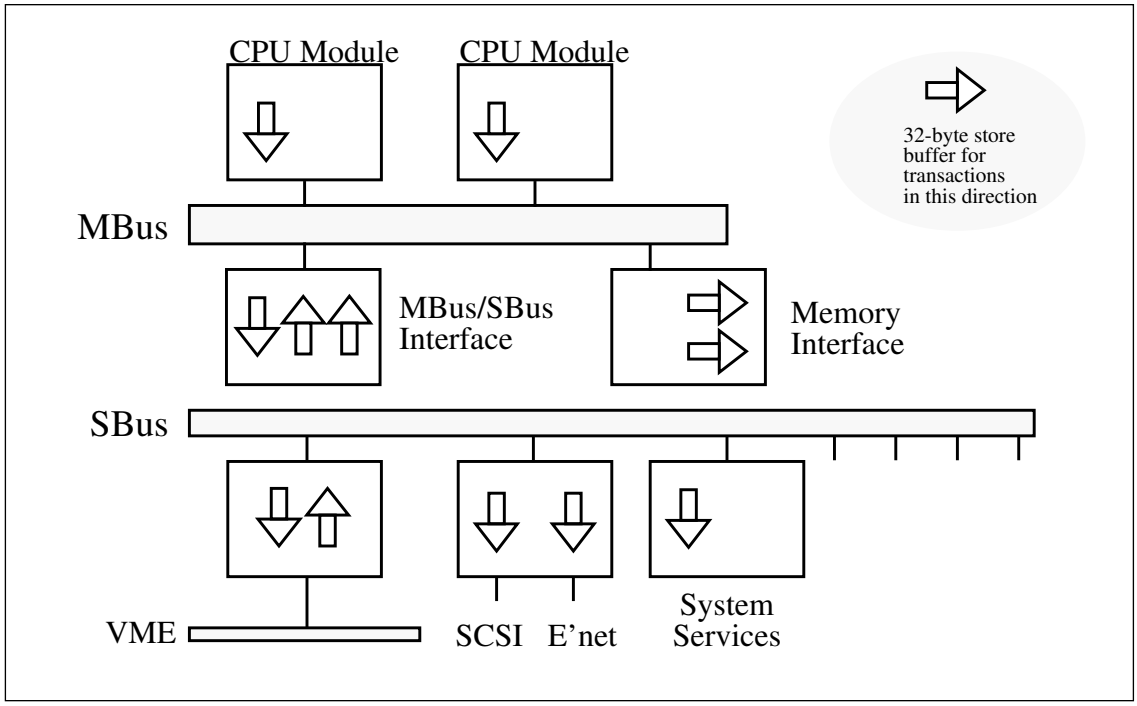

Figure 4-1 System Block Diagram with Store Buffers

4.4 Connectivity

Standard features on every 600MP system are an on-board Ethernet and SCSI channels and two synchronous serial ports. Optional features include additional buffered Ethernet and SCSI channels, additional serial and parallel ports, FDDI, and a wide range of IBM, DEC, and OSI connectivity products.

This wide range of options extends the configuration flexibility of the SPARCsystem 600MP series by supporting a variety of network topologies and fitting into environments with existing equipment.

writes into 32-byte packets, and doing 32-byte reads on behalf of multiple sequential VME reads. VMEbus activity through the IOC will provide a sustained 16 Mbyte/sec writes and 13 Mbyte/sec reads. Both numbers assume an ideal master.

4.3 *Optimizing Bus Transactions*

With many masters sharing the same buses, wait-states on a bus can degrade overall performance. The 600MP design minimizes bus occupancy through the use of store buffers (see Figure 4-1) and in some cases through split transactions. Store buffers can be found on each module. They are also provided in the memory controller (two 32-byte swing buffers) and between the MBus and the I/O section of the machine, as well as on the processor-to-VMEbus interface. DVMA stores to memory utilize dual swing buffers as well for stores to memory. The purpose of a store buffer is to accept a store operation, free up the master and the intervening bus, and then carry that operation to the next stage of bus or device. All store buffers in this design are strongly ordered and provide the capability for software synchronization if desired.

Performance Enhancements for Server Applications

A good server requires more than just lots of processing power: high I/O bandwidth and sufficient connectivity are key elements in server applications. The SPARCserver 600MP products were designed with careful attention to balancing speed and throughput.

4.1 SBus as Primary I/O Bus

An onboard SBus allows the user to take advantage of high-performance and low-cost peripherals from Sun and from third parties. There are three SBus slots, plus a fourth slot that can be used for a fourth SBus card in the absence of a second MBus module. With 32-byte activity the SBus provides sustained performance, assuming an ideal master, of 53 Mbyte per second writes to memory, and 30 Mbyte per second reads from memory.

4.2 VME and I/O Cache Improvements

VMEbus support provides a bridge to older configurations and to widely available VME peripherals. There are up to 11 VMEbus slots, depending on which model of the 600MP family you have. The VMEbus channel provides a 32 Kbyte I/O cache (IOC) to accelerate sequential activity between VMEbus masters and system memory. This IOC provides VME masters with an 8 Mbyte window of virtual address space with which to read and write memory, and enough buffers to support up to 1024 simultaneous transfers in progress. The I/O system is tuned to provide optimal performance for masters that do 32-byte burst accesses. IOC activity matches that burst size by buffering VME

3.2 Tools for Monitoring Multiprocessing

The SunOS shared memory multiprocessing model simplifies the job of administering and monitoring the SPARCsystem 600MP. Because one shared memory image of the SunOS kernel is running on all system CPUs, the theory of tuning SunOS system software for performance is unchanged from the SunOS 4.1 release and there are no new MP specific tuning operations.

All the standard SunOS 4.1 utilities and monitoring tools are included in SunOS 4.1.2 release on the SPARCsystem 600MP system. This reduces the learning time for administrators working with MP.

Two new utilities are provided in SunOS 4.1.2 system software. The `mps(1)` utility prints the status of every process and the last CPU on which it ran. The `mpstat(1)` utility prints the percent of CPU utilization for every configured CPU in the system.

SunOS 4.1.2 system software is binary compatible with all existing SunOS 4.X system software based applications. It is also binary compatible with all existing SunOS 4.1 software based device drivers which restrict themselves to published SunOS interfaces. Binary compatibility for existing applications and third-party peripherals further simplifies the task of integrating the SPARCsystem 600MP hardware product into production operations. Device drivers execute within a SunOS supplied wrapper routine which ensures correct execution without the need to modifying existing 4.1 based drivers.

In the future, SunOS 5.0 system product will provide the customers with the performance increases of fully symmetric multiprocessing in a standards based operating system—simply by upgrading software. SunOS 5.0 software-based releases will combine both an innovative multithreading architecture for advanced application development and standards POSIX P1003.4 multiprocessing libraries for application portability. Sun's future technology direction for providing a parallelized software development environment will be based on SunOS 5.0 system software. The SPARCsystem 600MP hardware was designed to efficiently support both SunOS 4.1.2 and 5.0 releases.

3.1 Scheduling

The scheduling of jobs is straight forward, using a small extension to the traditional SunOS timesharing scheduler. Scheduling is done in a manner which minimizes idle CPU cycles. SunOS maintains a single run queue for the entire system. The highest priority runnable process for the entire system is selected from the run queue and started on the available processor.

Processes are free to migrate from one CPU to another during execution. Because of the shared memory and cache coherency support in hardware, the cost of a granting a time quantum to a process on a new CPU is essentially the same as that of continuing it on the last CPU on which it ran. In this way, available CPU cycles can be used as processes become available.

The traditional UNIX utility, `nice(1)`, can increase the priority of processes to grant them additional CPU time.

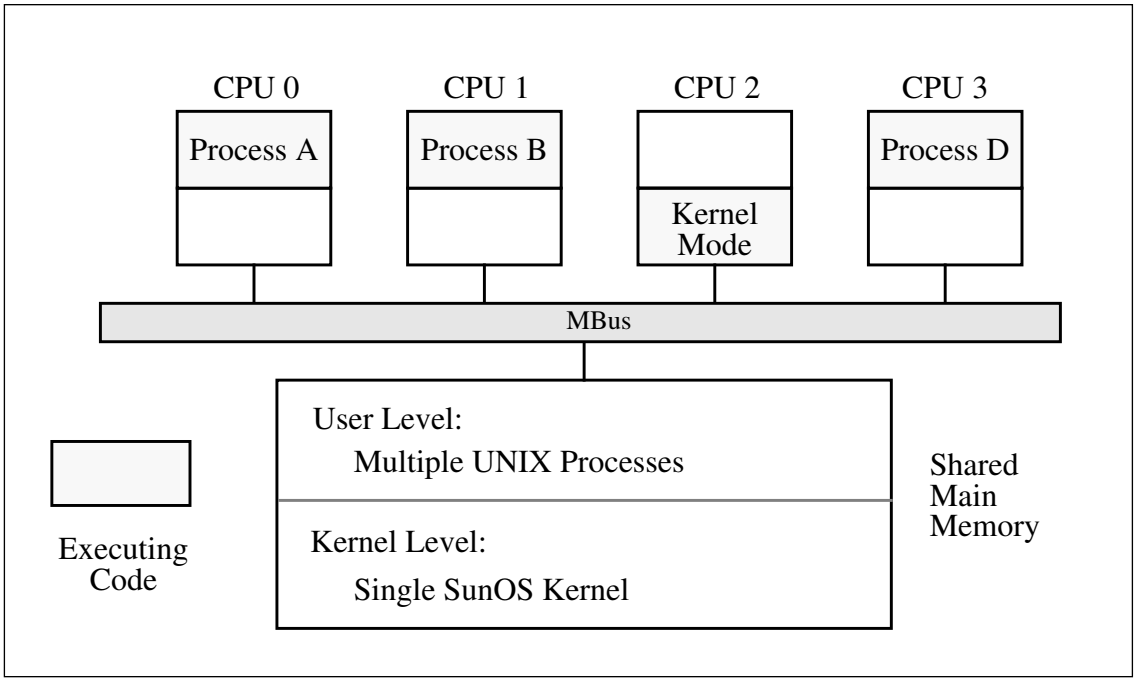

Figure 3-1 SunOS 4.1.2 Multiprocessing

Where multiple CPUs need to access a shared data structure in an orderly fashion, software primitives call *locks* are used to synchronize access to shared data, particularly the kernel data structures. These locks are often referred to as mutual exclusion locks, or as *mutexes*.

SunOS 4.1.2 system software uses a small number of kernel mutex locks for its operation; only one CPU at a time is executing kernel-level code. In contrast, SunOS 5.0 system software uses a larger number of locks and is able to support more threads of execution through the SunOS kernel. In particular, SunOS 5.0 system software offers better I/O and NFS performance than SunOS 4.1.2 system software because of the ability to execute multiple concurrent device drivers and NFS daemons.

Multiprocessing Software Implementation

The SunOS releases for the SPARCsystem 600MP series were designed to provide the simplest transition for customers moving to multiprocessor computers. The initial SunOS release for the SPARCsystem 600MP system product, SunOS release 4.1.2, provides binary compatibility for all SunOS 4.X software-based applications and a minimum of new administrative procedures. SPARCsystem 600MP systems can be integrated into existing networks in a few hours.

Looking ahead, SunOS 5.0 system software combines the benefits of symmetric multiprocessing with those of an operating system based on the industry standard: System V Release 4 (SVR4).

Figure 3-1 illustrates shared memory multiprocessing with SunOS 4.1.2 system software. One SunOS kernel image controls the entire system, with each CPU running a separate UNIX processes. All processes share a common pool of physical main memory managed by the SunOS virtual memory system. Individual CPUs are not generally visible to one another or to the processes executing on a different CPU.

Any process can execute on any CPU, and the SunOS kernel is free to execute on any CPU. Processes typically migrate from CPU to available CPU many times during their execution lifetimes. The single SunOS scheduler schedules all CPUs. For speed and flexibility, interrupts are mapped to and serviced by whichever CPU is currently executing kernel level code.

page. The IOMMU contains 16 TLB entries with least-recently-used (LRU) replacement. It provides up to 2 Gbyte of virtual address space to device drivers, extending the limit of 1MB that was imposed in the older, shared-MMU systems. DVMA through the IOMMU participates in system coherence with hardware support so no processor cache flushing is needed around I/O.

DVMA activity can come from VME and SBus devices (including on-board SBus devices such as SCSI and Ethernet). A DVMA address can come from an SBus master or a VMEbus master. DVMA can access main memory or any slave devices on the SBus. The only limitation is that DVMA cannot access devices on the VMEbus (although VMEbus masters can access VMEbus slaves since the CPU does not participate in those accesses). See Figure 2-3 for the different memory access modes supported.

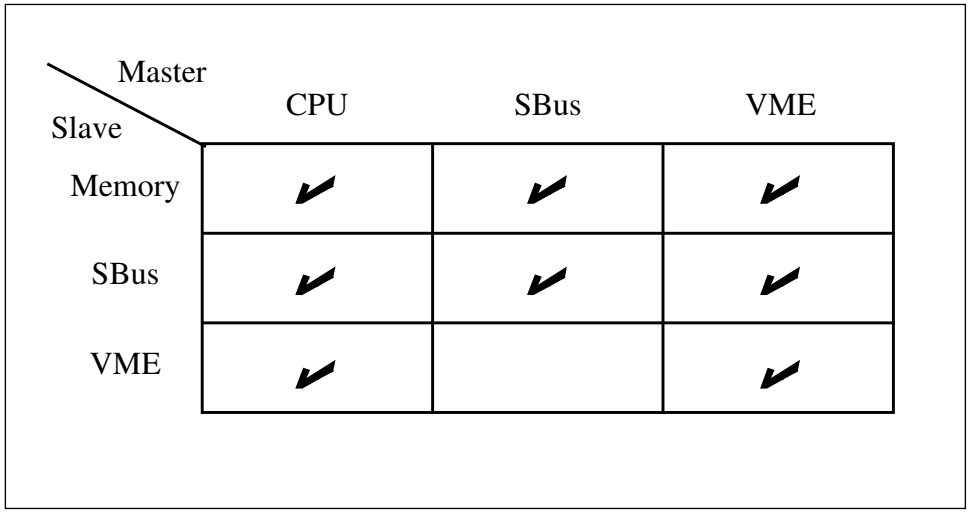

Figure 2-3 Memory Access Modes Supported

Interrupts from I/O devices can be steered to any processor in the system, meaning that the same hardware can support either a symmetric or asymmetric multiprocessing operating system.

CPU-to-memory transactions occur on the MBus, it's much less expensive and complex than the specialized synchronization buses used on older MP implementations. Appendix 'A' covers MBus cache coherency in greater detail.

2.3 Memory Management & DVMA in an MP Architecture

With multiple processors in a system, some system facilities can be shared while others need to be replicated per processor. An example of replication is the memory management unit (MMU). Older Sun systems contained a single MMU consisting of several levels of static RAM tables indexed by a context number and by some bits of the virtual address. This older MMU was shared between the processor and DVMA.

Multiprocessor servers place different requirements on memory management. It is desirable to have many more contexts than the traditional 8-64 supported in previous designs. It is also desirable to keep more pages mapped at any time in order to reduce the frequency of page faults. Lastly it is important that the MMU is close to the processor and cache for speed reasons; this implies that the MMUs must be on the module, so they must be physically small and lower in cost than an array of fast, expensive static RAM chips.

These needs are met through use of the SPARC Reference MMU (SRMMU). SRMMU is based on tables in memory that contain the multiple-level mappings. Each cache/MMU controller contains a context register and a base address register that points to the first level of tables in memory. Also integrated is a set of Translation Lookaside Buffers (TLBs) that contain the set of recently used translations. TLB misses are serviced by hardware, meaning that fewer traps into the operating system are needed to manage mappings once a page has been faulted in by the memory management code. Stale mappings get purged from the TLBs through use of MMU *flush* commands from the operating system.

The SPARCsystem 600MP MMU has 4096 contexts and 64 TLBs with a random replacement algorithm for each processor. See Appendix 'B' for a more detailed description of its organization.

The SRMMU in each processor is not accessible from DVMA, so another memory management unit is implemented to translate DVMA addresses to system physical addresses. This unit is the I/O Memory Management Unit (IOMMU). The IOMMU is very similar to an SRMMU, except that there are no contexts, only one level of table, and no "referenced" or "modified" status per

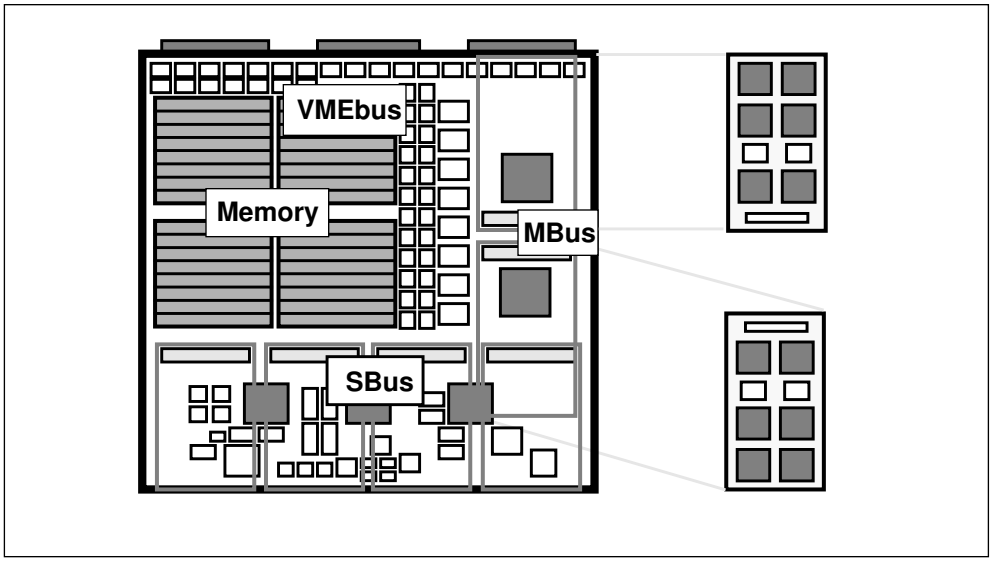

Figure 2-2 SPARCsystem 600MP System Board and SPARC Modules

2.2 Cache Coherency

When there are multiple local caches in a multiprocessing system, some mechanism must be included to ensure that local copies of data remain consistent. In order to maintain this data consistency, all caches in the system participate in a multiprocessor *cache coherence* (consistency) protocol. The SPARC MBus provides transactions that make it possible to implement such coherence in a simple way. DVMA is also able to participate in the coherence protocols, eliminating the need for processor cache flushing around I/O operations.

Because cache coherency is supported both in the MBus specification and in the cache control logic, there is little overhead and little impact on the performance of individual CPUs. With caches snooping the MBus for coherent transactions, and intervening to maintain consistency, copy-back (or write-back) operation is possible. The copy-back cache only issues stores to memory when flushing or reallocating a modified cache line, reducing MBus occupancy significantly compared to a write-through cache. Since all CPU-to-CPU and

2.1 Modular CPU Design

With processing power increasing at a rapid pace, one of Sun's primary implementation goals for its future systems is to separate CPU technology from the remainder of the product design. In the SPARCsystem 600MP design, the standard MBus connector provides the interface between two separate boards. The main system functions and I/O are on the system board, while CPUs are located on modular daughtercards, called SPARC modules. Providing a standard interface between CPUs and the rest of the system allows modular upgradability to future processor implementations. In the SPARCsystem 600MP design, it also provides a convenient way to add more processors to a basic system, providing inexpensive scalability to increase system performance as application needs grow. Adding or changing processors is as easy as installing a SPARC module. As future CPUs become available, customers can swap SPARC modules to achieve higher performance. With a stable system platform, both Sun and the customer benefit from a much faster development process and improved time-to-market.

With the flexibility to choose the number of CPUs, and scalability to add power with more or faster CPUs, the SPARCsystem 600MP offers the industry's best growth path and investment protection.

The MBus specification for this standard CPU interconnect is publicly available through SPARC International. It is widely available to the SPARC community of both component and system developers. With a potential product base larger than just Sun, CPU suppliers have greater incentive to develop new SPARC modules for the MBus. Also, they can spread the development costs over a larger number of units, with both Sun and its customers benefitting from lower product costs and quicker time-to-market.

SPARC modules are the size of an SBus card, and attach to the MBus connectors on the system board in very much the same way as SBus cards. A processor core consists of a CPU, an FPU, a memory management unit/cache controller (MMU/CC), and local cache. In its first implementation, the SPARCsystem 600MP SPARC module has two processor cores (each with a 64KB cache) and a single system board supports one or two modules for two- or four-way multiprocessing. See Figure 2-2.

Multiprocessing Hardware Implementation

The key to Sun's multiprocessing picture is the tightly-coupled, shared-memory model. See Figure 2-1. Processors are tightly coupled by a high-speed system bus, called the MBus, and share the same image of memory, although each processor has a cache where recently accessed data and instructions are stored. A single copy of the operating system coordinates the activities of all processors, taking care of scheduling jobs and coordinating access to common resources. All CPUs have equal access to system services, such as I/O and networking, unlike some designs where specific resources are attached to a specific CPU. The overall objective is a general-purpose system with sufficient compute and I/O resources to improve performance for a wide range of applications.

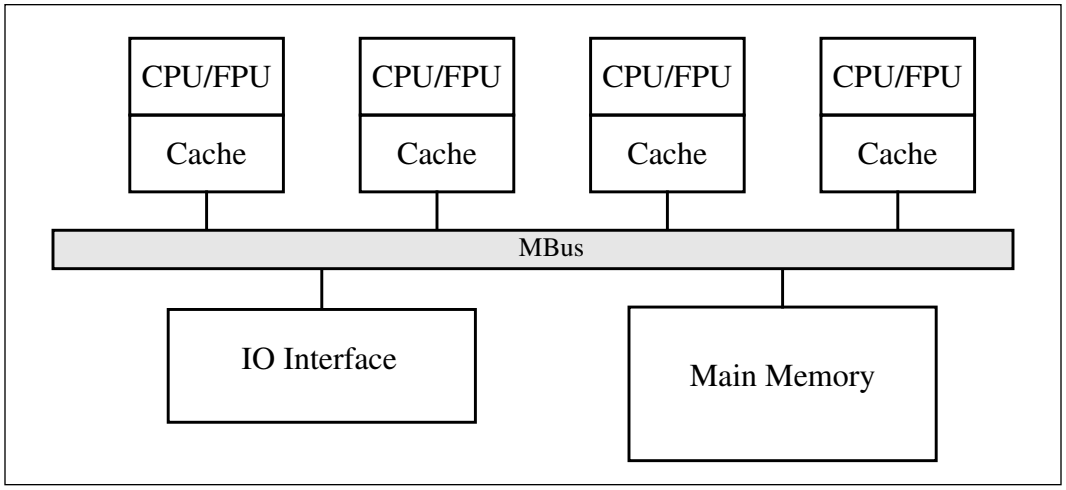

Figure 2-1 SPARCsystem 600MP Shared Memory Model

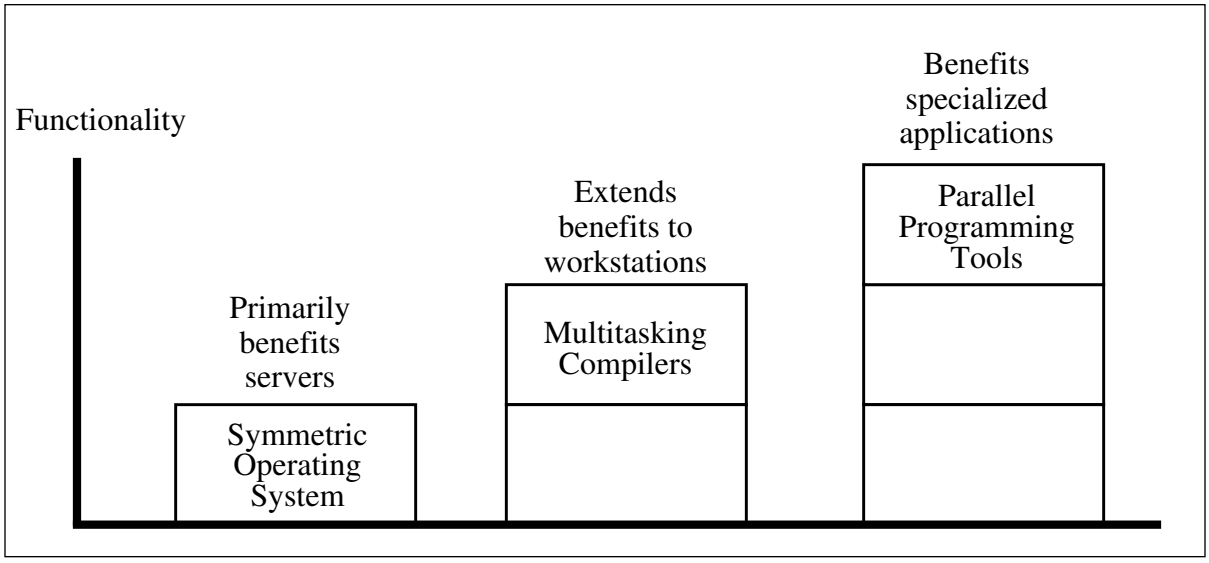

Figure 1-1 Sun's Multiprocessing Strategy

computing environment and larger technical departments, characterized by many users and large client/server networks. Extending the product line with MP systems is important for providing the flexible range of solutions and configurations customers are seeking to replace existing large, proprietary, centralized equipment. Tighter integration of CPU and memory technologies makes MP implementations more affordable.

Sun's long term objectives for multiprocessing directly focus on improving application performance for customers (see Figure 1-1). Sun's first MP implementation provides SunOS 4.1 binary compatibility with existing applications, so every MP system running multiple tasks can realize the immediate benefits of finishing more work in a given period of time. SunOS™ 5.0 system software, Sun's SVR4-based operating system, will improve the underlying implementation by offering a multithreaded kernel to increase not only user tasks but also system tasks. SunOS 5.0 system software will provide standard interfaces for MP applications, so that software developers can achieve portability.

As MP product costs come down through use of modular CPU designs, benefits will extend from multi-job servers to single-job workstations, and focus on reducing processing time for individual applications. Lower costs also mean increased volumes, which will motivate ISVs to develop or modify applications specifically to exploit MP.

Sun is also working on a full software development environment for parallel programming, with multitasking compilers to automatically paralyze existing source code, and parallel libraries and tools to support more sophisticated optimizations. The net result: more applications using affordable multiprocessing to support larger problems and handle more users to deliver solutions faster. Faster solutions are a competitive edge in any business.

Why Multiprocessing?

Sun's hallmark has been its commitment to high-performance systems, a recent example being its innovation of the SPARC RISC architecture. With SPARCs scalable architecture, customers have seen processing power increase with each new implementation, and future designs will continue to provide higher performance. Another way to increase available processing power is to allow more than one CPU per system, an approach called multiprocessing (MP). Multiprocessing has several benefits:

- Independent tasks can be handled by separate processors, increasing job throughput, the number of transactions, and/or the number of simultaneous users.
- Combining CPUs in one system lowers the overall cost of computing by sharing system resources like memory, disk, and network interfaces.
- Providing a high-speed interconnect between these multiple CPUs also provides the ability to achieve better coordination and faster interaction between related tasks.
- Depending on the type of application and tools available, a single large job can be decomposed into several smaller tasks that can run simultaneously for faster application time-to-completion.

Sun is introducing multiprocessing products now for a few important reasons. Multiprocessing technology is starting to move out of specialized high-end applications. This is indicated by two recent trends: the inclusion of MP in UNIX® standards, and the development of UNIX database products that take advantage of MP. Sun's products are increasingly being used in the commercial

Preface

SPARCsystem 600MP

Introduction

The SPARCsystem™ 600MP product family extends Sun's product line with new flexibility, scalability, and expandability. Multiprocessing is one of the key technologies introduced to provide customers with high application performance, modular upgradability, and a growth path to the future, without compromising SPARC™ binary compatibility for thousands of existing applications. This paper examines how Sun implemented multiprocessing and how it affects customers from the end user, system administrator, and software developer points of view. It also covers a few other features of the SPARCsystem 600MP series designed to optimize performance for real-world applications.

4.4 Connectivity 4-3

5. Summary 5-1

A. Cache Coherency A-1

B. SPARC Reference Memory Management Unit B-1

Contents

Preface v

1. Why Multiprocessing? 1-1

2. Multiprocessing Hardware Implementation 2-1

2.1 Modular CPU Design 2-2

2.2 Cache Coherency 2-3

2.3 Memory Management & DVMA in an MP Architecture 2-4

3. Multiprocessing Software Implementation 3-1

3.1 Scheduling 3-3

3.2 Tools for Monitoring Multiprocessing 3-4

4. Performance Enhancements for Server Applications 4-1

4.1 SBus as Primary I/O Bus 4-1

4.2 VME and I/O Cache Improvements 4-1

4.3 Optimizing Bus Transactions 4-2

© 1991 by Sun Microsystems, Inc.—Printed in USA.
2550 Garcia Avenue, Mountain View, California 94043-1100

All rights reserved. No part of this work covered by copyright may be reproduced in any form or by any means—graphic, electronic or mechanical, including photocopying, recording, taping, or storage in an information retrieval system— without prior written permission of the copyright owner.

The OPEN LOOK and the Sun Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 (October 1988) and FAR 52.227-19 (June 1987).

The product described in this manual may be protected by one or more U.S. patents, foreign patents, and/or pending applications.

TRADEMARKS

The Sun logo, Sun Microsystems, Sun Workstation, NeWS, and SunLink are registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Sun, Sun-2, Sun-3, Sun-4, Sun386*i*, SunCD, SunInstall, SunOS, SunView, NFS, and OpenWindows are trademarks of Sun Microsystems, Inc.

UNIX and OPEN LOOK are registered trademarks of UNIX System Laboratories, Inc.

PostScript is a registered trademark of Adobe Systems Incorporated. Adobe also owns copyrights related to the PostScript language and the PostScript interpreter. The trademark PostScript is used herein only to refer to material supplied by Adobe or to programs written in the PostScript language as defined by Adobe.

X Window System is a product of the Massachusetts Institute of Technology.

SPARC is a registered trademark of SPARC International, Inc. Products bearing the SPARC trademark are based on an architecture developed by Sun Microsystems, Inc. SPARCstation is a trademark of SPARC International, Inc., licensed exclusively to Sun Microsystems, Inc. SPARCsystem is a trademark of SPARC International Inc., used under license by Sun Microsystems, Inc.

All other products or services mentioned in this document are identified by the trademarks, service marks, or product names as designated by the companies who market those products. Inquiries concerning such trademarks should be made directly to those companies.

SPARCsystem™ 600MP New Technology for Flexibility, Scalability, and Growth

Technical White Paper

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 8xx-xxxx-xx
Revision X, September 1991