*Sun Microsystems*
*Computer Corporation*

A Sun Microsystems, Inc. Business


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

Australia: (02) 413 2666
Belgium: +32 2 759 38 11
Canada: 416 477-6745
Finland: +358-0-502 27 00
France: (1) 30 67 50 00
Germany: (0) 89-46 00 8-0

Hong Kong: 852 802 4188
Italy: 039 60551
Japan: (03) 3221-7021
Korea: 822-563-8700
Latin America: 415 688-9464
The Netherlands: 033 501234

New Zealand: (04) 499 2344
Nordic Countries: +46 (0) 8 623 90 00
PRC: 861-831-5568
Singapore: 224 3388
Spain: (91) 5551648
Switzerland: (01) 825 71 11

Taiwan: 2-514-0567
UK: 0276 20444
Elsewhere in the world, call
Corporate Headquarters:
415 960-1300
Intercontinental Sales:
415 688-9000



Printed in USA

4/92

# 4

## Conclusion

The new SPARC server architecture extends the commodity workstation production process and pricing model up into the high end server marketplace for the first time. The systems provide substantially higher throughput at a substantially lower cost than competitive high end server systems, using only two extensible packages to deliver a low cost growth path to the customer.

In conjunction with the Solaris operating system, these systems are designed to meet both the high capacity I/O and computationally intensive demands of a very wide range of applications. The SPARCcenter 2000 is one of the highest capacity Unix based servers available, and has a mid-range server price. The SPARCserver 1000 is in a class of its own for performance and expandability in a compact package for the office environment.

that controls access to the SCSI disk device driver can be split up so that there is a lock for each separate SCSI bus in the system, or for each disk in the system. In this way multiple CPUs can be active concurrently.

Solaris 2.1 was developed and tuned for machines with one, two or four processors, and it scaled well over that range. Solaris 2.2 has been developed and tuned for machines with up to eight processors, and achieves excellent scalability for a wide range of workloads over the full configuration range of the SPARCserver 1000. Work is now in progress to extend the tuning up to the full twenty processor configuration of the SPARCcenter 2000.

The primary reason the full SPARCcenter 2000 configuration is not supported in thee first release is that the number of possible configurations of a machine that has 40 I/O slots and between 2 and 20 CPUs is immense. To reduce the test matrix to manageable proportions, given the resources and the time available, a subset of the full configuration was tested. The test matrix will cover the full configuration in the next release.

## *Solaris 2 Release Process*

Solaris 2 now has a fixed release schedule which issues new versions twice a year. This enables rapid introduction of performance enhancements and new functionality. Since the base operating system conforms strictly to interface standards such as POSIX, XPG3 and the System V Interface Definition, application code from one release will run unchanged on subsequent releases.

at a time. Another application might have multiple threads and multiple LWPs because it needs to have several outstanding system calls or wants to execute on multiple CPUs simultaneously.

User mode threads share a common address space. A change in shared data by one thread is seen by the other threads in the process. The thread library provides a variety of synchronization facilities to allow threads to cooperate in accessing shared data. These facilities include mutual exclusion (mutex) locks, condition variables and semaphores.

Threads are the primary interface for application parallelism. Few multi-threaded applications explicitly use the LWP interface. Most programmers can program using the threads interface and let the programming library take care of mapping threads to the kernel primitives underneath. Multithreaded application programs can be written without regard to the number of CPUs configured on the target machine, as the thread library can detect that there are insufficient LWPs to run the program efficiently and can automatically start more LWPs, and each LWP can be scheduled to run on a separate CPU.

Creating, tracking and providing support for evolving industry standards has always been a key goal of Sun Microsystems. Providing both thread and LWP interfaces eases the implementation of standards. The POSIX P1003.4a Pthreads standard will be incorporated at the user library level rather than in the kernel. In this way, SunSoft can quickly respond to the evolving standards. The Solaris thread interface is a superset of the POSIX 1003.4a standard, and this superset has been adopted by Unix International as the standard thread interface for the multiprocessor extensions to Unix System V Release 4

## *Real Time Scheduler*

In the traditional Unix timesharing scheduler, all processes were executed using a round-robin scheduling algorithm based on priority. Solaris 2 allows multiple, customizable scheduling algorithms to be loaded, and, apart from the usual time-sharing scheduler a real-time scheduler is provided as standard. This can be used to improve performance of applications that rely on a central server process, for example database back-ends. Real-time scheduling improves performance by allowing processes to run at a priority above incidental activities as well as reducing involuntary context-switching. In addition, it is possible to lock a process to processor to minimize potential context switch time.

## *Performance Tuning and Multiprocessor Scalability*

As the number of CPUs increases the kernel can run into bottlenecks as multiple CPUs contend for access to a shared resource, such as a key data structure or device. As part of the on going tuning of Solaris 2 locks that cause contention are split up. For example a lock

# Solaris Operating System 4

## Solaris Features

While the SPARCcenter and SPARCserver systems offer many benefits, such as power, scalability, extensibility, it is the SunSoft Solaris operating system that enables these benefits. Solaris has been enhanced and highly optimized for the SPARCcenter and SPARCserver multiprocessor systems.

SunSoft's Solaris2.2 has a fully pre-emptible, real-time, symmetric multiprocessing (SMP) kernel that supports multiple threads of control within a the kernel as well as within a single application process. The kernel runs equally on all processors, in the tightly-coupled shared memory multiprocessor model. In addition, the kernels concurrent usage locks have been tuned and replicated so that the system can efficiently use multiple processors.

### Multiple Processors and Multiple Threads

Multi-threading is a technique that applications can use to decompose their task to take maximum advantage of multiple processors. Solaris supports threads at three levels, the kernel uses a thread to handle each system call and interrupt, so that multiple CPUs can accelerate kernel tasks; the user code to kernel interface uses multiple light weight processes (LWPs) so that a single user process can have multiple outstanding system calls and can be scheduled onto multiple CPUs; and a purely user mode thread library allows very lightweight thread switching and large numbers of threads, using LWPs as virtual processors to run the threads. Solaris allows an application to control how many threads and LWPs it needs to maximize its performance and minimize the kernel overhead. For example, one application might have multiple threads all using the same LWP. This is the way most existing thread libraries work on conventional uniprocessor Unix implementations and only one thread runs

miss count, number of shared writes, lock requests and number of different kinds of reads (ex: stream, PIO). The information provided by the monitoring hardware is sufficiently detailed to permit problem isolation and fine tuning of the operating system and applications.

The systems have hardware monitoring and environmental sensors provided to protect the system from hazardous conditions. Environmental sensors monitor the systems' temperature, fan operation and protect the system by informing the operating system when recommended operating conditions are exceeded.

SuperCache cache controller. The systems are expanded by adding system boards and populating functional units until the required capacity is achieved. As all units are equally accessible, the specific location of memory banks, NVRAM SIMM banks, and processors is not fixed. Automatic configuration ensures that a fully coherent system is booted.

## *Jumper-less Geographical Installation*

Because the XDBus is implemented with programmable device addressing, system boards may be plugged anywhere into the multi-slot XDBus backplane without physical re-configuration. There are no jumpers or switch settings associated with any configuration. This permits easier maintenance and reduces the likelihood of installation error.

## *Power and Packaging*

All systems have a resilient power supply that is impervious to power fluctuations, brown outs and even entire line dropouts. Furthermore, all systems use low voltage swing (0.4V to 1.2V signal range), low impedance Gunning Transceiver Logic (GTL). This logic is a significant improvement over the usual TTL logic, by providing faster switching times and eliminating dead time between cycles. As a result, the systems enjoy most of the benefits of ECL's high performance with the low power consumption increased integration and higher reliability of CMOS.

## *Fault Resilience And Performance Monitoring*

All storage systems are protected with either ECC or parity. Failing components can be re-configured by the boot diagnostics or by Solaris. In addition, Solaris maintains error logs to make diagnosis and repair simple. Also the Solaris on-line diagnostics, *prtdiag*, allows for diagnosing many problems while the system remains operational. *Prtdiag* identifies the appropriate field replaceable unit (FRU).

Perceiving the need to provide practical tools for performance tuning of the operating system as well as user-level multi-threaded programs, these systems were designed with programmable hardware event counters built into several of its subsystems. Event counters come with every machine and less intrusive than a hardware trace collection. These counters can be accessed through the built-in scan interface rather than having to use external testing equipment such as oscilloscopes or custom monitoring boards. The counters can be used to monitor the effectiveness and performance of many components. Counters are collected by each of the cache, bus watcher, memory queue handler and I/O cache, for example, the cache

# *Hardware Packaging Details* 3

The traditional approach to building multiprocessor servers involves a range of boards, all connected to a common backplane. There would be separate boards for CPU, memory and I/O Bus connectors and there would be a trade-off between the number of slots dedicated to CPU, memory and I/O such that the maximum configuration could not be physically configured in all dimensions simultaneously. In the new SPARC server architecture these components are sufficiently integrated that a combination can be put on one board, and each time the system needs to be expanded a single new system board provides more CPU, memory, and I/O. This greatly reduces inventories in both manufacturing and service organizations, leading to further cost savings. In addition to the system board there is a control board built into each system.

## *Control Board*

Each system has exactly one control board. It contains the central arbitration logic for the XDBuses, as well as all of the system-level logic -- all of the non-replicated system components. This includes the system clock generator and the IDPROM. The control board is physically mounted to the other side of the XDBus backplane from the system boards.

## *System Board*

The system board is the primary component of the implementation. A SPARCcenter2000 may contain from one to ten system boards; a SPARCserver 1000 may contain from one to four system boards. Each system board contains two processor units, a memory unit, an I/O unit, and an XDBus interface. In addition, the system board contains up to two daughter-cards, called the SPARCmodule, that contains the SuperSPARC processor and the

## *SBus Interrupt Processing*

Each SBus is provided with an interrupt target register that indicates which of a number of processors is to handle interrupts for the SBus. Each SBus may make arrangements for interrupt processing independently -- at any moment, any set of SBuses may be configured to send interrupts to arbitrary sets of processors. Groups of SBus target registers can also be updated in a single operation if this is desired. Normally this is the approach taken by the operating system. The interrupt target registers could also be configured to shield one or more processors from SBus interrupts. This might be useful in a real-time environment.

invalidate read buffers at the beginning of a DVMA read and flush write buffers at the end of a DVMA write. Stream mode is provided because full bandwidth can be achieved and sustained while the SBI buffers transactions into and out of the SBus. Unlike peripherals operating in consistent mode, each slot operating in stream DVMA mode may have more than one pending transaction. For example, one XDBus may be transferring into one of the stream buffers while the SBI is transferring the other buffer to the requesting device. In stream mode, each SBus can sustain 55 MB/sec when writing and 49 MB/sec when reading, using 64-byte bursts. Burst transfer speed is 80 MB/sec.

## *I/O Cache*

SBus transactions can be many different sizes, 1, 2, 4, 8, 16, 32 or 64 bytes. Some other systems do not implement all transfer sizes on the SBus, and many SBus cards do not implement all transfer sizes. The SBus protocol negotiates the largest possible transfer size for each transaction, and may break up large transfers into several smaller ones if necessary. The XDBus only performs 64 byte transfers so an I/O cache is used to match up the transfers. A 64 byte block comes over the XDBus into the I/O cache and can be written on the SBus in several transactions. If an SBus card is performing DVMA it may write small SBus transactions into the I/O cache and when the cache is full a single 64 byte XDbus transfer occurs. One 64 byte buffer per SBus slot is provided in each I/O cache ASIC, so the SPARCcenter 2000 has a separate cache for each XDBus. The I/O cache is kept coherent with main memory in the same way as the CPU cache, by bus snooping logic.

## *I/O MMU*

The SBus operates in virtual memory space, meaning that the SBus accepts virtual rather than physical addresses for main memory. Each SBus has a dedicated I/O memory management unit (MMU) for virtual address translation. The operating system is responsible for maintaining consistency between the processor’s MMU(s) and the other SBus I/O MMUs.

The I/O MMU provides each SBus with a 64 MB DVMA address space. The large DVMA address space provides the operating system great flexibility in choosing I/O buffers -- thereby simplifying the processing of I/O. Additionally, because the SBuses are completely independent, their DVMA spaces may be configured separately, potentially using over 640 MB of DVMA space in the SPARCcenter 2000 system and 256 MB for the SPARCserver 1000.

## Standard SBus Interface

The SBus complies with SBus Specification Revision B.0, including all burst sizes from 4 bytes to 64 bytes and parity support. The SBus operates at 20 MHz, independent of the system clock, so that the XDBus frequency can be increased without affecting the SBus. Parity support can be enabled on a per-slot basis, permitting the use of peripherals that do not support the parity extension. However all Sun peripherals do currently support parity, including the DSBE/S Differential Fast SCSI/Buffered Ethernet board (the primary disk and Ethernet interface on the SPARCserver 2000), the FSBE/S (used to interface tapes and CD on the SPARCcenter 2000 and the main interface on the SPARCserver 1000) and the FDDI/S network interface board.

## SBus Transfer Modes

The SBus implementation provides three transfer modes:

- programmed I/O,
- consistent mode direct virtual memory access (DVMA)
- stream mode DVMA.

While programmed I/O requires direct CPU intervention, DVMA permits large amounts of data to be transferred between the SBus and memory without using the processor. The SBI permits selection of transfer mode on a per-page basis. This permits a driver to use the transfer mode most appropriate to each task. For example, the driver for the differential SCSI host adapter uses programmed I/O for access to control registers but uses stream mode DVMA to transfer the large blocks of user data to and from the disk drives.

In Programmed I/O (PIO) mode, transfers are performed directly by the processor in physical address space and are not cacheable in either the processor's cache or in the I/O cache associated with the target SBus. This mode is provided for convenient access to memory-mapped I/O control registers and for transfer of small blocks of data.

Both DVMA modes use the I/O cache to achieve maximum performance. The two differ primarily in the way they present the memory model to the SBus. In consistent mode, all stores issued by an SBus board are guaranteed to be observed in issuing order by all processors. The practical impact is that the SBus permits only one pending transaction between each SBus board and system memory.

Stream mode DVMA uses buffers resident in the SBus interface chip. Each slot has its own buffers, used in a double-buffering arrangement. These buffers are not part of the system's shared memory image and therefore are not kept consistent by the XDbuses cache coherency mechanism. As a result, consistency must be maintained by the device driver that must

# I/O Subsystem

Each system supports multiple industry standard SBus I/O subsystems conforming to IEEE P1496 and SBus revision B.0. Each SBus delivers over 50 MB/sec sustained throughput and on a fully configured SPARCcenter 2000 the total SBus capacity (10 times 50 MB/sec) is matched to the internal XDBus capacity (500 MB/sec). All peripheral devices are connected to the SBus. There is an I/O unit (IOC) that provides buffering or a bridge between the SBus, its peripherals and the XDBus. The IOC is composed of cache chips and page tables. See Figure 2-3.The SPARCserver 1000 only uses one IO Cache ASIC as it only has to connect to a single XDBus.

In the case of the SPARCcenter 2000, each system board has a SBus with four SBus slots. With a maximum of ten system boards in one system, I/O capacity can be distributed over ten SBuses, with forty ethernets and forty SCSI buses. Each SCSI bus runs at 10 MB/sec and each ethernet runs at just over 1 MB/sec so the maximum aggregate throughput during DMA I/O transfers is approximately 450 MB/sec.

In the SPARCserver 1000, each system board has a SBus with three free SBus slots and one built-in FSBE/S combined fast SCSI and ethernet board. Therefore, with a maximum of four system boards per system, I/O capacity can be distributed over four SBuses, sixteen ethernets and sixteen SCSI buses, with a maximum aggregate throughput during DMA I/O transfers of approximately 180 MB/sec.

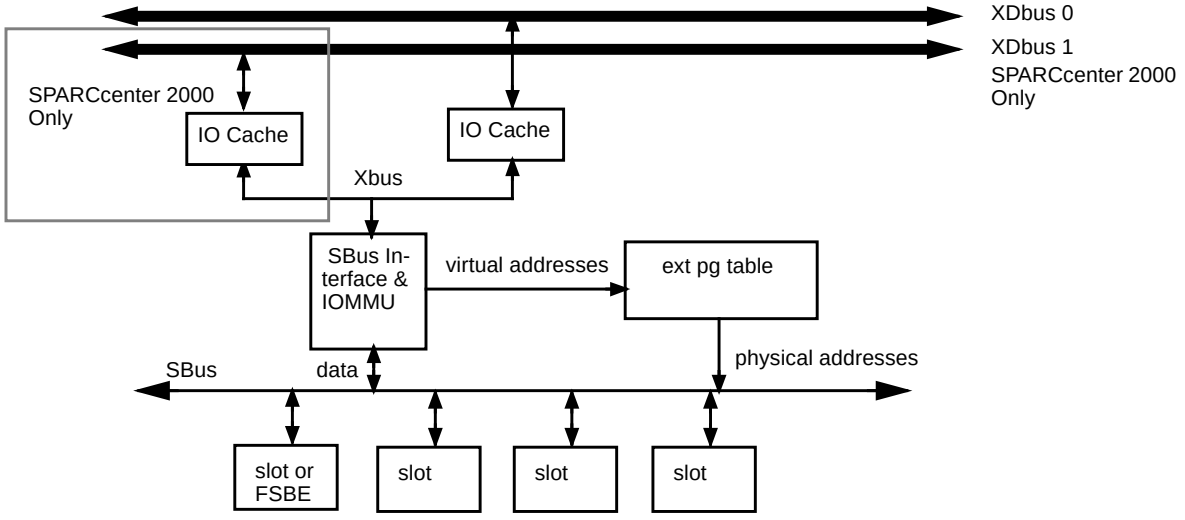


*Figure 2-3* SBus I/O Unit Structure

Two types of memory SIMMs are supported: standard DRAMs and battery-backed NVRAMs. Both DRAM and NVRAM are implemented on the same SIMM form factor. However, they cannot be mixed within a single bank. The NVRAM SIMM consists of an array of 1 Mbit SRAMs and a small lithium battery.

SIMMs must be configured in groups of four on the SPARCserver 1000 and eight on the SPARCcenter 2000 (one group of four in each bank is required to maintain the XDBus interleave). The batteries of each group of NVRAM SIMMs are wired in parallel. There is sufficient power in each group available to guarantee that no data will be lost even if one battery fails while the system is not in operation. Each NVRAM SIMM provides a battery status indicator. In the event of a battery failure during system operation, the operating system can force an immediate update of the disks and reconfigure the marginal NVRAM group out of the system.

A single XDBus interleaves addresses between memory banks on 64 byte boundaries so the maximum interleave on the SPARCserver 1000 is four way. The XDBuses used in the SPARCcenter 2000 are interleaved on 256 byte boundaries, producing an eight way interleave of the memory banks.

The interleaving not only reduces the latency for accesses to blocks of memory, but on average the access pattern of all the CPUs and I/O subsystems will be distributed uniformly across all the memory banks, minimizing the chance that a block sequential access from one device will conflict with the access from another. Specifically, within an MQH, the multiple SIMMs and DRAMs reduces latency for accessing a sub-block. Interleaving addresses across MQHs, and across XDBuses, balances the load and helps avoid hot spots.

## *Error Correction*

ECC is implemented in the MQH chip. Within a 64-bit memory word, it is able to detect and correct all single-bit errors, to detect all two-bit errors, and to detect three-bit and four-bit errors if all of the erroneous bits are in the same 4-bit nibble of the memory word. The SIMM crossbar ASIC orders memory bits so that each bit of a 64-bit memory word is in a different DRAM chip, so the failure of a chip causes a series of single-bit correctable errors. In such an event, the ECC fault handler can take appropriate action. Single-bit errors are written back to memory as the corrected data is delivered, ensuring that errors do not build up in memory. The MQH provides ECC for the entire memory subsystem, including the MQH itself, the memory crossbar on each SIMM, and on the DRAMs themselves.

## *Non-Volatile Memory (NVRAM)*

One of the most expensive operations for systems operating in an I/O-intensive environment is the synchronous disk write. Disk operations are performed synchronously, when data integrity is paramount, such as NFS write operations, updates to file system directory entries, and updates to a database management system's log file. Synchronous writes are posted immediately, then the invoking process is blocked until a successful completion is verified. These synchronous operations are expensive because they cause a process to wait while data is written to the physical disk drive and acknowledged. This process has much higher latency than the buffered write-behind scheme used by UNIX. With the optional non-volatile memory, the operation can be done in two phases: the data can be committed to the NVRAM, where it will survive a system crash, even one involving a power failure. Later, at an opportune time, the data is physically written to the disk drive -- improving responsiveness as well as disk utilization.

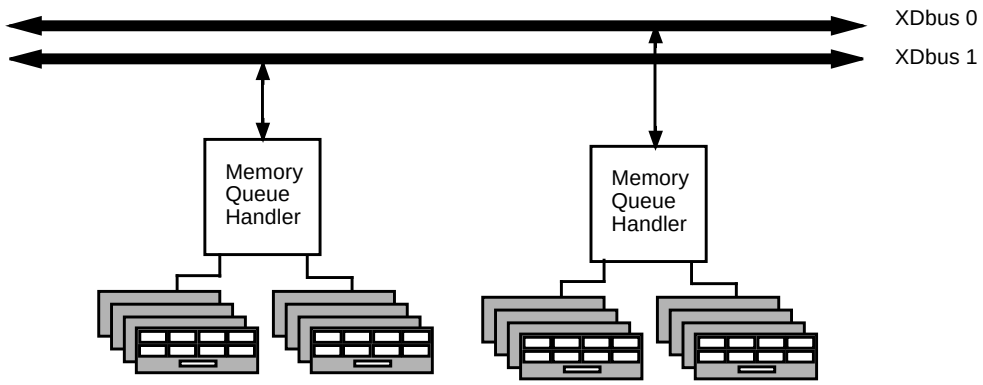


*Figure 2-2* SPARCcenter 2000 memory Unit Structure

Each memory bank is assigned a base address under software control. This allows the software to configure around missed or failed memory, as well as being able to handle heterogeneous memory -- such as NVRAM.

## *Memory Accesses*

Memory requests come in from multiple CPUs more quickly than main memory DRAMs can respond. Interleaved memory helps reduce the cost of memory access by distributing sequential requests from multiple CPUs across all the memory banks in the system, and permitting multiple memory components to operate in parallel.

A single bank consists of one to four groups of four SIMMs. When a request for data arrives at a MQH, it dispatches the request to a group of four SIMMs that in turn fetch or store data from the eighteen DRAMs on each SIMM. Each SIMM has an 18 bit interface so the group of four provides 72 bits (64 bits of data plus ECC). The 18 DRAMs on each SIMM provide four bits in each DRAM cycle, which is loaded into crossbar ASIC on the SIMM, such that the four bits are in successive words. After one DRAM cycle the MQH loads four complete 72 bit words from the crossbar, while the DRAMs cycle again to produce another four words. Without a break the next four words are loaded into the crossbar and read out by the MQH. The MQH reads the eight words out of the SIMMs at the same 40MHz rate that the words are transferred over the XDBus, it also arranges for the address that was requested to be transferred as the first word in the block. This allows the CPU to start processing as soon as the first word arrives, rather than waiting for all eight.

# Memory

In the current implementation, a memory bank consists of up to 16 SIMMs. SIMMs hold either 8 MB or 32 MB of RAM each so the maximum capacity is either 128 MB or 512 MB of RAM per bank. The SPARCserver 1000 system board has a single full bank of 16 SIMMS, and the SPARCcenter 2000 system board has two half-banks of 8 SIMMS so the maximum capacity of a single board is 512 MB in both cases. In the SPARCcenter 2000 with configurations of ten system boards, the memory capacity is 5120 MB, or five GB. A fully configured SPARCserver 1000 has a memory capacity of 2048 MB or two GB. Furthermore, the memory system is flexible enough to accommodate non-volatile RAM SIMMs (NVSIMMs), which are used in a similar fashion to a Prestoserve synchronous disk write accelerator. The NVSIMM is more efficient than the SBus Prestoserve because it operates at full memory speeds rather than SBus I/O speeds.

## Memory Structure

Each bank of memory in the family is an array of DRAMs controlled by an ASIC, called the Memory Queue Handler (MQH). The MQH implements address decoding, ECC error correction, XDBus access, and memory control in a single chip. It only transfers full blocks of 64 bytes.

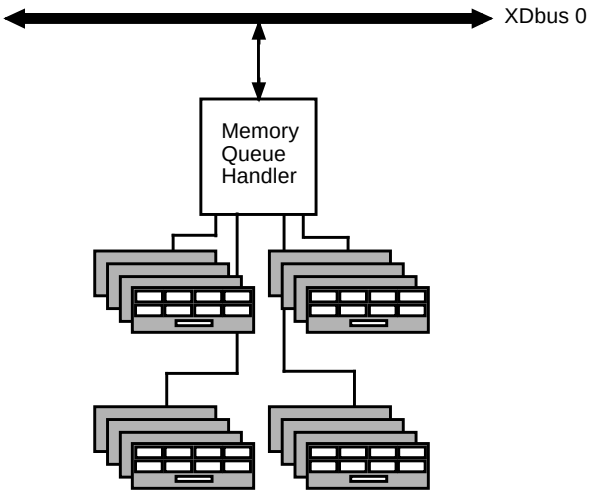


*Figure 2-1* SPARCserver 1000 Memory Unit Structure

## *Interrupts*

One of the crucial features in a symmetric multiprocessing system is the management of interrupts. Many SMP systems use a specialized mechanism to distribute interrupts. The SPARC server architecture uses the XDBus for most interrupt distribution, avoiding specialized mechanisms -- an interrupt is merely transported as a specific kind of XDBus packet. This arrangement dramatically simplifies the system's overall implementation while imposing very few demands upon the system backplane.

Most interrupts come from peripheral devices on an SBus that do not have direct access to the processor units. An XDBus device that wants to interrupt a processor simply prepares an interrupt packet and places it onto the XDBus. SBus devices can be programmed to direct their interrupts to specific processors.

When a device wants to interrupt, it sends an appropriate XDBus packet to the destination processor -- a specific processor or a broadcast to all processors. All bus watchers watch for interrupt packets addressed to its processor. When a bus watcher detects such an interrupt packet, it generates an XBus packet to the associated SuperCache controller describing the interrupt source. When the SuperCache controller receives this packet, it sets the appropriate bit in its pending interrupt register. At the end of each instruction cycle, the SPARC processor examines this pending interrupt register and takes appropriate action.

One useful consequence of using the standard XDBus as the interrupt transmission mechanism is that the interrupted processor always knows the source of the interrupt: the originator's XDBus device ID is part of the transmitted packet. This eliminates polling to determine the identity of the interrupting device, resulting in a significant increase in overall interrupt processing efficiency.

The SuperSPARC processor itself is a single highly integrated chip that includes two integer units, a floating point unit, a branch processor, a SPARC Reference MMU and a large, sophisticated on-chip cache. The processor is capable of executing three instructions each clock cycle when executing from the internal cache and accepts many combinations of instructions that would not be possible on other superscalar CPU's. This flexibility provides good performance on existing code that is not optimized for the SuperSPARC, although best performance comes from the SuperSPARC specific compiler option.

## *SPARCmodule Caches*

The SuperSPARC with its SuperCache chipset has a 20 Kilobyte, 5-way associative instruction cache and a 16 Kilobyte, 4-way associative data cache that are both loaded from a second level 1 Mbyte direct-mapped unified cache that loads from the memory system.

The instruction and data cache are a unified "on-chip" cache. On-chip refers to a design where the entire cache, including the control logic, is on a single chip. Splitting the on-chip cache between a separate instruction and data cache allows both caches to transfer during a single clock cycle. This speeds up both load and store instructions. Splitting the cache also means that instructions and data do not displace one another, assuring better cache performance.

The first-level cache is "set associative". In an associative cache, the addresses are tagged (as opposed to addressed via their full address). An n-way cache can have n different cache lines with the same address tag and a least-recently-used algorithm is used to allocate cache entries. Research has shown that associative caches perform as well as larger, direct-mapped caches. The SuperCache's on-chip design accommodates the control logic needed to implement the multi-way associative cache as well as the SRAM store to hold the cache tags. This highly-integrated cache design achieves the goal of fewer, more reliable components while still achieving a high cache-hit rate.

The SuperCache chip controls a 1MB, direct-mapped external cache that handles both instructions and data. The cache is always operated in a "write-back" mode. A write-back cache writes to memory when it is explicitly directed or when it needs to do so.

In addition to controlling the cache, the SuperCache provides hardware support for accelerated block copy (bcopy) and block zero (bzero) operations. The bcopy/bzero accelerator is crucial to the design of systems with high-capacity I/O subsystems. In such systems, the most common usage of bcopy is to copy whole I/O buffers between system and user space. Bzero is often used to create blank memory pages when allocating memory. The SuperCache cooperates with two bus watchers to implement the previously described XDBus cache coherency algorithms.

### *XBus And Cache Coherency*

In a system of multiple processors, data coherency becomes a problem. If each of the processors is caching (the same) data, then it is important to provide a cache coherency protocol among them. The SPARC server architecture implements a bus watcher mechanism on the XDbus. These bus watchers implement a cache coherency protocol that keeps all the caches synchronized with one another. In addition, there is a packet-switched bus, called the Xbus, that interfaces each of the caches to its bus watchers. The XBus, like the XDBus, is packet-switched allowing for asynchronous operation between the processor units and the bus watchers.

The family supports cache coherency in one of three ways: write-invalidate, write-broadcast and competitive caching -- a hybrid of the first two. The caching method is configured by the operating system software and selected based on expected work-loads. Write-invalidate works well when there is little write-sharing among the processors. Write-broadcasts works well when there is a lot of write-sharing among the processors. Competitive caching is a method for adapting between these two extremes.

Solaris 2.2 uses the write-invalidate mechanism, but a future version of Solaris may permit the use of the other protocols to allow users to optimize performance for their specific work loads.

## *Processors and Symmetric Multiprocessing*

The SPARC server architecture accommodates a large number of independent, field-upgradeable processor units. The SPARCserver 1000 has up to eight, and the SPARCcenter 2000 has up to twenty. Each unit consists of a highly integrated superscalar microprocessor, an external cache, a backplane interface and some support devices. To improve modularity and field upgrading, the processor itself, the cache, and part of the bus interface are located on a small (approximately 3”x5”), replaceable daughtercard -- known as a SPARCmodule. As processor technology advances considerably faster than other system technologies, the family accommodates at least three generations of processor models.

The first processor module incorporated into the SPARCcenter 2000 included a 40 MHz SuperSPARC processor. This has now been upgraded to 50MHz now that higher speed parts are available and an optional version with 2 MB of cache per processor has been added to the product line. The first module incorporated into the newly announced SPARCserver 1000 is the 50 MHz SuperSPARC processor. In the future, higher performance modules will become available and will simply plug in to the existing machines. One restriction is imposed by Solaris 2.2, all CPU’s are assumed to be identical, and mixed clock rate modules are not supported in a single machine.

bus, sends a request packet that specifies the target address and then releases the bus to make it available for other activity. While the request is being serviced, the bus is free to perform other activities. All packets on the bus are tagged so that a request can be associated with its reply. Furthermore, there is an added efficiency due to the pair of XDBuses in the SPARCcenter 2000 servers. The effect is a single bus with twice the bandwidth.

Although packet-switched implementations are somewhat more complex than circuit-switched buses of similar specifications, they provide considerably greater effective throughput. For example, the 40 MHz implementation of the MBus has a peak bandwidth of 320 MB/sec and an effective speed of about 105 MB/sec. The XDBus implementation in the SPARCcenter 2000 also operates at 40 MHz, and its peak bandwidth is also 320 MB/sec. Because arbitration for one packet is overlapped with actual transmission of the previous packet, it is possible to achieve nearly maximum efficiency: 250 MB/sec is often observed.

The SPARCcenter 2000 system has a pair of XDBuses. Should one bus fail, the system is capable of operating in a degraded mode with only one functioning XDBus -- providing fault isolation. In this circumstance only 1 MB of cache can be used, and the half of the main memory attached to the failed XDBus is not accessible. However the remaining parts of the system and the Solaris operating system are operational.

## *XDBus Arbitration and Flow Control*

With many devices competing to use the bus, some arrangement must be made to permit each device to make appropriate use of the bus. The XDBus is hierarchically arbitrated, using a central arbiter on the backplane itself and local arbitration for each segment. Each XDBus has an arbiter that grants requests to use the bus on a priority basis. Within each priority level the arbiter guarantees fair service to all devices. Bus allocation is non-preemptive since all packets are of fixed, small size.

The arbitration scheme is designed to maximize performance. The bus request and grant lines are implemented on dedicated lines, permitting transmission of one packet to be overlapped with arbitration for the next. This makes it possible to completely fill the bus with packets, even though arbitration is required for every packet. Additionally a device may make multiple requests before the first is granted, permitting a single device to use the bus to its maximum potential. The arbiter provides for a number of outstanding requests to each device in order to avoid resorting to explicit flow control whenever possible.

# The SPARC Server Architecture and Implementation in Detail

2

## Central Bus Complex

The SPARC server architecture's central bus complex, the XDBus, was designed for simplicity, high capacity, redundancy and efficiency. A single mechanism is used to transfer data between CPU and memory, between memory and the I/O buses, and to transmit interrupts -- all at rates that ensure a system remains balanced as the configuration is expanded.

### XDBus

The XDBus is a packed-switched, parity-protected, high-performance interconnect. A packet-switched bus, unlike the typical circuit-switched bus, eases the implementation of interleaved memory and allows the connection of arbitrarily slow devices without overall performance degradation.

As with all bus designs, only one requester or responder can be using the bus at a time. Therefore, it is important to minimize the amount of time that a device is in control of the bus and to eliminate non-optimal use. For example, in the case of a "circuit-switched" bus, the requester arbitrates for the bus, places the target address on the bus and then holds the bus while the request is being serviced. This is similar to placing a phone call. In the case of a long-distance call, you pay for the entire length of the call even though you may not be talking (using) the phone line (bus) the entire time.

The unique advantage of the XDBus is its "packet-switched" design. A packet-switched bus permits substantially greater overall throughput than comparable circuit-switched buses by separating bus requests from their corresponding replies. A requester arbitrates for the

## *SPARCcenter 2000 System Implementation Overview*

Taking the three basic building blocks already described, the SPARCcenter 2000 system board contains a dual up CPU block sharing a single BootBus, a single SBus I/O block and a single memory bank on each XDBus. The entire system board only contains nineteen highly-integrated ASICs: nine large 100K gate ASICs and ten much smaller chips. The backplane accepts up to ten system boards for a total of 20 CPUs, 40 SBus slots, and 5120MB of RAM. The SPARCcenter 2000 uses a modified form of the Sun rack-mount server packaging that was used on the previous generation SPARCserver 690.

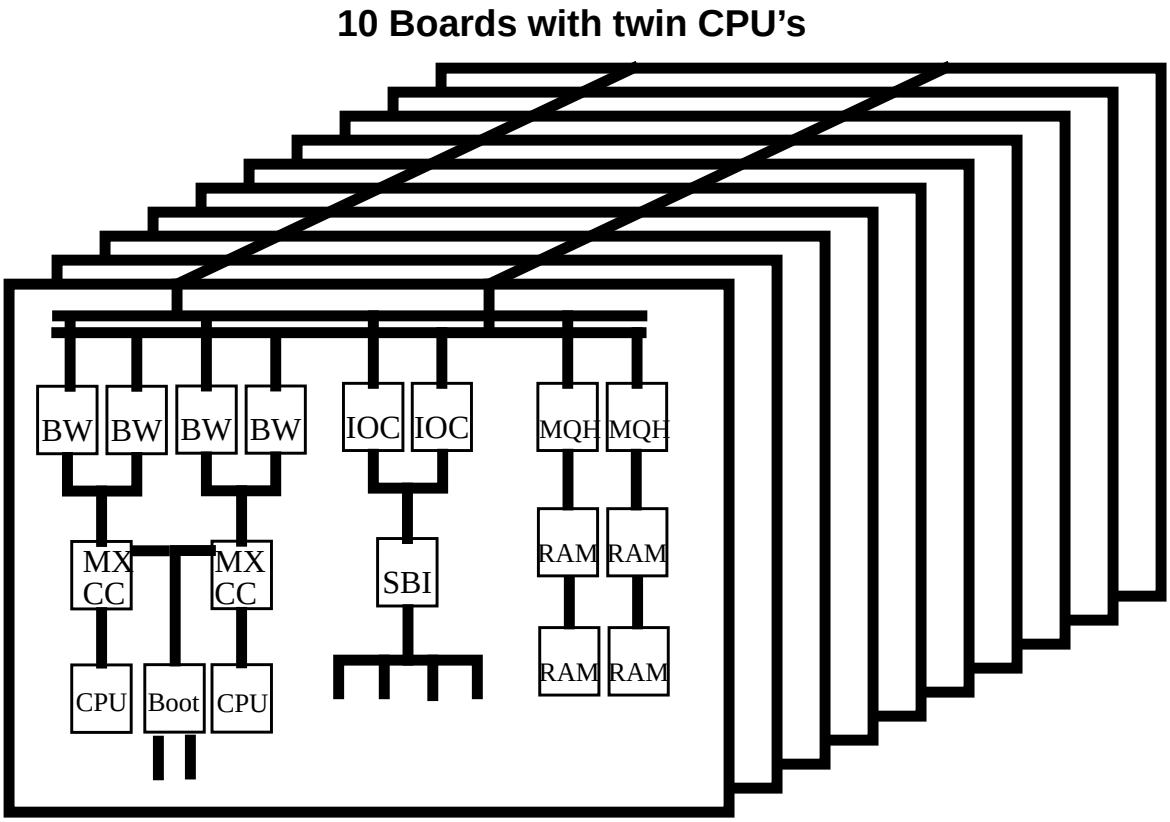


*Figure 1-6* SPARCcenter 2000 System Board Configuration

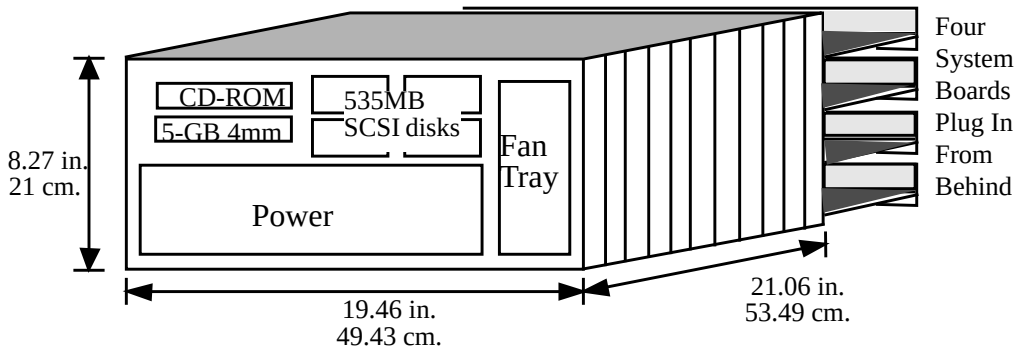


*Figure 1-5* The SPARCserver 1000 Package

The package is extremely compact; four system boards lie flat and make up the full height and width of the box. The disks, DAT tape and CDROM fit in front of the XDBus backplane and the system board is L-shaped to fit around them. The power supply and fan tray squeeze into the gaps so there is virtually no unused space.

To provide more flexibility in the same package a special board containing low profile disk drives fits in the same physical space as a system board. It contains four 1” high 535MB SCSI drives and connects to an SBus SCSI controller mounted on a system board. In this way the standard package can be configured with a single system board and three disk boards, which taken together with the four disks mounted at the front provides over 8GB of storage. A special version of the package that leaves out the front mounted items can be stacked with the system unit to provide external mountings for disk boards if more system boards are needed together with a large disk capacity.

The side panel is removable, and this allows the package to be rack mounted in any standard 19” rack. For maximum disk capacity the rack mounted disk units used in the SPARCserver 2000 can be configured, with the SPARCserver 2000 sitting inside or on top of the mass storage rack. These racks take eight trays of six 2.1GB disks, and using eight SBus Differential SCSI interfaces about 100GB of storage can be configured.

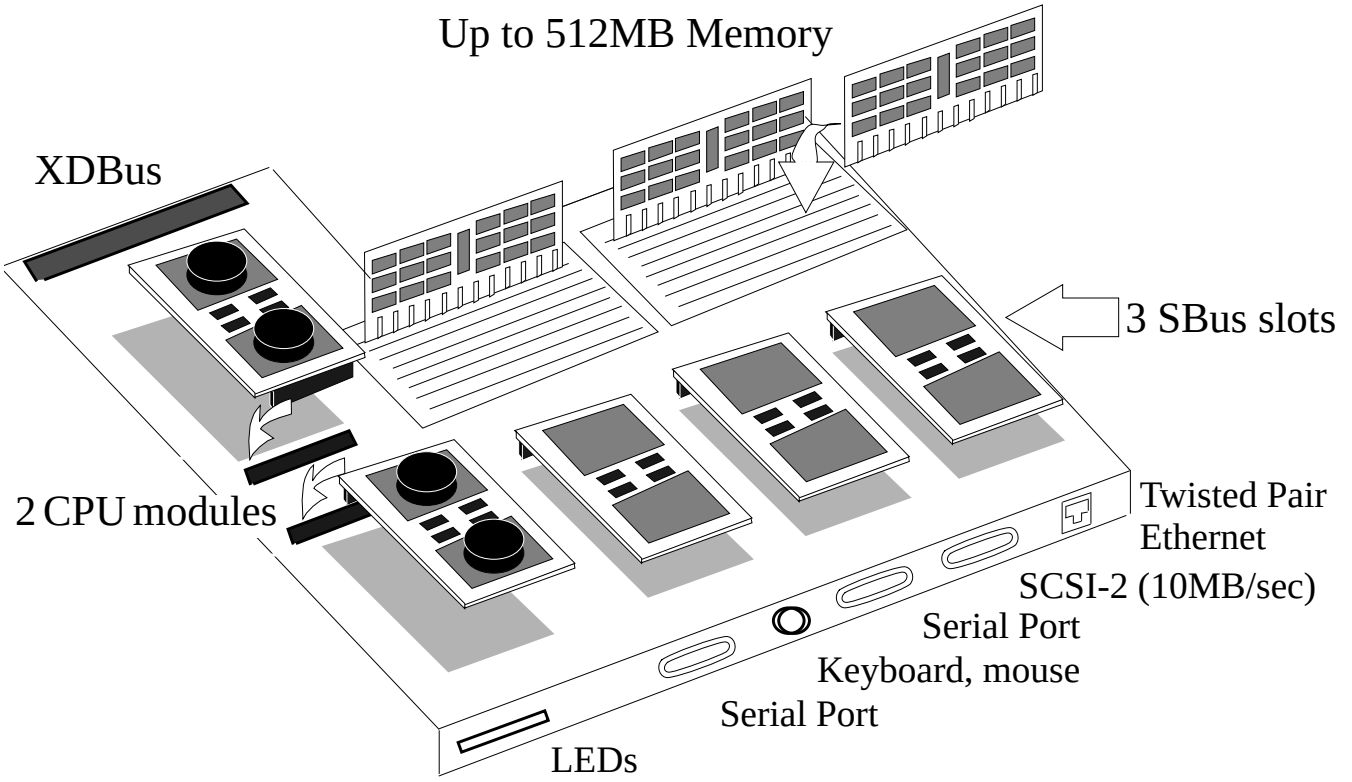


*Figure 1-4* Physical Layout of the SPARCserver 1000 System Board

The system board acts as an infrastructure to support plug in components. The memory subsystem uses the same SIMMs as the SPARCcenter 2000. The I/O subsystem uses the same SBus cards as the rest of Sun's product line and the CPU modules are the same as those used in the SPARCstation 10 Model 51 and Model 512MP.

## *SPARCserver 1000 System Implementation Overview*

The design objective for the SPARCserver 1000 was to take the architecture into an office environment and to produce a low cost entry point into the range. This was achieved by using a very compact package, about the same size as an office laserprinter, that can be put on a desktop, stacked up on the floor, or rack mounted.

Taking the three basic building blocks already described, the SPARCserver 1000 system board contains twin CPU blocks sharing a single BootBus, a single SBus I/O block including an integrated FSBE/S interface and a single memory bank on one XDBus. The backplane accepts up to four system boards for a total of 8 CPUs, 16 SBus slots, and 2048MB of RAM.

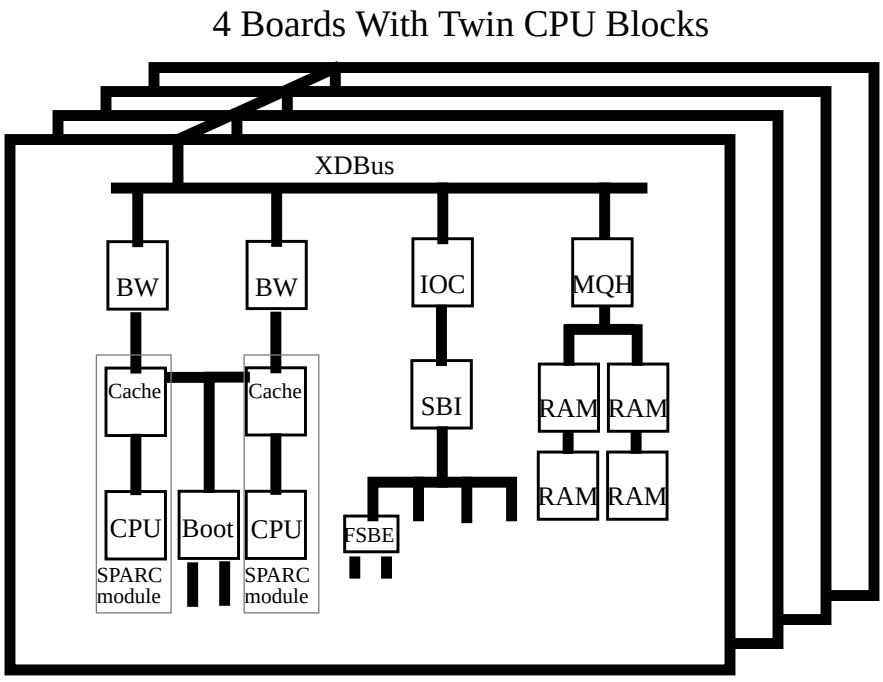


*Figure 1-3* SPARCserver 1000 Configuration

## *Reliability, Availability And Serviceability*

The system contains an extensive power on self test (POST) capability which uses multiple processors to test and configure all functional units in the system. In this way, the time taken to do the testing does not increase unduly for the largest configurations. The concept of a “minimum hot core” is used, and if a single SPARCmodule is working, it can read the POST prom from its directly connected BootBus, set LEDs to indicate any problems and send messages to a serial port, all using the BootBus. It then explores the rest of the system and units that pass the test are attached to the XDBus and configured into the address space. Probing and configuration uses the IEEE JTAG interface (normally used in IC testing during manufacture), so that full internal testing of the main ASICs in the system is done independently of the XDBus interconnect. If a component fails, this is logged into nonvolatile memory and the operator is notified. The system can configure around failed components, including a failed or nonexistent XDBus, and there is essentially no difference between a SPARCserver 1000 and a SPARCcenter 2000 that has configured out one XDBus due to a failure. If neither XDBus is available the system reverts to a single board mode, using only the local part of the XDBus to access memory and the SBus. Unlike most other systems, this architecture ensures that the system can always reboot very quickly after a failure and it has a very high availability rating. In the past dual systems with shared disks have been used to provide high availability, a single SPARCserver 1000 or SPARCcenter 2000 can be configured to provide better inherent reliability and a comparable availability rating in a single system, without the complexity of a dual machine.

## CPU Building Blocks

Attached to the XDBus to form a symmetric shared memory multiprocessor are a number of CPUs. These are packaged in removable SPARCmodules identical to those used in the high end workstation product line, containing a 50MHz SuperSPARC+, SuperCache controller[1] and 1MB of external cache. The module detects which type of machine it is plugged in to and uses a more advanced XBus packet switched interface protocol that can take advantage of the full XDBus bandwidth. A large ASIC called the Bus Watcher (BW) connects each CPU to each XDBus. A special feature of the SPARCcenter 2000 is support for a unique SPARCmodule with 2MB of cache which requires two Bus Watchers per CPU to operate. Systems can be extended in the field by adding SPARCmodules or upgrading them to higher clock rates. Although other vendors presently offer large-scale symmetric multiprocessing, it is not offered in well-balanced systems with commensurate memory bus bandwidth or I/O channels. For example, some systems are offered with even more processors than the SPARCcenter 2000, yet these systems provide a tenth of the overall backplane bandwidth.

## I/O Building Blocks

Each I/O block implements an industry standard SBus interface that is accessed via the XDBus. A single large ASIC implements the SBus interface (the SBI) and the SBI is connected to each XDBus via an I/O Cache ASIC (IOC). Each SBI delivers over 50 MB/sec throughput on the SBus and supports four slots.

## Memory Building Blocks

Main memory is configured with two levels of interleave. Accesses from multiple CPUs and I/O devices are processed in parallel by multiple memory banks, and each memory bank can also queue multiple requests and process them in parallel on different groups of SIMMs. Memory is managed and connected to one XDBus only by a single large ASIC called the Memory Queue Handler (MQH). The access time is the same for all of memory, no matter where it is physically located on the XDBus. The memory address space and interleave is configured automatically by the power on self test (POST) code in the boot prom in order to optimize performance and program around missing or failed memory.

1. The SuperCache controller is also known as the MXCC, or MBus/XBus cache controller. XBus is used within the system board, it is similar to a single XDBus.

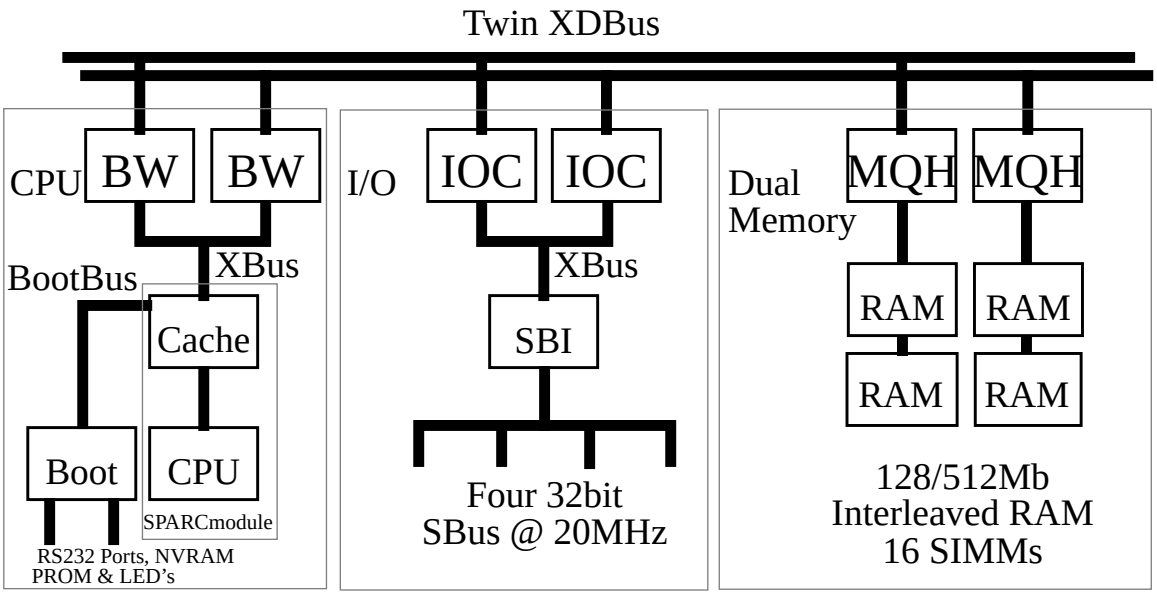


*Figure 1-2* The CPU, I/O and Dual Memory Building Blocks of the SPARCcenter 2000

## *XDBus System Interconnect*

At the heart of the SPARCserver 1000 is an XDBus. A high-speed packet-switched backplane bus that provides very high data transfer bandwidth. A single instance of this bus has a nominal bandwidth of 320MB/sec. In addition, its packet-switched design means the bus will be able to deliver a high proportion of this bandwidth, typically sustaining 250MB/sec. By comparison, typical circuit-switched buses can only provide a fraction of their peak bandwidths under load. The SPARCcenter 2000 uses twin interleaved XDBuses providing even higher bandwidth and higher availability, since the SPARCcenter 2000 can continue to operate even when one of its XDBuses has failed. In practice, the paired buses provide overall throughput of approximately 500 MB/sec. The architecture supports up to four XDBuses in a single machine, so that larger configurations than the SPARCcenter 2000 offers could be produced in the future using basically the same chip-set.

# Architectural Overview

## Building Blocks

The architecture is based on a small number of building blocks which are combined in two configurations to produce the SPARCserver 1000 and SPARCcenter 2000. This section looks at the individual blocks, the next section will examine the combinations used to build each system board and the way system boards are connected. One key feature of the design is the unprecedented level of integration of the circuitry. In most cases a single ASIC implements the complete functionality of a major subsystem and there is virtually no support logic or “glue” in the design. Competing designs often have hundreds of individual chips that implement inferior functionality at much higher cost and lower performance. The effect of compressing a huge amount of functionality into a very small space can be seen in the compact dimensions of the SPARCserver 1000 and the high capacity of the full size SPARCcenter 2000 rack.

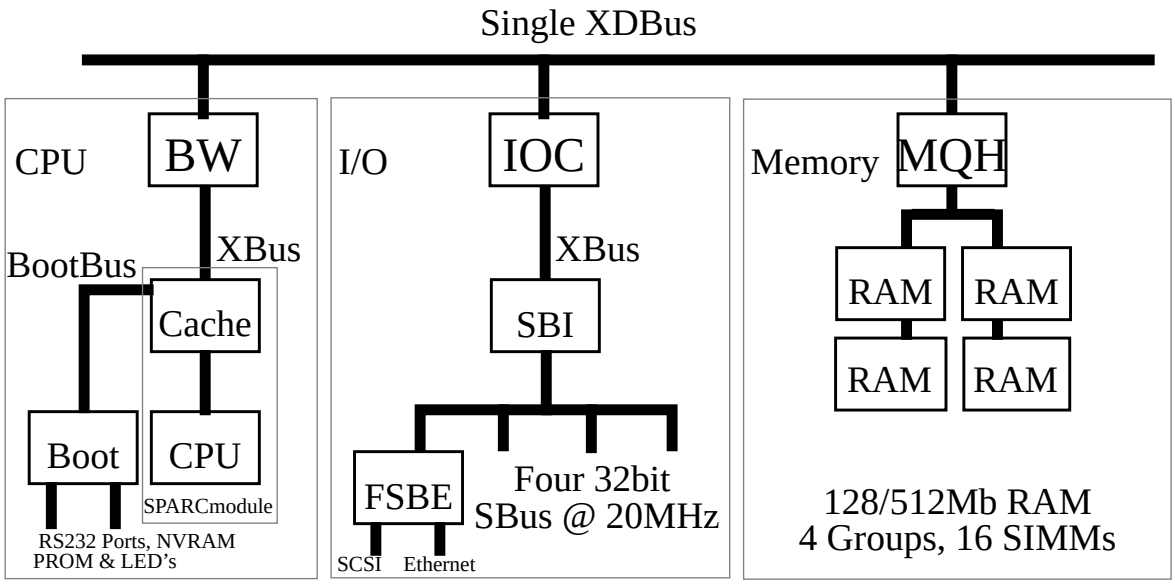


*Figure 1-1* The CPU, I/O and Memory Building Blocks of the SPARCserver 1000. BW = Bus Watcher, IOC = Input Output Cache, MQH = Memory Queue handler, SBI = SBus Interface, FSBE = Fast SCSI Buffered Ethernet.

# 1

# Architectural Requirements

A server architecture must have high ultimate capacity. Even when a small configuration is used there must be headroom to expand as the workload increases. The system must be optimized for throughput. The ability to process a large number of concurrent jobs or transactions is the top priority. Applications running on servers are often a critical part of the infrastructure of a company so reliability and high availability must be designed into the architecture.

*The new SPARC server architecture meets these goals.*

## *Capacity*

The architecture consists of multiple high performance SuperSPARC CPUs connected to multiple memory controllers and many independent SBuses for I/O. The primary limitations to the capacity of an implementation are the physical constraints of available board area and package dimensions, and even a fully loaded SPARCcenter 2000 does not implement the largest configuration allowed by the architecture.

## *Throughput*

There are multiple high capacity packet-switched XDBuses connecting the functional units in the architecture. To keep the throughput of the system in balance with the number of CPU and I/O buses systems can be implemented with one, two or four XDBuses. Memory banks are interleaved across multiple XDBuses to balance the workload and improve performance by distributing sequential requests across multiple memory banks. An XDBus is usually configured to support approximately ten SuperSPARC CPUs and five SBuses. To manage the throughput of the hardware the multithreaded Solaris 2.2 kernel allows all CPUs to share the interrupt load and I/O processing requirements.

## *Reliability And Availability*

Throughout the design the use of highly integrated components provides exceptional reliability. A novel new transmission line technology called GTL is used on all buses, allowing much higher clock rates and better noise immunity than the usual CMOS. The architecture provides error correcting code (ECC) protected memory with parity protection on all buses for both the data path and control signals to detect errors. On power-up, or after the system detects an error and reboots, an advanced system testing and configuration process is used. Failed components can be identified and configured out of the system, enabling continued operation for high availability applications.

Compared to a SPARCserver 10, with one to four processors connected to one memory bank and one SBus, the additional throughput of the SPARCserver 1000 and SPARCcenter 2000 comes from the use of many more processors interconnected via substantially faster buses to multiple independent memory banks and multiple SBuses.

*Table 1-1* SPARCserver System Expansion Comparisons

| Machine | SPARCserver 10 | SPARCserver 1000 | SPARCcenter 2000 |
|---|---|---|---|
| SuperSPARC Clock rate | 36, 40, 45, 50 MHz | 50MHz | 50MHz |
| CPU External Cache | Optional 1MB | Standard 1MB | 1MB or 2MB |
| Max Number of CPUs | 2@50MHz, 4@45MHz | 8@50MHz | 20@50MHz |
| Total Memory | 32MB to 512MB | 32MB to 2048MB | 64MB to 5120MB |
| Memory Banks | 1 with 512MB | 4 with 512MB each | 20 with 256MB each |
| SBus I/O Slots | 4 at 20MHz | 12 at 20MHz | 20 at 20MHz |
| Independent SBuses | 1@40MB/s sustained | 4@50MB/s sustained | 10@50MB/s sustained |
| Interconnect Throughput | MBus@105MB/s | XDBus@250MB/s | XDBus@500MB/s |
| Internal Disk Capacity | 2 x 1.05GB | 4 to 16[1] x 535MB | 18 x 2.1GB |
| Other Internal Storage | 1.44MB 3.5” Floppy | CDROM, 5GB DAT | CDROM, 3x5GB Exabyte |
| Max Disk Capacity | About 40GB | About 100GB | About 500GB |

1. Four are packaged, additional internal disks take the place of system boards.

# 1

## Background

The high end server marketplace is a particularly challenging one to design for with several conflicting trade-off's to be made. In particular there is a very wide range of capacity and performance requirements, but relatively low production volumes compared to the workstation marketplace. For most vendors this leads to a product line that includes many low volume machines, each spanning a small performance range, and with high upgrade costs for users moving up the range. The design costs for each model are amortized over a relatively small production run, the manufacturing process is less automated, and the pricing is correspondingly higher than volume workstation products.

Sun has been moving into the server marketplace for some time, using the high volume economics of the workstation production process to push commodity pricing into the low end and mid-range server markets. After launching the multiprocessor SPARCserver 600MP range in 1991 Sun became the #1 vendor of multiprocessor Unix servers by both unit shipments and installed base during 1992, surpassing several companies who had been specializing in this type of machine for many years.

With its new SPARC server architecture, Sun now brings this approach to the high end of the server marketplace. Two packages are required to deliver the architecture, one low cost package that can sit on or beside a desk in an office environment, and one package for the data center that maximizes expansion capacity. These two packages contain scaled versions of the same architecture, and the investment in IC designs and operating system porting and tuning is shared by both machines.

The desktop package used in the SPARCserver 1000 has been designed in a very elegant, modular manner that allows automated high volume production and simplifies servicing. The data center package used in the SPARCcenter 2000 also has a small number of modular components but configurations are built to order, due to the physically large package and huge number of configuration options.

A key theme of the new SPARC server architecture is that it is built from a small number of highly integrated components which are replicated to produce the required configuration. Several of these components, including processor modules and SBus I/O cards, are identical to those used in the high volume workstation product line, which reduces per-processor costs and means that there is a wide choice of I/O options. The system design that connects processors, I/O and memory over multiple high speed buses is implemented in a small number of complex ASICs, which reduces the number of ICs on each board, easing manufacture, reducing cost and increasing reliability.

# *Introduction* 1

## *Overview*

This paper describes the architecture of one of the most flexible and powerful ranges of general purpose computers available today. To start with some of the background issues that shaped the design are explored, then put into perspective in relation to other products. The main features of the architecture and the way it is implemented and packaged are covered, then subsequent chapters explore the details of each part of the architecture and the Solaris operating system implementation.

In summary, the new SPARC server architecture extends the commodity workstation production process and pricing model up into the high end server marketplace for the first time. The systems provide substantially higher throughput at a substantially lower cost than competitive high end server systems, using only two extensible packages to deliver a low cost growth path to the customer.

In conjunction with the Solaris operating system, these systems are designed to meet both the high capacity I/O and computationally intensive demands of a very wide range of applications. The trend in the industry for down-sizing means that applications are migrating within the corporate data centers, on to high capacity Unix based servers such as the SPARCcenter 2000. Applications are also moving out of the data centers, and the SPARCserver 1000 is in a class of its own for performance in an office environment as a departmental server.

# *Contents*

Please
Recycle

# *A Scalable Server Architecture*

*From Department to Enterprise - The SPARCserver 1000 and the SPARCcenter 2000*

*SMCC Technical Marketing*

**Sun Microsystems Computer Corporation**
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No.: 8xx-xxxx-xx
Revision 2, May 1993