Gigaplane-XB: Extending the Ultra Enterprise Family

Alan Charlesworth (alan.charlesworth@west.sun.com), Andy Phelps, Ricki Williams, Gary Gilbert
Sun Microsystems, Inc.

Abstract

The Gigaplane-XB interconnect of the Starfire system extends the range of Sun's Ultra Enterprise SMP server family by 4x. It uses multi-level address and data routers to provide 167 million global snoops per second, and 10,667 MBps of uniform-memory-access bandwidth. The use of point-to-point routers allows the system to be dynamically reconfigured into multiple hardware-protected operating system domains, with hot-swappable boards. Comparisons are made with the bus-based technology of the smaller family members.

1. Introduction

When a new processor debuts, we hear mostly about its SPECint95 and SPECfp95 performance. It is probably more important to know how well a processor's power is balanced its memory bandwidth and latency. From a system designer's perspective, the main reason to have an instruction set architecture is to get the opportunity to engineer into the processor chip a good interconnect architecture.

Table 1 compares the memory-port architectures of current microprocessors. A combination of a 16-byte wide data path, a 100 MHz system clock, and separate address and data paths has made Sun's Ultra Port Architecture (UPA) among the highest-bandwidth memory ports. The UPA is used across all Sun's UltraSPARC systems: from desktop workstations to the 64-processor Starfire (UltraEnterprise/High Performance Computing 10000) server.

Table 1. Memory-port characteristics.

Instruction set architecture	Interconnect architecture	Current CPU model	Max Port MHz	Data width (bytes)	Separate address and data?	Data duty factor	Peak data bandwidth (MBps)
SPARC	UPA [9]	UltraSPARC-II	100	16	Yes	100%	1,600
Alpha	See [3]	21164	88	16	Yes	100%	1,408
Mips	Avalanche [18]	R10000	100	8	No	84%	840
PA-RISC	Runway [1]	PA-8000	120	8	No	80%	768
PowerPC	See [2]	PPC 604	67	8	Yes	100%	533
Pentium Pro/II	See [5]	Pentium Pro/II	67	8	Yes	100%	533

2. Technology foundation

We had four building blocks for the Starfire design:

- Our previous design experience with large servers.
- The UltraSPARC processor.
- The Ultra Port Architecture.
- The UE6000 member of the Ultra Enterprise family.

2.1 Previous experience

Our team has implemented three previous generations of enterprise servers, and our ongoing strategy has been to build as large a system as will fit on one centerplane.

When we started the Starfire design in the fall of 1993, our product was the 64-processor CS6400. It used the XDBus [4], which pioneered the technique of bus interleaving to increase snoop bandwidth. The four XDBuses of the CS6400 could do 110 million snoops per second, and 1,280 MBps of data bandwidth. This was the most capacity of its generation. Smaller members of the family used one and two XDBuses, and all models employed the same set of interconnect ASICs. The CS6400 was packaged on large 16" x 22" system boards, which held four processor modules, 576 DRAMs, and four SBus I/O cards. The system cabinet was about 5' tall x 2' wide x 4' deep.

To improve serviceability, the CS6400 had a System Service Processor (SSP) which provided intelligent and known-good diagnostic support. The SSP had back-door connections to all the ASICs in the system to aid in fault diagnosis.

To improve availability, the CS6400 system boards could be hot-swapped in and out for service, and Solaris dynamically reconfigured on and off of these boards — all while the system remained operational. Because the interconnect was bus-based, the system had to be quiesced for a few seconds while the board was physically removed from the cabinet.

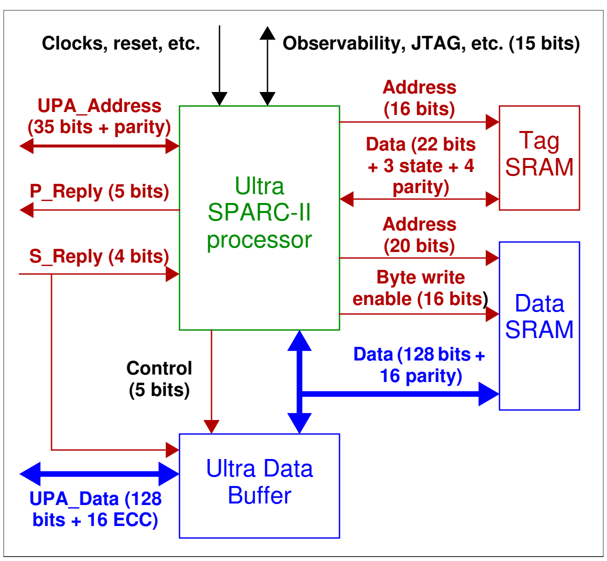
To increase system flexibility, the CS6400 could have two single-board *domains* in addition to the main system. These were useful as isolated environments for safely testing new software. A domain ran a separate instance of Solaris, and was protected from software faults in other domains. Domains could be dynamically configured and deconfigured on the fly.

We found that the scale of the CS6400 worked well, and decided to carry over many of its concepts to the new UltraSPARC / UPA technology generation.

2.2 UltraSPARC processor

The UltraSPARC processor family [11] was first delivered in late 1995 at 167 MHz. The second-generation UltraSPARC-II has been scaled up to 300 MHz in the most recent products [12]. It has an estimated SPECint95 of 12 and a SPECfp95 of 18.1 when configured with a 2 MB external cache in a workstation.

Figure 1. The UltraSPARC-II processor.



The processor can execute four operations per clock, including one floating-point multiply and one floating-point add. It supports software-initiated prefetches. There may be up to three outstanding memory transactions, including one instruction-cache miss, up to three load/prefetch misses, and up to two writebacks of modified cache lines.

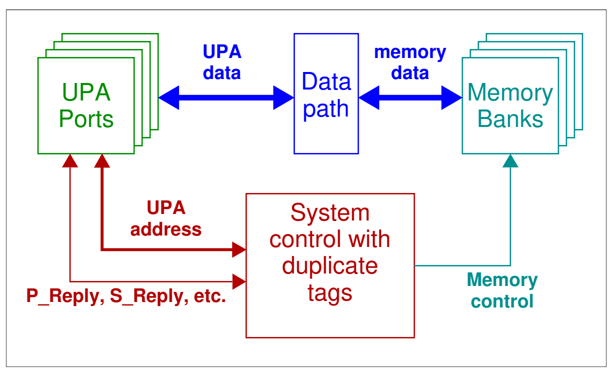
The UltraSPARC family implements the Visual Instruction Set (VIS) multimedia extensions [6]. These include block load and store instructions that transfer data directly between an aligned 64-byte block of memory and an aligned group of eight double-precision floating-point registers. VIS block stores do not cause a write-allocate in cache, and so reduce the load on the interconnect. Block accesses avoid polluting the caches with non-reused vector data, and so avoid sweeping code out of the instruction cache. There may be up to three block accesses in progress, including one block store.

The processor, external cache, and data buffer chips are all mounted on a module, so faster processors and larger caches can be productized without affecting the underlying system board.

2.3 Ultra Port Architecture

The UPA [9] allows a wide range of implementations, ranging from uniprocessor desktops to multiboard symmetric multiprocessor (SMP) systems. The UPA does writeback MOESI (exclusively modified, shared modified, exclusively clean, shared clean, and invalid) coherency on 64-byte-wide cache blocks.

Figure 2. The Ultra Port Architecture model.



To allow scaling beyond traditional snoopy buses, the UPA defines a point-to-point packet-switched protocol between the distributed system control function. For example, the sequence for a load-miss to memory is:

1. The UPA master port issues a read transaction request (P_Req) on its UPA address bus to the system control.
2. The system control does a snoop operation on the duplicate tags, and at the same time starts a memory cycle.

3. The datapath is scheduled, and the data is delivered to the requesting UPA port with an S_Reply.

The UPA does not dictate snoop timing, so large systems can trade-off snoop latency for greater bandwidth.

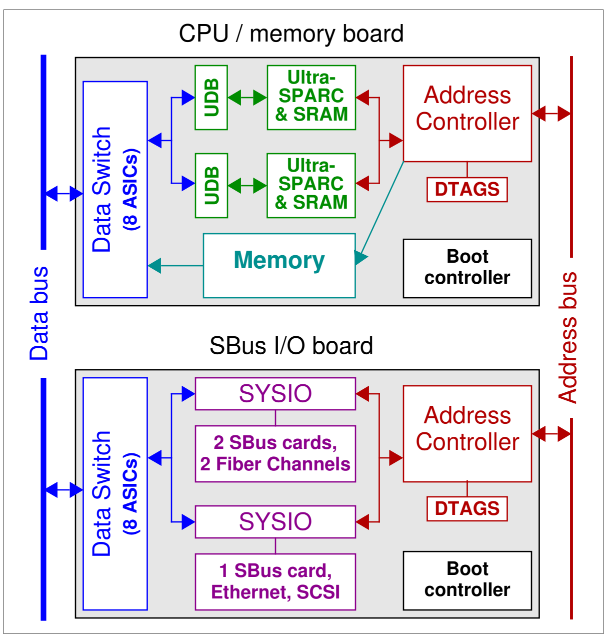
The UPA maintains a set of duplicate coherence tags for each cache in the system. It performs a fast SRAM lookup in parallel with memory reads. In contrast, directory-based systems which maintain coherence states in main memory must incur a read-modify-write penalty for every memory read.

The UPA's departure from conventional snoopy-bus and directory-based approaches allows processors to achieve the lowest possible latency on cache misses, and effectively pipelines the interconnect to allow maximum and often bubbleless utilization of address path, datapath and main memory. Cache fill transactions and dirty-victim writeback transactions proceed in parallel and independently of each other, further reducing cache miss latencies.

2.4 Ultra Enterprise 3000–6000

The UE6000 [10] has the largest bus-based interconnect of current systems. Its address bus does a snoop every-other clock, while its 36-byte-wide error-correcting code (ECC)-protected data bus transmits a 64-byte data block every-other clock. At an 83.3 MHz system clock, this is 41.7 million snoops per second, and a 2,667 MBps data rate.

Figure 3. UE3000–6000 bus-based interconnect.



The UE6000 is implemented with eight board slots on either side of its centerplane. Smaller cabinets (UE4000 and UE3000) hold eight and four boards respectively. The boards are 11.5" x 10" in size, and have a 2" spacing on the centerplane.

There are three board types. The CPU/memory board contains two processor modules, 16 DIMMs, and 10 interconnect ASICs of three different types. The SBus I/O board provides three SBus slots on two SBuses, as well as built-in 10 and 100 Mbit/sec Fast Ethernet, Fast/Wide SCSI-2 and two Fibre Channel sockets. The Graphics I/O board substitutes a high-bandwidth framebuffer connection for one of the SBus slots.

The UE3000 can be configured with up to six processors, the UE4000 up to 14 processors, and the UE6000 with up to 30 processors.

3. Starfire design choices

We set three primary product goals for the Starfire:

1. Scale up the Ultra Enterprise server family by providing 4x the bandwidth of the UE6000.
2. Introduce mainframe-style partitioning to Unix servers with Dynamic System Domains.
3. Increase the system reliability, availability, and serviceability (RAS).

3.1 Providing 4x greater bandwidth

We again chose to do a uniform-memory access (UMA) centerplane interconnect. We believe that one should not add the complications of nonuniform memory access (NUMA) and multi-cabinet cable topologies unless one is forced to by market requirements. We observe that a system with 64 processors, 64 GB of memory, and several TB of disk is enough computing for almost all enterprise and scientific-computing customers. In limiting ourselves to a single-cabinet interconnect we did not feel that we were missing much of the market. For larger needs, Sun provides a complete range of clustering solutions which apply to Starfire like the rest of Sun's servers.

3.1.1 Four interleaved address buses. To quadruple the snooping bandwidth, we chose to interleave four snoop buses, as we had done on the previous generation. Each bus covers 1/4 of the physical address space, based on the two low-order address bits. Like on the UE6000, the buses snoop every-other cycle, and update the duplicate tags in the between cycles. At an 83.3 MHz clock, the system snoop rate is 167 million per second, which given the 64-byte cache line width, is enough for a 10,667 MBps data rate.

3.1.2 Data crossbar. We couldn't quadruple the UE6000's databus width, because it was already a half-cacheline wide. Instead we settled on an inter-board data crossbar. To reduce the signal count, we made the crossbar port 18 bytes wide, which conveniently is the same width as the UPA databus from the UltraSPARC processor module.

At an 83.3 MHz clock, the peak data rate is 21,300 MBps. For simplicity of implementation, the data crossbar is not buffered, so a full-duty-factor random address stream will yield 69% of the peak rate. For a full 16-board configuration, only about 50% of the data bandwidth is necessary to match the snooping rate. The balance point between Starfire's address and data interconnects is about 13 boards — smaller configurations don't fully saturate the address buses, and larger systems don't fully utilize the data crossbar.

3.1.3 Active centerplane. Each system board has 1,296 signal and ground pins, for a total of nearly 21,000 pins that go to the global interconnect. The only way to get this many signals to the interconnect ASICs was to mount them on the centerplane. Because that made a potential single-point of failure, both the address and data interconnects can operate in degraded mode until a field-swap is made of a failed centerplane. We investigated socketing the centerplane ASICs, but determined that would reduce reliability.

3.1.4 Pipelined coherency. When a processor does a *ReadToShare* UPA transaction after a load miss, the UPA protocol allows for optimizing the case where the processor later wants to modify the data. It gives the requester *exclusive* access if no other processor already has the data cached. This optimization allows the processor to write the data without further interconnect transactions.

We could not pipeline a conditional reply from the system, and so we chose to have our implementation always reply with a *ReadBlockShared*. The processor will never enter the *exclusive* state, but instead enters the *shared clean* state. If a processor later decides to modify data, it must do a *ReadToOwn* coherency transaction to get the cache line into the *exclusively modified* state, which invalidates other cached copies.

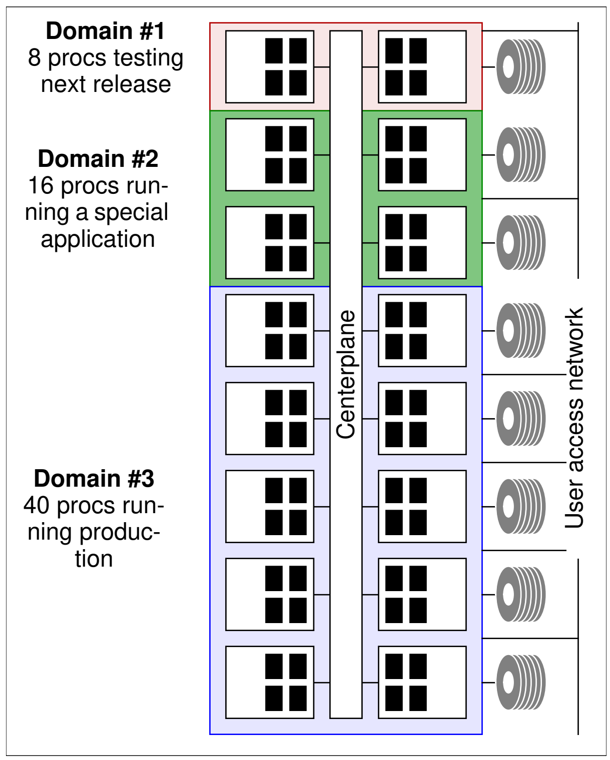
3.2 Providing dynamic system domains

Partitioning flexibility has been a widely desired highend feature, but has previously been limited to the mainframe world. Partitioning allows the system administrator to subdivide the system to test new software releases, to isolate different workloads, or to consolidate smaller server functions into a single, more flexible and manageable solution [17].

A domain extends across one or more system boards, along with the peripherals that are connected to the SBus cards on those system boards. Each domain runs its own instance of Solaris.

3.2.1 Point-to-point routing. We wanted to keep failures on one system board from affecting the other system boards. To electrically isolate the domains, we used point-to-point router ASICs for all of the interconnect: data, arbitration, and the four address "buses." The routing components enforce hardware and software isolation between domains by filtering the address requests. We avoided bussed signals, so that one failed system board could not hang the entire machine.

Figure 4. Example of Starfire system domains.



3.2.2 Faster clock. Point-to-point connections also reduce electrical loads to a minimum, allowing us to design for a 100 MHz system clock.

3.2.3 Dynamic reconfiguration. Dynamic reconfiguration empties work out of a board, then introduces the board to another domain, or leaves it idle so it can be removed. The Solaris scheduler is informed that no new processes should start on the board. Meanwhile, any pending I/O operations are completed, and user and kernel memory is moved into other boards. A switchover to alternate I/O paths takes place so that when the system board is removed, its I/O controllers will still be available. Reintroduction is the reverse process, where the scheduler assigns new processes to the board, and lets memory fill up again.

3.2.4 Inter-domain networking. We wanted to provide for optional fast communication between domains in a domain *group*. We allowed for a range of physical memory to be shared between the domains in a group. This facility is used by the operating system to implement Inter-Domain Networking (IDN). IDN provides a fast network device that supports the TCP/IP standard Data Link Provider Interface (DLPI). This is a standard API which all network devices export in order to provide TCP/IP services.

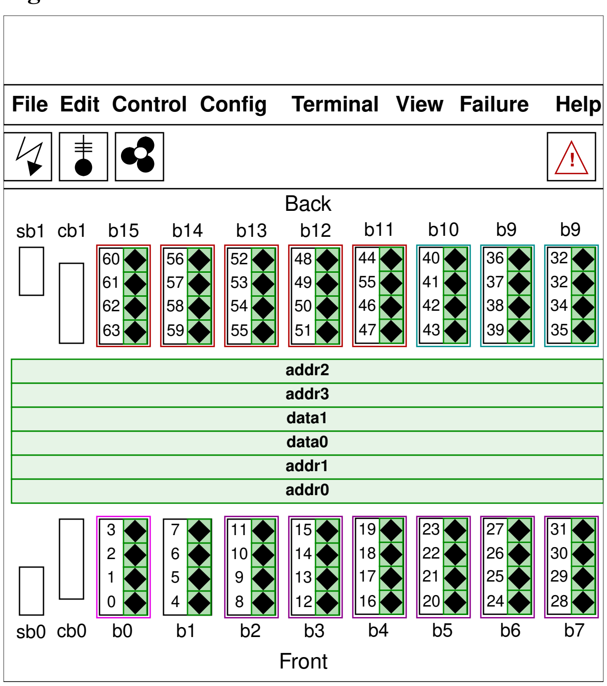
Please see [15] for more information on Starfire domains.

3.3 Providing increased RAS

Systems vendors must meet the uptime demands of business-critical applications by providing computing platforms that are Reliable, Available for use when needed, and easy to diagnose and Service (RAS).

3.3.1 System Service Processor (SSP). The SSP is fundamental to our RAS strategy. It is the access port for service providers, using programs like *hostview*, shown in Figure 5.

Figure 5. SSP *hostview* main screen.



The SSP provides a level of intelligence physically separate from Starfire. This approach enables many features, such as being able to power the Starfire on and off remotely. The SSP sequences the boot process, assists in configuration decisions, monitors environmental conditions, helps the Starfire recover from interrupts, and sends information to customer service regarding interrupts.

The SSP is connected via Ethernet to the Starfire's control board. The control board has an embedded control processor which interprets the TCP/IP Ethernet traffic and converts it to JTAG control information.

3.3.2 ECC and parity. We ECC protected the entire datapath from the UltraSPARC module to the memory and back. Single-bit errors that are detected by the interconnect ASICs along the way are logged for fault-isolation. The ECC protection is designed so that the failure of an entire DRAM chip will result in correctable, single-bit errors.

The global address interconnect is ECC protected, while the UPA address bus from the UltraSPARC, and its local path to the global interconnect interface are parity protected.

3.3.3 Optionally redundant components. We made it possible to configure a Starfire to have 100% hardware redundancy of critical components: control boards, centerplane support boards, system boards, disk storage, bulk power subsystems, bulk power supplies, cooling fans, peripheral controllers, and System Service Processors.

The centerplane can operate in a degraded mode in the event of an ASIC failure there. If one of the four address buses fails, the remaining buses can be used to access all the system resources. The data crossbar is divided into separate halves, so it can operate at half-bandwidth if an ASIC fails.

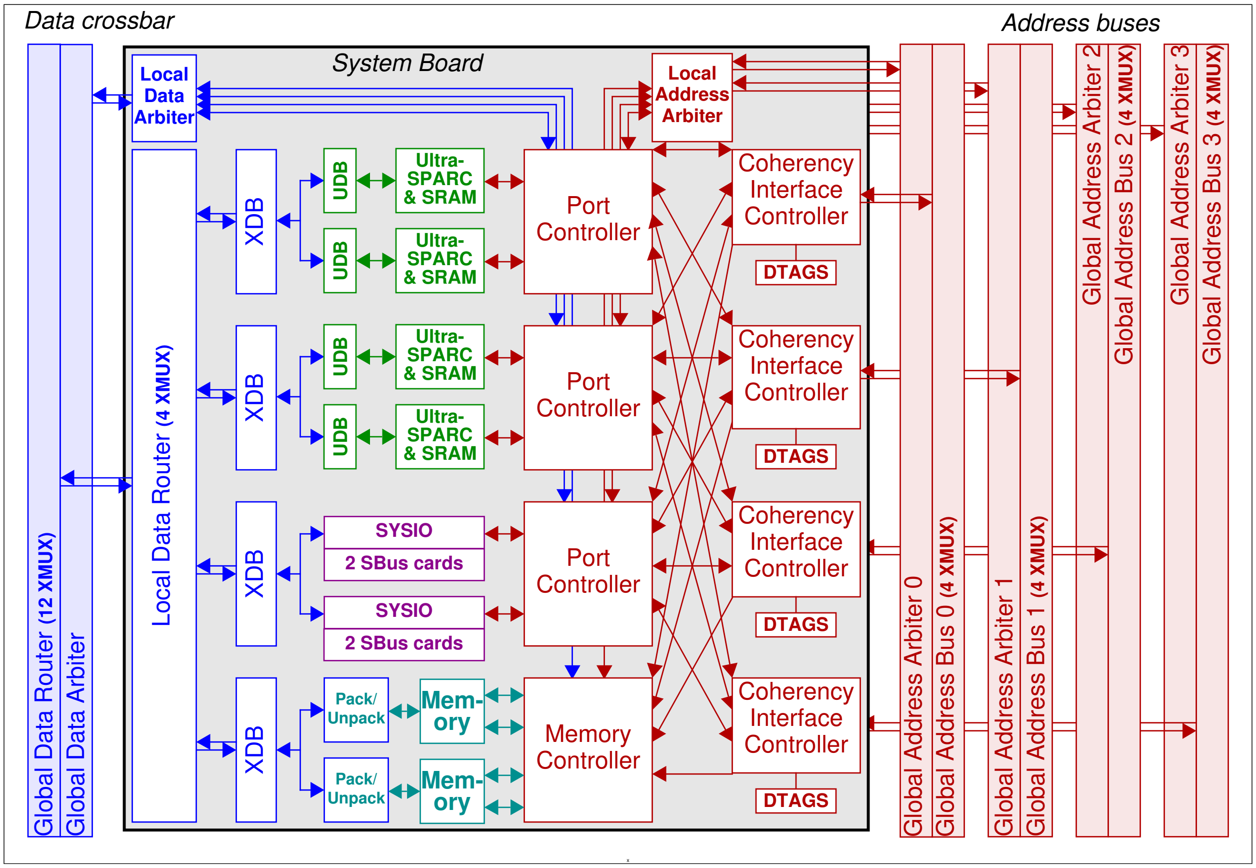
A fully redundant system will always recover from a system crash, utilizing standby components or operating in degraded mode.

3.3.4 Hot swap. We provided each board with its own DC-to-DC voltage converters, so a board can be powered on and off individually. The point-to-point interconnect allows the rest to the system to keep running undisturbed while a powered-off board is removed or reinserted.

3.3.5 Robust packaging. We designed the system boards for convenient and error-free handling during service. They are encased in a metal housing, but are under 30 pounds in weight. Board connectors are keyed so that boards cannot be plugged in upside down. The control and centerplane support boards are keyed such that they cannot be placed in each other's slots. The system boards have LEDs that indicate when the power to the board has been turned off, making it safe for removal from a running system.

Please see [16] for more on Starfire RAS.

Figure 6. Starfire router-based interconnect.



4. Starfire Interconnect

Both the UE6000 and Starfire have two-level interconnects.

1. The intra-board interconnect serves to conduct traffic between the on-board processors, I/O interfaces, memory banks, and the off-board address and data ports.
2. The inter-board interconnect is a bus in the UE6000, and point-to-point routers in the Starfire.

The larger number of processors, I/O interfaces, and memory banks in Starfire required an extra layer of interconnect chips on both the address and data portions to handle the increased fan-in and fan-out.

- **Address**. The snooping and memory control functions of the UE6000's Address Controller are split into the Starfire's Port Controller, Memory Controller, and System Controller.
- **Data**. The XDB buffer chips are interposed between the processors, I/O interfaces, memory banks, and the off-board port.

4.1 Address interconnect functional units

Address packets take two cycles to convey address information over one of the address buses. As explained above, the address buses act *logically* as buses, but are *physically* implemented with point-to-point connections and broadcast router ASICs.

1. The **Port Controller** (PC) accepts processor and I/O-bus requests and forwards them to one of the four Coherency Interface Controllers on its board. It typically uses the two low-order block address bits to determine which address bus to use.
2. The **Coherency Interface Controller** (CIC) maintains cache coherency for the caches on its board. It generates ECC for outgoing addresses sent to its

address bus, and checks and corrects the ECC for incoming addresses. It reads and writes the Duplicate Tag RAM on alternate cycles. It does domain-group filtering to remove addresses that are outside the shared-memory ranges of a group of domains.

3. The **Memory Controller** (MC) filters the memory addresses from all four address buses to determine which ones are targeted at its four memory banks. It generates the signals to read and write the DRAMs.
4. The **Local Address Arbiter** (LAARB) arbitrates requests from the units on its board to access the address buses. A mode of the XARB ASIC.
5. The **Global Address Arbiter** (GAARB) arbitrates requests for the use of its address Bus. It does domain filtering to isolates one domain-group's transactions from another's. A mode of the XARB ASIC.
6. The **Global Address Bus** (GAB) connects together a Coherent Interface Controller from each system board. It functions like a snoopy bus for coherency purposes, but is really a point-to-point router. There are four address buses in the system. Each bus is implemented from four XMUX ASICs.

4.2 Data interconnect functional units

Data blocks require four cycles to move 64-bytes of data over the data interconnect. In the case of a load miss, the missed-upon quadword is sent first.

1. The **UltraSPARC Data Buffer** (UDB) buffers data blocks moving between the UltraSPARC processor and the UPA databus. It generates ECC for outgoing data blocks, and checks and corrects ECC for incoming data blocks. It is located on the standard UltraSPARC Module. Implemented from two UDB chips.
2. The **Pack/Unpack** assembles four incoming 18-byte pieces from the interconnect into a complete 72-byte block for storing into memory. It also disassembles outgoing 72-byte blocks from memory into four 18-byte pieces for transmission on the interconnect. Two XMUX ASICs.
3. The **Xfire Data Buffer** (XDB) buffers outgoing data blocks through a two-location FIFO, and incoming data blocks through a 256-location buffer. It checks ECC.
4. The **Local Data Arbiter** (LDARB) arbitrates requests from a system board to access the data interconnect. A mode of the XARB ASIC.
5. The **Local Data Router** (LDR) connects the four Xfire Data Buffers on a system board to the Global Data Router. Four XMUX ASICs.
6. The **Global Data Arbiter** (GDARB) arbitrates requests for use of the crossbar. Two are used in a system for redundancy.
7. The **Global Data Router** (GDR) is a 16 x 16 crossbar that connects together the Local Data Routers on each system board. Twelve XMUX ASICs.

4.3 Interconnect ASICs

Six of our seven ASICs (PC, MC, CIC, XARB, GDARB, and XDB) implement an entire functional unit on a single chip. The seventh ASIC (XMUX) is a multiplexer part that is used to implement the local and global routers.

The ASICs are implemented in 3.3 volt, 0.5 micron CMOS technology. The largest die size is 9.95 x 10 mm, with up to five metal layers. They are all packaged in 32 x 32 mm Ceramic Ball Grid Arrays (CBGAs) with 624 pins. There are 18 ASICs on each system board, and 34 on the centerplane.

4.4 Boards

Three board types plug into the centerplane:

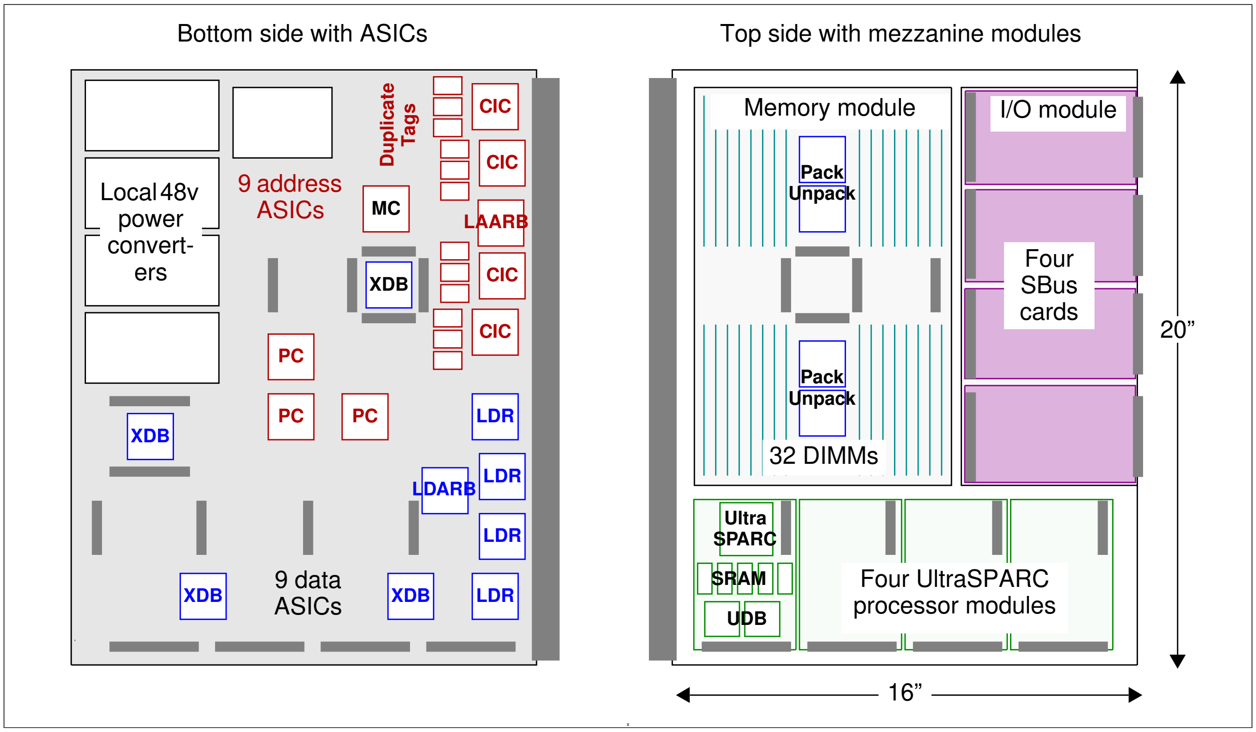
4.4.1 System board. The system board accepts three types of mezzanine modules on its top side: the memory module, the I/O module, and four processor modules. The bottom side of the system board has nine address ASICs, nine data ASICs, and five 48 volt power converters. The board is 16” x 20” with 24 layers.

4.4.2 Centerplane support board. Each of the two centerplane support boards provides resources for one-half of the router ASICs on the centerplane. It provides the secondary clock tree, JTAG interfaces, and DC-to-DC power conversion for the centerplane logic. It is 7.75” x 16”, and has 14 layers.

4.4.3 Control board. The control board provides JTAG scan information to the System Service Processor via an Ethernet connection. The control board provides fail-safe logic to shut down the machine when an emergency event is detected, and the SSP cannot do so.

The control board is the source for the system clock and system reset. It controls the bulk power supplies, and the cooling system, and has eight programmable LEDs. For redundancy, a Starfire may be configured with two control processor boards, each of which feeds clocks and JTAGs to the system boards. It is 12.75” x 16”, and has 14 layers.

Figure 7. System board.



4.5 Mezzanine modules

Three types of mezzanine modules mount on top of the system board.

4.5.1 Processor module. Starfire uses the same Ultra-SPARC processor module as the rest of the Ultra Enterprise server family. Currently, the module has a 250 MHz processor, 4 MB of external cache SRAMs, and two UltraSPARC buffer chips. It is approximately 3.5” x 5”.

4.5.2 Memory module. We placed Starfire’s memory on a mezzanine module to leave more room on the system board for routing traces. The memory module contains four 576-bit-wide banks of memory, each composed of eight 72-bit-wide DIMMs. The memory module contains up to 1 GB of memory with 16-Mbit DRAMs, and up to 4 GB of memory with 64-Mbit DRAMs. It is approximately 14.5” x 9.5”.

A memory module can do a 72-byte access every four system clocks (50 ns), which is a bandwidth of 1,280 MBps. This matches the bandwidth of both a cross-bar port and a processor’s UPA databus. ECC is generated and corrected by the processor module; the memory simply contains the extra storage.

An individual DRAM bank can do a read cycle every 11 clocks, or a write cycle every 13 clocks. The worst-case memory module bandwidth — if all the traffic were targeted to only one of the four banks — is 444 MBps.

4.5.3 I/O module. We placed Starfire’s I/O interface on a mezzanine module to allow us the option to connect to different I/O buses without affecting the system board. It is approximately 14.5” x 6”.

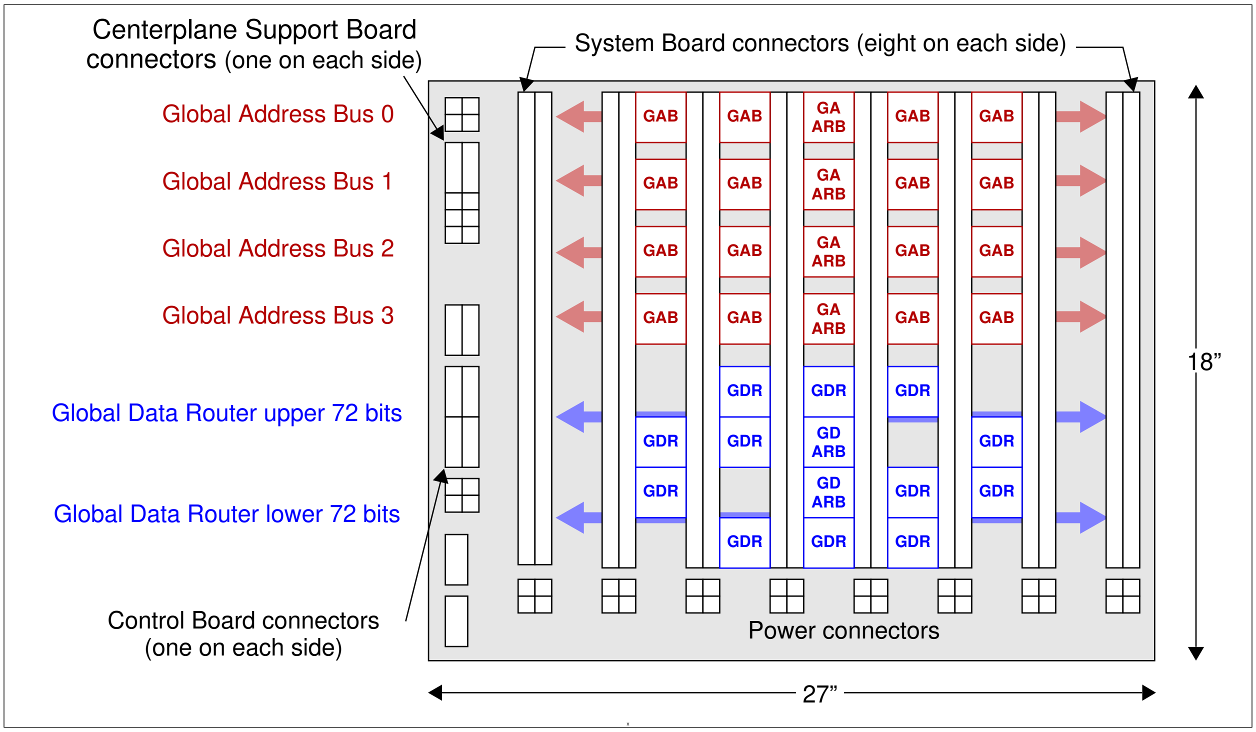
The current I/O module interfaces between the UPA and two SBuses, and provides four SBus card slots. It bridges between the UPA and SBus with the same SYSIO ASIC as the other Sun workstations and servers use. Each SBus has an achievable bandwidth of 100 MBps.

4.6 Other cards

4.6.1 DIMMs. 32 Dual Inline Memory Modules (DIMMs) plug perpendicularly into a memory mezzanine module. Starfire uses a JEDEC-standard 72-bit wide, 168-pin ECC DIMM. It holds 18 fast-page-mode DRAMs, and provides 32 MB of memory with 16-Mbit chips, and 128 MB of memory with 64-Mbit chips.

4.6.2 SBus cards. Four SBus cards lay flat on top of the I/O mezzanine module. Starfire uses the same commodity SBus cards as the other Sun workstations and servers. They are approximately 3.5” x 7.5”.

Figure 8. Centerplane.



4.7 Centerplane

The centerplane is 27” wide x 18” tall x 141 mils thick. It has an aspect ratio of 10, and 28 layers (14 signal and 14 power). It has space for 34 ASIC packages, and sockets on each side for eight system boards, a centerplane support board, and a control board.

The net density utilization was nearly 100%. Approximately 95% of the nets were routed by hand. There are 14,000 nets, approximately two miles of etch, and 43,000 holes. The board is manufacturable using standard process technology, and it is capable of rework.

4.7.1 Centerplane ground rules. The centerplane ground rules included:

- triple tracking through CBGA pin fields
- quad tracking through connector pin fields
- 50 +/- 5 ohm allowed impedance variation
- 0.008” buried vias (used between signal pairs)
- 0.012” plated thru-hole vias (finished)
- 0.028” via anti-pads (22 physical + 5 keep-out)
- 3 mil lines with 5 mil spaces

4.7.2 Board spacing. The board spacing is 3” to allow enough air flow to cool the four 45-watt processor modules on each system board.

To accommodate 16 CPU board slots and to run the channels at a 10 ns cycle time, the channel's physical length had to be minimized. The distance between adjacent boards (front-to-back) is approximately 1”. The maximum wire length is approximately 20”.

4.7.3 Clocking. The clock sources are distributed through the centerplane to each system board ASIC. Skew between clock arrivals at each ASIC is minimized by routing all traces on identical topologies. Every clock is routed through equivalent buffering and over the same 100 ohms differential impedance trace length, adjusted for loading differences.

LVPECL clocks are distributed through three layers of buffers and two layers of board routing. Worst case skew is less than 0.9 ns, which includes output skew of three layers of buffering and dielectric constant variation between system boards and centerplane. Clocks are buffered on each ASIC by a phase-locked loop (PLL). The PLL allows us to zero out the on-chip clock insertion delay.

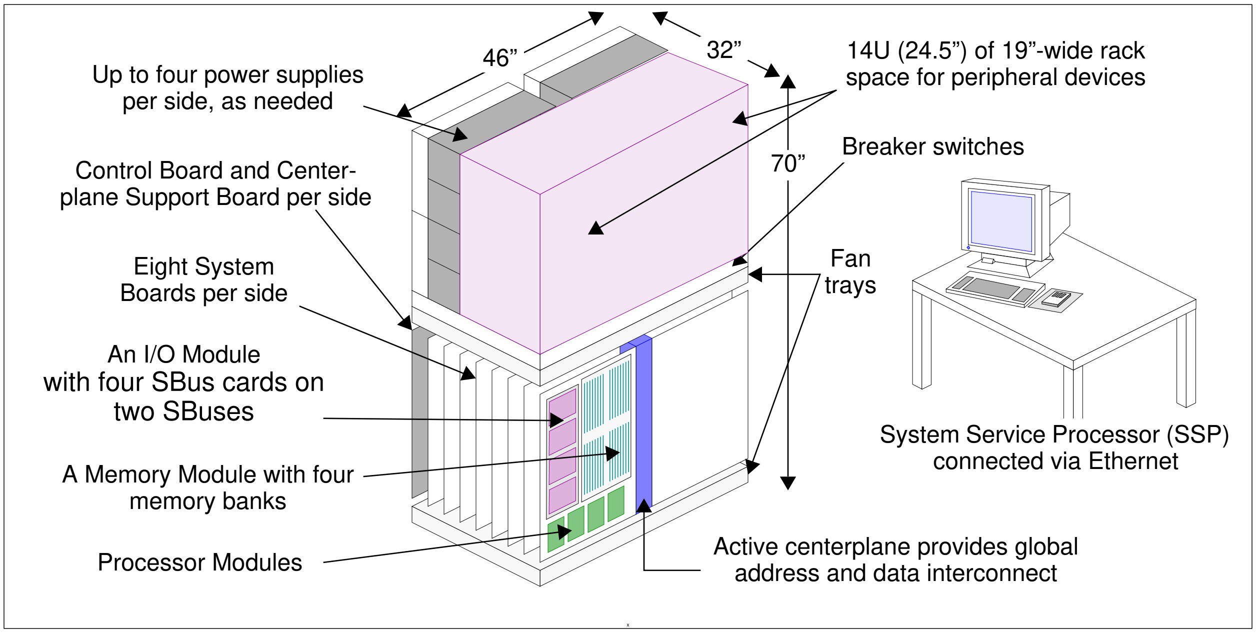
4.7.4 Electrical design. The system is designed to run at 100 MHz under the worst-case voltage and temperature conditions. Uni-directional point-to-point source terminated CMOS was used to realize the 144-bit-wide 16 x 16 data cross bar and the four 48-bit-wide address broadcast routers.

The system clock must be 1/2, 1/3, or 1/4 of the processor clock. The current 4 MB-cache processor module runs at

250 MHz, which determines the system clock to be 83.3 MHz.

During the design phase, we ran hundreds of SPICE electrical simulations with different combinations of impedance, voltage, process, temperature and switching patterns to optimize the system noise margin. We did extensive cross talk analysis, and developed and implemented a novel method for absolute-crosstalk minimization on long minimally-coupled lines.

Figure 9. Starfire system cabinet layout.



4.8 System cabinet

The system cabinet is 70” tall x 32” wide x 46” deep. It holds processors, memory, I/O interfaces. It has 24.5” of 19”-wide rack space, which can be configured with three Removable Storage Modules. These can each hold seven SCSI drives, which at the current maximum drive size of 9 GB makes a total of 189 GB that can fit in the system cabinet. The rest of a system’s peripherals are housed in standard Sun peripheral racks.

4.8.1 Power. Starfire’s power system is organized into three layers:

1. Up to four 220 VAC line cords connect through line filters to a maximum of eight 48 VDC bulk power supplies.
2. The 48 VDC is protected and distributed to each slot.
3. The boards locally convert from 48 VDC to 3.3 VDC and 5 VDC.

Starfire connects via standard IEC309 plugs to between one and four 30 Amp, 180 – 264 VAC, 50 or 60 Hz, single-phase circuits. Each AC circuit powers a pair of bulk AC to 48-volt-DC power supplies. The maximum configuration of eight supplies can power a maximum system with *N*+1 redundancy. Power supplies are added as needed when the system is upgraded.

We selected 48-volt distribution because of its safety — voltages of less than 60 volts are defined by the safety agencies as *extra-low voltages* that pose no shock hazard.

The outputs of the power supplies are connected to two power distribution boxes, each of which provides 11 circuits: eight for system boards, one for the control board and centerplane support board, and one each for the top and bottom fans.

Each board receives a 48-volt circuit from the power distribution box, which is protected by a 20 amp circuit breaker. The board converts the 48 volt power to the 3.3 volt and 5 volt levels required for the logic.

4.8.2 Cooling. Starfire uses four fan shelves: one each above and below the front and rear card cages. Three fan trays hot plug into each fan shelf. Each fan tray is two fans deep and one fan wide, and cools approximately two system boards and a portion of the centerplane. Fan speed is automatically controlled relative to component and ambient temperature — to reduce acoustic noise in normal environmental conditions.

5. Interconnect operation

Starfire's interconnect operates in four phases:

1. The **address** phase broadcasts the target address over an address bus, and maintains cache coherency.
2. The **read** phase gets the data block from the source unit.
3. The **data** phase moves the data block from the source board to the destination board over the data interconnect.
4. The **write** phase puts the data block into the destination unit.

To illustrate the operation of the interconnect, the subsections below show the steps involved in performing a load-miss request to memory.

5.1 Address phase

In the address phase, the address is broadcast to all the Coherency Interface Controllers on a particular bus, so that they can pass the address on to their memory controllers, and can maintain coherency between all the caches in the system.

1. The **processor** has a load miss, and makes a UPA address bus request to its Port Controller.
2. The **Port Controller** evaluates the requests from both of its processors, and picks this one to pass to the Local Address Arbiter and to the Coherency Interface Controller for the address bus being used.
3. The **Local Address Arbiter** evaluates all this board's requests, and picks this one to send to the Global Address Arbiter of the bus being used.
4. The **Global Address Arbiter** evaluates all the requests for its bus, and picks this one. It sends *grants* to the Local Address Arbiter and the Coherency Interface Controller.
5. The **Coherency Interface Controller** generates ECC for the address, and sends it to its Global Address Bus.
6. The **Global Address Bus** broadcasts the address to all of its Coherency Interface Controllers.
7. All the **Coherency Interface Controllers** pass on the address on to their Memory Controllers.

   All the **Coherency Interface Controllers** check and correct the address ECC. They look up the address in their Dtag SRAMs. They all send their snoop results back to the Global Address Arbiter for their bus.
8. The **Global Address Arbiter** ORs the snoop results together, and returns the global snoop result to all of its Coherency Interface Controllers.
9. All the **Coherency Interface Controller** use the global snoop result to decide what to do locally. If there had been a hit for this address, then the CIC for the target memory would have needed to *abort* the memory access, so the block would come from the owning cache instead.

5.2 Read from memory

Memory supplies data to its local Xfire Data Buffer, which is its gateway to the data interconnect.

1. The **Memory Controller** checks and corrects the ECC of the address. It recognizes the address as one of its own. It arbitrates for the datapath between the Pack/Unpack unit and the DRAM bank. It generates the signals to make the DRAM do a read cycle. It requests the data interconnect from the Local Data Arbiter.
2. The **DRAM** bank reads the data block to the Pack/Unpack.
3. The **Pack/Unpack** unpacks the 72-byte-wide data block into four 18-byte wide chunks, and sends them to the Xfire Data Buffer.

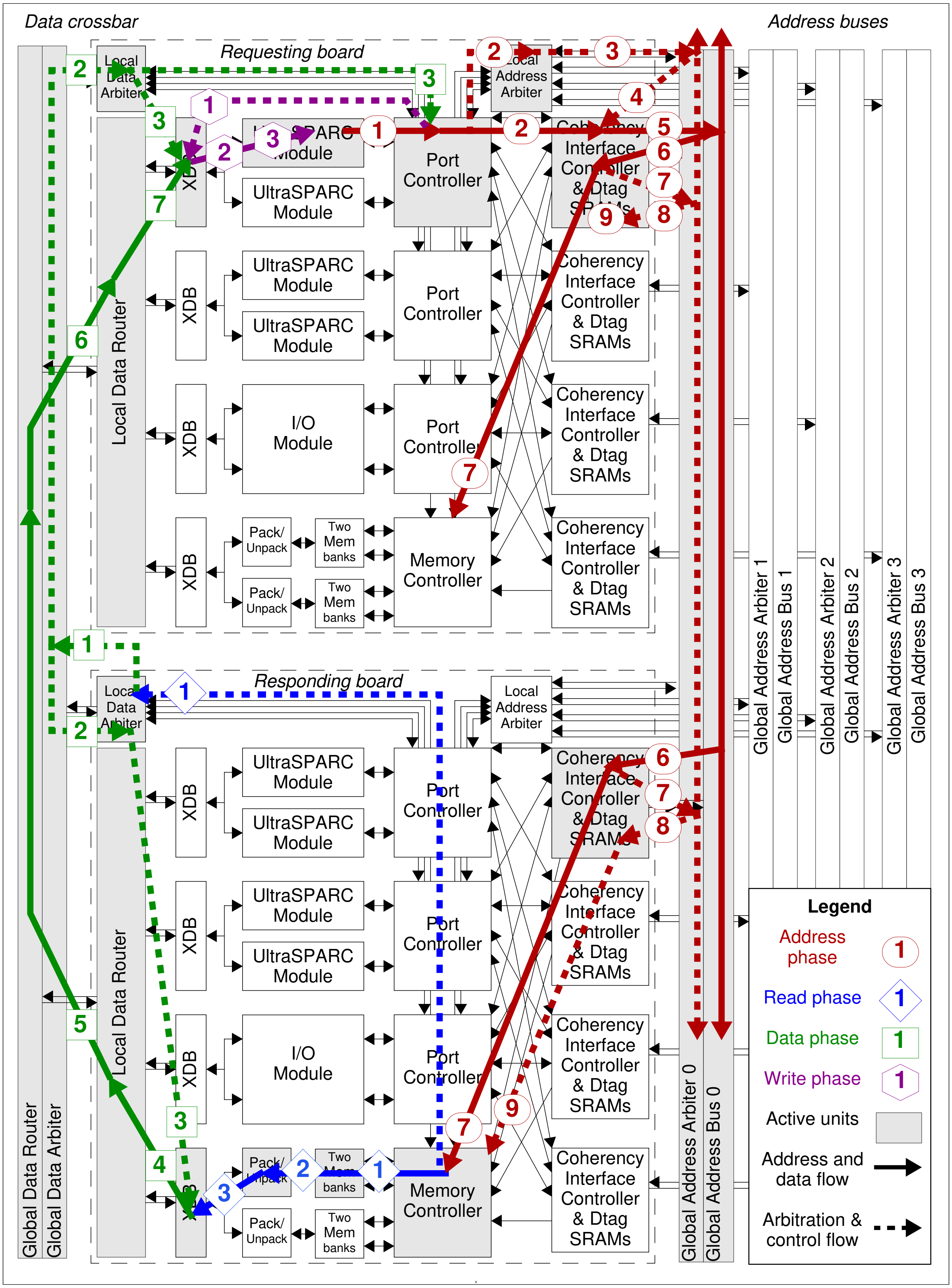
5.3 Data phase

The data interconnect moves the data block from the Xfire Data Buffer on the source board to the Xfire Data Buffer on destination board.

1. The **source Local Data Arbiter** evaluates all the local requests for the crossbar, and picks this one to send to the Global Data Arbiter.
2. The **Global Data Arbiter** evaluates all the system-wide requests for the crossbar, and selects this one to transmit. It sends a *grant* to the source Local Data Arbiter, and a *notify* to the destination Local Data Arbiter.
3. The **source Local Data Arbiter** tells the source Xfire Data Buffer to begin the data transfer.

   The **destination Local Data Arbiter** notifies the destination Port Controller that data is coming, and tells the destination Xfire Data Buffer to get ready.
4. The **source Xfire Data Buffer** checks ECC, and sends the data to its Local Data Router.
5. The **source Local Data Router** sends the data to the Global Data Router.

Figure 10. Interconnect steps for a load-miss from memory.



6. The **Global Data Router** sends the data to the destination Local Data Router.
7. The **destination Local Data Router** sends the data to the destination Xfire Data Buffer.

5.4 Write into processor

The processor gets the data needed to satisfy its load miss.

1. The **Port Controller** sends the Xfire Data Buffer the address for writing and then reading the incoming data block. It evaluates this request for its UPA databus, and selects this one.
2. The **Xfire Data Buffer** checks ECC and sends the data to the UltraSPARC Data Buffer.
3. The **UltraSPARC Data Buffer** checks and corrects ECC, and sends the data to the UltraSPARC processor.

5.5 Latency comparison: bus and routers

Table 2 compares the minimum timeline for a load-miss to memory for the two interconnects. This *pin-to-pin* latency starts with the cycle that the address request is sent out on the processor's address pins, and lasts until the cache block is returned on the processor's data pins.

Table 2. Timeline for a load-miss to memory.

Cycle number for	UE6000	Starfire
Address out of processor	1	1
Address onto target board	4	10
Data out of memory	12	26
Data onto requesting board	14	34
First 16 bytes into processor	18	38

A bus takes only one clock to move information from the sender's output pins to the receiver's input pins. A routed point-to-point connection take three clocks: a cycle on the wires to the router, a cycle inside the routing chip, and a another cycle on the wires to the receiving chip.

Buses have lower latency than routers, and are the preferred interconnect topology for as large as they can be implemented.

Relative to the UE6000, Starfire's larger number of UPA ports forced an additional layer of address and data ASICs on each board. Each extra chip layer costs at least two cycles: one on the wires into the chip, and another inside the chip.

Starfire's designers used routers with point-to-point interconnects to emphasize bandwidth, partitioning, and RAS.

The UE3000-6000 designers used a bus to optimize for low latency and economy.

The Ultra Enterprise family combination of bus topology from 4 to 30 processors, and point-to-point interconnect from 32 to 64 processors has proven quite successful in the marketplace against the more exotic NUMA architectures.

6. Interconnect performance

This section compares the measured latency and bandwidth of the router-based Starfire interconnect with the bus-based UE6000 interconnect.

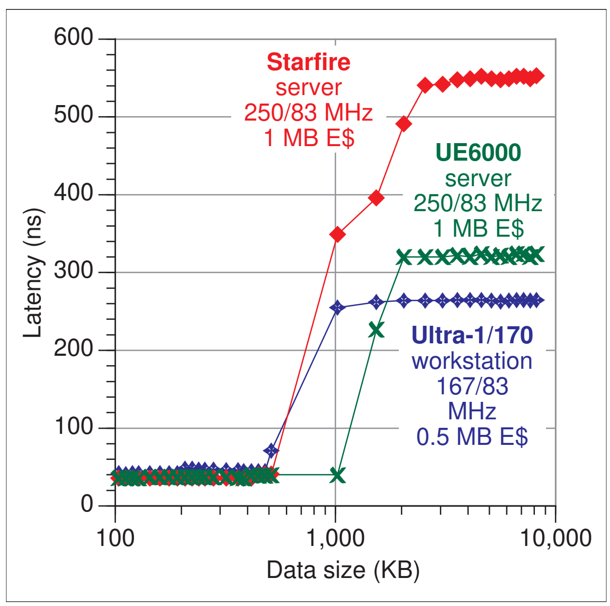
6.1 Latency

The *lat_mem_read* C-language kernel of lmbench [8] is the defacto-standard latency benchmark. It measures back-to-back load latency for a *single* processor. This is the time per load, assuming that the instructions before and after are also cache-missing loads.

The benchmark creates a ring of pointers that points forward by the stride size. Traversing the array is done in a *for* loop of p = (char **)*p.

Looking at the profile of latency versus data size in Figure 11, we can see the external cache and memory-latency plateaus. The external cache has a nine processor-clock latency.

Figure 11. lmbench latency profiles.

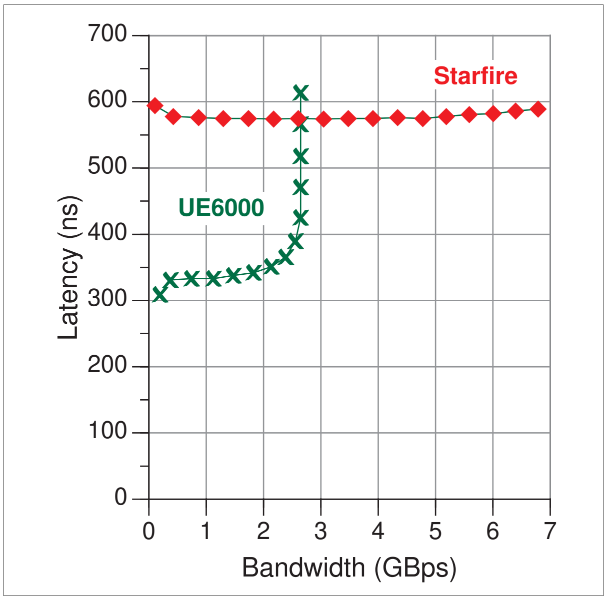


Memory latency is shortest for the workstation, because that system is small enough to fit on one board, and there is only one processor to connect to a few DIMMs. Servers have longer latencies because they need off-board wires and several layers of logic to connect multiple processors to large amounts of DRAM.

The bus-based UE6000's *lmbench* latency is about 300 ns, and the router-based Starfire's latency is about 550 ns.

Figure 12 shows the latency when we ran simultaneous copies of the pointer-traversing loop on multiple processors. The total bandwidth available to the processors increases linearly, and the latency stays flat until the interconnect saturates.

Figure 12. Latency versus bandwidth for multi-processor pointer-chasing.

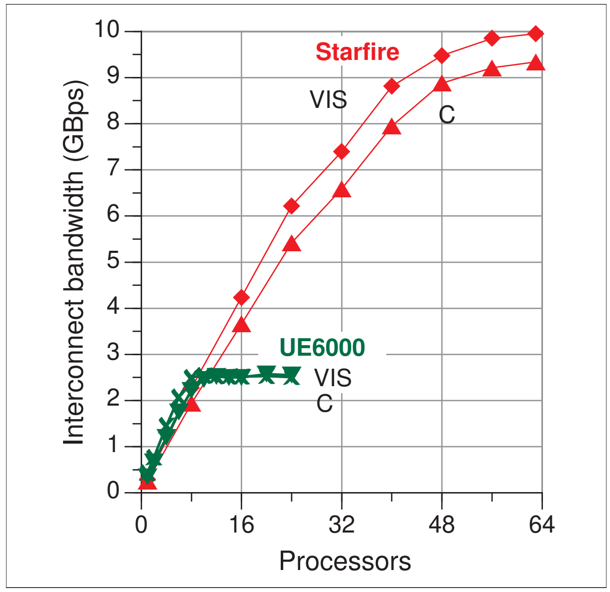


6.2 Bandwidth

The Stream benchmark [7] is the defacto-standard memory-bandwidth benchmark. It is available in both C and Fortran versions. Stream measures stride-1 memory bandwidth using four vector loops: copy, scale, add, and triad. Multiprocessor operations are done in parallel do/for loops. These measurement were done using SunPro C 4.0.

Figure 13 shows the average interconnect bandwidth generated by the four Stream vector loops, including traffic generated by write-allocates in cache caused by store misses. The bandwidths rise smoothly to the respective peak bandwidths of 2,667 MBps for the UE6000, and 10,667 MBps for Starfire.

Figure 13. Stream interconnect bandwidth.



The VIS assembler code is more latency tolerant than the C code, which was compiled with a version of the compiler that did not generate software prefetches. The VIS bandwidths rise somewhat faster to the peak bandwidths. We observed bandwidths up to 97% of peak on the Starfire, and up to 99% of peak on the UE6000.

It must be remembered that Stream measures a worst-case scientific-computing vector workload. Database applications typically use less bandwidth. Bus-counter measurements on the UE6000 show that 83% of its peak bandwidth was consumed by 24 processors running TPC-C.

6.3 System benchmarks

The Starfire has set several application-benchmark world records, including the 300 GB and 1 TB sizes of TPC-D, but these results are beyond the scope of this paper. For more information on the Starfire, see Sun's web site at http://www.sun.com/servers/datacenter.

7. Conclusion

Starfire's router-based implementation of the Ultra Port Architecture has extended the Ultra Enterprise family bandwidth by a factor of 4x. The error isolation of point-to-point wires makes possible the flexibility of dynamic system domains, and improves system availability and serviceability.

Our challenge for future enterprise-server generations will be to continue scaling bandwidth upward, and to keep latency as low as possible.

We have found that the combination of snoopy coherency for the lower portion of the product family, and routed coherency for the upper portion of the family to be a good strategy.

Starfire's high-bandwidth centerplane router, dynamic system domains, and enhanced RAS have proven to be a success in the marketplace. We are carrying forth these technologies into future generations.

8. References

[1] William R. Bryg, Kenneth K. Chan, and Nicholas S. Fiduccia, "A High-Performance, Low-Cost Multiprocessor Bus for Workstations and Midrange Servers," *Hewlett Packard Technical Journal*, Feb. 1996. http://hpcc997.external.hp.com:80/hpj/feb96/fb96a2.pdf.

[2] Marvin Denman, Paul Anderson Mike Snyder, "Design of the PowerPC 604e Microprocessor," *Proceedings of Compcon 96*, pp. 126-131, http://www.mot.com/SPS/PowerPC/library/technical_papers/compcon/604compcon.pdf

[3] David M. Fenwick, Denis J. Foley, William B. Gist, Stephen R. VanDoren, Daniel Wissell, "The AlphaServer 8000 Series: High-end Server Platform Development," *Digital Technical Journal*, Vol. 7 No. 1 1995. ftp://ftp.digital.com/pub/Digital/info/DTJ/v7n1/The_AlphaServer_8000_Series_H_01jul1995DTJH04P8.ps.

[4] Linley Gwennap, "Sun, Xerox to License XDBus Technology," *Microprocessor Report*, March 8, 1993.

[5] Intel, *Dual Independent Bus Architecture*, 1997, http://developer.intel.com/design/PentiumII/prodbref/dibtech.htm.

[6] L. Kohn, G. Maturana, M. Tremblay, A. Prabhu, G. Zyner," The Visual Instruction Set (VIS) in UltraSPARC." *Proceedings of Compcon-95*, pp 462-469, San Francisco, March 1995.

[7] John McCalpin, "Memory bandwidth and Machine Balance in Current High Performance Computers," *IEEE Computer Society Technical Committee on Computer Architecture Newsletter*, pp. 19-25, December 1995, http://www.computer.org/tab/tcca/news/dec95/dec95_mccalpin.ps, and http://www.cs.virginia.edu/stream/.

[8] Larry McVoy and Carl Staelin, "Lmbench: Portable Tools for Performance Analysis," *Proceedings: USENIX 1996 Annual Technical Conference*, January 22-26, 1996, pp. 279-294, http://reality.sgi.com/employees/lm/lmbench/lmbench-usenix.ps.

[9] Kevin Normoyle, Zahir Ebrahim, Bill VanLoo, Satya Nishtala, "The UltraSPARC Port Architecture", *Proceedings Hot Interconnects III*, August 1995.

[10] Ashok Singhal, David Broniarczyk, Fred Cerauskis, Jeff Price, Leo Yuan, Chris Cheng, Drew Doblar, Steve Fosth, Nalini Agarwal, Kenneth Harvey, Erik Hagersten, "Gigaplane: A High Performance Bus for Large SMPs." *Proceedings Hot Interconnects IV*, August 1996.

[11] Sun Microsystems, *The UltraSPARC II Processor Technology*, July 1997, http://www.sun.com/ultra30/whitepapers/ultrasparc.pdf.

[12] Sun Microsystems, *The Ultra 30 Architecture*, July 1997, http://www.sun.com/ultra30/whitepapers/u30.pdf.

[13] Sun Microsystems, *Ultra Enterprise X000 Server Family: Architecture and Implementation*, 1996, http://www.sun.com/servers/ultra_enterprise/arch.pdf

[14] Sun Microsystems, *The UltraEnterprise 10000 Server*, January 1997, http://www.sun.com/datacenter/whitepapers/E10000.pdf.

[15] Sun Microsystems, *Ultra Enterprise 10000: Dynamic System Domains*, January 1997, http://www.sun.com/datacenter/whitepapers/domains.pdf.

[16] Sun Microsystems, *Ultra Enterprise 10000 Server: SunTrust Reliability, Availability, and Serviceability*, January 1997, http://www.sun.com/datacenter/whitepapers/suntrust.pdf.

[17] Sun Microsystems, *Consolidation and Recentralization with the Ultra Enterprise 10000*, http://www.sun.com/servers/datacenter/whitepapers/consolidate.pdf.

[18] Kenneth C. Yeager, “The Mips R10000 Superscalar Microprocessor,” *IEEE Micro*, Vol. 16, No. 2, April 1996.