Gigaplane™: A High Performance Bus for Large SMPs

Ashok Singhal[1], David Broniarczyk, Fred Cerauskis, Jeff Price, Leo Yuan, Chris Cheng, Drew Doblar, Steve Fosth, Nalini Agarwal, Kenneth Harvey, Erik Hagersten
Sun Microsystems, Inc.
Bjorn Liencres[2]

1.0 Introduction

Snooping buses provide a well-understood and relatively simple interconnect for building cache-coherent shared-memory multiprocessors (SMPs). However, it is commonly believed that buses have "run out of steam", and that one can get at most two of the following three properties in bus-based SMP:

1. Large number of devices (CPUs)
2. High bandwidth
3. Low latency (e.g. for cache misses)

Consequently, numerous manufacturers have resorted to building large shared-memory systems using clusters of SMP building blocks, each with only a small number of processors on relatively low bandwidth bus, connected by a network with cache coherence implemeted using directory-based "distributed shared memory" (DSM) mechanisms. While DSM-based clusters may indeed provide sufficient aggregate bandwidth to scale to a very large number of processors, they incur rather severe penalties (for example, in software complexity, hardware complexity and remote access latencies) when applied to the number of processors that can be accomodated on a bus. Our design philosophy, therefore, was to push bus technology as far as we could and only then resort to clustering.

This paper describes Gigaplane, a snooping bus designed to support all three: a large number of CPUs, high bandwidth and low latency.

Gigaplane mechanical and board designs, described in Section 2.1 , result in a modular, cost-effective and highly configurable system.

One reason for the high bandwidth of the bus is the ability to switch bus drivers without inserting a dead cycle, which results in 33% savings in bandwidth. The careful electrical design that made this possible despite a long and heavily loaded bus is described in Section 2.2 .

---

1. Email: ashok.singhal@Eng.Sun.Com, phone: (415)-786-6091, fax: (415)-568-9603
2. Contributed to this work while at Sun Microsystems.

Careful logical bus protocol design, described in Section 3.4 , results in very low latencies as well as excellent scalability even in the presence of contention and hot-spots. Apart from raw performance, the Gigaplane logical protocol design includes a number of innovations:

- A split transaction snooping protocol with no transient states, which minimizes the interactions between different transactions,
- A protocol that permits a large number of outstanding transactions (112 system-wide, 7 from each board) with out-of-order responses allowed, which supports devices with multiple outstanding transactions (and non-blocking caches), permits a great deal of pipelining and minimizes the effects on average latency of contention, hot-spots and long-latency devices,
- Live insertion ("hot-plug") of system boards into a running system.

2.0 Gigaplane Physical Design

2.1 Mechanical and Board Design

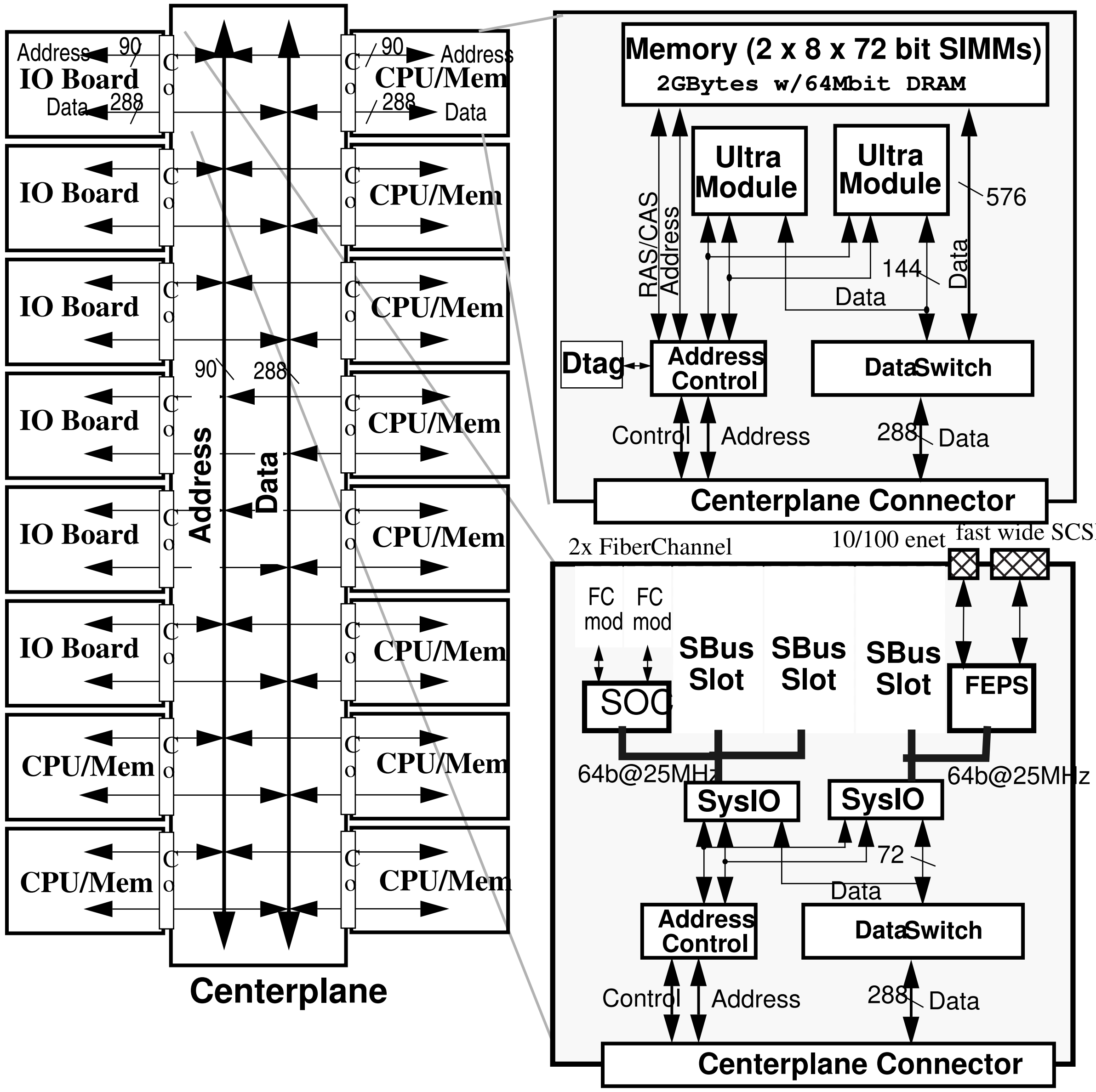
Gigaplane is designed as a centerplane with 8 board slots in the front and 8 board slots in the back (smaller configurations are also available). Figure 1 shows the physical structure of a Gigaplane-based, Ultra Enterprise 6000 system. There are currently 3 board types:

- a CPU/Memory board that can support up to two UltraSPARC CPU modules and up to two banks of memory. Each bank can be 64MB, 256MB or 1GB.
- A dual-SBus IO Board has 2 SYSIO SBus controllers, each of which provides a 64-bit, 25 MHz SBus with a sustainable bandwidth of over 100 MByte/sec along with a set of standard on-board devices including 10/100 MBit/s Ethernet, fast-wide SCSI, and two full-duplex Fiber Channel ports. The board provides 3 SBus slots.
- A SBus/FFB IO Board is similar to the dual-SBus board, except that one SYSIO and one SBus slot are replaced by a slot for a UPA port used for Creator or Creator3D frame buffers.

All board slots can accomodate any type of board.

Each board contains an Address Controller (AC) and an 8-way bit-sliced Data Controller (DC) that interface the Gigaplane to an on-board UPA (UltraSPARC Port Architecture) bus and to memory.

FIGURE 1. Gigaplane System Structure



2.2 Electrical Design

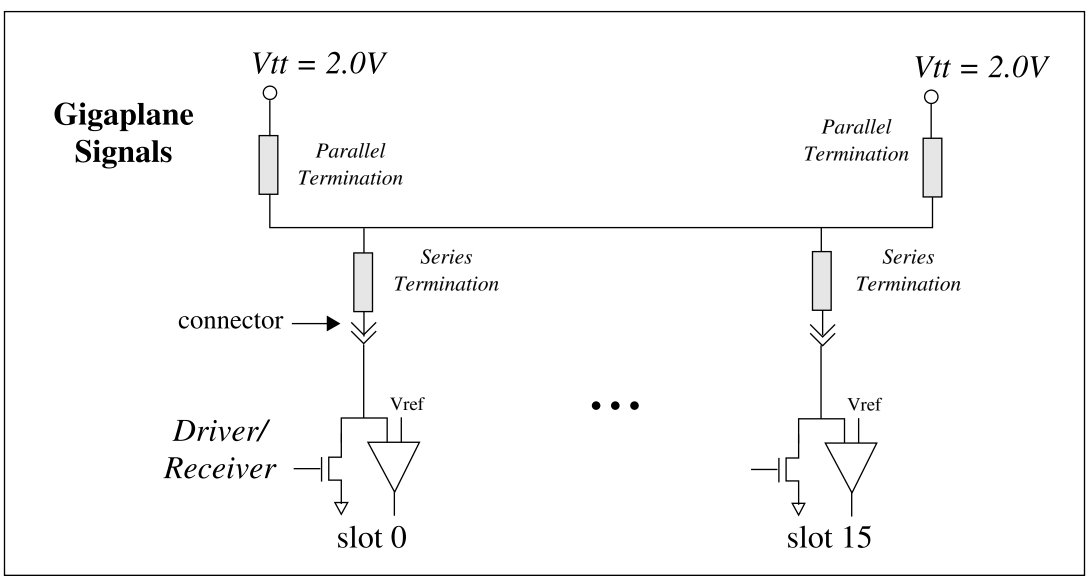
Gigaplane is designed to run at 83.5MHz under the worst case conditions to achieve the sustained bandwidth of 2.67 GByte/sec. A custom designed low voltage swing CMOS open drain type of transceiver was tuned to minimize the wire-or glitch in the Gigaplane bus such that the dead cycle can be eliminated when changing masters. During the design phase, thousands of SPICE simulations were run with different combinations of impedance, terminations, voltage, process, temperature and switching patterns to optimize the system noise margin.

Due to cooling considerations, the distance between boards is designed to 2.0" in order to allow enough air flow to cool two 45W processor modules on one single CPU board. To accommodate 16 board slots (up to 15 CPU boards and one I/O board) and run the bus at 12ns cycle time, the bus's physical length has to be minimized. A 20.5" wide centerplane was designed such that 8 boards can be plugged from either side of centerplane. The distance between adjacent loads is approximately 1.0". The actual bus length is approximately 16.0" including two terminators and 16 board loads.

2.2.1 Impedance and Termination

Gigaplane is a heavily loaded bus. To minimize the bus loading delay, the intrinsic impedance of the unloaded trace is designed to be around 27 ohms. This low impedance centerplane reduces bus loading delay by 46% compared to a nominal 50 ohms impedance backplane. To minimize the wire-or glitch in the bus, the signal low-to-high transition needs to be controlled. The slower the low-to-high transition, the lower the wire-or glitch noise. The signal's low-to-high transition is slowed down but still ensures first incident switching under the worst case conditions. Both parallel and series terminations methods are applied to the Gigaplane bus. The bus signal's low-to-high transition is controlled by the RC time constant of the parallel terminations and loaded bus capacitance. By using different values of parallel termination resistors, the signal low-to-high transition time can be customized. The bus cycle time is relatively small compared to the electrical length of the bus, therefore the reflections take more than one clock cycle to settle. The series termination is used to damp the reflection in the bus as well as to increase the source impedance of the driver to further reduce the wire-or glitch.

FIGURE 2. Gigaplane Signal Termination



2.2.2 Clocking

Gigaplane 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. These LVPECL clocks are distributed through two layers of buffers and three layers of board routing. Worst case skew is less than 0.6ns, which includes output skew of two layers of buffering and dielectric constant variation between system boards.

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. The Gigaplane clock source is programmable.

3.0 Gigaplane Logical Design

3.1 Signals

Gigaplane consists of an Address Bus, which carries a 2-cycle address packet (transaction request) and a 288 bit wide Data Bus (256 data + 32 ecc), which carries a 64-byte data packet (transaction response) in 2 cycles. An address packet includes a 41-bit physical address and a 7-bit SourceId that uniquely tags the transaction. Since data packets may be sent in a different order from the address packets, a data packet also carries a DataId tag to indicate which address packet is associated with. Other signals include snooping state signals, arbitration lines, clocks and parity.

3.2 Arbitration

Arbitration is required for both the Address and Data buses. Arbitration is required for the data bus since data packets need not be in address packet order. Arbitration is not required for other signals because they are driven with a fixed timing relationship to either the Address bus or the Data bus. An arbitration bus consisting of 16 dedicated lines, one per board, is used to implement a distributed, fair algorithm. Since address and data packets are two cycles each, arbitration for the two buses is done in alternate cycles on the same arbitration bus.

The distributed arbitration algorithm has a worst case latency of 2 cycles for the winner. This is sufficiently small that it can be completely hidden for the data bus. However, the Address Bus arbitration adds to the transaction latency. To reduce this latency, Gigaplane uses two 0-cycle arbitration techniques. The first is a straightforward bus parking method where the previous winner is allowed to drive the bus without arbitration. This technique's benefit is limited to the previous winner. The second technique allows any device to take advantage of a 0-cycle arbitration latency. In this mode, any device may drive the bus. If multiple devices happen to drive the bus, the "collision" is detected and the packet ignored. In parallel, the normal 2-cycle distributed algorithm is used to compute the new winner. At first glance, it may seem that this technique wastes bus bandwidth due to collisions. However, with the normal 2-cycle algorithm the collision slot would not have been used anyway so the bandwidth is not truly wasted.

A special mechanism is provided to synchronize the distributed arbitration state in order to bring a “hot-plugged” board on-line and in sync with the other boards. This can be done in a running system without the need for software to quisce the system.

3.3 Data Transfer

FIGURE 3. Signal Timing for a ReadToShare transaction with fast Address Arbitration

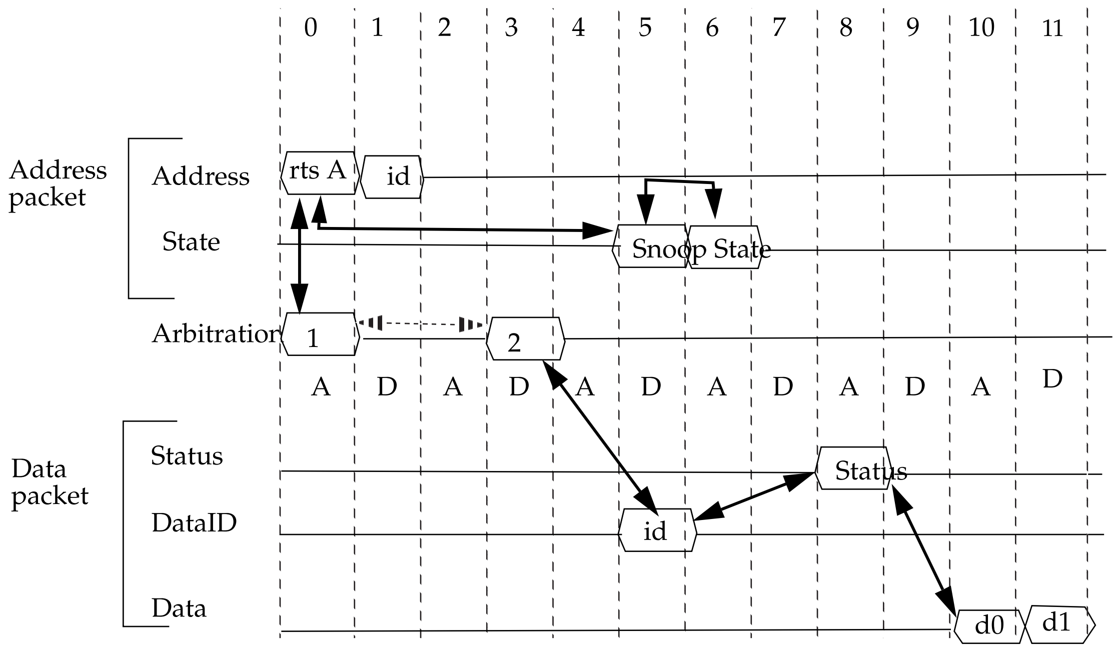


Figure 3 shows the transaction for a cache data read miss request. Solid arrows represent fixed timing relationships, dotted arrows represent variable timing relationships. The gap between the address and the data packets can vary (the figure shows the minimum spacing) and data packets can be in a different order from the request packets.

Note that the snoop state is associated with the address packet, not the data packet. Also, note that the DataId appears 5 cycles before the data and this allows sufficient time for the id match before the data arrives, therefore no additional delays are incurred because of out-of-order data responses. The data arbitration occurs well before the data which allows time for DRAM access. DRAM access can be speculatively started (to minimize read latency) before the snoop response is known, and a memory request for Owned data can be cancelled by asserting a Status signal 2 cycles before the data. Data bus bandwidth can be “wasted” by cancelling these speculative reads, but these cancellations do not occur for data packets that are delayed after the snoop response, either by a busy memory controller or by a busy data bus. There are also some transactions that do not require a data packet (e.g. a write miss with valid data but without exclusivity). In practice, on a busy system data bus utilization remains well matched to address bus utilization even for applications that involve a substantial number of cache misses that are satisfied from other caches instead of from memory.

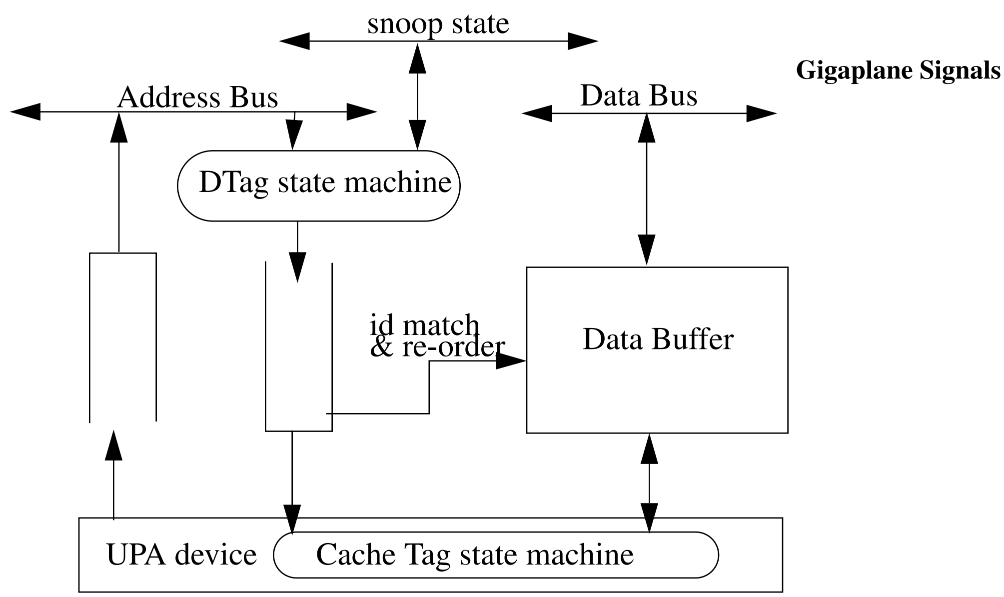
3.4 Transactions and Snooping Protocol

Gigaplane implements an invalidation-based snooping cache coherence protocol. The Address Controller (AC) has duplicate tags (DTags) for snooping for each device. Gigaplane implements a simple 3-state protocol in its DTags with states Owned, Shared and Invalid. The device cache implements a 5-state protocol with states Modified, Owned, Exclusive, Shared and Invalid.

3.4.1 A Protocol with no Transient States

In principle, the state machines of most snooping protocols are simple if the transitions are atomic. However, state transitions in moderm SMPs are not atomic because of split transactions and separate tag copies for snooping and the cache. In practice, snooping protocol state machines can be quite complicated because of transient states[8].

FIGURE 4. Protocol State Machine Operation



Unlike most split transaction snooping buses, the Gigaplane protocol has no transient states. The DTag state transitions appear logically atomic and occurs when the address packet appears on the address bus. The state change is actually written to the DTag SRAM a few cycles after the the address packet appears, but simple pipeline bypasses are used to present the logical appearance of instantaneous DTag transition. This is possible because snoop state timing is fixed relative to the address packet. Similarly, the cache tag transition occurs logically atomically and is associated with the data transaction at the UPA device alone. This is illustrated in Figure 4.

The key advantage of a protocol with no transient states is that it is simpler to pipeline multiple transactions, and to allow a large number of outstanding transactions (even if they are to the same address), since it is not necessary to keep track of the transient states for each of the outstanding transactions. This allows Gigaplane to support up to 112 outstanding transactions with out-order data responses.

3.4.2 Out of Order Data Responses

Many SMP bus protocols require that the data response be in the same order as the address request (examples: DEC Alphaserver 8400 and Pentium Pro buses). This results in higher latency in the presence of hot-spots, contention and long latency operations such as cache interventions and programmed IO reads to IO bus devices. Transactions following a long latency operation must be delayed. Techniques such as re-tried or pended transactions can be used to reduce the effects on latency, but these techniques increase the effective bus bandwidth used and result in additional complexity. Without the appropriate bus protocol support, allowing out-of-order data responses can be difficult. Quoting from [2], "With as many as 16 outstanding transactions (8 in the AlphaServer 8400/8200) active in the system at any one time, the task of producing a logic structure capable of retiring transactions in order is enormous. Furthermore, the retiring of transactions out of order complicates the business of maintaining coherent, ordered memory updates."

The Gigaplane bus protocol allows out-of-order data responses. The effects of contention, hot-spots and long latency operations are therefore limited to the transactions that directly involved with those operations or devices experiencing contention.

3.4.3 Large number of Outstanding Transactions

Most SMP buses also permit only a relatively small number of outstanding transactions (for example, DEC AlphaServer 8400 and SGI Challenge support only 8).

Gigaplane supports a total of 112 outstanding transactions, 7 from each board. Each board can support up to 14 transactions from its UPA devices. Multiple outstanding cache misses (including those with writebacks of dirty victims) and prefetches from a UPA device are allowed. This allows each UPA device to take advatage of over 1GByte/s UPA bandwidth.

Writebacks of dirty victims are handled after the read miss, with the data transfer for the writeback completely overlapped by the memory read for the miss.

3.4.4 Hot-Plug Capability

Gigaplane supports hot-plug and hot-unplug of boards into a running system without requiring that the machine is quisced by the operating system. This greatly improves the availability and serviceability of the system. Special long pins in the connector trigger an early warning to the system that a board is being plugged in. The bus interface ASICs then suspend all activity on the Gigaplane so that when the signal pins make contact, the electrical noise due to the contact does not result in errors. Special short pins in the connector cause activity to resume after all the signal pins have made contact. A similar procedure is used while unplugging boards.

Hot-plugged boards are not immediately incorporated into the system. An interrupt is generated to the operating system when a board is hot-plugged. Firmware and software must initialize the board, re-synchronize the arbitration state machines and then bring on-line the devices on the board.

The system also provides redundant power and cooling with hot-pluggable power and cooling modules.

3.4.5 Error Detection and Re-Configuration

Gigaplane provides ECC (single bit error correction, double-bit error detection, 4-bit nibble error detection) over the data path, and parity protection over all other signal paths. The system provides extensive diagnostics as well as automatic reconfiguration around failed components, including boards.

4.0 Gigaplane Performance

4.1 Latency

The Gigaplane design optimizes not only the latency of a transaction on the bus, but also enables efficient pipelining of the transaction request from the processor to the Gigaplane Address Bus and the response from the Gigaplane Data Bus back to the processor. This minimizes the latency seen by the processor pipeline.

TABLE 1. Cache read miss progress from/to processor module pins

Action	System Clock (12ns)
UltraSPARC cache miss request on the UPA Address Bus	1
Cache Miss request on Gigaplane Address Bus	4
First 32 bytes on Gigaplane Data Bus	14
First 16 bytes on UPA Data Bus (critical word first)	16
Last 16 bytes on UPA Data Bus	19

Table 1 shows the progress of a cache miss in system clocks (12ns) from the processor module and back. The critical word is available on the UPA (processor module) in 16 system clocks or 192 ns. The cache miss latency as seen by the processor pipeline is higher. If the data loaded is used immediately (as measured by lmbench [6] by following a chain of pointers, each of which results in a cache miss and memory reference), the latency seen by the pipeline is 306 ns. The high throughput, large number of transactions and out-of-order responses results in low latency even when there is contention. Figure 5 shows the e latency when multiple copies of the equivalent program are run simultaneously on multiple processors resuling in bus contention (bandwidth). Latency increases steeply only when the bus bandwidth limit is reached. Since the latency increases slowly with bandwidth, the total bandwidth available to the procesors running the program increases almost linearly until the bus bandwidth limit is reached, as Figure 6 shows.

FIGURE 5. Effects of contention on latency

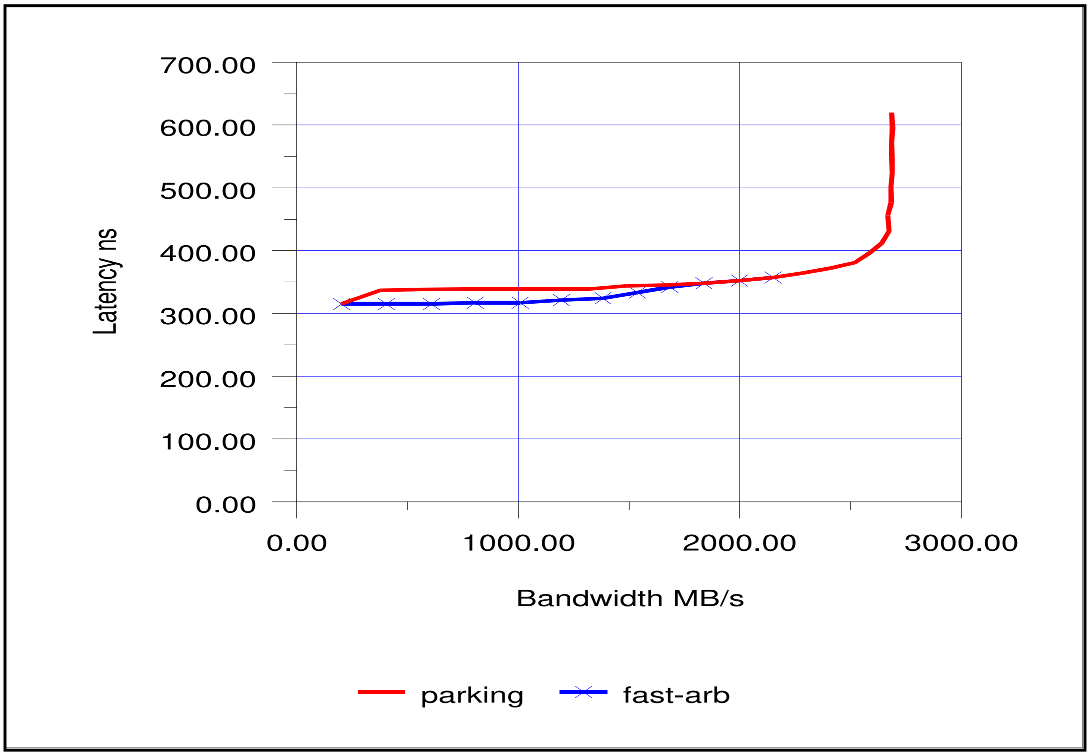
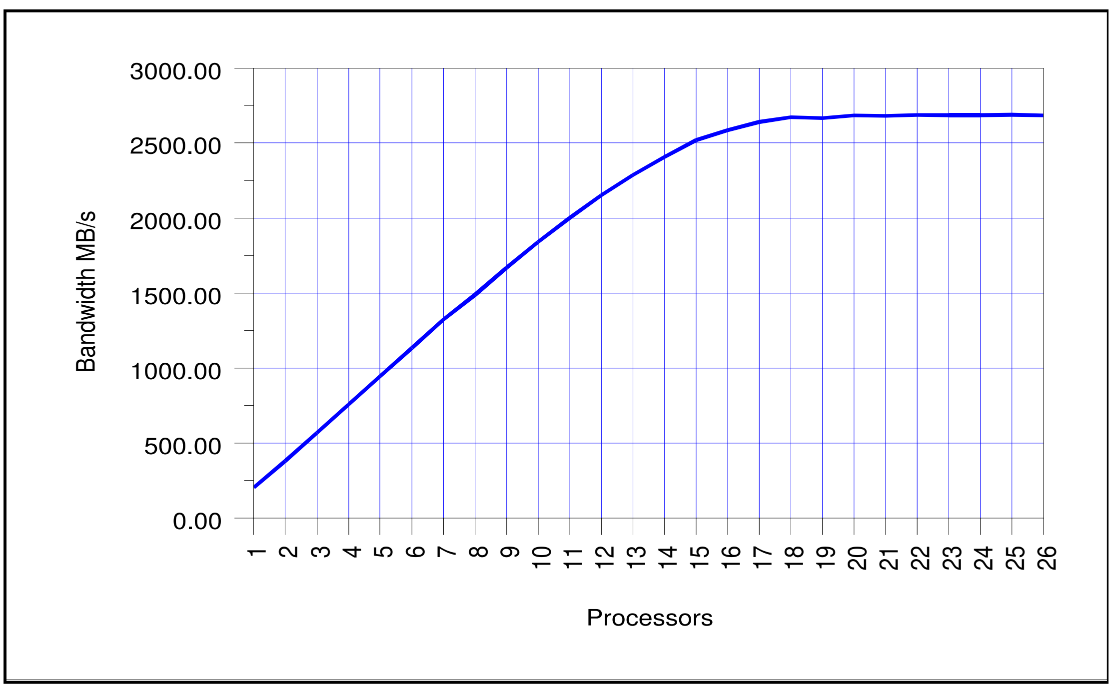


FIGURE 6. Bandwidth scaling with processors



4.2 System Transaction Throughput and Data Bandwidth

While system specifications usually quote the system interconnect data bandwidth, in reality both the transaction throughput and the data bandwidth are important performance figures. Transaction throughput, which represents the maximum number of cache misses that the system can process, is important for applications in which only a small amount of a cache line is useful data. For such applications, longer cache lines inflate the data bandwidth figure but add little to the performance of the system. For other applications, notably those with sequential data access patterns, data bandwidth is more important.

Gigaplane address and data packets each occupy 2 cycles of the Address Bus and Data Bus respectively. At a system clock frequency of 83.5 MHz, the system can sustain up to 41.75 million transactions/sec. Each of these transactions can be a cache miss. Since each transaction can carry 64 bytes of data, sustained data bandwidth is about 2.6 GByte/sec.

Each memory bank can sustain a bandwidth of over 500 MByte/sec and the system supports up to 16-way interleaving among memory banks to prevent bank contention.

4.3 Single Processor Transaction Throughput and Data Bandwidth

In addition to providing a high total system throughput and data bandwidth, we designed each board to provide high sustainable throughput and bandwidth to an individual processor. The UPA bus on each board is also clocked at 83.5 MHz and has a data width of 144 bits (128 + 16 ecc). Sustainable data bandwidth on the UPA on each CPU/Memory board is over 1 GByte/sec. The Gigaplane and board implementation support sufficient numbers of outstanding transactions from each CPU (7) and on the Gigaplane (7)(including multiple block reads or cache misses and multiple writebacks of dirty victims or multiple block writes) so that the entire UPA data bandwidth is available to a single UPA device provided it can issue sufficient requests.

4.4 Comparison with Other SMP Buses

Table 2 is a comparison between Gigaplane and buses used in a few other large SMPs.

TABLE 2. Comparison with other SMP Buses

System/Bus/CPU	Max # CPUs	Sustainable System Data B/W (MB/s)	System Transaction throughput (M trans/s)	Read[a] Latency (ns)
Sun Ultra Enterprise 6000/ Gigaplane/UltraSPARC-1 @167 MHz	30	2670	41.75	306
DEC AlphaServer 8400 5/300/ [2]/Alpha 21164 @300 MHz	12	1600	25	400
SGI Challenge/ POWERpath-2 [3]/MIPS R10K @200 MHz	36	1200	9.5	1115

TABLE 2. Comparison with other SMP Buses

System/Bus/CPU	Max # CPUs	Sustainable System Data B/W (MB/s)	System Transaction throughput (M trans/s)	Read[a] Latency (ns)
HP T 520/ [5]/PA-7150 @ 120 MHz	14	960	30	?
Sun SC2000E/2xXDBus [4]/ SuperSPARC-2 @85 MHz	20	625	9.09	1200

a. As measured by lmbench [6](following a chain of pointers, each of which results in a cache miss and memory read). Numbers other than Ultra Enterprise 6000 are reported in [7].

5.0 Summary

By a combination of aggressive packaging and electrical design, protocol innovations, and extensive pipelining Gigaplane achieves higher bandwidth, higher transaction throughput and lower latency than other competing buses, while providing support for up to 30 UltraSPARC processors. In addition to high performance features, the modular system structure provides great flexibility in configuration. Reliability, availability and serviceability features (such as hot-plug capability of system boards) make Gigaplane systems extremely attractive server platforms.

References

[1] UltraSPARC Programmers Reference Manual

[2] "AlphaServer 8200 and AlphaServer 8400", Technical Summary, Digital Equipment Corporation.

[3] "Symmetric Multiprocessing Systems", Technical Report, Silicon Graphics

[4] Jean-Marc Frailong, Michel Cekleov, Pradeep Sindhu, Jean Gastinel, Mike Splain, Jeff Price and Ashok Singhal, "The Next-Generation SPARC Multiprocessing System Architecture", Proceedings COMPCON 93.

[5] HP T Series product information at http://www.dmo.hp.com/cgi-bin/fe.pl/gsy/2a2.html (on 3/15/96)

[6] Larry McVoy and Carl Staelin, "lmbench: Portable tools for performance analysis", USENIX January 96.

[7] lmbench results at http://reality.sgi.com/employees/lm_engr/lmbench/lmbench-summary (on 3/15/96)

[8] John L. Hennessey and David A. Patterson, "Computer Architecture A Quantitative Approach", Appendix E, "Implementing Coherence Protocols".