MICROPROCESSOR REPORT

THE INSIDERS' GUIDE TO MICROPROCESSOR HARDWARE

VOLUME 7 NUMBER 3 MARCH 8, 1993

Sun, Xerox to License XDBus Technology

Small Signal Swings, Packet-Switched Design Used in Sun Dragon

By Linley Gwennap

Sun and Xerox are willing to license the XDBus technology they developed for Sun's high-end Dragon system, expected to begin shipping next month as the SPARCcenter 2000 *(see **0615MSB.PDF**)*. The new bus, also known as Dynabus, is designed to connect large numbers of processors and other devices requiring high bandwidth. The basic bus specification was developed at Xerox's Palo Alto Research Center (PARC) and is owned by Xerox. The two companies have implemented the specification in a chip set being built by LSI Logic.

Unlike previous Sun buses such as SBus and MBus, XDBus will not be made a public (i.e., free-of-charge) specification immediately. The companies do not believe this is appropriate because the potential market for such a high-end design is much smaller than that for a workstation bus. Since there are no immediate plans to put I/O devices directly on the XDBus, Sun does not need to encourage third-party card vendors to use this specification. Sun's motivation for licensing is to increase the proliferation of SPARC; the SuperSPARC module is the only currently-available processor that can be connected easily to XDBus, so the new technology can provide high performance for members of the SPARC family.

Neither Sun nor Xerox is willing to state the license fees or conditions, saying that such issues are being discussed on a case-by-case basis. Both vendors say they have limited resources and can support only a small number of licensees initially. Several companies have indicated an interest in the technology, but none has yet completed an agreement.

Potential customers have a variety of possibilities for using the new technology. One option is to license XDBus from Xerox and build a set of interface chips. To make it easier for customers, Sun will also license its chip designs and allow customers to buy the parts from LSI. Since customers may wish to use a different system design than Dragon, a third option is to license the chip design files from Sun and modify them to suit.

The XDBus itself is a high-performance multiprocessor bus that compares well with Futurebus+ and with high-speed proprietary designs. Since the two companies are willing to share their technology, anyone designing a high-performance system with two or more processors should seriously consider XDBus against other alternatives.

XDBus Designed for MP Systems

Using a low-voltage GTL interface (see sidebar below), XDBus makes it easier to construct long buses with many devices. The new bus also uses a "packet-switched," or split-transaction, protocol that makes efficient use of the total bus bandwidth in a multiprocessor system. The packet-switched protocol also permits the bus to be pipelined and split into segments, further extending the physical length without degrading the switching time. These advances are integral to the design of the SPARCcenter 2000.

Like its predecessor MBus (see µPR 8/8/91, p. 8), XDBus multiplexes address and data onto a single 64-bit path, as shown in Table 1. MBus includes no parity bits, but since XDBus is intended for large commercial applications, it uses eight parity lines (one for each data byte) for data integrity. MBus specifies 40-MHz operation, the same speed as the XDBus in the initial SPARCcenter system. Thus, the peak bandwidth of both buses is the same: 320 Mbytes per second. As we shall see, however, XDBus makes better use of this raw bandwidth.

Both buses are synchronous and both rely on a central arbiter to assign ownership of the bus. While both define a set of transactions that provides cache consistency in a multiprocessor environment, XDBus has additional transactions that provide faster TLB consistency and more efficient I/O support. Another new transaction eliminates the need for interrupt signals, helping to keep the total number of signals to just 88.

The synchronous design allows the bus to be pipelined, extending the bus length without increasing the switching time. Figure 1 shows a long bus broken into three segments. Without pipelining, the switching time would be the maximum transmission delay from one end

of the bus to the other. By breaking the bus into smaller segments, this switching time is greatly reduced. The registers move the data from one segment to the next on each clock; although the transaction latency is slightly longer than in the non-pipelined case, two of the three segments can be used for other transfers on any given cycle, increasing the overall throughput of the bus.

Packet Switching Improves Utilization

Although a fast cycle time provides high peak bandwidth, this bandwidth must be used effectively to deliver high performance. A drawback of the MBus is its *circuit-switched* design, which permits only a single transaction to be in progress at a time. For example, a processor could gain ownership of the bus and request data from memory. The processor would hold the bus during the memory latency period, preventing any other bus accesses until the memory delivers the needed data. This design is simple and provides reasonable performance if there is one processor and one memory controller. In a system with several of each, however, the dead time greatly reduces the deliverable bus bandwidth.

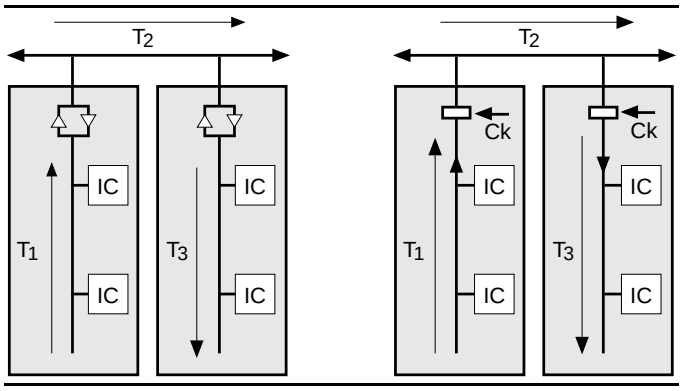


Figure 1. In a non-pipelined bus (left), the cycle time must be the sum of T_1, T_2, and T_3. In a pipelined bus (right), data is transferred from segment to segment, reducing the cycle time to the maximum of T_1, T_2, and T_3.

XDBus overcomes this problem with its *packet-switched* design, which splits transactions into request packets and reply packets. In the previous example, the processor would transmit its request to the memory, and then release the bus for other traffic during the memory latency period. Once the requested data becomes available, the memory controller would arbitrate for the bus and deliver the data to the processor. XDBus delivers data with the "critical word first," which means that the requested doubleword is sent first but the complete transaction provides an aligned block of data.

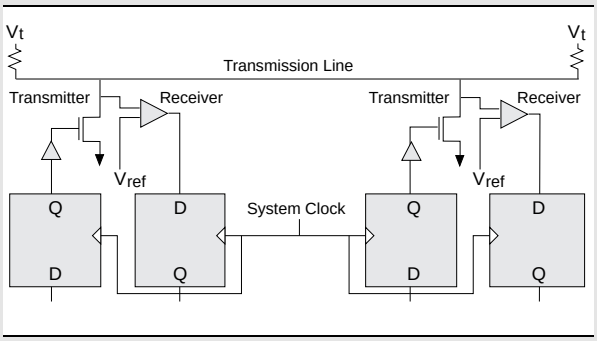
All XDBus packets are either two or nine cycles in length and use the first cycle to transmit a packet header, shown in Figure 2. For example, a read-request packet takes two cycles and sends the requested address to the target device. A read-reply packet consists of a header cycle followed by eight cycles of data. In this case, the header contains the device ID of the requester and the address of the requested data, so the reply packet can be unambiguously matched with the original request. There is no specific limit to the number of transactions that can be in progress at any given time.

One problem with the packet design is that it increases the complexity of the bus interface. In particular, slave-only devices such as memory must now be able to act as bus masters to generate return packets. Thus, this design is best suited to buses that connect only high-performance devices with VLSI interfaces and can bear the cost of a complex interface.

The ability to overlap transactions, however, yields a significant gain in overall performance. In an MBus system with a memory latency of six cycles, a 32-byte read transaction takes a total of eleven cycles—one address, four data, plus the latency—meaning that only

GTL Permits Longer Buses

GTL (Gunning transceiver logic) devices use a small (800 mV) voltage swing to achieve fast switching times. All transmission lines are terminated to reduce signal settling times. As shown in the figure below, data is transferred synchronously from a register at the sender to a register at the receiver, allowing the entire clock cycle to be used for the transfer without wasting time for synchronization. This technology allows XDBus to operate at frequencies up to 80 MHz in small configurations. Conversely, by holding the frequency to 40 MHz, a long bus with many devices can still meet the specification.



GTL also provides for low power, since V_t is just 1.2 volts. Most of the power is consumed in the external terminating resistors, not in the interface itself. The open-drain CMOS driver consumes virtually no power when off and only 9 mW when turned on, compared to 71 mW for a BTL (Futurebus+) transceiver and 125 mW for an ECL device. This low power means that 160 transceivers—enough for a full unidirectional XDBus interface—dissipate just 1.5 watts, permitting a complete interface to be integrated into a single VLSI chip without external bus transceivers.

36% of the cycles are used to transfer data. In an XDBus system with the same latency, eight out of eleven cycles transfer data, resulting in a 73% efficiency.

Future trends should increase the advantage of packet switching. Cycle times have been decreasing much faster than memory access times. Thus, memory latency, expressed in cycles, is increasing. If this trend continues, the efficiency of circuit-switched buses will decline, while the XDBus efficiency will remain at 73%.

XDBus Includes Robust Transaction Set

Figure 2 gives a complete list of XDBus transactions. Since XDBus uses a MOESI-type protocol (see µPR 6/20/90, p. 12) to maintain consistency among multiple caches, all transactions define their effect on all caches on the bus. For example, Read Block and Write Block transfer cache-coherent data to and from memory; all devices snoop these transactions and update their caches accordingly. Non-Cacheable Read Block and Flush Block also transfer memory data, but are not snooped.

"Block" transactions take nine cycles to transfer 64 bytes of data. "Single" transactions take only two cycles, transferring one double-word during the second cycle. These shorter transactions are used for either shared memory data (WSU, WSI, SSU, SSI) or access to I/O devices (IORS, IOWS, IOSS). XDBus supports a separate 36-bit I/O address space in addition to the 36-bit physical memory space. A separate set of transactions accesses this I/O space.

XDBus uses a single Interrupt transaction to communicate interrupts to processors. This eliminates the need for dedicated interrupt lines on the bus; MBus, in contrast, uses 14 signals to determine interrupt destination, priority, and other information. Since all XDBus devices must be able to master the bus, all devices can use the Interrupt transaction. Transaction-based interrupts are more easily extended to larger systems and have the additional advantage that the target can be selected dynamically; for example, interrupts can be sent to the most lightly-loaded processor. One problem is that interrupts cannot be transmitted if the bus is busy or "hung" by a hardware failure. XDBus uses a watchdog timer to detect and correct such situations.

Two transactions allow for remapping virtual pages on the fly. Normally, software must interrupt all processors and ask them to remove an entry from their TLB before remapping that page. XDBus allows one processor to broadcast a DMI transaction indicating the address to be demapped. All other processors signal that they have removed the entry by sending a DMT. Once all DMTs are received, the original processor can proceed with the remapping. By performing this operation in hardware, numerous cycles are saved on each CPU.

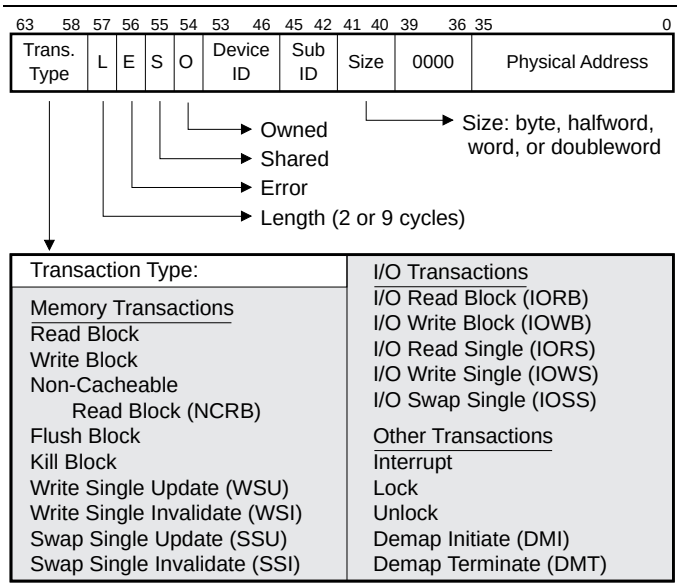


Figure 2. Format of an XDBus packet header. Some fields are not used by all types of transactions.

Arbitration Occurs in Parallel

Like most high-performance buses, XDBus uses a dedicated set of arbitration signals, listed in Table 1, to perform bus arbitration in parallel with other transactions. This prevents wasting bus cycles when selecting the next bus master. XDBus, like MBus, uses a central arbiter to grant bus ownership for each transaction, but the new design provides more information to the arbiter. Instead of a single request signal, XDBus uses three lines, REQ[2:0], to indicate the priority level of the request. This allows, for example, replies to take precedence over requests, guaranteeing forward progress since, during busy periods, pending transactions are completed before new transactions are begun.

Signal Name	Description	Direction
Data[63:0]	Multiplexed address/data	Bidir.*
DataParity[7:0]	Parity for Data[63:0]	Bidir.*
DataIn[63:0]	Multiplexed address/data**	Input
DataInParity[7:0]	Parity for DataIn[63:0]**	Input
Req[2:0]	Arbitration request (level 0–7)	to Arbiter
ReqOwner	Transaction includes owned data	to Arbiter
ReqShared	Transaction includes shared data	to Arbiter
ReqParity	Parity for signals in this section	to Arbiter
Gnt	Arbitration request granted	from Arbiter
GntType[2:0]	Type of request granted (0–7)	from Arbiter
GntOwner	Logical OR of all ReqOwner	from Arbiter
GntShared	Logical OR of all ReqShared	from Arbiter
GntParity	Parity for signals in this section	from Arbiter
Clock	Bus data clock	Input
nClock	Inverse of Clock	Input
BidEn	Bidirectional mode enable	Input

Table 1. XDBus defines a total of 88 signals, with an additional 72 signals (marked **) used only in unidirectional mode (*Output only for unidirectional mode).

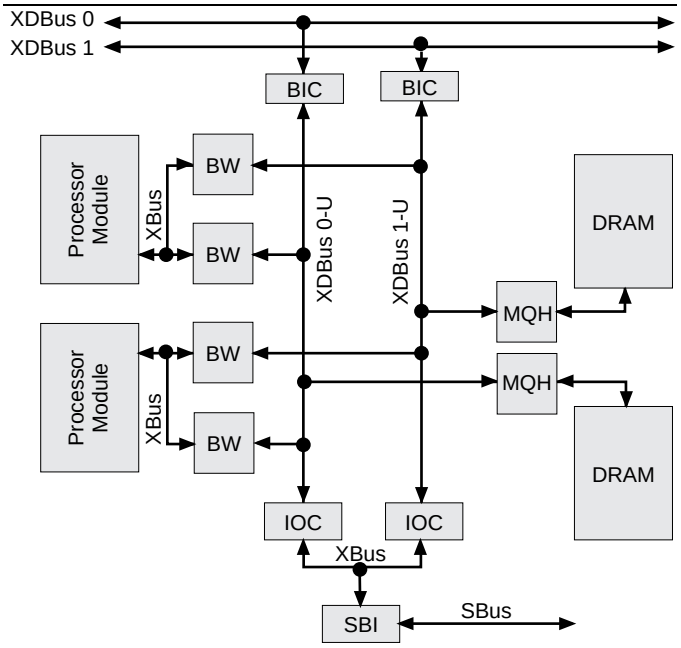


Figure 3. A SPARCcenter 2000 board supports dual XDBuses with two processors, two memory controllers, and one SBus interface. The on-board XDBuses operate in unidirectional mode ("U"), providing separate data-in and data-out ports.

Other encodings instruct the arbiter to temporarily suspend new transactions, preventing a device from being overrun by requests. In extreme cases, a device can ask that the bus be halted. This flow-control method eliminates the need for separate "ready" or "halt" lines.

Each cycle, the arbiter examines the arbitration requests from each device and decides which one to grant. Although XDBus does not specify a particular arbitration procedure, it does assume that the request priorities are respected and that the arbiter uses some "fairness" criteria to prevent individual devices from being starved from lack of bus access. The arbiter indicates its decision by asserting the selected device's GNT signal and indicating the granted transaction type.

SPARCcenter 2000 Implementation

Sun's SPARCcenter is the first product to use the XDBus. The system uses two XDBuses to double the bandwidth. Each memory controller is assigned to only one of the two buses, while processors and I/O devices can access either bus. Physical memory addresses are interleaved between the two XDBuses on 256-byte boundaries, increasing the chances that both can be used during a series of sequential accesses.

Because the Sun system provides for up to 40 devices connected to a single XDBus, the bus is partitioned using bus pipelining. Figure 3 shows a single system board. The XDBuses at the top run across the backplane between up to 10 system boards. The registered BIC interface provides an electrical buffer and a one-cycle logical buffer. It also demultiplexes the data bus so the on-board XDBuses can operate in unidirectional mode, with separate lines for data out and data in. The advantages of this arrangement are discussed below.

Each system board supports two processor modules. In the current design, these are the same SuperSPARC modules used in Sun's multiprocessor workstations. These modules, however, support only a single set of cache tags; in the workstations, this forces the cache controller to "steal cycles" from the CPU when performing bus snooping. For a larger system with more processors, this performance degradation is unacceptable. The SPARCcenter implements a duplicate copy of the cache tags in the bus watcher (BW) chips. XBus, a simplified version of XDBus, is used to maintain consistency between the two copies, interrupting the processor only when it is necessary to update the tags. SuperSPARC modules can connect to either MBus or XBus directly.

The system board also supports two memory controllers (MQH), one on each XDBus, as shown in the figure. Each controller can handle up to 256 Mbytes of memory using 16-Mbit DRAMs. The two I/O controllers (IOC) connect to a single SBus via the SBus interface chip (SBI). The IOCs and SBI, like the processor and its bus watchers, use XBus to communicate.

Unidirectional Mode Increases Parallelism

Breaking a single logical XDBus into several parts increases the opportunities for parallelism while keeping the bus frequency up to a reasonable rate. The use of pipelining helps keep the frequency at 40 MHz, as discussed previously in Figure 1. To take advantage of this pipelining, packet transmissions must be overlapped so that multiple packets are in flight at any given time.

Implementing the on-board XDBuses in unidirectional mode creates opportunities to overlap transactions. Figure 4 depicts a typical situation. A two-cycle packet (D1–D2) is being sent from device A to B. Although the second double-word is still on the backplane, the arbiter has already told device C to begin sending the next packet (D3–D4). In the following cycle, device B receives D2, while D3 reaches the backplane. Thus, the backplane is fully utilized with no dead cycles. The arbiter understands the structure of the buses and schedules packets accordingly.

If the on-board XDBuses were bidirectional, in many cases device C could not begin sending its packet because it (or other devices on that board) would need to snoop the previous transaction. By splitting the incoming and outgoing traffic onto two different sets of lines, devices B and C in the previous example could even be on the same board and still overlap the two transactions.

The SPARCcenter uses the two-level arbitration scheme shown in Figure 5. Each of the four devices on a single board sends its arbitration signals to the board

arbiter (BARB), which selects one request based on priority and fairness. Up to ten BARBs then transmit their requests to the central arbiter (CARB), which selects a single request. The CARB informs the selected BARB, which informs the selected device.

The two-level scheme greatly reduces the number of signals on the backplane. Since each device has 11 signals that must be individually routed to or from the arbiter, a single-level arbiter would require 440 signal pins and an equal number of backplane traces. By segmenting the arbitration, the backplane requires a more manageable 110 arbitration lines, and the CARB can be implemented as a single chip. The CARB and BARB chips are relatively simple logic designs, using less than 100,000 transistors each.

Futurebus+ Provides an Alternative

There are few open standards for high-performance system buses. The most discussed of these is IEEE standard 896, better known as Futurebus+ *(see **060806.PDF**)*. While Futurebus+ has many possible configurations, it can be configured to be similar to XDBus with a split-transaction protocol and a 64-bit data path. Instead of adding additional buses to increase bandwidth (the SuperSPARC module can interface with up to four XDBuses), Futurebus+ allows for data widths of up to 256 bits.

Futurebus+ multiplexes address and data on the same lines, but it uses a separate set of eight signals to communicate transaction type, size, and similar information, allowing a more efficient use of the data bus. The data bus is used to transmit byte masks when needed, providing more flexible support for sub-word transactions than XDBus. Transactions are of variable length and can be up to 64 words, longer than the 64-byte fixed length of XDBus transactions.

One major drawback of Futurebus+ is its use of high-power BTL (backplane transceiver logic) drivers, which use eight times the power of GTL transceivers. As a result, a Futurebus+ interface requires external transceiver chips. These are currently available in a maximum width of 9 bits, requiring eight chips for a 64-bit implementation.

The initial standard, published last year, is very broad and complex. To simplify designs, subsets of the standard are being defined but have not yet been completed, delaying the availability of third-party cards. DEC is the only major system vendor that has announced a product with Futurebus+, and that product uses it as a 25-MHz I/O channel. At this time, it is difficult to tell if Futurebus+ will live up to its promise.

Some Proprietary Buses are Faster

From a product standpoint, XDBus competes against proprietary high-end system buses from other system vendors. For example, Hewlett-Packard's recently-announced Corporate Business Servers use a bus called PMB to connect up to eight processors with up to 2G of memory and 16 I/O interfaces. According to a paper presented at Compcon last month, the PMB uses a 64-bit data path (expandable to 128 bits) and a 60-MHz clock to provide 50% more peak bandwidth than a single XDBus. Furthermore, the HP design includes a separate bus for address and arbitration cycles, so the data bus can be fully utilized for transfers.

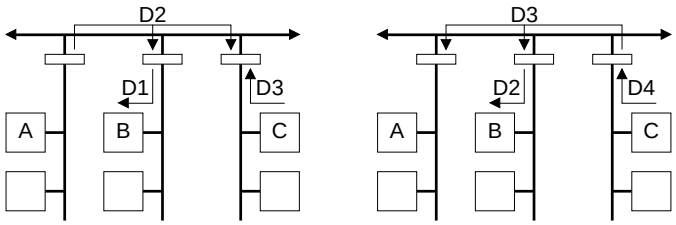


Figure 4. On the left, "B" is receiving data while "C" begins a new packet. On the subsequent cycle (right), "B" get its last data while the new packet reaches the backplane.

HP eschewed the complexities of a packet-switched design in favor of what the paper calls a pipelined bus. This is different than Sun's pipelined XDBus design. The PMB devices are programmed with a standard memory latency at start-up, for example, 13 cycles in the current implementation. During normal operation, data appears on the data bus exactly 13 cycles after an address is placed on the address bus, as shown in Figure 6. Devices can indicate "busy" or "wait" if they cannot deliver the requested data in time. The fixed transaction length of four cycles makes it easy for the address and data buses to stay synchronized.

The HP bus offers better utilization than XDBus, but the separate address and data buses greatly increase the number of pins needed for an interface.

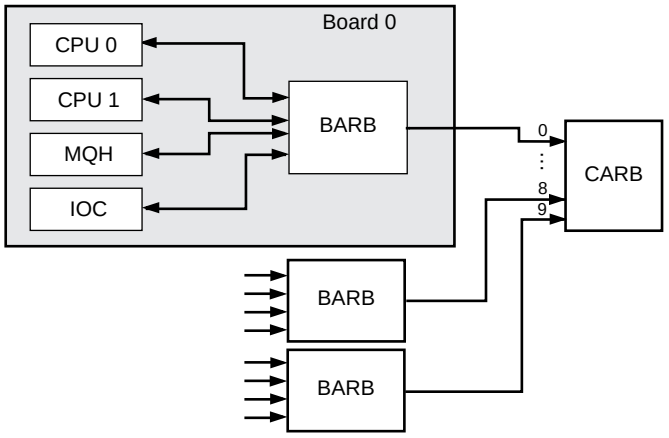


Figure 5. Two-level arbitration scheme resolves priorities first at the board level (BARB) and then at the central level (CARB).

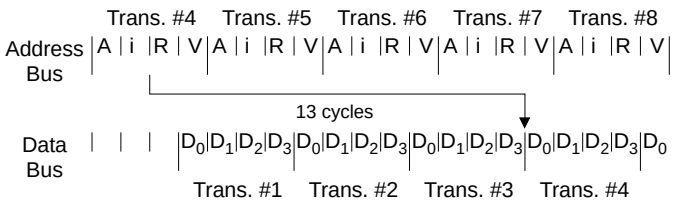


Figure 6. HP's PMB overlaps transactions so that the data bus lags the address bus by exactly 13 cycles.

Furthermore, even though the PMB uses open-drain CMOS transceivers, its 3V logic creates a higher power dissipation than GTL. As a result, PMB devices use a set of external transceiver chips, increasing the cost. It is the custom design of these transceivers, however, that helps the bus achieve its 60-MHz clock rate across a backplane with a similar length as the Sun system's (about 17").

XDBus Will Proliferate

XDBus overcomes many of the problems of previous designs for multiprocessor systems. Although its first implementation is in a high-end server, the GTL interface allows for single-chip interfaces, making this technology well suited for lower-cost servers. Nearly any system with two or more processors would benefit from the packet-switched technology. Although Sun would not comment on future product plans, XDBus will probably appear in other Sun systems over time, particularly as CPU performance increases beyond what can be effectively supported on MBus. As processor bandwidth requirements grow and the cost of integrating an XDBus interface falls, this design could be used in high-end workstations.

While other proprietary buses may offer better performance than a single XDBus, the dual XDBus design is competitive with other systems in its price class. (Mainframe crossbar buses are another story.) It also offers a degree of fault tolerance. The dual design can be easily scaled back to a single bus for less expensive systems, leveraging the same interface chips and system design. HP, on the other hand, uses a totally different bus for its lower-cost servers. Sun should find that its investment in XDBus will be fruitful.

For More Information

Sun and Xerox are working to make the complete XDBus specification public, but it is not available at this time. Technical information is available in *XDBus: A High-Performance, Consistent, Packet-Switched VLSI Bus*, by P. Sindhu *et al*, presented at Compcon. For a copy of this paper, contact Gladys Petel, Sun Microsystems (SMCC), UMTV16-10, 2550 Garcia Avenue, Mt. View, CA 94043; 415/336-1190.

If you are interested in getting a copy of the XDBus specification (under NDA) or in discussing licensing terms, contact Mr. Jean Gastinel at Xerox, 2225 E. Bayshore Road, Palo Alto, CA 94303; 415/812-5500.

For information on Sun's XDBus chip set, contact Chet Silvestri, VP Technology Sales, Sun Microsystems (SMCC), UPAL01-315, 2550 Garcia Avenue, Mt. View, CA 94043; 415/336-0329.

If Sun and Xerox push XDBus as an open standard, it will create an interesting competitor to Futurebus+. Although it has the advantage of being an IEEE standard, the overcomplicated Futurebus+ looks like a bus that was designed by committee and it has not yet caught on in the market. XDBus is simpler and should be strongly considered by anyone looking for a high-performance multiprocessor system bus.

It is not yet clear how willing Sun is to license this technology. If XDBus is simply a tool to sell SPARC processors, Sun is unlikely to license its chips to anyone who wants to use it with a different CPU. The company also says that one of its criteria is the licensee's potential for success, and admits that it considers companies that compete head-to-head with Sun as less likely to succeed. This could restrict potential customers to niche vendors willing to use SPARC. Xerox, on the other hand, has fewer conflicting corporate interests and may be more willing to license the XDBus specification broadly. Without Sun's chip designs, however, potential customers would have to to start from scratch. If Xerox can sign up one or two major system or semiconductor vendors, XDBus could become widely used. ♦

