TURBOchannel and SBus

Ron Levine

June 15, 1992

0. Purpose

This white paper is intended to organize and clarify the issues in the debate on the relative merits of the TURBOchannel and SBus I/O interconnects. It strives to be objective and to correct some of the misinformation that has appeared in the trade press or circulated informally through such channels as the Notes files. It is not intended for circulation outside of Digital. Suitably sanitized, it may provide a basis for a trade journal article.

1. Introduction

Among I/O interconnects, TURBOchannel and SBus may be considered to form a class apart from other I/O channels and buses. Both have unique, innovative features that distinguish them from the majority of other buses and interconnects, and there are substantial similarities in the way that they differ from the bus mainstream. Still, there are important differences between them. The primary purpose of this paper is to delineate the differences between them and to try to clarify the significance of the differences, the implications of the differing feature, and the tradeoffs that the differences represent. For the most part, it is understood that we are dealing with the original specifications of the two interconnects, except in Section 17, which considers the existing proposals to double their potential bandwidths. To set the stage for this compareand-contrast exercise, we begin by pointing out how TURBOchannel and SBus are alike in the ways that they differ from the rest.

2. How TURBOchannel and SBus are similar.

First, SBus and TURBOchannel are similar in their primary intended application, viz., as high-performance I/O interconnects for desktop systems. They are primarily intended for connecting workstations to such high-bandwidth peripherals as graphics subsystems, IPI disk controllers, FDDI and ATM network interface. In most implementations, they also support more conventional peripherals such as SCSI controllers or parallel printer interfaces. This intended application drives the common design goals, and is thus responsible for much of the similarity in their architectures.

On the other hand, the intended application may also represent to the customer one of the most important distinguishing factors between the two interconnects--that the intended desktop systems are, for one, DECstations, VAXstations, and forthcoming Alpha-based systems, and, for the other, SPARCstations and their clones. That is, there may well be, for any given customer, other factors that discriminate these two product families and overshadow whatever differences there may be in the I/O interconnects. Also, for any potential option designer, the installed base of the target workstations is likely to the be most important factor determining his option development priorities. This paper is restricted to questions of technical merit of the interconnect architecture, and does not further consider advantages based only on market considerations.

As desktop expansion buses, typically supported completely on a mother board, SBus and TURBOchannel are together quite different from backplane buses, such as VME, Multibus II, and Futurebus+. They are both a little more like the daughter card expansion buses, such as

ISA, NuBus, MCA, and EISA, but they differ together substantially from these traditional microcomputer expansion buses:

in their partially radial topologies and the consequent simplicity of their protocols,

in their attainable performance, in

cost/performance ratio,
in making use of more recent and more highly integrated circuit technologies,

in their providing for self-configuration.

Both SBus and TURBOchannel, in their standard configurations, have peak instantaneous transfer rates of 100 MB/s. Of course, because of necessary overheads, this rate is an asymptotic maximum, which, it is guaranteed, can never be attained as an average rate, even over a single transaction. How closely various average transfer rates can approach this bandwidth cap is an important measure of the technical merit of the interconnect that this paper will analyze. Other measures of merit to be discussed include the ease and cost of designing to the interconnect specification, the production cost of expansion modules, and the applicability of the interconnects to demanding applications.

3. Radial topology

Both TURBOchannel and SBus have partially radial topologies; that is, in each architecture, the interconnect has a distinguished participant with control functions (called *system* in TURBOchannel and *controller* in SBus), and some of the signals are point-to-point, on private lines between the controller and the other participants. Radial signals are contrasted with the shared or "bussed" signals, which are on lines shared by all the bus participants. All of the traditional busses consist entirely of bussed signals.

The radial architecture enjoys several advantages over a bus architecture. First, it allows a much simpler protocol, with much less overhead for address decoding, contention resolution, and other handshaking activity required on shared bus lines. Second, the radial architecture requires substantially fewer signal lines per module. The overhead reduction implies higher effective bandwidth. The protocol simplification and line count reduction contribute to simplifying the design of expansion modules and reducing their production costs. Further, the radial architecture provides an improved environment for signal integrity, eliminating many electrical problems that must be considered when signal lines are shared among many participants.

The TURBOchannel architecture is completely radial, in that it allows implementations in which all the TURBOchannel signals are radial. In any case, seven of the 44 TURBOchannel signals must always be radial. In all implementations so far, the address/data lines are bussed. The broadcast control signals, such as *clk*, *~write* and *~reset*, which are driven only by the system and which do not distinguish between the options, are also shared in the existing implementations.

A completely radial TURBOchannel implementation, with separate address/data lines for each option, would have enormous potential performance. This upward scalability may well be

needed to provide adequate I/O capacity for forthcoming desktop systems based on Alpha or other hot processors. There is even a potential benefit in the radial implementation of normally broadcast signals; namely, radial implementation of these signals is essential for doubling the maximum clock rate on a slot-specific basis, as discussed in Section 17.

The SBus architecture is only partly radial. Of the 82 SBus signals, only three control signals (***BusRequest****, ***BusGrant*, SlaveSelect*)** are radial. Most of the lines, including the address and data lines, are always shared.

Thus, the first distinction between the TURBOchannel and SBus interconnect architectures is that the former is substantially more radial than the latter. Consequently, the TURBOchannel protocol is substantially simpler than the SBus protocol, and the protocol overhead costs are lower. This difference implies higher performance, simpler option design and lower option production costs for TURBOchannel than for SBus.

4. Types of participants and transactions

A TURBOchannel implementation has a single ***system*** and a number of *option* slots.

T

The TURBOchannel controller function always resides in the system. All transactions are between the system and one of the options. I/O ***transactions*** are initiated by the system and transfer only one word of data per transaction. ***DMA transactions*** are block transfers and initiated by options. All options must be able to participate in I/O transactions, but not all options need be capable of DMA transactions.

An SBus implementation has a ***co****ntroll****er*** and a number of other modules which may be ***masters*** or *slaves* or both. The SBus equivalent to a TURBOchannel transaction is called a bus ***cycle***. Masters initiate bus cycles and slaves respond. The controller is involved in every bus cycle, primarily for arbitration and address translation. The host normally has both master and slave functions. In practice thus far, the controller is implemented as part of the host, but it is logically distinct from the host *qua* bus participant, and it may be implemented outside of the host.

Thus, with respect to the classification of bus participants, SBus is somewhat more complex than TURBOchannel. Further, SBus is, in principle, much more symmetrical than TURBOchannel, in that the host is treated logically just like any other participant and data can be transferred directly between non-host participants, so long as one is a master.

In practice, aside from such peer-to-peer communication, there is a fairly direct correspondence between the two interconnects: non-host SBus masters correspond to TURBOchannel options that have DMA capability, and non-host SBus slaves correspond to TURBOchannel options without DMA capability. However, the logical differences entail differences between the protocols, with the SBus protocol for any given bus cycle having somewhat greater complexity and greater overhead than the TURBOchannel protocol for the corresponding transaction.

TURBOchannel has two distinct styles of transaction--DMA and 1/O transaction--each ruled by its own very simple protocol. SBus has just one defined bus cycle protocol, which is more complex both because SBus is less radial and more symmetrical and because it uses virtual addressing, as discussed in Section 6, below.

An SBus advocate might argue that having two different transaction protocols somewhat dilutes the claim that TURBOchannel is simpler. But even if it takes more work to describe two very simple protocols than one complex protocol, the TURBOchannel interaction is unequivocally

simpler at run-time, because the determination of the transaction type has no run-time cycle cost, but is manifest from the participant that initiates the transaction.

Having to implement both transaction protocols in a TURBOchannel DMA option represents no more complexity than having to implement both master and slave options in an SBus master. Dividing the complex compound behavior into two non-overlapping simple behaviors actually simplifies the design task.

5. Asymmetry and peer-to-peer communication

The SBus architecture is less radial and more symmetrical than the TURBOchannel architecture. In the previous section we argued that this entails greater complexity and cost, and that it has a negative effect on performance. What benefits, if any, follow from SBus' greater symmetry?

One benefit that has been mentioned is the possibility of direct peer-to-peer communication, which is not possible with TURBOchannel. It remains to be seen how important this will be in practice. Direct peer-to-peer communication has rarely been used in previous buses that permit it, such as UNIBUS and Q-bus, probably because of the difficulty of providing the needed operating system support. Further, applications that really require direct peer-to-peer data transfer are likely to involve specific pairs of devices. In these special cases, the requirement could be better met by private direct connections between the communicating devices.

Fully symmetric multi-master buses, such as VME, are popular and conceptually satisfying because they do not require a special controller node. They are especially appropriate for multiprocessor systems. However, the desktop computer world is inherently asymmetrical; there is an essential difference between the host system and its peripherals.

Thus, SBus advocates have criticized TURBOchannel because it does not allow the host to initiate DMA transactions. But, on most I/O channels most of the time, DMA transactions are normally initiated by the device controllers, which must, after all, deal with external factors such as mechanical motion in the device or asynchronous activity on a network. A dumb frame buffer is a possible exception to this claim, and indeed, some SPARCstations contain a slaveonly SBus slot intended to accommodate the frame buffer. But if the graphics subsystem includes any graphics acceleration processing, then it, too, should act as the DMA master because of the asynchronous nature of such graphics processing.

Of course, while the device controller determines the precise clock cycle to initiate the transaction, most I/O activity is ultimately controlled by software running in the CPU, which issues I/O requests. In this regard, TURBOchannel has an advantage because such I/O requests are passed to the device controller by means of CPU-initiated I/O transactions, which have no counterpart in SBus and are much quicker than the quickest SBus master/slave bus cycle.

6. Addressing method

One of the most striking differences between the SBus and TURBOchannel architectures is that the former uses virtual addressing and the latter uses physical addressing. What are the advantages of each choice? What are the tradeoffs?

The operating system passes to the device drivers virtual addresses of virtually contiguous buffers. However, eventually in every bus cycle, physical addresses must be used to address memory. The issue İS who will do the virtual-to-physical address translation, and who will deal with the gather/scatter problem that arises when virtually consecutive addresses cross a physical page boundary.

This is an issue because it requires access to the virtual memory map in the course of an I/O operation, perhaps even during a bus cycle.

In SBus, masters initiating bus cycles put virtual addresses on the data lines. It may require several bus cycles to translate the virtual address to a physical address. Address translation and gather/scatter are functions of the controller only. TURBOchannel has no comparable overhead in its transactions. The SBus controller may take an arbitrary number of clock cycles to perform address translation.

The use of physical addressing in TURBOchannel has the benefit of eliminating the translation cycle overhead of SBus. However, it can complicate the design of options, giving them the need to watch for page faults and handle the gathering and scattering. It also impacts the design of a device driver, which must set up address translation tables in memory that can be accessed

by the DMA device. However, these complications are independent of the function of the option, and so need to be solved only once. Thus, the TURBOchannel Interface ASIC provides the gather/scatter function for any option that makes use of it. Similarly, the required functionality can be easily included in new software drivers for new options by borrowing example code written once.

7. System configurations

For both SBus and TURBOchannel, the I/O channel may also be used as the system memory bus. SBus uses the nomenclature "host-based system" for this configuration.

In the DECstations lxx and the Personal DECstations 2x, TURBOchannel is indeed embedded in the system bus, so these platforms would correspond to the SBus notion of host-based system. But in the DECstations 2xx, the host system has its own memory interconnect independent of the TURBOchannel, which is reserved for I/O activity only.

In a host-based SBus configuration, there can be small overhead savings for transactions in which the CPU is the master, because the address translation cycle happens locally and does not use bus cycles. On the other hand, a host-based system must be regarded as I/O-poor, in a sense, because the same data pathways are forced into double duty as I/O channel and system memory interconnect and I/O channel.

One important limitation imposed by the radial topology is that any particular implementation has a fixed number of available slots. This is in contrast to a full bus topology, which can accommodate larger numbers of participants, limited only by electrical factors such as capacitive load, and which allows provisions for further expansion. This limitation exists independently of the number of radial signals, and so applies to both interconnects.

An SBus implementation can have at most eight masters. This limit is arbitrary and not related to any particular architectural feature. The SBus specification claims that having a defined maximum limit to the number of slots is useful to an option designer as a basis for estimating worst case latency limits.

The TURBOchannel architecture does not define an absolute limit to the number of option slots that an implementation may provide. In principle, physical address space is a resource that potentially limits the number of slots, because each slot takes a fixed range of physical addresses, and the address space must be partitioned among the option slots and the system memory. In any TURBOchannel implementation, the physical address slice size is the same for all option slots, but this common option address slices size may differ among implementations.

8. Signal count

Cost is directly related to signal count, which is 44 for TURBOchannel and 82 for SBus. This is a clear and tangible win for TURBOchannel: 38 fewer signal lines. Also, TURBOchannel protocol does not suffer for the smaller signal count; TURBOchannel protocol is actually simpler than SBus's protocol.

Both interconnects have 32 data lines. In the TURBOchannel these are multiplexed in the transaction, carrying first the physical address, then the data. In the SBus, they are also multiplexed, carrying the virtual address during the translation cycle and the data during the slave cycle.

The largest element of the difference between the two signal counts is in the SBus' PhysicalAddress[27:0], which is used only during the slave cycle.

The next largest component of the difference is the fact that there are seven shared interrupt level lines running to every SBus slave, but only one radial interrupt line to every TURBOchannel option. The implications of the difference in the number of interrupt lines are discussed in the next section.

Where TURBOchannel has a single acknowledgement line per option, SBus has three ACK lines used to encode several different acknowledgement states and error conditions, necessitated by the relative complexity of the SBus protocol. Section 13 below contains further germane comments on the complexity of the SBus acknowledgement scheme.

Finally, SBus has a 3-bit size indicator, which encodes the length of a burst and the different partial word transaction possibilities. But TURBOchannel indicates the burst length by the number of cycles for which a single defined signal (~wReq or ~rReq) is asserted. Note that, while the SBus protocol demands that the size lines be stable through the slave cycle, these lines are really only needed for one clock cycle for the controller to read the size from the master, and so these lines have a relatively short duty cycle in the SBus protocol. By contrast, the single length-indicating line in the TURBOchannel carries significant information for almost all of the clock cycles of the transaction. Thus, in a sense, TURBOchannel makes more efficient use of the signals it includes; each signal is, on the average, involved in bus control for a greater fraction of the time than in SBus.

9. Interrupts

TURBOchannel has one radial interrupt line per option. SBus has seven shared interrupt lines running to each slot, in order to allow an interrupting slave to indicate the priority level of its interrupt. This difference of 6 lines is a significant fraction of the total signal count difference between the two architectures. So on SBus, while the controller is instantly aware of the priority of an asynchronous interrupt, it does not necessarily know the interrupting device. The SBus protocol requires that a slave asserting an interrupt must set a bit in an internal register readable by the CPU to indicate that it has set the interrupt. The CPU must poll the bus participants to determine the source of the interrupt.

In the TURBOchannel protocol, on the other hand, since there is just one radial interrupt line to each option, the system knows immediately which slot has generated the interrupt. Thus, the TURBOchannel protocol enjoys some economies in interrupt service time. TURBOchannel interrupt priority is determined on a slot-specific method by the system implementation.

10. DMA burst size

The larger the DMA burst size, the more closely the data transfer rate may approach the asymptotic maximum, because there are more data words over which the per-transaction overhead can be amortized. On the other hand, allowing very large DMA bursts can potentially increase latency when multiple modules are contending for bus access. Also, larger DMA bursts require larger buffer capacity in the participating bus nodes.

For SBus, the maximum DMA burst size is 16 words. This maximum is determined architecturally by the 3-bit size field on the bus. The burst size for any transaction is set by the master, so any generally applicable slave must be capable of handling the maximum 16 word DMA transfer size, either by containing enough buffering or by making use of the DMA flow control provision discussed in Section 17, thereby slowing down the bus throughput.

For TURBOchannel, the maximum DMA burst does not seem to have an architectural limit. But the system acts as the slave in DMA transactions, so each system defines a maximum burst size, which is 128 words in all systems implemented to date. An option may use DMA bursts of any size up to the system-defmed maximum, or may eschew DMA transactions entirely.

Thus, in present practice, the TURBOchannel DMA burst size is eight times larger than for SBus. This substantial difference is the major factor in the differences in protocol limited bandwidth, discussed below.

11. Protocol-limited bandwidth

The ratio of the number of data words transferred in a transaction to the number of clock cycles for the transaction, multiplied by the clock frequency, gives a data transfer rate. Using the maximum DMA burst size gives a measure of interconnect performance that we may call the "protocol-limited bandwidth". TURBOchannel beats SBus on this measure, both because its maximum DMA burst size is larger and because its DMA transaction overhead is smaller.

For general SBus masters, a bus cycle can transmit at most 64 bytes in at least 20 cycles. For a 25 MHz clock, this yields 80 MB/s for the maximum transfer rate averaged over a bus cycle. A CPU-master in a host-based system can avoid using the bus for the translation cycle, and so improve this measure to 64 bytes in 18 cycles, or 88 MB/s. However, as noted above in Section 7, the host-based SBus configuration is likely to have poorer performance in terms of total throughput, and in any case, as we discuss in Section 5, the CPU is not normally the master in a DMA transaction.

On TURBOchannel, the DMA burst size can be as large as 128 words and the DMA transaction requires only the number of words plus three clock cycles. Therefore, the maximum protocol limited transfer rate is (128.4/131)x25 = 97.7 MB/s. This is 22% faster than for the general case for SBus.

To be a little more fair to SBus, we note that a TURBOchannel option that is capable of 128word bursts must contain more buffering than the 16-word option, and so be more expensive. So we compare the protocol limited transfer rates, using the same 16-word burst size for both TURBOchannel and SBus. TURBOchannel still wins because its simpler protocol has fewer overhead cycles. In this case the TURBOchannel is 19% faster than the general SBus master.

On any interconnect, a certain number of single-word (or smaller) transactions are needed for control functions, as when the controller reads or sets option control and status registers. So comparing the two interconnects' performance for the worst case of single-word transfers is also informative.

TURBOchannel I/O transactions tranfer one data word per transaction, with a possible minimum number of three clock cycles for an I/O write and four cycles for an I/O read, perhaps more, depending on latency in the option. An ordinary SBus cycle has a minimum of six clock cycles, so half the transfer rate of a TURBOchannel I/O write. In both architectures, the single-wordtransfer cycle count can be extended arbitrarily by slave latency.

12. Throughput rate

The peak instantaneous transfer rate (100 MB/s), and the protocol-limited bandwidths (80-98 MB/s) are simply raw indicators of maximum channel capacity, but are not the most practical performance indicators from the point of view of the end user. Averaging the transfer rate over longer time periods, such as the time for an I/O operation like reading a disk block, may be better because it folds in other costs, such as software overheads and interrupt service time, that are indirectly affected by the interconnect architecture but difficult to determine analytically. However, such tests typically do not tax the full capacity of the interconnect, but are limited by the much smaller bandwidths of the particular devices.

The performance measures that are meaningful to the end user and also tax the capacity of the interconnect are usually in terms of *throughput,* the total data transfer rates achievable under mixed workloads involving several devices operating independently. Typically, throughput increases with increasing workload until it reaches a maximum, and then decreases with further workload increments, because of contention and congestion on the interconnect. Such a performance measure includes at least two numbers--the maximum throughput and the workload at which it is attained.

No systematic throughput studies are available for either of the interconnects under discussion. Moreover, whether systematic or informal, such studies would reflect not only the relative merits of the bus architectures, but also the particulars of the system memory implementations, the file systems, and operating system details. Nevertheless, there are reports of informal studies on existing implementations that indicate throughputs on TURBOchannel about double the best acheivable on SBus. It is most likely the case that the limitations of the existing SBus performance are due not so much to the bus architecture as to other implementation details.

13. DMA flow control

While the SBus clock may run as fast as 25 MHz, a slave is not required to be able to accept or deliver one word per clock. Rather, the slave paces the data transfer by acknowledging each transferred word, half-word, or byte. (Data Acknowledegment). The slave may have up to 255 cycles to complete and acknowledge the transfer. Some Sun marketing literature has claimed that this feature of the protocol has the benefit of accomodating slower slaves and allowing the design of slaves that economize on buffers. But, of course, such tolerance of slow pokes will drastically increase the bus latency and bring down the throughput. Moreover, since the SBus maximum DMA burst size is only 16 words, the possible economies from scrimping on slave buffers are minimal.

This kind of flow control is not needed in TURBOchannel, in which the DMA slave is always the system. TURBOchannel is designed for high performance, and requires every DMAcapable option to be able to keep step with the one-word-per-clock transfer during the transaction, perhaps after an initial startup latency. It does not allow an arbitrarily slow option to bog down the interconnect for all the participants.

Some designers of SBus modules have complained about another feature of the SBus flow control scheme that considerably complicates the interface design. Namely, if an SBus slave is unable to complete a transfer within the timeout, it may issue a "Rerun acknowledgement". In response to a rerun acknowledgement, the master must reliquish the bus, request mastership again, and upon regaining mastership must perform the identical ransfer that caused the slave to issue the rerun acknowledgement. This requires the master to save considerable state information about the entire transaction until it is completed. The SBus specification recognizes that use of rerun acknowledgements can adversely affect system performance and suggests ways of avoiding using them. But even if never used by a slave in a particular system, SBus masters must incorporate the functionality for dealing with them.

14. Board Size

SBus specifies a smaller board size than TURBOchannel (19.1 sq. in. vs. 26.1 sq in). The difference is 27% of the area of a TURBOchannel board, 37% of the area of an SBus board.

You can put more circuitry, and so more functionality, on the larger TURBOchannel board. It is easier to design a board with given functionality for more available real estate than for less available real estate. The cost, of course, is not in the area of the board but on the number of lines and integrated circuits mounted on it. So the larger board doesn't cost you more than the smaller one if you don't fill it up. Therefore, there is no advantage in the smaller board, unless you can get more of them into a desktop machine AND almost all conceivable single-board TURBOchannel options could easily fit on the smaller area; two "ifs" which do not seem particularly likely. One might try to argue that levels of integration are increasing, and so future circuits will do a lot more in the same board area. But if history is any guide, then the market will demand more functionality, at a comparable price, to fill up the available area.

Sun marketing literature points out a benefit of the smaller board size: it makes the board more suitable for laptop and notebook computers. This is a valid point. The same marketing literature will then assert, without justification, that the smaller board size does not, on the other hand, limit the potential of SBus for high-end applications, such as for very high performance options (e.g. array processors), or in servers. This claim is dubious on its face. If it were true, then there would be no need for providing for double-width SBus boards within the specification.

15. Power and cooling

The TURBOchannel specification provides for more electrical power per board (26 watts) than SBus (10 watts). Moreover, TURBOchannel requires an implementation to provide forced air cooling (150 linear feet per minute), while the SBus specification allows an implementation to use convective cooling alone. Like board size, power and cooling limit functionality and performance. That is, more power and cooling per board means you can do more on a board-more circuits and faster circuits. Thus, as for board size, the TURBOchannel specification for power and cooling are oriented towards higher performance domains than the SBus specification.

Sun marketing literature attempts to put the best face on the power and cooling limitations, presenting them as advantages for small-scale implementations, such as SBus-based notebook computers. But, for the high-performance option designer, these limitations are not advantages at all; rather, they present design constraints. Ten watts does not suffice to support interesting logic, and convective cooling is not sufficient for VRAMs, or for DSPs and other highbandwidth coprocessors.

16. Self-configuration code

Both TURBOchannel and SBus provide for automatic self-configuration at system boot time through information encoded in ROM on the expansion boards. This ROM information may well include diagnostic and initialization procedures to be executed by the host CPU. Both interconnects aim to be applicable to more than one processor architecture family, including future processor architectures not yet defined. Thus, both interconnect specifications incorporate schemes for defining procedures to be run on arbitrary processors.

The SBus specification makes use of a version of the Forth programming language for ROMbased procedures. Thus, every SBus-based system is obliged to incorporate a Forth interpreter for running such procedures. Forth is appropriate for this purpose because it is a machineindependent high-level language (relative to assembler), but not a very high-level language and so capable of the sort of register-level operations that are needed for diagnostics and initialization. Moreover, Forth is one of the fastest-running interpretive languages, because of its "threaded" structure. And finally, Forth interpreters are easily ported to new machine architectures because most Forth commands are defined by threading together other Forth commands, and the amount of new machine code that must be written is small.

The TURBOchannel solution of the configuration code problem is simply to use a defined subset of the MIPS R3000 machine language. One small possible advantage of this approach is that the option designer can write his configuration code in the C language, and run it through a MIPS C compiler to produce the ROM code. This is an advantage only because C is the most widely used system programming language in the present era, and Forth is relatively unfamiliar to system programmers.

However, every non-MIPS-based TURBOchannel-based system must incorporate [a] MIPS emulator. The principal objection to this approach is that such emulation of one machine architecture by another is very slow and inefficient. This is a real objection that has already arisen in the implementation of the TURBOchannel-based VAXstations.

Thus, the SBus solution to the configuration code problem is arguably superior to the TURBOchannel solution. It will be possible to repair this deficiency in the future by extending the TURBOchannel specification. It is ironic that SBus, which has the better solution for accomodating new processor architectures, has yet been implemented only on the SPARC architecture, while TURBOchannel has been extended already from its orginal MIPS platform to two new architectures, VAX and Alpha.

17. Accelerated versions

SBus has defined an extended transfer mode, which doubles the potential bandwidth, using the same clock rate, but doubling the data path width to 64 bits. It doubles the data width by multiplexing the 28 physical address lines and four other control lines between their address/control functions and data carrying during the slave cycle. Note that this scheme now makes use of previously unused duty cycle on these lines, and thus ameliorates the inefficiency of the SBus mentioned in Section 8. Note also that this scheme adds substantial complexity to the SBus protocol logic, and the compatibility issues are not clear. In any case, SBus extended transfer mode systems and devices are still exotic. A forthcoming DMA interface ASIC from Motorola promises to make them more common.

Proposed upgrading of the TURBOchannel bandwidth involves doubling the clock frequency for some options. This can be done in a compatible way by making the system implementation completely radial. The proposed TURBOchannel bandwidth enhancement does not require

changing the interface logic, but just making new options that want to use the higher bandwidth capable of running at 50 MHz.

19. Summary

For simplicity and elegance of design, TURBOchannel has the clear edge. It has fewer signal lines and makes more efficient use of them. It has a much simpler protocol. It is easier to design to and cheaper to implement. Expansion board designers who have implemented boards for both interconnects concur in this assessment. The producer of one high-performance board reports that the interface portion of of the SBus version ofhis product is 50% more costly, in terms of board area and circuitry, than the TURBO-channel version.

TURBOchannel also wins on performance, at least at present. Its protocol-limited bandwidth is significantly higher. The greater complexity of SBus on several counts entails greater overhead costs in software and interrupt servicing which are likely to limit throughput. The SBus flow control features add considerable complexity but have no function other than accomodating lowperformance expansion modules which can reduce the throughput for all other modules. Performance measurements of existing **implementations show the TURBOchannel implementations to have about twice the maximum sustainable throughput of the SBus implementations; however, this difference is probably not due entirely to interconnect architectures, but results from other details of the implementations.**

It is expected that SBus performance can be improved in subsequent implementations. Moreover, the SBus enhanced transfer mode will be available earlier than the 50 MHz TURBOchannel. While this enhancement will double the peak instantaneous bandwidth, it entails still greater complexity so may well fall short of doubling the sustainable throughput.

TURBOchannel, because of board area, power and cooling, provides a better environment for high-end applications such as multimedia, specialized coprocessors, and high-speed networking. Moreover, because of its upward scalability, TURBOchannel promises to extend this edge in the future. This scalability is important because processor performance is still increasing rapidly, doubling every two years, but successful open I/O interconnect architectures must have much longer life cycles than processor architectures.

SBus is more suitable for use in small systems, such as notebooks, because of its smaller board area and power and cooling limitations. SBus' use of a high level language for configuration code is a better solution than TURBOchannel's use of MIPS code, but this TURBOchannel deficiency can be repaired by extending the specification.