SLAC - PUB - 5244
April 1990
(I)

APPLICATIONS FOR THE SCALABLE COHERENT INTERFACE*

David B. Gustavson
Stanford Linear Accelerator Center, Stanford, CA 94309

ABSTRACT

IEEE P1596, the Scalable Coherent Interface (formerly known as SuperBus) is based on experience gained while developing Fastbus (ANSI/IEEE 960-1986, IEC 935), Futurebus (IEEE P896.x) and other modern high-performance buses. SCI goals include a minimum bandwidth of 1 GByte/sec per processor in multiprocessor systems with thousands of processors; efficient support of a coherent distributed-cache image of distributed shared memory; support for bridges which interface to existing or future buses; and support for inexpensive small rings as well as for general switched interconnections like Banyan, Omega, or crossbar networks.

This paper reports the status of the work in progress and suggests some applications in data acquisition and physics.

A COMPUTING-INDUSTRY PROBLEM

The coming generation of supercomputers-on-a-chip has raised computation price/performance expectations, wreaking havoc in the supercomputer industry. However, actually delivering the performance these chips promise requires new supporting infrastructure.

Several supercomputer manufacturers have gone out of business recently; the reason is certainly not lack of demand for computation, but rather that supercomputer mainframes do not appear to be cost-effective when compared with workstations or network compute servers. (However, some problems really *do* require the resources of a supercomputer, particularly their large main memory and high I/O bandwidth.)

In order to produce a successful supercomputer, one has to develop new technology. Unfortunately, this is dangerously unpredictable, expensive and time consuming. By the time the technology is debugged and the final machine is ready to deliver, the faster-moving integrated circuit technology has produced processors which are so fast that a thousand dollars worth of new chips promises a significant fraction of the throughput of the multi-million-dollar supercomputer. The supercomputer can only survive because of the valuable software and other infrastructure which supports it, and because of a small number of customers who absolutely need the resources of these biggest machines regardless of the price.

SOLUTION STRATEGY

A way is needed to harness multiple inexpensive fast processor chips and apply them to large problems. A large part of this problem is software. Enormous effort has been applied to this problem over the last decade or so, and progress has been made; the problem is not trivial, but it is not insurmountable either. I will not discuss the software here, but consider only the hardware aspects of the problem (while keeping in mind that hardware must provide certain facilities to make the software problems manageable).

* Work supported by the Department of Energy, contract DE-AC03-76SF00515.

Invited talk presented at the Eighth Computing in High Energy Physics Conference, Santa Fe, New Mexico, April 9–13, 1990

Suppose that the interface between the processor part and the rest of a (large) system could be standardized. Then a supercomputer manufacturer would spend its effort on developing an efficient interconnect (switch), I/O, cooling, power system, and multiprocessor software. All these are technologies which change less rapidly than processor chip technology, and are not susceptible to quick obsolescence. When everything else is ready, the manufacturer would buy a large number of the latest standard processor and memory boards and plug them in. Though this is a slight oversimplification (e.g., part of the system software probably *does* care about details of the processor chip), I think it represents a strategy which is becoming practical, and greatly improves the chance of producing a cost-effective product.

SOLUTION IMPLEMENTATION

IEEE P1596, Scalable Coherent Interface, aims to be exactly the needed standard interface between processor and interconnect. SCI standardizes the physical characteristics of the processor board and connector, the electrical signalling, power distribution, and signalling protocols. In addition, SCI defines a fiber optic implementation of the protocols (SCI-FI) which operates at lower speeds (due to current fiber-optic technology limitations) but over longer distances, and plans to define a personal-computer form factor for those who might like desktop gigaflops.

In fact, though I introduced SCI in the context of supercomputers here, an important aspect of SCI's design is its suitability for use in inexpensive small computers. It is the high volume production which can be supported by the small-computer market which we all need in order to bring the prices down for the large-computer market. One of the most attractive aspects of SCI is the smooth growth path it provides, allowing buyers to expand their systems as future needs require, while preserving their initial investment.

HARDWARE ARCHITECTURE

Computer bus structures cannot cope with the high speeds that will soon be needed to support fast integrated processors. Buses have electrical limitations caused by distributed capacitance due to imperfect transceivers, connectors, etc., and by the finite speed of light (signal propagation delay). In addition to the electrical problems, buses

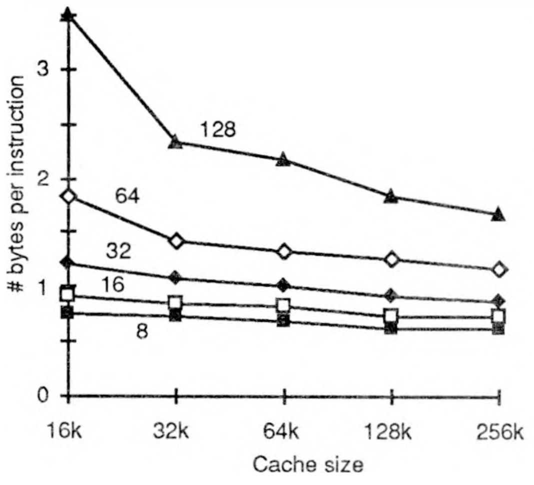

Figure 1. Average number of bytes moved between cache and memory per executed instruction as a function of cache size and cache line size.[8]

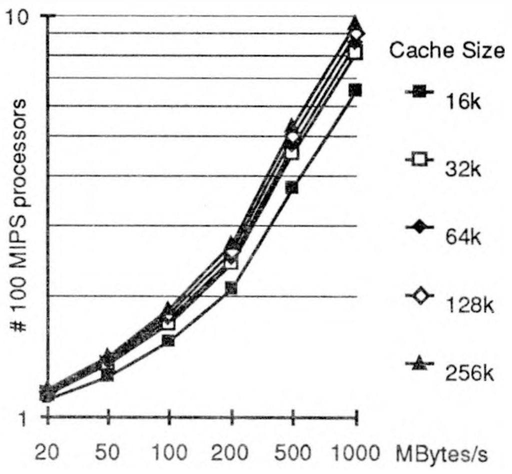

Figure 2. Number of 100 MIPS processors needed to saturate a bus system as a function of bus transfer rate.[8]

inherently become bottlenecks because they are shared resources. For example, Kristiansen[8] calculates that Futurebus+ in its 64-bit implementation can only support about 200 MIPS worth of processors, based on extensive studies using traces in a multi-level cached environment (see Figure 1 and Figure 2). More explanation of the problems of buses is available in the references.

Instead of using a bus structure, SCI simulates a bus by using a large number of independent unidirectional links. That is, SCI can be thought of as a high-performance bus replacement, from the user's point of view, even though technically it is not a bus.

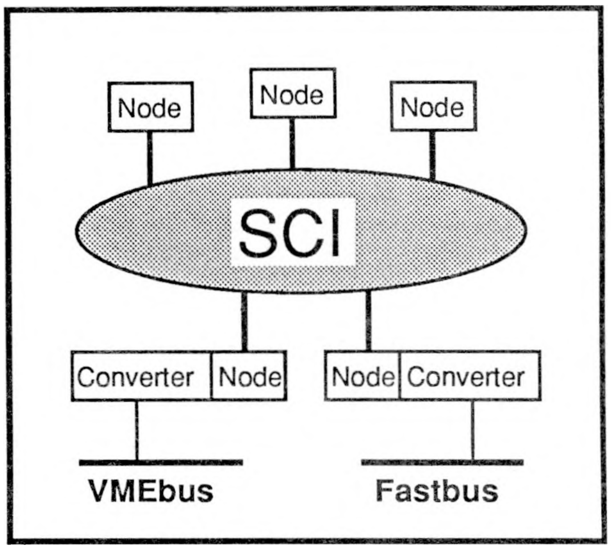

Figure 3. Typical SCI configuration, with bridges to other buses, e.g. VME and Fastbus.

Since each SCI node has one input and one output link, the protocols have to provide for the output data flow to be controlled somehow by the input data. This implies a loop or ring of some sort in order to provide the necessary feedback. This ring might be a large simple one, where several modules are connected output link to input link, or a trivial one with two modules, one of which is really the interface to a bidirectional switch. In turn, the switch might be made of a mesh of rings connected by bridging nodes. It is easy to generate butterfly-like switch characteristics in this way, and also more complex switches.

Thus the inherent association of SCI with ring structures does not appear to be a serious limitation. Other switch-based machines have to provide feedback in some way, too. Except for some which operate synchronously (and thus have inherent speed and size limits), the main difference is that SCI's feedback path is full bandwidth, where others have instead added a special low-bandwidth reverse signal on each link. That introduces scaling problems and makes signal regeneration more difficult.

Even a single SCI ring, with up to (say) eight 100-MIPS processors, will far outperform a bus because of the high bandwidth and fast arbitration and addressing mechanisms (cache controllers on buses often slow address cycles for all when the directories are busy because of on-board processor activity). SCI's cost should also be much lower, because of the narrow interface and the elimination of complex snooping cache controllers. To be fair, though, we should admit that initial implementations (using ECL gate arrays) will dissipate perhaps 20

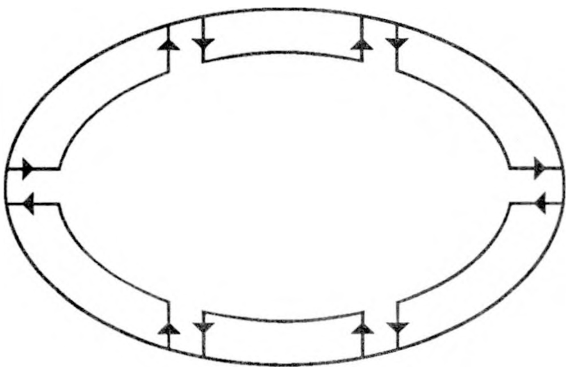

Figure 4. Ring interconnect.

watts. This should improve with time, and one must keep in mind that this one package contains the entire 'bus' interface including transmitters, receivers, fifos, and much of the coherence-management logic.

Unidirectional links effectively remove the speed-of-light barrier to system growth: the system size and clock rate are decoupled, and there are no cycle-by-cycle handshakes. Physical signalling problems are greatly simplified because there is always one driver and one receiver, at opposite ends of the link. Signals operate steadily whether there is data flowing or not, which makes it easy to use phase locked loops for data extraction if desired (no start-up preamble is required).

SCI's official scope, in attempting to show that SCI is not a local area network, limits SCI to distances less than 10 meters. Fortunately, however, this limitation cannot be enforced. The user is free to make the links as long as desired, accepting a trade-off between length and latency as appropriate for the application. Furthermore, one can use SCI for networking purposes (very efficiently) by layering networking protocols on top of it.[9] SCI moves data efficiently in a shared-distributed-memory model, looking to the user like a memory bus. This model supports networking and message passing very well.

SCI does not specify a switch design, though we do place some constraints on such a design. There is much room for research and experience in this field; perhaps someday it will make sense to standardize a particular switch, but not now. Meanwhile, one can organically grow interesting switches out of ringlets and nodes which connect two ringlets selectively.

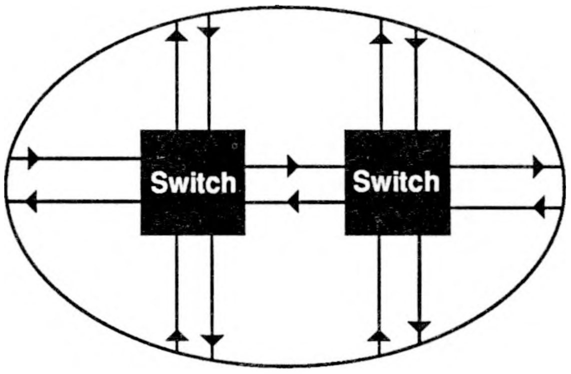

Figure 5. Switch interconnect.

SIGNALLING

Differential ECL (Emitter Coupled Logic) signalling works well at SCI rates. It rejects noise, survives attenuation, and results in constant current flow between connected modules, enormously simplifying the ground distribution problem compared to bussed systems.

SCI uses a narrow 16-bit data path (plus clock and one flag bit) at 2 ns/word (250 MHz clock, both edges active), to control the interface IC pin-count problem and make switch elements more practical. Note that 'differential' implies 2 pins per signal, and 'unidirectional' implies 2 links, one for input and one for output, so we have 72 pins for each SCI interface. A circuit for making switch networks must have at least twice that many, and preferably four or eight times, so the importance of a narrow data path becomes obvious. We believe that speed will become cheaper much faster than pins do, and thus chose to push the speed near currently practical limits in order to keep the pin count reasonable. Note that the differential signalling is not as wasteful of pins as it seems, because at these speeds one would otherwise need a large number of ground (or signal return) pins to keep noise under control; there are already high speed driver chips on the market which have one ground next to each output pin.

In this signalling scheme, the practical limits on signal transmission become attenuation, distortion, and skew. The nature of the unidirectional link makes it easy to regenerate and re-time signals with simple repeater circuits as needed along a cable. Preliminary experiments have transmitted data 6 meters on inexpensive ribbon cable without repeaters. This did not appear to be near the practical limit; however, twisted pair and shielded flat cables did not work as well — high-frequency behavior of the cable is important.

The SCI protocols include occasional patterns which reveal bit boundaries for dynamic skew compensation. The receiving circuit provides an on-chip delay chain, and adjusts the delay on each bit until the observed skew is effectively eliminated. It is not yet certain whether the initial chips will include this circuitry, as it does not appear to be essential and there is competition for available gates. When such chips become available, however, inexpensive cabling can be used and the skew changes which occur as it is moved or handled can be dynamically compensated.

The unidirectional nature of the SCI links is important. There is a certain DC component to the signal current through a link, even though it is essentially constant with differential signalling. If the link were to be turned around, the drivers at one end would have to be turned off, the signals would flow to the other end, and drivers at that end would turn on. During this process, the direction of the average DC current would have to reverse. In large high-speed systems, this causes significant electrical noise and ground-shift problems. The SCI scheme has non-zero ground currents, but they are constant and thus cause little trouble.

Another design choice made by SCI is the lack of a reverse flow-control signal in the link. It is tempting to add one signal in the reverse direction, as some switches have done, to be used for flow control, i.e. to ask the transmitter to stop sending data temporarily. However, at SCI speeds the amount of data stored in transmission lines can be large compared to a packet. This means that the time of arrival of the 'stop' request relative to the packet is unknown, depending on the length of the cable. Handling overruns due to this delay, and interpreting the timing of the arriving 'stop' signal, add considerably to the complexity of a system and place limits on its growth. Thus this method was rejected by the SCI working group in favor of flow control via packets which eventually reach the transmitter through its normal input-link mechanism.

Actually, of course, any packet delivery mechanism can be used to get the SCI packets from place to place. The signalling just described is our initial copper signalling standard, but we also envision fiber optic or coaxial cable (bit serial) mechanisms of various speeds. (Hans Wiggers, of Hewlett Packard Laboratories, 415-857-2433, is our task group coordinator for these.) Our protocols are designed to permit mixing packet delivery mechanisms and speeds within a system (queues and buffers are provided where needed).

CLOCKING

The SCI interface is synchronous, but the phase of the incoming data stream with respect to the local clock depends on propagation delays and clock frequency differences and thus is arbitrary. Receiver circuits insert delay before sampling the data in order to synchronize it to the local clock.

We support several models of clock distribution. In the simplest, one can use a central clock as a 250 MHz reference so that phase differences are constant. This is inconvenient to manage, however, in large systems or when reconfiguring small ones (e.g., when adding a second subrack or crate), and is completely impractical in Fiber-SCI systems. Therefore, we specify a mechanism which allows each interface to generate its own clock, and we insert or delete elasticity symbols as needed to compensate for differing clock rates.

Much of this compensating mechanism would have been needed in any case, as it is critical to avoid sampling the incoming data too near its transitions in order to avoid triggering metastable states in the sampling circuits. Thus we have to observe the incoming clock phase relative to our own clock and adjust the sampling time to account for drifts. Small systems can use the input link's clock at each node, regenerating it for output to the next node, but care must be taken to avoid excessive buildup of phase jitter.

CONNECTOR

Modelling and experiments have shown that several commercial connectors are able to meet our requirements. We intend to follow Futurebus+ in adopting the DuPont Metral 2mm connector, in a 4×48 size. The pinout is organized as a row of differential outputs, a row of grounds, a row of differential inputs, and a row of differential static signals such as geographic address. At one end of the connector we allocate 24 pins for 48 V supply, 48 V return, power control, precharge, and electrostatic discharge. Three pin lengths are specified, with a single length in each row for inexpensive mass production. This spreads out the peak insertion force points of the various rows, reducing the insertion force considerably. Choosing short pins for the power converter enable ensures that little current is being drawn when a module is inserted or removed.

A happy consequence of using point-to-point connections is that extender-board operation is easy with SCI. With a high-performance bus, plugging a passive extender board into the backplane causes reflections and grossly affects the bus behavior. Active extenders also change the timing of the extended board's operation. The most practical method is to move the board being examined to the end of the backplane, and either access it through cut-out end panels in the crate or remove the bus terminators and put them on the end of a passive extender board. This also changes system timings, which may cause a problem under investigation to disappear. With SCI, all connections are transmission lines which are merely lengthened by an extender; in fact, an extender cable can be used instead of a board if that is more convenient. Furthermore, because of the nature of the SCI packet mechanism, timing changes due to extension are very unlikely to change system behavior in a way which will cause problem behavior to be altered. This should make SCI devices much more convenient to troubleshoot than bus-connected devices.

Tests revealed that the card-edge connector used in IBM's PS/2 MicroChannel has excellent electrical characteristics. A variant of this, which automatically connects the input link to the output link when the board is removed, is being considered for the PC version of SCI. Thus empty sockets do not break the ring.

We were dismayed to learn that existing bus standards can be unreliable because they only define the gross mechanical shape of the connector, not the pin construction or metallurgy. Even specifying the normal forces or insertion forces is not sufficient. In particular, the common situation where half of the mating pair is made by one vendor and half by another can lead to totally unsatisfactory performance. We plan to follow the lead of IBM reliability researchers, and specify the shape of one side, the metallurgy, and the stress which should be applied by the other side. This should result in long-term reliable operation even if multiple vendors are involved. Futurebus+ has now decided to take this approach too, so we will reference the Futurebus+ connector specification documents.

POWER

SCI specifies on-board DC/DC power conversion, distributing only 48 VDC. This makes power-on insertion and removal of modules easy, which can be very important in large systems. (When multiple voltages are supplied to a module through its connector, it is very difficult to guarantee a safe sequence of power-contact making or breaking as modules are inserted or removed.) Also, uninterruptable power is almost trivially simple to provide in the context of on-board conversion.

Furthermore, it is not clear which supply voltages will be important in future designs; ECL wants –2 V and –5.2 or –4.5 V, TTL and CMOS want +5 V, and 3.3 V or lower seems important for future CMOS and GaAs technologies. On-board conversion avoids guessing future requirements, and enormously reduces the number of power pins which would need to be allocated to provide for full current at each of many potentially useful supply voltages.

The primary disadvantages of on-board conversion are the converter's space and added heat load on the board. Though modern converter technology has reduced this problem considerably, it still causes us to specify a somewhat larger board than would otherwise be required. We expect the converters to be located along the top edge of the board, in the waste heat from the active application circuitry.

BOARD AND MODULE

We have chosen to specify a single board size, the Eurocard 6U (233.35mm) × 280mm. The module width, or pitch, is 30.48mm. Boards can use side-mounted or straddle-mounted connectors, providing for double-sided surface mount or single-sided tall components, at the cost of dual card guides or moveable card guides.

The mechanics for a whole related family of board sizes and corresponding backplanes and subracks is detailed in IEEE P1101.2, an adaptation of IEEE 1101 but using the new 2mm connector family. Development of a new all-metric standard is under way (IEEE P1301.1), and we support that in principle, but won't wait for it to be completed.

PROTOCOLS

SCI has been careful to keep its protocols lean and efficient. All transactions are 'split', with a request subaction followed by a response subaction. A small set of transactions has been defined which permit fixed length transfers of 0, 64 and 256 bytes, variable lengths from 1–16 bytes, and several special transactions associated with coherence list maintenance and generalized multiprocessor locks. Fixed length packets simplify the very fast logic required in the SCI interface chips. Block transfers are short enough to limit switch blockage, while providing reasonable data transfer efficiency. All packets are a multiple of 8 bytes in order to facilitate internal demultiplexing so that slower but wider on-chip logic paths can be used.

We have included several forward-progress mechanisms in order to prevent starvation of certain nodes:

1. Our arbitration protocols guarantee fair access to a small fraction of the system bandwidth (important for guarantees of forward progress), with optional priority allocation of the rest.
2. A batched retry mechanism guarantees fair access to a saturated node.

To avoid certain common kinds of deadlock, we specify separate request and response queues, and forbid mechanisms in which responses generate dependent requests. This breaks a cyclic dependency which otherwise could severely impact system performance. (E.g., with combined queues all entries could be filled with requests, making responses impossible.)

COHERENCE

Cache memories essentially take snapshots (called 'cache lines') of small regions of memory, and keep them close to the processor on the assumption that the same data or nearby data may be used again soon. By making the cache memories very fast, the effective speed of memory can become nearly that of the cache, assuming effective strategies for cache management are used. The coherence problem arises when two or more caches take snapshots of the same region of main memory, and then one of the processors modifies the data. All the old snapshots are suddenly wrong, and somehow the other caches have to be told to get a new copy if their processor needs the data again.

Bus-based systems generally use snooping to accomplish this. For example, during a Futurebus+ address cycle every cache controller on the bus looks up the current address in its directory (which keeps track of all the snapshots in its cache), and decides whether the bus operation affects the validity of any snapshot in its cache. If so, it may participate in the data transfer on the bus or modify it in some way to ensure that invalid snapshots are discarded or updated. This directory may be shared by the processor, and may be busy checking processor accesses when the bus cycle comes along, so it can delay the end of the address cycle as necessary to finish the processor access and then do the bus address check. Because cache access interacts with the bus in this way, the slowest cache controller design in the system sets the system speed, placing a premium on high speed (and cost). In addition to this disadvantage, snooping only works when there is a bottleneck in the system, through which all traffic passes serially (like a bus). Such a mechanism could not hope to perform well in a large multiprocessor environment. (It is possible to extend it further than one might think based on the above discussion, however, by careful use of a hierarchy of caches).

Therefore, for SCI we have developed a basic cache coherence mechanism which maintains a distributed directory of users of each data item (our snapshot, a 64-byte cache line), so that only those who care have to be notified when shared

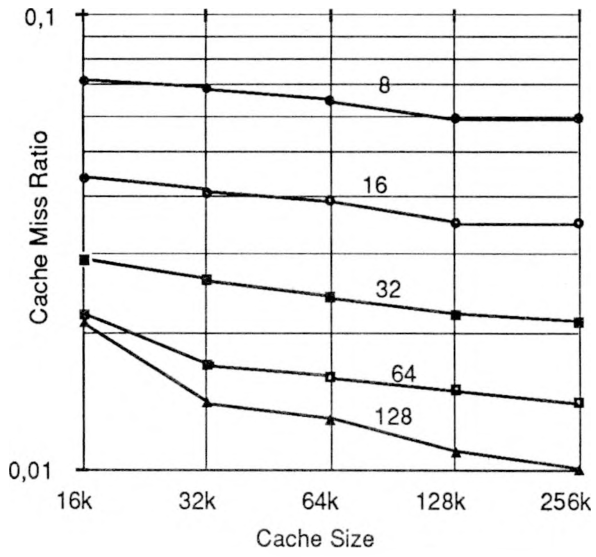

Figure 6. Cache Miss Ratio as a function of cache size and cache line size[8]

data is modified. By storing this directory as linked-list pointers in each participating cache, the storage required does not have to be preallocated and there is no intrinsic limit to growth. This mechanism appears fairly simple at first, but the hazards can be subtle, as it involves maintaining a correct distributed structure which is built and modified by many processors, potentially at the same time. We hope to verify the correctness of this mechanism by formal means[2] as well as by careful design and simulation tests. In other ways, however, this mechanism is much simpler than the snooping mechanism, and there is less premium on speed because it has no analog to the snooping delay of a bus address cycle. SCI packets come and go, and if the cache controller is slow in servicing one it affects only the latency of the associated operation, not of the whole system.

Extensions to this basic mechanism are being developed which may approach the ultimate N/log N performance desired for very large systems. Request-combining allows reduction of traffic at system hotspots, and approximate tree pointers allow spreading information to sublists faster than linear lists permit, while avoiding the overheads associated with maintaining balanced trees.

SCI emphasizes coherent cache/shared memory operation only because that is the difficult service to provide, and is the most general processing model. However, coherent and non-coherent operation can coexist, along with message passing, for programs which work well in those simpler environments.

SOFTWARE

Though the SCI standard will not provide multiprocessor software for the user, software implications of our protocol design are constantly being considered. Multiprocessor systems need efficient synchronization primitives which do not cause excessive communication traffic or congestion. Resource allocation must be free of deadlocks. Livelocks must be avoided (where several processors become synchronized in such a way as to interfere with each other's operation so that no real work gets done). SCI specifies separate queues for requests and responses, forward progress mechanisms which allocate request-queue space fairly during saturation, and a bandwidth allocation mechanism which guarantees that no one gets blocked indefinitely by higher priority users.

A rich set of lock primitives is also provided (read-modify-write doesn't work if you don't have a single bus!), though they are mainly needed for interfacing with other systems. (Within cache-coherent SCI, a processor can lock a cache line so that it has exclusive write access to it, perform any operation it likes, then release the line again. Thus an unlimited variety of lock mechanisms can be implemented in software. However, when a transaction involves a foreign bus without cache coherence, like VME, the lock primitive is needed in order to inform the bridge what it must do. For example, the bridge may translate a lock into a read-modify-write sequence on VME.)

The goal of the SCI coherent distributed cache model is to make caching invisible to software, so that the effect is merely that memory appears to be faster than it really is. Shared memory also makes efficient message-passing easy.

Most of the SCI specification is being expressed in "C", so that the actual specification code can be tested in simulation. Fortunately, SCI is synchronous except for a few well-defined points (where incoming signals are deskewed and re-synchronized to the local clock), and this makes simulation of SCI systems easy and reliable. Simulation of busstyle interfaces is enormously more difficult, especially if one includes the arbitration process and snooping-cache-directory effects on the address cycle.

In the course of this development, it has become clear that existing processor chips could be much more suited to multiprocessor applications if they had some improved features. For example, it would be nice if they directly supported a good set of lock primitives (like SCI's). Each generation of processor seems to improve some of the features we want: for example, the 68040 greatly improves multi-level hierarchical cache operation by providing the external signals and controls needed to keep the internal and external caches consistent. We are starting an education program to make processor designers aware of the needs of the interconnect, which will be the key to achieving high performance multiprocessor systems in the future. Meanwhile, there are ways to get the job done with less-than-perfect processor chips at some cost in elegance.

As part of the SCI project, we started a task group to consider the appropriate architecture for I/O and Control and Status Registers. I know from experience with other buses that this is a difficult area for standardization, because so much is arbitrary and designers don't want their freedom restricted. However, multiprocessor considerations reduce the arbitrariness considerably, and our task group coordinator, David V. James, brought experience in this area which showed us that some schemes are demonstrably better than others. For example, some DMA architectures are much better than others when one considers the implications of shared control structures, multiprocessor completion interrupt service, etc.; some types of control or status data structures require lock variables if they are to be accessed in a multiprocessor environment, and others do not.

As a result of the general usefulness of this work, we decided to offer it to the rest of the community as work which could be shared, and asked the IEEE Microprocessor Standards Committee to establish it as a separate project, which is now known as IEEE P1212. This work has been participated in by workers from Futurebus+, VME, etc., and is now seen as providing a software interface model to the various buses which may facilitate eventual migration from one to another.

For further information, contact David V. James, IEEE P1212 Chairman, Apple Computer, MS 76-2H, 20525 Mariani Avenue, Cupertino, CA 95014, 408-974-1321.

David is also the SCI Vice Chairman and Logical Task Group Coordinator (i.e., chief architect). He has done an excellent job of combining new ideas and inputs from SCI working group members with his own experience and integrating them into a coherent whole much better than the sum of its parts.

PHYSICS APPLICATIONS

What is the laboratory market for SCI? Technical work often includes large computations. For example, airframe designers would like teraflop machines for studying aerodynamic behavior without using wind tunnels. Tomographic imaging, which involves deducing internal structure based on the observation of many projections, is compute intensive. Real-time 3-D Magnetic Resonance Imaging involves enormous data handling problems. Integrated circuit design requires large detailed simulations to verify design correctness before investing in circuit masks.

Modern theoretical physicists are often compute-limited. Currently, dedicated multiprocessors are being used for lattice-guage-theory calculations at Fermilab, for example.

Multiprocessor farms have been built at many particle physics laboratories to provide cost-effective analysis for the enormous volumes of experimental data. The software problem for particle physics event analysis is particularly easy, because the events are statistically independent of one another; one merely distributes them to independent processors for analysis, and combines the results later.

Data production at the Superconducting SuperCollider is estimated at over 10^{14} bytes/second, with enormous computation problems associated with triggering, filtering, storing and analyzing the data. At these data rates, the aforementioned independence of events breaks down because of detector time constants, so analysis may require examination of one or more events preceding the given event. This will probably require enormously increased interprocessor communication. SCI's coherence mechanisms may be helpful in managing this problem, where simple message-passing was adequate in the past. The ability of the SCI architecture to grow seamlessly as budgets permit will be a great convenience as well.

SCI's fiber-optic implementation may be useful for moving data out of detectors. This uses little of SCI's special capabilities, but there may be some advantage to having the data arrive in SCI-format packets if SCI is to be used in the data filtering and analysis system.

SCI's first priority is commercial success. For this reason, many features are incorporated to support multiprocessor coherent and message-passing operation, and few (if any) features are included specifically to support physics applications. To be successful, SCI must have commercial interface chips, and special physics optimizations would reduce the chance of getting these produced. However, the needs of commercial multiprocessor operation appear to be an adequate superset of the needs of physics laboratories. Furthermore, the laboratories may find some of SCI's features more useful than they initially appear (e.g., cache coherence) as they gain experience with them.

FASTBUS, FUTUREBUS+, OR SCI?

I am often asked the question, should the next generation of equipment be designed for a bus (and which one) or for SCI? I have an internal conflict on this subject, as I have participated in the designs of Fastbus and of Futurebus+ from their inception and want them to succeed. Nevertheless, time moves on and needs and economics change, and so do the right answers to such a question.

Fastbus supports parallelism in the traditional physics way, without cache coherence, but with many independent bus segments which only link together (causing contention) as needed. To be really efficient, Fastbus users need to move from Segment Interconnects to Buffered Interconnects, which would be available soon if only we could get the manufacturer to deliver the gate arrays. The Fastbus cable segment has its problems, but is very useful and will be much improved if we can ever get the manufacturer to deliver a monolithic

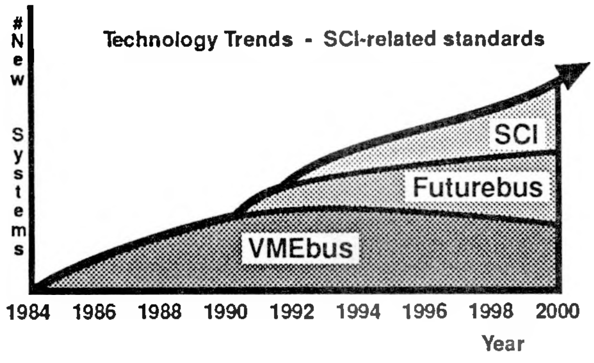

Figure 7. Technology Trends, Market Size

transceiver chip. Fastbus is as good as Futurebus+ if you don't need cache coherence and can live with 32-bit width. Fastbus is much simpler, and is faster than Futurebus if the board is an ECL design. (Futurebus uses special transceivers, called BTL, which require signal translation to get from ECL, or TTL for that matter, to the bus and back. That adds delay and power consumption; Fastbus can use any ECL logic device as a transceiver, eliminating the translation circuits and delays.)

Futurebus+ is intrinsically about the speed of Fastbus, based on transmission quality, but suffers from the aforementioned signal translation burden and from complexity in several areas. Arbitration is generally much slower than Fastbus. However, if it becomes widely accepted by industry and supported in silicon, it may be very attractive. It intends to support a no-handshake burst transfer mode ('packet mode') which can be faster than the Pipelined Block Transfer of Fastbus, in which each line sends a start bit as a timing reference, followed by data bits. This makes it possible to eliminate the effects of skew (different propagation delays on each data signal line) and reach transfer burst rates at the bus bandwidth limit, typically 100 MBits/sec/signal line on a full-size crate. Futurebus+ also defines wide versions with 64, 128 and potentially even 256-bit widths. These might become interesting if they are supported well in silicon.

However, Futurebus+ (despite some claims to the contrary) does not really support multiple bus segments, or cable segments. Though these are possible in principle, they were not designed in from the beginning as they were in Fastbus. Futurebus+ bridges in principle could connect multiple segments and maintain cache coherence, but the complexity and cost may make them uninteresting. At present, there is a lot of marketing hype going on, so the user must investigate carefully to make sure products are **really** available before making serious system design decisions. The most practical way to connect Futurebus+ segments is probably through bridges to SCI; the Futurebus+ and SCI cache coherence models were designed to make this practical, and SCI handles cables (or fiber optics) well.

So, I would say that it makes sense to continue expanding Fastbus systems as needed, gradually introducing Buffered Interconnects and better Cable Segment transceivers. For cache-coherent computing clusters of 200 MIPS or less, use Futurebus+ when it becomes available. For more powerful systems or more extended systems or systems which might grow past bus limits in the future, use SCI (when the chips become available) with either copper or fiber signalling. The SCI cache coherence directory model also works well on Fastbus without requiring any bus protocol changes.

I have not mentioned VME (IEEE 1014) in this discussion, because while it is very useful due to its large market presence, it has some severe limitations. The mechanism VME uses for signal delivery is technically poor: transmission lines are badly defined, grounds are inadequate, and it only works when the receiver waits long enough for the reflections and noise to die out before it looks at the signals. That time depends on the number and type of modules in a VME crate, so it tends to be set short in the development lab (and in the shipped product) to get good performance numbers. When actually used in a large system, however, the module can become unreliable as the signal settling time becomes longer than anticipated. Furthermore, though there are various intercrate buses for VME, VME was not designed to support such things, resulting in occasional deadlocks. Though a change to the VME spec is being made to fix this problem, there are backward compatibility issues. Furthermore, VME speed is low by modern standards.

Despite these criticisms, the SCI project is going to some effort to include the special features in our protocols which are necessary to support a bridge to VME. There are many useful VME boards on the market, and if we are careful to live within the VME limitations we can use these boards to good advantage.

THE CREATION OF STANDARDS

We used to think there was an inherent conflict between the desire of the consumer for open standards and the desire of the producer for proprietary solutions. The consumer gains from open standards because they make possible free competition among multiple vendors, but of course this competition tends to make vendors work harder for their money and hence they prefer to avoid it. From the vendor's point of view, the best solution seemed to be proprietary technology which a customer would purchase sole-source, locking the customer to that vendor for the foreseeable future. However, some vendors have learned that an open standard makes it possible for many companies to contribute a variety of complementary products which can satisfy the needs of a wider range of customers, increasing the usefulness of the system to the consumer and thus increasing the size of the market, benefitting producer and consumer alike.

Open standards generally arise when one proprietary solution is second-sourced by many other vendors. The secondary vendors join in a standards effort to improve compatibility (which will increase their acceptability and thus market share) and to avoid arbitrary changes to the specification being forced on them by the original vendor.

Another mechanism occasionally generates open standards, i.e. origination by users within a standard-producing organization, based on anticipated need. This mechanism applies to Fastbus (IEEE 960, IEC 935), Futurebus (IEEE P896), and SCI (IEEE P1596). This process tends to be difficult, as the problem is not adequately constrained by existing practice, leaving an enormous range of design possibilities for the working group to consider. Such projects tend to converge slowly if at all, partly because not enough resources are available and partly because they are not sufficiently concentrated to get the job done rapidly.

Detailed designs of complex standards like these must be carried out by a small number of technical experts, who take the goals, directions, suggestions and criticisms from the larger working group and incorporate them into a coherent solution which meets the requirements as well as possible, in an iterative and interactive process. This resembles the design process in industry, where a few designers respond to requests from many customers (filtered by the marketing department) for product features—and the final vote is the customer's purchasing dollar. Fastbus discovered this mode of operation and used it during critical periods; Futurebus has used it from time to time; and SCI has had the good fortune to be able to use it consistently, applying several near-full-time workers to the effort from the start. The SCI effort has also included an intense meeting schedule, averaging 3–4 days per month, in addition to active electronic communication with interested parties between meetings (or with parties who were unable to attend). This concentrated effort results in a disproportionate increase in productivity, which we think will lead to a standard for SCI in an unusually short time. Participation by committed commercial implementors has also helped SCI to test proposals for realism and practicality, and keeps designers aware of the value of time.

CONCLUSION

The SCI project has met its milestones on schedule, strongly driven by its commercial implementors. The current plan aims for an approved standard covering the physical signalling, logical protocols, and cache coherence mechanism in 1990. The first commercial implementor (Dolphin Server Technology, Oslo, Norway) expects to have working prototypes within a few months of the standard's approval, and has promised to make the interface chips available to others.

SCI's performance seems such a large step ahead of the current state of the art in computer buses that it has some difficulty in appearing credible. The best answer to these doubts will be the existence of working silicon, available at reasonable prices, being used in working systems.

The complexity of SCI is approximately the same as that of a split-cycle bus system (like the VAX BI or Futurebus+ or Fastbus with Buffered Interconnects) in small applications, and is much less than that of a bus system for large applications. Nevertheless, relatively few designers have experience with split-cycle bus design issues and therefore there will be some need for training as they move to SCI.

SCI has no evident competition in terms of open systems which could hope to deal with the massive computation needs for the next generation of data acquisition, analysis, and general computation.

For details, or to participate, please contact the author:

David B. Gustavson, IEEE P1596 Chairman
Computation Research Group, bin 88
Stanford Linear Accelerator Center
Stanford, CA 94309 U. S. A.
tel: (415) 926-2863 fax: (415) 926-3329
Email: DBG@SLACVM.bitnet or dbg@slacvm.slac.stanford.edu

REFERENCES

1. K. Alnes, E. H. Kristiansen, D. B. Gustavson, D. V. James, "Scalable Coherent Interface", CompEuro 90, Tel Aviv, Israel, May 1990.
2. S. Gjessing, S. Krogdahl, E. Munthe-Kaas, "Formal Specification and Verification of SCI Cache Coherence", NIK89, Stavenger, Norway, November 1989.
3. D. B. Gustavson, "Scalable Coherent Interface", COMPCON Spring 1989, San Francisco, CA, February 27–March 3, 1989.
4. D. B. Gustavson, "IEEE P1596, A Scalable Coherent Interface for Gigabyte/sec Multiprocessor Applications", Nuclear Science Symposium, Orlando, Florida, November 9–11, 1988.
5. D. V. James, "Scalable I/O Architecture for Buses", COMPCON Spring 1989, San Francisco, CA, February 27–March 3, 1989.
6. E. H. Kristiansen, K. Alnes, B. O. Bakka and M. Jenssen, "Scalable Coherent Interface", Eurobus Munich, May 1989.
7. P. Sweazey, "Cache Coherence on SCI", IEEE/ACM Computer Architecture Workshop, Eilat, Israel, June 1989.
8. E. H. Kristiansen, "Scalable Coherent Interface", New Backplane Bus Architectures, pp 67–75, CERN CN/90/4, March 22–23, 1990.
9. L. Wirbel, "NASA gets memory-based net", *Electronic Engineering Times,* pp 33–38, April 23, 1990.