30
1-21-90 JT (2)

SLAC-PUB--5119

DE90 006454

THE SCALABLE COHERENT INTERFACE, IEEE P1596, STATUS AND POSSIBLE APPLICATIONS TO DATA ACQUISITION AND PHYSICS*

DAVID B. GUSTAVSON

Stanford Linear Accelerator Center, Stanford University, 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 32-bit 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 repeaters 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 presents a summary of current directions, reports the status of the work in progress, and suggests some applications in data acquisition and physics.

I. INTRODUCTION

A. 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.)

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 infrastructure which supports it, and because of a small number of customers who absolutely need these biggest machines regardless of the price.

B. 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).

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 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 now, and greatly improves the chance of producing a cost-effective marketable product.

C. 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 will define 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 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.

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

Presented at the IEEE Nuclear Science Symposium, San Francisco, CA, January 15-19, 1990

II. THE SCI SOLUTION

A. Hardware Architecture

Computer bus structures cannot cope with the high speeds that will soon be needed to support fast integrated processors. In addition to the electrical problems, buses inherently become bottlenecks because they are shared resources. Instead, SCI uses a large number of independent unidirectional links. More explanation of the problems of buses is available in the references.

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 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.

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).

B. 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.

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 switches delay in or out 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.

C. 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.

D. 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. There is enough extra room to isolate the power pins by leaving blank columns. Short pins for the power control ensure that little current is being drawn when a module is inserted or removed.

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. 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.

E. Power

SCI specifies on-board DC/DC power conversion. 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 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 several 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.

F. Board and Module

We have chosen to specify a single board size, the Eurocard 6U (233.35mm) × 280mm. The module width, or pitch, is 25.4mm. 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.

G. 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.)

H. Coherence

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

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.

Efficient multiprocessor locks have been developed. Most of these are needed mainly for controlling the interface to existing buses. For multiprocessor operation in a coherent SCI domain, specific lock operations are not needed: the processor exclusively locks a cache line briefly and does what it wishes.

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.

III. Physics Applications

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.

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.

IV. Conclusion

The SCI project has met its milestones on schedule, strongly motivated 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 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) 323-3626
bitnet: DBG@SLACVM

V. 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.