SLAC-PUB--5699

DE92 010728

Scalable Coherent Interface: Links to the Future*

David B. Gustavson** and Ernst Kristiansen***

**Stanford Linear Accelerator Center, P.O. Box 4349, MS 88, Stanford, CA 94309.
***Dolphin Server Technology, P.O. Box 52, Bøgerud, N-0621 Oslo 6, NORWAY.

Abstract

Now that the Scalable Coherent Interface (SCI) has solved the bandwidth problem, what can we use it for? SCI was developed to support closely coupled multiprocessors and their caches in a distributed shared-memory environment, but its scalability and the efficient generality of its architecture make it work very well over a wide range of applications. It can replace a local area network for connecting workstations on a campus. It can be a powerful I/O channel for a supercomputer. It can be the processor–cache–memory–I/O connection in a highly parallel computer. It can gather data from enormous particle detectors and distribute it among thousands of processors. It can connect a desktop microprocessor to memory chips a few millimeters away, disk drives a few meters away, and servers a few kilometers away.

1: Introduction

Communication among the various parts of a computer system has used buses for several generations, except in the very highest-performance situations. But computer buses have reached practical and fundamental limits, such as the speed of light, the capacitance of transceivers and connectors, and the one-talker-at-a-time bottleneck.

However, the demands of real-world applications continue to grow without regard to these limits, surpassing the ability of buses to accommodate them.

For example, building a large multiprocessor by connecting hundreds or thousands of powerful microprocessors requires a high degree of parallelism for which buses are impractical.

Engineering simulations can saturate any conceivable machine, and run so slowly on present machines that users are forced to overly simplify their design models.

The next generation of particle physics experiment (at the Superconducting Super-Collider in Texas or the Large Hadron Collider in Geneva) will generate raw data at 10^{14}-byte/sec rates, calling for vast amounts of specialized processing power and high capacity data transmission.

Recognizing this problem, Paul Sweazey (who was the cache coherence task group coordinator for Futurebus) convened a SuperBus study group in 1987 to consider what could be done. This became the Scalable Coherent Interface, IEEE project P1596, in 1988. The general strategy was clear, but many details remained.

The approach taken was to use a large number of 2-byte-wide point-to-point links, initially running at 1000 Megabytes/s each. A packet protocol would be developed to allow a collection of links to provide bus-like services, but taking care to choose only scalable algorithms that guarantee forward progress, avoiding starvation and deadlock. The protocol also had to efficiently maintain consistency among a large number of hierarchical caches. A fiber-optic link was required too, but for practical reasons it would initially be slower, at 1000 Megabits/s.

This is a relatively new kind of standards work: rather than codifying existing practice, the standardization process creates new technology as needed. This approach is essential if standards are to be completed before they are obsolete, in a fast-moving field like ours.

The SCI work was essentially complete in January 1991. Draft 1.0 was distributed to a balloting body numbering about 150 persons, and passed with a 92% approval. However, a large number of suggestions was made by the voters, which resulted in a few minor technical changes but a significant amount of reorganization and editorial change to the document. The resulting Draft 2.0 was redistributed to the balloting body in December 1991, and the expectation is that the standard will receive final IEEE approval early in 1992.

Chip design proceeded in parallel with the standards work, with heavy involvement of industry in the architectural development.

The serial link chip saw working silicon at Hewlett Packard in October of 1991. However, it relies on other chips (not yet available) for the protocol. The same chip also supports Serial HIPPI and can transparently replace 20 parallel signals in each direction with a pair of fibers.

* Work supported by the Department of Energy, contract DE-AC03-76SF00515.
** Email: dbg@slacvm.slac.stanford.edu
*** Email: ehk@ifi.uio.no

Invited paper presented at COMPCON Spring '92, San Francisco, CA, February 25–27, 1992

The first to announce a chip that supports the parallel SCI links was Dolphin Server Technology, with a family of interface chips that they will use in products but also sell to others. These chips should become available in the first half of 1992. A single chip incorporates the receivers, transmitters, fast buffer storage, protocol logic, most of the cache-coherence protocol support, and a general-purpose interface to the user's processor.

SCI is agnostic with regard to processor religion, working equally well with all denominations. However, in the development of SCI we learned that interconnects have become the limiting part of modern multiprocessors, and the next generation of processor could change a few features that would make multiprocessing work much better. We will be writing more on this subject this year.

2: Solving the problems

Starting from a computer bus point of view, let us consider how one can make buses better. The speed of light limit allows a bus to run faster if it is shorter and connects fewer devices. Sadly, this makes it less useful.

The one-at-a-time bottleneck can be improved by reducing the number of devices, ultimately to just one talker per bus. Then to connect a large number of devices, one has to interface, or bridge, a large number of buses.

Throughput can be improved by using split-cycle operations that don't tie up the bus, intermediate paths or bridges while waiting for the response to a request.

The connector and capacitance problems can be eliminated by using proper transmission lines, with one device on each end so that there are no disruptive stubs in the middle. Connectors work much better when signals pass *through* them, not *past* them as they do in buses.

Bus turn-around delays (the time from turning off the drivers on one end, letting the signals propagate to the other, turning on the drivers on the other end and letting the signals propagate back to the start) can be eliminated by using one transmission line for each direction.

Differential signalling reduces sensitivity to noise and virtually eliminates ground-return-current noise, because the signalling current is constant — the same current flows either on one wire or its neighbor. If signals are transmitted continuously, the returning current does not stop, start, or change sign. Furthermore, it is easier to receive signals at very high speeds when they run continuously, not stopping and starting unpredictably.

Cache consistency, or coherence, has been maintained in small bused systems by taking advantage of the bus bottleneck — every cache controller observes every transaction in the system, "snooping" to catch transactions that might invalidate cached data. That approach can be scaled up to a few buses by making the bridges rather sophisticated, but it does not work in highly parallel systems.

A cache coherence scheme that scales to large systems requires a directory that keeps track of which data are being used by which caches, so that the appropriate caches can be updated as necessary. Earlier schemes have used a directory but kept it in memory. SCI maintains it as a distributed doubly linked list of caches instead, with the head pointer at a memory controller and the link pointers stored in the cache controllers. This has the virtue that the correct amount of storage is always available for the directory structure, no matter how many caches are sharing copies of a particular line of data. It also spreads the maintenance traffic across the system, rather than concentrating it at the memory.

Pursuing these ideas led us to SCI.

3: The SCI solution

SCI has several aspects. Its *protocols* describe packets and their behavior, and work without regard to the packet transport mechanism currently in use.

Three *links* are defined in the present standard, so that products from various vendors can communicate. Future technological improvements will bring new link standards, faster or cheaper or both, but the SCI protocols will still work. Cable connectors, pinout, and signals are specified.

A *module* and subrack have been defined, based on the international (metric) IEEE Std 1301.1-1991, with connector, power and cooling specified so that interchangeable modules can be built by multiple vendors.

3.1: Protocols

SCI packets are efficient. They are designed for fast routing through switch networks or bridges: when cross-traffic permits, a packet can start out the other side before it has been entirely received.

The packets are short, to match the needs of the processor and cache. (SCI specifies a 64-byte cache line size. Other line sizes may be used in caches between the processor and the SCI-visible cache.) Short packets also result in less latency when one blocks another's access to a shared path.

Longer packets that can transfer 256 bytes are defined as an option, but are not likely to be implemented in the first generation of support chips.

SCI-to-SCI bridges are simple. They merely forward packets based on the address, and are not directly involved in the cache coherence mechanism.

SCI also provides noncoherent transactions, as needed for certain I/O operations, Control and Status Register[20] accesses, and interfaces to noncoherent buses like VME.

3.2: Links

SCI links run continuously. The startup process may take as long as necessary for the receiver to synchronize to the transmitter, but then synchronization is maintained continuously. If some catastrophe causes the link to lose synchronization, the protocols reset and resynchronize the links.

Every SCI interface has one input and one output link, or a multiple of pairs of links. Low-cost systems can connect outputs to inputs to form rings. High performance systems connect the outputs and the inputs to a switch network that routes the packets directly (by means outside the scope of the SCI standard). The protocols are the same in either case, so the SCI device, or node, does not have to know what kind of interconnect it is using.

Some extra cost was incurred in the interface to support ring connections, because nodes in a ring have to pass along packets not addressed to them. This additional cost for address recognition, storage and other logic was worthwhile because the system cost for a ring connection is very low. This allows SCI nodes to be used in high-volume low-cost applications, bringing that economy of scale to the low-volume high-performance applications.

Rings turn out to be better performers than one might expect, because the SCI protocols discard the packet upon receipt. Thus it only travels, on average, half way around. Furthermore, rings can be interconnected to make high-performance switch fabrics.

The GigaByte/s link uses 18 differential ECL signals for 16 data bits, a clock, and a flag bit that delimits packets. The clock runs at 250 MHz, clocking data at each edge, so the data rate is 2 bytes every 2 ns. Our strategy is to keep links narrow and fast, because pins, connectors and cables remain expensive while speed gets cheaper.

The Gigabit/s link can use fiber optics, for long distances (kilometers), or electrical coaxial cable, for short distances (tens of meters). The signals are encoded as 16 bits plus flag in a 20-bit frame, DC-balanced so that AC-coupling techniques can be used to break ground loops etc. Coaxial cable is inexpensive and convenient for the short distances typical of a room full of computing equipment. When requirements change, the optical transceivers can be added to a socket already in the device, or a separate package can be used as a translater between the electrical and the optical signalling domains.

Technological change or economic change will require definition of new link standards from time to time, but the protocols were designed to be independent of the link speed and signalling details. However, SCI does assume that the error rates are very low; if a link technology with high error rates has to be used, mechanisms transparent to SCI will be needed that make the link appear reliable.

3.3: Mechanical Package

Some applications have special needs that require the use of special-purpose or existing system packaging. However, a wide variety of applications can benefit from the enormous effort that has gone into defining the IEEE 1301 standard. Many users underestimate the difficulty of designing satisfactory packaging, and underestimate the cost as well.

The module size selected by SCI is based on 300 mm depth, 300 mm height, and 30 mm width. Actual board dimensions are less, to allow for guides, clearances, etc.

The connector is the 2 mm EIA-64 (often referred to as Metral®) modular family. Modules of 4 rows of 6 pins are available as building blocks. SCI mounts the pins on the backplane, sockets on the module board. Six modules are used for signals (18 pairs in, 18 out, 36 grounds for impedance control and isolation, and static signals that can be used for coded location information if desired.)

SCI distributes 48 V power. The voltage requirements of high performance chips are expected to change rapidly in the near future, so there is no way to specify enough power pins for each relevant voltage. Furthermore, SCI is expected to be used in large systems where it is important to be able to replace a module without turning the power off. That is only practical when a single supply voltage is used, and then only with some help from the connector, using pins of several lengths.

Using 48 V means that ordinary signal pins can be paralleled to handle enough current. Thus the SCI power connector is just a seventh signal module. The pins are arranged in rows of various lengths: a long row serves as ESD discharge and early ground; medium rows connect the main power; and short pins enable the on-board power converter operation. Thus there is never any significant current flowing in the power pins when contact is made or broken.

The remaining space on the backplane and back edge of the module can be used for 14 more connector modules (336 pins) which can be used for custom I/O, more SCI links, or fiber-optic and coaxial connectors.

4: Support for simulation and verification

The SCI specification includes a lot of tutorial explanation, because it is a new approach in the computer communication field, but relies on executable C-code programs for the detailed specification.

This approach is valuable for testing the specification in simulation, for verifying the design of SCI chips, and for simulating the performance of SCI systems.

The cache coherence mechanism was particularly important to model and understand precisely. Multiple processors are maintaining shared data structures, the directory linked-lists, concurrently with no semaphore or lock variables. (The protocols use indivisible compare&swap transactions instead.)

Work is in progress at the University of Oslo[11,16] to verify the correctness of this specification by mathematical proof. Though this is unlikely to be completed before chips are built, the intense scrutiny it has brought to the specification has greatly increased our confidence in it. Simulation has also been used extensively.

5: Conclusions and further work

SCI began as a research project carried out in the standards-development environment. It was not obvious at the start whether protocols with the desired properties existed. It is remarkable how simple and elegant the solutions to some of the difficult problems turned out, and how widely applicable SCI will be as a result. We often found that the requirement for finding scalable solutions initially made things more difficult, but then they became easier because the same mechanism could be extended beyond its original purpose to fill other needs.

In the course of SCI development we have seen the need for several related projects. Five of these have been started as official standards projects already, and more are in the planning stages. The existing projects are:

5.1: SCI/VME bridge

This project, P1596.1, is defining a bridge architecture for interfacing VME buses to an SCI node. This provides I/O support for early SCI systems via VME. Products are likely to be available in 1992. Chaired by Ernst Kristiansen, Dolphin Server Technology, Oslo, Norway, ehk@ifi.uio.no, phone +47-2-627000, fax +47-2-627313.

The main decisions involve the mechanism for mapping addresses between VME and SCI, which versions of VME to support, and how to handle interrupts, mutual exclusion, and cache coherence.

Current thinking would place the SCI bridge in the controller positions of four half-length VME backplanes, arranged back to back. This should give excellent performance; dedicating the controller positions seems a reasonable compromise for an I/O system. The bridge may also support transfers from one VME bus to another. We expect to support most of the VME sizes.

5.2: Cache optimizations for kiloprocessors

This project, P1596.2, is developing request combining, tree-structured coherence directories and fast data distribution mechanisms needed for systems with thousands of processors, compatible with the base SCI coherence mechanism. Chaired by Ross Johnson, University of Wisconsin, Madison, WI, ross@cs.wisc.edu, phone 608-262-6617, fax 608-262-9777.

This working group is developing and extending ideas that came up during the development of the base SCI standard, but which we felt could be postponed to avoid delaying SCI's introduction. That is, the protocols defined by P1596 seem adequate for systems with perhaps hundreds of processors (enough for a year or so), and include hooks for adding these optimizations later.

When a large number of requests is addressed to the same node, the interconnect becomes congested and performance suffers. Request combining allows several requests to be combined into one when they meet (waiting in queues in the interconnect). All but one of these requests generates an immediate response, which tells the requester to get the data from that one's cache instead.

SCI nodes are designed to handle this kind of no-data response already, because it is used in the basic coherence protocol. The remaining request goes forward, eventually resulting in a response that provides the data to that cache, where they are read by the other nodes. This spreads out the traffic, reducing congestion.

Note that these immediate responses relieve the interconnect from retaining any information about the combining, which enormously simplifies the process compared to previous implementations.

Once the data become available, the time needed to distribute them to all the requesters becomes important. The linear linked lists of the base SCI standard result in times proportional to the number of requesters, which can be a performance problem in large systems.

Instead of linear lists, one would like to maintain a tree structure, which could distribute the data in time proportional to the logarithm of the number of requesters.

At first we thought it would be impractical to maintain binary trees in the distributed coherence directory because the overhead would be too high. The schemes we had seen others use were not acceptable for SCI because they involved setting lock variables to get mutual exclusion while tree maintenance was done, scaling poorly and violating an SCI design principle. (Lock variables also introduce a variety of complications, such as what to do when the process that holds the lock fails.)

Thus we first considered using approximate or temporary pointers that would form shortcuts along the

linear directory lists, but which would gradually become inaccurate as processors rolled out cache lines etc. Whenever a temporary pointer was used it would be checked for validity, and the algorithm would drop back to following the (always valid) linear list when necessary.

But at the August 1991 meeting, Ross Johnson presented a method for maintaining correct trees at all times, without using lock variables and without adding much overhead. Though details need to be worked out and some corner cases need more study, we feel that the remaining questions can be resolved.

Open issues concern the worst-case scenarios (when things happen in the worst possible sequence) and how to reduce the likelihood of having these occur.

5.3: Low-voltage differential signals for SCI

This project, P1596.3, is specifying low-voltage differential signals suitable for high speed communication between CMOS, GaAs and BiCMOS logic arrays used to implement SCI. The object is to enable low-cost CMOS chips to be used for SCI implementations in workstations and personal computers, at speeds of at least 200 MBytes/s. Chaired by Gary Murdock, National Semiconductor, Santa Clara, CA, phone 408-721-7269, fax 408-721-7218.

Faster signalling requires smaller signals, if edge rates and currents are to be kept reasonable. Smaller signals require differential signalling (or at least their own reference independent of system ground). At first glance, differential signalling seems to cost a factor of two in signal traces and pins, but the real cost is much smaller because far fewer ground pins are needed, far less system noise is created (or picked up), and the higher signalling speeds reduce the number of parallel signals needed.

SPICE modelling has been done that shows we can signal at SCI speeds with contemporary CMOS technology. MOSIS test chips have already reached nearly this performance. In fact, the hardest part of the problem is how to provide or accept the data at the signalling rate!

Present modelling is based on a 250 mV voltage swing, centered on 1 V. This provides a little headroom for common-mode rejection at the receiver, while allowing use of 5 V, 3.3 V and eventually 2 V technologies.

The working group is choosing certain signal levels and rates to be supported as signal interchange (link) standards for CMOS implementations of SCI, and will also define an 8-bit and possibly a 4-bit link to complement the 1596-defined 16-bit and 1-bit links. This work should complete in 1992.

5.4: High-bandwidth memory chip interface

This project, P1596.4, is defining an interface that will permit access to the large internal bandwidth available inside dynamic memory chips. The goal is to increase the performance and reduce the complexity of memory systems by using SCI signalling technology and a subset of the SCI protocols. This work was started by Hans Wiggers of Hewlett Packard Laboratories, and is now chaired by David B. Gustavson, Stanford Linear Accelerator Center, Computation Research Group, P.O. Box 4349, MS 88, Stanford, CA 94309, USA, dbg@slacvm.slac.stanford.edu, phone 415-926-2863, fax 415-961-3530.

A serious problem with present memory systems is the need to use a large number of memory chips in parallel banks to get the bandwidth needed for today's powerful microprocessors. As the capacity per chip increases, the smallest memory configuration with adequate bandwidth reaches a point where it has an unreasonably large capacity (and needlessly high cost). Furthermore, the increments for expansion are too large. We hope to get much higher bandwidth from far fewer chips by using SCI-like signalling technology. This will lower the entry cost for low-end systems while raising the performance of high-end systems.

Several models are being considered [19]. One of the most promising uses several RAM chip ringlets attached to a single controller by 8-bit-wide point-to-point links. The name RamLink is becoming popular for this approach. Details of the signalling are still being worked out, but there seems to be general agreement to use small signal voltages.

5.5: Shared-data formats optimized for SCI

This project, P1596.5, is specifying data formats for efficiently exchanging data between byte-addressable processors on SCI. SCI supports efficient data transfers between heterogeneous workstations within a distributed computing environment. Current systems require conversions among large numbers of vendor- or language-dependent data formats; specifying a single transfer format greatly reduces the complexity of this conversion problem. In addition to simplifying the data-interchange problem, standard data formats provide a framework for the design of future processor instruction sets and language data types. Chaired by David V. James, Apple Computer, Cupertino, CA, dvj@apple.com, phone 408-974-1321, fax 408-974-9793.

The specification defines integer and floating-point sizes, formats, and address-alignment constraints. Bit

fields are supported as subcomponents of a larger byte-addressable integer datum. Work on this project has just begun, but it should not take long to complete, since much of the groundwork has been done earlier in conjunction with development of the CSR Architecture[20].

6: Acknowledgements

The development of a standard of this sophistication in such a short time was only possible due to several fortunate circumstances.

One key was having a core of people with the right experience available full-time, which kept the momentum high. We would particularly like to acknowledge the contributions of David V. James, formerly of Hewlett-Packard and now of Apple Computer, who has been our vice chairman and chief architect. The particular experience he brought to bear on this problem, and his particular talents, were invaluable.

Another key was having active participation from industry. That kept us from designing impractical protocols, and kept us thinking about the value of timeliness. In particular, Dolphin Server Technology has been extremely helpful in this work, supporting it with experienced and talented people and with a large investment in supportive products.

7: References and Bibliography

1. D. B. Gustavson, "IEEE P1596, A Scalable Coherent Interface for Gigabyte/s Multiprocessor Applications", Nuclear Science Symposium, Orlando, Florida, November 9–11, 1988.

2. D. V. James, "Scalable I/O Architecture for Buses", COMPCON Spring 1989, San Francisco, CA, Feb. 27–March 3, 1989.

3. D. B. Gustavson, "Scalable Coherent Interface", COMPCON Spring 1989, San Francisco, CA, February 27–March 3, 1989.

4. E. H. Kristiansen, K. Alnes, B. O. Bakka and M. Jenssen, "Scalable Coherent Interface", Eurobus Munich, May 1989.

5. P. Sweazey, "Cache Coherence on SCI", IEEE/ACM Computer Architecture Workshop, Eilat, Israel, June 1989.

6. S. Gjessing, S. Krogdahl, E. Munthe-Kaas, "Formal Specification and Verification of SCI Cache Coherence", NIK89, Stavenger, Norway, November 1989. Also available as Informatics Research Report No. 142, University of Oslo, 1990.

7. E. H. Kristiansen, "Scalable Coherent Interface", New Backplane Bus Architectures, pp 67–75, CERN CN/90/4, March 22–23, 1990.

8. J. E. Smith and J. R. Goodman, "Restricted Fetch&Φ Operations for Parallel Processing, by Gurindar S. Sohi," Computer Sciences Technical Report #922, University of Wisconsin – Madison, March 1990.

9. K. Alnes, E. H. Kristiansen, D. B. Gustavson, D. V. James, "Scalable Coherent Interface", CompEuro 90, Tel Aviv, Israel, May 1990.

10. D. V. James, A. T. Laundrie, S. Gjessing, G. Sohi "New Directions in Scalable Shared-Memory Multiprocessor Architectures: Scalable Coherent Interface," *Computer* Vol. 23, No. 6, June 1990, pp. 74-77.

11. S. Gjessing, S. Krogdahl, E. Munthe-Kaas, "Approaching Verification of the SCI Cache Coherence Protocol", Informatics Research Report No. 145, University of Oslo, August 1990.

12. "SCI – Scalable Coherent Interface, P1596/D2.00 18Nov91," Draft for Recirculation to the Balloting Body. Prepared by the P1596 Ballot Review Committee of the Microprocessor Standards Committee. IEEE, New York, N.Y., 1991.

13. S. L. Scott, "A Cache Coherence Mechanism for Scalable, Shared-Memory Multiprocessors," Computer Sciences Technical Report #1002, University of Wisconsin – Madison, February 1991.

14. P. J. Woest and J. R. Goodman, "An Analysis of Synchronization Mechanisms in Shared-Memory Multiprocessors," Computer Sciences Technical Report #1005, University of Wisconsin – Madison, February 1991.

15. S. L. Scott and J. R. Goodman, "Performance of Pipelined K-ary N-cube Networks," Computer Sciences Technical Report #1010, University of Wisconsin – Madison, February 1991. This paper shows that when pipelining is permitted, as is the case for SCI, one gains performance rapidly (relative to synchronous systems) by increasing the dimensionality of the interconnect, and the resulting performance is much higher than for synchronous systems.

16. S. Gjessing, E. Munthe-Kaas, "Formal Specification of Cache Coherence in a Shared Memory Multiprocessor," Informatics Research Report, Univ. of Oslo, Nov. 1991.

17. G. Delp, D. Farber, R. Minnich, J.M. Smith, M. Tam, "Memory as a Network Abstraction," *IEEE Network Magazine*, July 1991, pp. 34–41.

18. J.W. Bothner, T.I. Hulaas, "Various interconnects for SCI-based systems," proceedings of Open Bus Systems '91, Paris, 26-27 November 1991.

19. S. Gjessing, G. Stone, H. Wiggers, "RamLink: A High Bandwidth Point-to-Point Memory Architecture," COMPCON Spring 1992, San Francisco, CA, Feb. 24–28, 1992.

20. IEEE Std 1212–1991, Standard for Control and Status Register (CSR) Architecture for Microcomputer Buses. This standard is used by SCI, SerialBus (P1394), and Futurebus+ (IEEE Std 896.1 and 896.2–1991).

21. J.M. Mellor-Crummey, "Concurrent Queues: Practical Fetch-and-Phi Algorithms," *Technical Report 229*, November 1987, University of Rochester, Computer Science Department, Rochester, New York 14627. (J.M.M-C. is now at Rice University).

END

DATE
FILMED
5106192