SLAC - PUB - 4798
November 1988
(I)

# Scalable Coherent Interface*

**David B. Gustavson**

**Stanford Linear Accelerator Center**
**Stanford, CA 94309**

## Abstract

The Scalable Coherent Interface project (formerly known as SuperBus) is based on experience gained during the development of Fastbus (IEEE 960), Futurebus (IEEE 896.1) and other modern 32-bit buses. SCI goals include a minimum bandwidth of 1 GByte/sec per processor; efficient support of a coherent distributed-cache image of shared memory; and support for segmentation, bus repeaters and general switched interconnections like Banyan, Omega, or full crossbars.

SCI abandons the immediate handshake characteristic of the present generation of buses in favor of a packet-based protocol. SCI avoids wire-ORs, broadcasts, and even ordinary passive bus structures, except that a lower performance (1 GByte/sec *per backplane* instead of *per processor*) implementation using a register insertion ring architecture on a passive "backplane" appears to be possible using the same interface as for the more costly switch networks.

This paper presents a summary of current directions, and reports the status of the work in progress.

## Introduction

SuperBus was the working name adopted by a Study Group under the auspices of the Microprocessor Standards Committee of the Technical Committee on Mini and Microcomputers in the IEEE Computer Society. The SuperBus Study Group began work in November 1987 under the leadership of Paul Sweazey of National Semiconductor. Its charter was to consider the need for and feasibility of a very high performance "backplane bus," to be at least an order of magnitude more powerful than the existing standard buses.

A consequence of the physical and logical constraints such a system must meet in order to be successful was the new name, SCI (Scalable Coherent Interface), because it became clear that traditional *bus* structures would not be able to meet the demands of the next decade: the real goal is to interconnect many powerful processors productively, so that the total power of a system can be increased by merely adding more processors.

Our examination of the needs for compute power to handle real engineering problems (e.g. aerodynamic simulation or simulation of large circuit designs) or physics problems (e.g. event reconstruction in the Superconducting SuperCollider) showed that a single bus, even at 1 GByte/sec, would be completely inadequate. Many buses (segmentation for parallelism) joined by selective repeaters would be necessary. Or, better yet, no buses at all, but rather some more general interconnection mechanism.

Many architectures which would be perfectly satisfactory for a single bus become ugly, inefficient or impractical for assemblages of multiple buses; i.e., they do not scale well. Thus "Scalable" reflects our constraint that the system be smoothly extensible.

"Coherent" refers to our requirement for a distributed cache-memory coherence mechanism, similar in concept to that developed for the Futurebus, which can greatly reduce the performance cost of interprocessor communication.

"Interface" reflects the generality of our specification, which permits a given module to connect to an unspecified generalized interconnection mechanism, which might be a switching network of any of various kinds, a passive "backplane" forming a register insertion ring, or conceivably even an active bus (i.e. transceivers directly on the backplane).

The SCI standardization project was authorized by the IEEE Standards Board in October 1988, and was assigned the number P1596.

## Conventional Buses Are Near their Limits

Present bus systems are running close to physical limits; one cannot speed them up much by turning up the clock frequency or increasing transceiver speed or power, unless one shortens them correspondingly. For example, the Next machine uses NuBus (IEEE 1196) protocols at 25 MHz, 2.5 times the 1196 clock rate, but allows only four sockets instead of the 1196's sixteen. If a bus is short enough and is lightly loaded, transceiver and logic speeds do dominate among its various limits, and so its clock rate can be increased.

The fundamental physical limits are the speed of light, which limits the propagation velocity of signals and thus adds delay to handshakes; the capacitance of connectors and transceivers, which so disturbs a bussed signal transmission line that the "ideal transmission line model" is a very poor approximation indeed; and skew, differences in propagation time among a number of parallel signals which threatens to blur the boundary between successive data items.

Other physics problems, such as crosstalk between adjacent signals, are much easier to deal with and have become more economic than fundamental. Distribution of power and ground

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

Invited paper presented at COMPCON Spring 89, San Francisco, CA, February 27–March 3, 1989

(nontrivial in the face of very rapid changes of current flow) is also in this category, and so is cooling.

Multiprocessor systems have other inherent problems. For example, when many processors operate in parallel to solve a given problem, they need to be able to communicate efficiently with one another in order to share resources or to divide the work. This intercommunication can be a significant bottleneck, perhaps using a large fraction of the system bandwidth just accessing one shared semaphore variable over and over.

Furthermore, fast processors require fast local storage, so they need their own local copies of data, some of which needs to be shared. These local "cached" copies create logical problems if they are modified, because the various copies can become different or incoherent. Somehow, when one processor modifies data which other processors are using, the other processors have to be notified that their local copies are no longer valid so that they can get a fresh copy.

The cache coherency mechanism developed for Futurebus (and now being adapted to Fastbus) requires each cache controller to observe all other traffic in the system in order to determine whether some of its own data might affect or be affected by the current bus operation. Such a "Snoopy Cache" scheme cannot be generalized to highly parallel systems, though it may still be useful for implementing islands of coherency, which may then intercommunicate via more explicit mechanisms such as message protocols.)

## SCI Avoids these Limits

Futurebus and Fastbus have gone about as far as is feasible in the use of shared transmission lines which form buses. The most practical way to do better would be to use an active backplane, which has transceiver chips connected directly to the bus transmission lines with no connectors or stubs between. This would minimize the capacitance, and would result in uniform and constant loading which would make it possible to compensate for the loading and significantly improve the transmission line behavior. The connectors would be between the modules and the transceivers, so the presence or absence of a particular module would have no effect on the transmission line loading.

An active backplane scheme could also make live insertion and removal feasible, if module power is controlled by the backplane. However, most customers find the active backplane frightening because of the difficulty of replacing it if a failure should occur and thus it has received little support so far.

Not all backplane physics problems are solved by the active bus mechanism: the wire-OR glitch would still create delays whenever multiple drivers are permitted to be active on a single line, and bus turn-around (changing from one driver to another, as when changing from read to write or when changing mastership) would require delays for similar reasons.

A bus is inherently a bottleneck because it is shared by too many processors. Processor throughput is so high even today that a few processors can saturate any bus. Heavy loading subjects the users to long waits, slowing the whole system.

Therefore SCI intends to abandon the bus mechanism in favor of high speed unidirectional links. Two models are being supported for using these links. The high performance (and high cost) model uses the links to communicate between the module and a fast switch network, resulting in one GigaByte/sec *per module*.

A lower performance model connects the input and output links of adjacent modules to form a register insertion ring, which can be implemented in printed wiring on a passive backplane structure at low cost but results in only one GigaByte/sec throughput *per backplane*.

Even this low-cost version is much faster than any existing backplane bus system, so it seems attractive especially as a transition model for the short term while processors proliferate and costs decrease.

We expect to standardize on one module which can operate equally well in either environment, so that processors from many vendors can be developed and used effectively in small quantities at first, and then be moved into a switch environment unchanged when switches become available or necessary or economical for the given application.

This provides a nearly unbounded upgrade path for system growth, and should create an attractive market for the manufacturer (high volume) and for the user (low cost due to high volume and competition among manufacturers).

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 (there is no start-up preamble required). That would make it possible to eliminate skew completely by encoding clock timing with each data bit transmitted, although we do not think this will be necessary yet at our initial 1 GigaByte/sec transmission rate.

A central clock will be distributed as a frequency reference so that only phase differences have to be compensated during data extraction. Differential signalling, probably ECL but perhaps current-steering drivers instead, results in constant current flow between connected modules, enormously simplifying the ground distribution problem compared to normal buses.

We plan to use a narrow 16-bit data path 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 are talking about 64 pins minimum for each SCI interface circuit just on its fast end. 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. Actually, the 16 data bits will be accompanied by a clock, a flag bit, and probably a parity bit, so the numbers are somewhat larger than stated above.

Modern ECL circuits appear to be able to handle point-to-point transfers at these data rates, but some care will be required

with layout and connectors.

We are addressing the logical problems in several ways, trying to keep the system efficient by appropriate choice of protocols and trying to prevent starvation or deadlocks by providing forward-progress mechanisms.

The protocol efficiency can be affected by the format of the packets, which should be designed to provide the information in the best way for very fast processing in the interface; by the command set, which should not only match the needs of SCI but also provide the necessary mechanisms for communicating with other buses through interfaces from SCI; and by the choice of interprocessor communication primitives, the semaphor or lock mechanisms.

We are assuming 64-bit addressing, with the high 16 bits used for module selection (to be examined at very high speed).

SCI packets transfer a minimum of 16 bytes, but permit use of any contiguous subset. SCI also supports aligned block transfers of 32, 64, 128 and 256 bytes. The 16-byte packet provides for lock operations on 1, 2, 4 or 8-byte variables. The supported locks are *load and store*, which returns the original value and stores the new one; *load and add*, which returns the original value and adds the provided increment; and *test and store*, which returns the old value and if it matches a test value replaces it with a new one (useful for linked-list append).

Forward-progress mechanisms try to guarantee allocation of resources in such a way that large classes of trivial deadlocks cannot occur. For example, some sort of emission control is needed to prevent one user from hogging all of the data transmission capacity of a switch trunk or an insertion-ring "backplane." Some sort of selective acceptance of packets is necessary to prevent a saturated popular server from devoting all its resources to one user. And, some rejection mechanism may be needed in order to free space in filled queues for more important traffic. Separate queues are maintained for requests and responses, so that an overload of requests cannot block the responses which must be sent in order to free server resources.

We are developing a cache coherence mechanism which maintains a distributed directory of users of each data item, 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.

The proposed mechanism seems simple enough that it should work, but it is not trivial. We must carefully check corner cases, such as what happens if one node decides to remove itself just as another is trying to add itself onto the list. Additional system traffic is required for maintaining coherence, but it is proportional to the information transfer traffic (about double for cached items). This seems a reasonable cost in exchange for the much larger factor of parallelism it makes possible, and for moving spin-wait traffic into caches.

## Acknowledgements

Many have contributed to SCI's development already; though I cannot list them all, I wish to acknowledge a few contributions which seem to me to be particularly significant.

Paul Sweazey of National Semiconductor had the audacity to think that we might be able to do still better than our best—some of us had just finished Fastbus and Futurebus, which we thought to be limited mainly by the speed of light. Paul also brought a thorough understanding of the cache coherency problem, due to his work coordinating that task for Futurebus.

Paul Borrill of National Semiconductor, Futurebus chairman, was instrumental in our escalation of goals to much higher system bandwidths and increased parallelism through the use of switches instead of shared buses. He and other veterans of recent bus standards helped us to understand the essential limits and thus to move away from buses for SCI.

David James, originally of Hewlett Packard and recently of Apple Computer, has brought great insight into the appropriate system architecture for SCI's needs, from register and I/O architecture to distributed cache coherency and forward progress mechanisms. David is our Logical Task Group Coordinator and has also written most of our working documents.

John Moussouris, a founder of MIPS Computers, has provided critical insights into the directions we need to take in order to rendezvous with future technology, has helped put us in touch with the appropriate experts, and has helped expose problems and errors in our various prototype gedanken models.

Ernst Kristiansen of Norsk Data has provided insight from the point of view of the implementor, considering the implications on actual chip and system design.

Phil Ponting of CERN in Geneva has provided effective and vital communication and redistribution services to our many European participants.

Hans Wiggers of Hewlett Packard Laboratories has helped us examine various physical layers, and is considering the implications of an optical fiber implementation of SCI.

## Conclusion

The SCI project is moving rapidly, and has attracted participants from many of the high-performance computer companies. The proposed signalling mechanisms appear to be technically feasible (though not entirely trivial), and there appear to exist logical protocols which are compatible with our goals.

The next phase will be a more careful study of the effects of various compromises and optimizations that could be applied to our logical protocols, and the selection of suitable connectors and packaging mechanisms. There is a lot of work to be done, but the enthusiasm level is high and progress has been rapid, so we are optimistic that we can achieve a workable specification in record time.

If you would like to participate in this work, or if you would like more detailed information, 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