SCI-Based Local-Area Shared-Memory Multiprocessor *

Hermann Hellwagner, Wolfgang Karl, Markus Leberecht, Harald Richter
Lehreinheit für Rechnertechnik und Rechnerorganisation /
Parallelrechnerarchitektur (LRR–TUM)
Institut für Informatik, Technische Universität München,
D–80290 München, Germany
{hellwagn,karlw}@informatik.tu-muenchen.de

Vaidy S. Sunderam
Department of Mathematics and Computer Science
Emory University, Atlanta, GA 30322, USA
vss@mathcs.emory.edu

Abstract

This paper describes a new interconnect standard, the Scalable Coherent Interface (SCI, ANSI/IEEE Std 1596-1992), and new developments in parallel computer architecture that become possible by this technology. For instance, using SCI, shared-memory multiprocessors can be built from standard desktop machines at low costs. The technical features and applications of SCI, novel architectural opportunities it opens, and its advantages in terms of low costs and high performance, are described. In addition, a project at our institution (LRR-TUM) is outlined that sets up, and develops software for, an SCI-based distributed shared memory multiprocessor built from standard desktop computers.

1 Motivation

1.1 Demand for low-cost parallel processing

In the future, parallel processing – besides high-bandwidth communication networks – will play a key role in providing advanced services in the information marketplace or in developing sophisticated products that will succeed in an increasingly competitive and global economy. However, parallel computers must become cheaper and more convenient to program in order to be able to enter the computing mainstream.

In the quest for low-cost and widely available parallel processing facilities, networks of workstations (NOWs) have increasingly been used in recent years. This has been fostered by the public availability of packages like PVM, p4, and NXLib [14], for instance, as well as public domain implementations of MPI, which allow a NOW to be utilized as a virtual parallel computer.

In particular, small and medium enterprises, universities, and research institutions follow this approach to provide low-cost parallel computing facilities, while saving and utilizing their existing investment in desktop machines. This approach also represents an opportunity for industrializing countries to proliferate parallel computing.

Manufacturers of parallel computers currently appear to meet the demand for low-cost machines by increasingly using standard components in their products, e.g. popular RISC microprocessors or even standard workstation nodes like in the IBM SP2, rather than adopting proprietary solutions. This approach enables the manufacturers to take advantage of the low costs of mass-produced standard components as well as to exploit the rapid technical advances that are being made with the components.

*Appeared in: Proc. Int'l Workshop on Advanced Parallel Processing Technologies (APPT'95), Beijing, China, Sept. 26-27, 1995, Publishing House of Electronics Industry, China, ISBN 7-5053-3304-6, 1995.

1.2 Potential of SCI-based local-area multiprocessors

In this paper, a project is described that carries on this trend to provide low-cost, yet technically advanced and powerful parallel computers. The project attempts to build a parallel system (and develop software for it) that has the following characteristics:

Low cost by use of standard components. The machine developed in the project is to consist of industry-standard PCs (based on Pentium and the PCI bus) and a standard interconnect, the Scalable Coherent Interface (SCI) [12]. The SCI standard and interconnect system plays a key role in this project. As outlined below, it has novel properties that facilitate sophisticated architectural features to be provided at low costs.

High-bandwidth and low-latency interconnect. From a hardware perspective, the current state of parallel computing in NOWs has two severe drawbacks: insufficient interconnect bandwidth and high latencies. Today's Ethernet LANs provide a raw bandwidth of 10 Mbits/s, much of which is consumed by the communications libraries and the Unix networking protocols (TCP and UDP). The communications software also incurs considerable latencies, in the range of 1000 μs. Consequently, only parallel programs with low communications requirements can make good use of Ethernet-based NOWs. Emerging new LANs based on FDDI, Fibre Channel, or ATM switches (will) deliver an order of magnitude more bandwidth (100–155 Mbits/s). However, it has been argued that this is not sufficient for efficient parallel processing in a NOW; at least two orders of magnitude of bandwidth increase will be required [11], with a comparable decrease in latency.
SCI is defined to provide 1 Gbit/s or 1 Gbyte/s bandwidth, while offering the chance for low-latency communications software if specific SCI protocol features, e.g. guaranteed data delivery, are exploited judiciously. As an example, Dolphin Interconnect Solutions, a Norwegian company offering SCI products, has demonstrated application-level communications latency below 10 μs in an SCI-based cluster of SPARCstation20s [8].

Flexibility in terms of use or configuration. From a software point of view, there is another drawback to NOWs: poor programmability. This is due to the fact that parallel programming in a NOW almost exclusively uses message passing as the underlying communication and synchronization model, mainly based on PVM. Use of Linda, p4 monitors, or other shared-data programming constructs is insignificant, although a shared-data model is widely held as being more convenient.
SCI provides an unprecedented flexibility in terms of use or configuration of a NOW, and in programming model support. On the one hand, SCI protocols provide message-passing transactions. More importantly, however, SCI allows distributed shared memory (DSM) systems to be built, on the basis of (non-coherent or coherent) memory access and atomic synchronization transactions defined in the standard.
In summary, SCI provides a unique migration path for the use of a NOW: it may be used as a conventional LAN, as a distributed memory parallel system, and even as a non-coherent or coherent (distributed) shared memory machine.

The focus of the project is to develop the hardware and software for a PC- and SCI-based shared memory parallel computer. The SCI technology enabling such a system as well as the work that is being or will be carried out in the project are described in the following.

2 The Scalable Coherent Interface "Bus"

2.1 SCI features

Having emerged from the discussion around the Futurebus+ standard, SCI tries to avoid the typical problems of backplane buses while still offering the same functionality. Yet it manages to keep a high degree of scalability. This is done by incorporating several remarkable features within the standard:

- Electrical signalling problems are avoided by the use of unidirectional, point-to-point links that connect SCI nodes. Because of this, high transmission speeds are possible. The current SCI standard defines 1 Gbit/s or 1 Gbyte/s transmission bandwidth (depending on the medium in use) over each link.

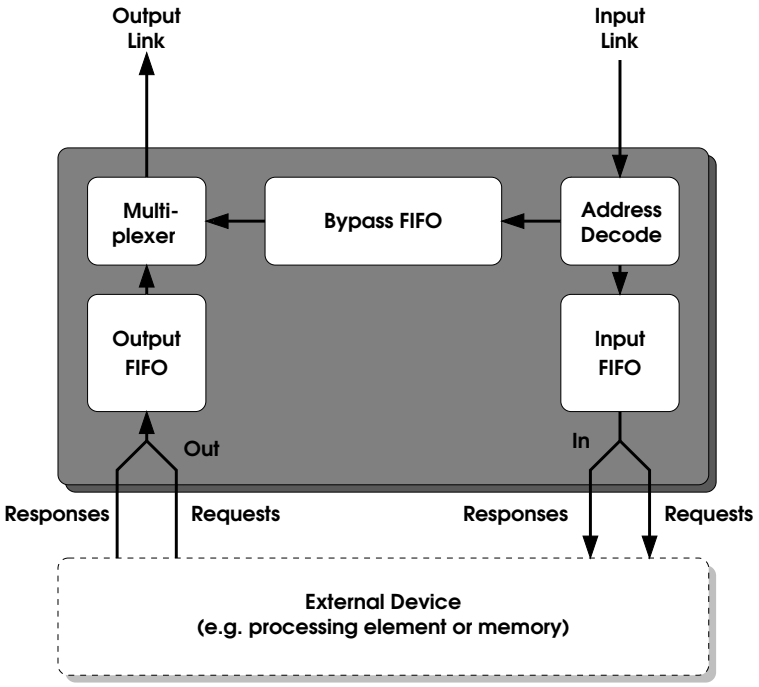

Figure 1: Block diagram of an SCI node

- Each node is required to have at least one incoming and one outgoing link. The block diagram of a typical SCI node is shown in fig.1 [10]. A packet arriving on the incoming link is inspected and, if required, forwarded via the outgoing link. By means of the Bypass FIFO, this can happen concurrently to the node transmitting a packet of its own. Dual request and response queues are used to avoid deadlocks.
- Due to its distributed nature, SCI features split transactions and provides a global 64 bit address space. The 16 most significant address bits determine the node number while the remaining 48 bits are used for node-internal addressing.
- SCI defines shared-memory protocols that, seen from a processor, look like those of backplane buses, but are implemented using packet protocols. These include memory read, write, and lock transactions. Together with the global physical address space, these protocols provide the basic mechanisms for setting up a DSM multiprocessor.
- Performance increase can be obtained by using SCI's optional built-in cache coherence protocol. It is based on distributed directories and is thus as scalable as the whole concept. Both NUMA and CC-NUMA machines are therefore made possible through SCI.

2.2 SCI applications

The SCI standard defines a general-purpose set of protocols that can be used in a variety of applications. Applications that have been proposed or demonstrated, include [11]: optical local-area networks, I/O interfaces, data acquisition systems [5], desktop workstation busses, and highly parallel, tightly coupled multiprocessors [6].

When targeting a tightly coupled DSM multiprocessor, SCI's flexibility can be exploited. Since its communication protocols support both message–passing and shared–memory programming models, architectural support similar to what has been investigated in projects like the Stanford FLASH [13] and

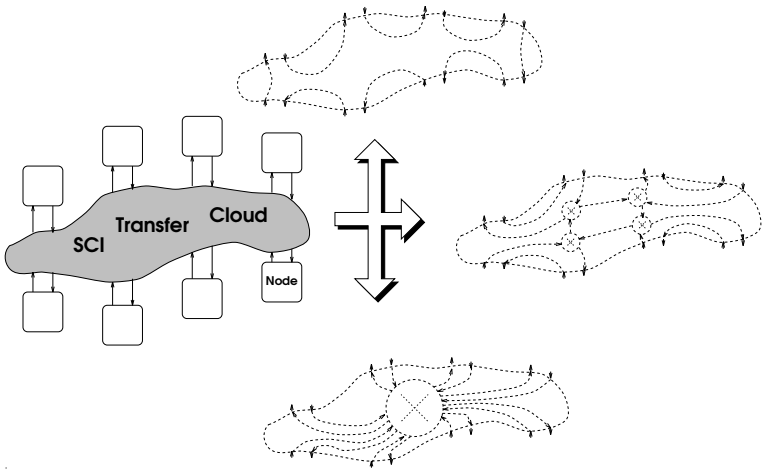

Figure 2: SCI systems: rings, ringlets connected by switches, multi-stage interconnection networks

the MIT Alewife architectures [1] is available. Shared memory can be provided in both of its NUMA or CC-NUMA variants, given that the SCI implementation incorporates the cache coherence protocols.

Various architectural design decisions of parallel computers are possible; see fig.2. Lowest-cost, small SCI systems typically will use a ring topology connecting one node's output to its neighbor's input [10]. More sophisticated SCI systems will be based on ringlets connected via switches or multi-stage interconnection networks.

3 The SMiLE Project

3.1 Objectives

At LRR-TUM, a project called SMiLE (Shared Memory in a LAN Environment) commenced in which we aim at developing hardware and software of a DSM parallel computer based on Pentium PCs and SCI. The general objectives of the project are:

Demonstration of SCI-based PC cluster. Development and demonstration of a low-cost, yet powerful SCI-based multiprocessor using standard PCs (Pentium and PCI-based), is one of our primary goals. Such a "poor man's version" of a DSM parallel computer could be employed by almost any educational institution for practical parallel processing courses and by small and medium-size companies for compute-intensive simulations in product developments.

Development of software for SCI-based multiprocessors. Software development for SCI multiprocessors represents a major research challenge. Software has to be provided on the levels of the operating system, parallel programming environments, and tools for parallel systems.

Research into efficient parallel programming of DSM systems. In the long term, the SCI multiprocessor will provide a vehicle for experimental research into efficent use of a DSM parallel machine. The key to efficiency is to provide and exploit data locality, and to minimize false sharing. Tools for performance analysis, and run time mechanisms to move or replicate data and to migrate processes are important components to improve locality of data accesses and, thus, to increase efficiency of a DSM machine.

3.2 Experimental Platform

For initial software development (see below), an SCI "ring" of two SPARCstation2s (SS2) has been set up using 25 MHz Dolphin SBus-SCI adapter cards [7]. The cards and cables are advertized to sustain application throughput of 10 Mbytes/s[1]. With faster machines (SS20), round-trip, ping-pong latencies of 8 μs have been reported [8].

The device driver for the SBus-SCI card supports two elementary software interfaces. The *raw channel interface* provides a set of `ioctl()` calls for establishing and controlling connections between processes on different machines. Data can then be transferred across channels using `write()` and `read()` functions, which are supplemented by a `poll()` system call in the most recent version of the driver (V1.4). Transfers are handled via DMA, with only a single intermediate buffer in kernel space being involved. The *shared-memory interface* consists of `ioctl()`s for constructing shared-memory segments as well as setting up address mappings in the interface so that remote segments can be accessed transparently. Processes can include shared segments in their virtual address space using the `mmap()` system call.

3.3 Current work

Work is currently being pursued on a hardware, a performance analysis, and a software track.

3.3.1 Hardware

The hardware track comprises the design of PCI-SCI interface cards. These interfaces will be used to set up a local compute cluster using standard Pentium based PCs. Further software development will be carried out on this platform.

Although Dolphin has announced a PCI-SCI adapter card, we commenced our own PCI-SCI interface design using VHDL-based design and simulation tools. The motivation for this work is that the performance analysis tool PATOP and the debugging tool DETOP, both developed at LRR-TUM [15], are desirable to be ported to and extended for an SCI platform, and be supported by monitoring hardware. The PCI-SCI interface monitoring "hooks" have to be identified and designed. Both message-passing and a DSM programming paradigm are to be supported.

3.3.2 Performance analysis

SCI specifies a topology independent communication protocol with the possibility of connecting up to 64k nodes. Therefore, the work in this area is to evaluate SCI-based multicomputer systems with different topologies and various configurations.

At the European Organization for Nuclear Research (CERN), Geneva, Switzerland, an SCI simulation system called SCILab has been developed [3] [17]. SCILab consists of an SCI modelling program and a configurator to generate SCI systems for simulations. It is based on MODSIM II, a general-purpose, block-structured, high-level programming language which provides direct support for object-oriented programming and discrete-event simulation [4].

Using these tools, our studies aim at investigating high-performance SCI-based systems. The evaluation of the influence of parameters such as link speed, interface design, and SCI protocols for different topologies will have a strong impact on our own hardware design.

Among the topologies that are investigated in this project, hierarchical structures are the most promising since they optimally reflect data locality. A hierarchical multistage network based on CLOS nets and a recursive single-stage network based on rings of rings are our main focus. In the quest for performance optimization of process-to-process bandwidth and latency, we will address the issues of adaptive routing and hot spot elimination by incorporating additional features into the SCI switches.

3.3.3 Software

In the software area, work is carried out on the SS2 SCI cluster as described above, to acquire initial programming experience with the low-level SCI driver interfaces, to develop basic concepts and interfaces

[1] The best achievable bandwidth is only a tenth of the theoretical maximum according to Dolphin, due to a simple request-response transaction protocol implementation used currently; multiple outstanding transactions will soon be available.

for parallel programming and development tools, and to port existing environments onto "native SCI" for higher efficiency. More specifically, the following work packages are currently being pursued:

- *SCI Programming Experience and Measurements.* This task ports the communication benchmarks from the PARKBENCH suite [16] and a small relaxation algorithm to the programming interface delivered with the SBus-SCI adapter card and is to measure the resulting communications performance. Initial programming experience and basic programming patterns and interfaces will be the major results of the task.

- *Port of PVM to native SCI.* Since the *Parallel Virtual Machine (PVM)* system [9] has emerged as the *de-facto* standard for message-passing libraries on NOWs, a port of PVM onto native SCI is of particular interest. The goal of this work is to provide efficient message passing on PVM application level by directly implementing the PVM communications routines on the raw channel interface. Significant reduction of message transfer latencies is expected through the use of the native SCI interface rather than TCP/IP.

  Since at present the PVM port is presumably the most interesting part of the project, we report our early results for this task. In the initial phase of the task, the design and a first implementation of `pvm_scisend()` and `pvm_scirecv()` calls have been completed. These are SCI counterparts of `pvm_psend()` and `pvm_precv()` routines with the direct-route option, which provide the fastest communication method in PVM over Ethernet. These routines transfer single data type buffers and, with direct routing, use direct process-to-process stream (i.e. TCP) connections for message exchange.

  Elementary implementations of `pvm_scisend()` and `pvm_scirecv()` were tested on the experimental platform. Communication tests consisted of up to 1000 cycles of sending messages of different lengths from a sender to a receiver, with a final acknowledgement from the receiver marking the end of the timing test.

  In a first, skeletal version of the calls, the sender-receiver pairs, message lengths, and protocol handshakes were "hardcoded" into the routines. Thus, this test measures the best possible "steady-state" performance that would be obtained by an optimized implementation. The performance results are shown in table 1 in the columns labelled *Best*. Lowest latencies for near-zero byte messages are about 80 μs, approximately a factor of 20 better than the best possible with native `pvm_psend()` and `pvm_precv()` using TCP/IP over Ethernet. Latencies are, of course, dominated by software overheads, so we expect to achieve optimal figures of 10–20 μs when operating on fast workstations, e.g. a 60 MHz SS20. The bandwidth obtained is also high; for large messages, throughput approaching 9 Mbytes/s could be achieved, only about 10% less than the figure published by Dolphin.

Message size (bytes)	Bandwidth (kB/sec)		Latency (ms)	
	Best	Actual	Best	Actual
2	24	10	0.08	0.21
32	134	61	0.24	0.53
256	247	94	1.02	2.74
4096	2767	912	1.48	4.43
32768	8811	3650	3.71	8.97

Table 1: Initial PVM-SCI performance results

While the above figures are promising, they do not represent true performance, because a fully operational PVM implementation would necessarily have to performing multiplexing, avoid blocking, and dynamically negotiate connections with multiple partners. Incorporating these facilities into the routines, however, significantly degrades performance, by a factor of 2 to 3, as shown in table 1 in the columns labelled *Actual*.

While these figures are still a factor of about 8 better than the optimum of `pvm_psend()` and `pvm_precv()` over Ethernet, they do not yet reflect the true SCI potential. At the time of this

writing, we are investigating the causes for the loss in performance. An educated guess suggests that improvements are possible along the following lines: tuning of our implementation, e.g. by fine-tuning retransmission delays when write buffers are full; improvements in device driver software and hardware SCI protocol implementations; and exploiting shared memory and other specific SCI protocol features such as guaranteed data delivery. We believe that, with these improvements, a full implementation of `pvm_scisend()` and `pvm_scirecv()` can deliver an order of magnitude better performance than the native PVM counterparts over Ethernet.

- *Design of a DSM programming model.* Initial studies on a DSM programming model are being done. The model must suitably take into account (i.e. abstract from, or expose to the programmer for performance reasons) the (physically) distributed nature of the (logically) shared memory. Work like SVM FORTRAN [2] appears to be a good starting point.
- *Performance analysis tool PATOP for a DSM environment.* This work is planned to extend the tool PATOP which has been developed and refined at LRR-TUM for distributed-memory systems [15], for a DSM platform. The functional specification of monitoring functions to capture the usage of shared variables in a DSM already commenced.

4 Conclusion

In this paper, we outlined the SCI technology, its flexibility and potential applications, and the SMiLE project at LRR-TUM that aims at setting up, and providing software for, a DSM multiprocessor from standard desktop machines and an off-the-shelf SCI interconnect. At present, however, due to the importance of the PVM system, our focus is on a PVM port onto an experimental SCI cluster. Preliminary results were reported for this task.

Clearly, our experience and results are too preliminary to draw conclusions about the promise of the SCI technology and the SMiLE approach from. However, we can still be confident that the concept of an SCI-based local-area shared-memory multiprocessor is viable and capable of yielding high-performance parallel processing facilities at low costs.

References

[1] Agarwal, A., Chaiken, D., Johnson, K., Kranz, D., Kubiatowicz, J., Kurihara, K., Lim, B., Maa, G., Nussbaum, D.: The MIT Alewife Machine: A Large-Scale Distributed-Memory Multiprocessor. Technical Report MIT/LCS/TM-454, MIT Laboratory for Computer Science (1991).

[2] Berrendorf, R., Gerndt, M., Nagel, W.E., Pruemmer, J.: SVM FORTRAN. Technical Report KFA-ZAM-IB-9322, Research Center Jülich (KFA), Central Institute for Applied Mathematics (1993). `ftp://ftp.zam.kfa-juelich.de:pub/zamdoc/ib/ib-93/ib-9322.ps`.

[3] Bogaerts, A., Wu, B.: The SCILab Cook Book. Internal Note, CERN, Geneva, Switzerland (1993).

[4] CACI Products Company: MODSIM II, The Language for Object-Oriented Programming. Reference Manual, Tutorial, CACI Products Company, La Jolla, CA 92037.

[5] CERN: Information on the RD24 and related projects. `http://www1.cern.ch/HSI/sci/sci.html`.

[6] Convex Computer Corporation: Convex Exemplar SPP1000 Systems Overview. Order Number 080-002293-000, Convex Computer Corporation (1994).

[7] Dolphin Interconnect Solutions: 1 Gb/s SBus-SCI Cluster Adapter Card. Dolphin Interconnect Solutions, White Paper (1994).

[8] Dolphin Interconnect Solutions: Dolphin Breaks Cluster Latency Barrier with SCI Adapter Card. Dolphin Interconnect Solutions, Press Release (1995). `http://www.dolphinics.no`.

[9] Geist, G.A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V.S.: PVM - Parallel Virtual Machine: A Users Guide and Tutorial for Networked Parallel Computing. M.I.T. Press, November 1994.

[10] Gustavson, D.B.: The Scalable Coherent Interface and Related Standard Projects. IEEE Micro 12(2) (1992) pp. 10–22.

[11] Gustavson, D.B., Li, Q.: Local-Area MultiProcessor: the Scalable Coherent Interface. SCIzzL, The Association of SCI Local-Area MultiProcessor Users, Developers, and Manufacturers (1994). `http://sunrise.scu.edu`.

[12] IEEE Std 1596-1992: Scalable Coherent Interface. IEEE CS Press (1993).

[13] Kuskin, J., Ofelt, D., Heinrich, M., Heinlein, J., Simoni, R., Gharachorloo, K., Chapin, J., Nakahira, D., Baxter, J., Horowitz, M., Gupta, A., Rosenblum, M., Hennessy, J.: The Stanford FLASH Multiprocessor. Proceedings of the 21st Annual International Symposium on Computer Architecture. ACM Computer Architecture News 22(2) (1994).

[14] Lamberts, S., Stellner, G., Ludwig, T.: NXLib Users' Guide. V1_1_3 edition. Technische Universität München (1994). `ftp://ftpbode.informatik.tu-muenchen.de:NXLIB/NXLibUgV1_1_3.ps.Z`.

[15] LRR-TUM: Annual Report 1994, Lehrstuhl für Rechnertechnik und Rechnerorganisation (Prof. Dr. A. Bode). Institut für Informatik, Technische Universität München, TUM-INFO, April 6, 1995. `http://wwwbode.informatik.tu-muenchen.de/`.

[16] PARKBENCH Committee: Public International Benchmarks for Parallel Computers. Technical Report (1994). `http://www.netlib.org/parkbench/parkbench.ps`.

[17] Wu, B., Bogaerts, A.: SCILab – A Simulation Environment for the Scalable Coherent Interface. Proceedings MASCOTS'95. IEEE CS Press (1995) pp. 242–247.