Tutorial: the Scalable Coherent Interface & Local Area MultiProcessor

David B. Gustavson, ANSI/IEEE Std 1596 (SCI) Chair
Executive Director: SCIzzL, the association of SCI developers, manufacturers and users
and the SyncLink Trade Association
Santa Clara University
1946 Fallen Leaf Lane
Los Altos, CA 94024-7206
tel: 415-961-0305
fax: 415-961-3530
email: dbg@sunrise.scu.edu

Faster and faster access to
– more and more data
– at lower and lower cost

We need not just to move information, but to share information.

• Distributed databases
• Distributed equipment
• Control systems
• Workstation clusters
• Powerful I/O systems (disk arrays, video, sensor arrays, etc.)
• Massively Parallel Processors
• Supercomputers

Supercomputers are Expensive!
(and inconvenient for use in military vehicles…)

• Custom, low-volume design
• Cost-effective for fewer and fewer problems
• Competing with microprocessors is tough
• Multiple microprocessors must be the way
  – but Clusters/NOW performance limited
    • poor latency
    • low bandwidth
    • hard to rewrite applications
Technology advances

by leaps and bounds, and gradual refinements: punctuated evolution

New technology is barely feasible

• so it starts with "bottom-up" designs.
• iterate design and
• iterate goals until OK
• very Expensive!

Goal: Economical solutions

• reduce these costs
• take maximum advantage of technology
• use standards

Method: Scalable architecture

• technology insensitive, yet practical
• anticipate needs
• apply RISC principles
  – to maximize bandwidth
  – to minimize latency
Why can we do this now?

- Technology has advanced enough
- We’ve learned from experience
- Not very many solutions are possible
  – So we can agree and standardize!

Economic scalability

- Need building blocks
- Standard interfaces
- Large numbers lower costs
- Parallel processing is the key

Interface standards

- Essential for multiple sources
- Essential for building-block components
- Essential for competition
- Efficient because well understood

Fundamental limits

- Speed of light is constant
- Physical size is constant
  – so latency is constant
  – but costs more and more!
    (measured in wasted cycles, as clock speeds rise)
Non-fundamental limits

- Bandwidth
- Frequency
- MIPS
- Memory
- Disk space
- Cost

High performance requires hiding distance.

You can’t beat the speed of light!

- Minimize accesses to remote information.
- Fetch *blocks* of data to optimize bandwidth usage.
- Cache data near where it is needed (usually near a processor).

Problem: Caches imply multiple copies.

Copies must be kept consistent!

- Manage them in software (very difficult, and slow)
- Support coherence in hardware (easy if designed into the system)

Intelligent synchronization, mutual exclusion

- atomic loads and stores
  - for certain standardized sizes, e.g. 4 and 8 bytes (software *needs* to know!)
- swap
  - add work items to lists
- compare & swap
  - remove items from lists
- fetch & add
  - delegate work to processors
  - barrier synchronization
Message passing

- Efficient MPP operation
- Compatibility with today’s systems
- Modularization/isolation of subsystems
- but, the software must be written for the specific architecture and configuration used

Shared memory

- Uniform model easier for software
- Eases transition from workstation to MPP
- Efficient protocol
  - Packet communication is part of a load or store instruction, not a call to a library subroutine.
  - So, latency is several orders of magnitude less!

Multiprocessors require Forward Progress

- Fair allocation of bandwidth
- Fair access to shared resources
  - Queue space
  - Services

Interconnect independence

- Don’t place costly burdens on interconnect
  - don’t require in-order delivery
  - don’t require reliable broadcast or multicast
    - but support it as an option for cases when it’s practical
  - don’t require uniform or specific topology
- Do support interfacing to other worlds
  - buses
  - networks
  - I/O systems
Include Powerful I/O

- Devices
  - disks
  - displays
  - workstations
  - specialized processors
  - existing buses
- Cables and Fibers are required
- I/O must be coherence-aware

Networking

- Efficient underlying protocol
- Low software overhead
- Backward compatible to ease transition
- Minimize wasted moves

Broadcast and Multicast

- Very useful if reliable, but
- Extremely expensive if reliable
- so:
  - don’t require in all systems
  - don’t rely on for essential protocols (e.g. coherence)

Conclude we need:

- Standard interface for MPP building blocks
- Optimize for lowest possible latency cost
- Scalable for big or small systems
- Versatile
- Technology independent
- Integrable, fit on part of one VLSI chip
The solution:

- Transmission lines, without taps (electrical or optical point-to-point links)
- No reverse flow-control signals (scaling problem)
- Packetize (scaling)
- Unidirectional continuous transmission (scaling)
  - Request/response, split transactions
- Memory architecture (versatile, simple to use, efficient)

Practical optimizations:

- Narrow fast links save cost and space, enable integration
- Electrical links use differential signaling (fast, low-swing, create little noise, reject incoming noise)
- Unidirectional continuous transmission (low noise, low latency, rapid synchronizing)
- Support ring connections in addition to switched connections (low cost, good performance)

The result: SCI

- Scalable
  - Size, speed, cost, technology
- Coherent
  - Where and when needed
    - Costs nothing for coherence of unshared data
    - Or, choose dynamically, based on address
    - Or, add it later, pay nothing now
- Interface
  - Standards allow: intercommunication, interoperability, interaction, modularity, multiple sourcing, competition.
Node interface structure

Idle deletion

1) Before idle discard (time = t + 0)
   - 2-symbol delay
   - Save go bits

2) After idle discard (time = t + 1)
   - 2-symbol delay

Idle insertion

1) Before insert (time = t + 0)
   - 2-symbol delay

2) During insert (time = t + 1)
   - 2-symbol delay

3) After insert (time = t + 2)
   - 2-symbol delay

Transaction Phases
Transaction Formats

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>readxx*</td>
<td>header</td>
</tr>
<tr>
<td>writexx*</td>
<td>header 16,64,256</td>
</tr>
<tr>
<td>movexx*</td>
<td>header 0,16,64,256</td>
</tr>
<tr>
<td>eventxx*</td>
<td>header 0,16,64,256</td>
</tr>
<tr>
<td>locksb</td>
<td>header 16</td>
</tr>
</tbody>
</table>

Note: xx represent one of the allowed data block lengths (number of data bytes, on the right after the header)

Send-packet format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flag</td>
<td>data (16 bits)</td>
</tr>
<tr>
<td>1</td>
<td>targetId</td>
</tr>
<tr>
<td>1</td>
<td>flowControl</td>
</tr>
<tr>
<td>cmd</td>
<td>sourceId</td>
</tr>
<tr>
<td>1</td>
<td>timeOfDeath</td>
</tr>
<tr>
<td>sequence</td>
<td>addressOffset/status</td>
</tr>
<tr>
<td></td>
<td>(optional) data</td>
</tr>
<tr>
<td>0</td>
<td>CyclicRedundancyCode (CRC)</td>
</tr>
</tbody>
</table>

Masked from CRC calculation (0 assumed)

64-Bit Addressing Model

<table>
<thead>
<tr>
<th>Node Type</th>
<th>Memory Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0]</td>
<td>256 Tbytes</td>
</tr>
<tr>
<td>[4K-1]</td>
<td>2Kbytes</td>
</tr>
<tr>
<td>[64K-1]</td>
<td>4Kbytes</td>
</tr>
<tr>
<td>private</td>
<td></td>
</tr>
</tbody>
</table>

Directly accessible from 32-bit CSR Architecture systems

Resource bottlenecks

<table>
<thead>
<tr>
<th>Requester Type</th>
<th>Resource Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>requester[0]</td>
<td>interconnect bottleneck</td>
</tr>
<tr>
<td>requester[1]</td>
<td>responder[0]</td>
</tr>
<tr>
<td>requester[N-1]</td>
<td>responder[N-1]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requester Type</th>
<th>Resource Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>requester[0]</td>
<td>shared-responder bottleneck</td>
</tr>
<tr>
<td>requester[1]</td>
<td>responder[0]</td>
</tr>
<tr>
<td>requester[N-1]</td>
<td>responder[N-1]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requester Type</th>
<th>Resource Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>requester[0]</td>
<td>(many-to-one) queue allocation</td>
</tr>
<tr>
<td>requester[1]</td>
<td>responder[0]</td>
</tr>
<tr>
<td>requester[N-1]</td>
<td>responder[N-1]</td>
</tr>
</tbody>
</table>
Idle encoding

| ipr ac cc hg lg old lt ipr* ac* cc* hg* lg* old* lt* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |

Error checking

Fair allocation of bandwidth

(1) \( \text{sendEnable} = \text{idle.go} \)
(2) \( \text{while (sending)} \) \( \text{save.go} \|= \text{idle.go} \)
(3) \( \text{while (not empty)} \) \( \text{save.go} \|= \text{idle.go} \)
(4) \( \text{if (done)} \) \( \text{idle.go} \|= \text{save.go} \)

Local transaction components

Local transaction busied
Queue allocation is needed

Queue reservations

A/B labels

Remote transaction
Switch interface

Remote transaction, echo lost

Distributed cache directory 1

Distributed cache directory 2
Distributed cache directory 3

processors

prepend queue

wannabe

wannabe

head

tail

execution
unit

cache

memory

• coherent line

• noncoherent line

CPU_D

CPU_C

CPU_B

CPU_A

Distribution cache directory 4

processors

head

mid

mid

tail

execution
unit

cache

memory

• coherent line

• noncoherent line

CPU_D

CPU_C

CPU_B

CPU_A

SCI module

a Scalable interface

• Distributed
  – Many communication links
  – Many memories ( = many banks)
  – Distance-independent protocols
• Message passing
• Shared memory
• Cache coherent
Scalable means —

- Low-volume LARGE systems share economies of high-volume small systems.
- Packets, not cycles
  - Address, control, data: all pipelined, in point-to-point links, length independent.
- 64-bit address (room to grow)
- 64K nodes (processors, clusters, modules, ...)
  - but still cost-effective for one processor.
- Options interoperate — affect performance, not correctness.

SCI offers:

- 1 Gbyte/s links
  - 16-bit differential ECL
  - 250 MHz squarewave max
- 1.25 Gbit/s links
  - coaxial cable
    - 0.7 V
    - 20 meters
  - fiber optics
    - kilometers
- IEEE Std 1301.1 Modules

SCI offers:

- Simple bridges
  - No cache coherence overhead
- Live insertion and removal
- Executable specification, C-code
  - Logical layer
    - symbol processing
    - packet processing
  - Coherence protocol
    - node behavior
  - Convenient for
    - simulation
    - design verification
  - Essential for unambiguous specification

Competitive approaches to solving the problem

- PCI (tight physical constraints, slow, bottleneck, bridge deadlocks, little support for caching)
- Mbus (tight physical constraints, bottleneck)
- RamBus (tight physical constraints), not a system interconnect
- RamLink
  (single-controller, simple SCI derivative, but supports processor-to-SCI-bridge connection)
Competitive approaches to solving the problem

- **FibreChannel**
  - Combines prior experience with Networking
  - I/O Channels
  - to solve all those well-known problems.

- **But, this is “Preparing to fight the previous war!”**
  - The old networking and I/O-channel models are too inefficient for tomorrow’s multiprocessor systems.
  - Inevitable software overhead for translating from application’s load/store paradigm to FC’s paradigm.

Competitive approaches to solving the problem

- **FibreChannel**
  - I/O model
    - move large blocks (poor latency characteristics)
    - software support adds latency
  - no support for memory model
  - no support for caching
  - mostly not yet an ANSI standard (first parts became a standard in 1995)
  - Many products announced, but not available or can’t talk to each other because built to different drafts
  - But: can transport SCI packets

- **ATM**
  - I/O model
  - connection based
  - excellent for wide area applications
  - good latency characteristics
    - constant, good for telephone
    - could be short, unfortunately isn’t
  - no support for memory model
  - no support for caching
  - free to drop 48-byte packets when congested
    - can cause larger blocks to be retransmitted
  - not a standard yet

SCI Status

- Started November 1987, SuperBus study group
  - Paul Sweazey, National Semi (then Apple, now National again), Futurebus Cache Coherence Task Group Leader
- Became P1596 Working Group in July 1988
  - David Gustavson, Stanford University (Fastbus, Futurebus,…, VME)
- Finished January 1991, Draft 1.00 to ballot
  - Passed April 1991, 92%
  - But many requests, comments, clarifications, etc.
- Revised November 1991, Draft 2.00 recirculated to voters
  - Incorporated minor technical, major editorial changes
  - One negative vote remains: we won’t change C to Pascal!
- Ballot recirculation passed January 1992
- Approved by IEEE Standards Board (March ’92), ANSI (Oct.’92)
- Published August 1993, with C-code on diskette, simulation
- IEC DIS 13961, awaiting publication in 1995
Gordon Bell says:

- “SCI is the only game in town. Computer companies ought to be implementing it right now. It probably isn’t going to be the ultimate answer, but we aren’t going to find out what changes might be desirable until we get some experience.
- There’s nothing else around that’s even close. As a minimum, SCI makes message passing, paging and other mechanisms run well.”
- (certified quote, Hot Chips V, August 1993)

SCI will be used by the military

- US Navy
  - Next Generation Computing Resources, High Speed Data Transfer Network
  - Lachenmaier says: “SCI can change the way computers are designed.”
- Canadian Navy
  - Next Generation Signal Processor
- Aircraft
  - SAE Avionics Systems Division, sensor and video
- Joint Advanced Strike Technology (Navy, Marines & Air Force)
  - SCI for main system interconnect
  - SCI for subsystem-internal use
  - Replaces 6 or 7 interconnects used previously

SCI is being used commercially

- H-P (Convex) shipping Exemplar 1Q95
- Unisys announced, not delivering yet
- Sequent revealed, Quad-P6’s not announced
- Data General, revealed, not announced
- IBM AS/400 revealed, not announced, building fast chips
- NCR revealed, not announced
  Stronger after AT&T breakup!

Commercial Users

- Siemens Nixdorf announced, not delivering
- Intel supporting others in interfacing SCI to P6 bus, Quad P6 board (Sequent), Byte mag
- Sun: not talking, but working on SCI, did significant public benchmark for SCI
- SGI: not talking, but working on SCI-like?
- Cray: SCX (GigaRing) is major SCI implementation, dual rings, fast wide CMOS, for I/O initially, moving to standard SCI-2
Extensions and Support for SCI

- Work in progress or completed
- Work being considered
- Related projects

Work in progress or completed

- P1596.1 SCI Bridge Architecture
- P1596.2 Extensions for Kiloprocessor Systems
- P1596.3 Low-Voltage Differential Signals (passed Sponsor Ballot)
- P1596.4 RamLink high speed RAM (passed Sponsor Ballot)
- IEEE Std 1596.5 Data Interchange Formats (approved June, 1993, published)
- P1596.6 RealTime Working Group
- P1596.7 SyncLink high speed RAM Sept’95

P1596.1 SCI Bridge Architecture

- Chair Clay Hudgins, Harris
- Vice Chair Khan Kibria, Unisys
- Study general bridge issues
- June 1994 changed directions to cover general switch and bridge issues, originally was looking only at SCI/VME bridges.

P1596.2 Extensions for Kiloprocessor Systems

- Started by Ross Johnson, Univ. of Wisconsin
- Chaired by Stein Gjessing, Univ. of Oslo
- Vice Chair David V. James, Apple Computer
- Tree-structured (STEM) and topology-optimized (GLOW) directories
  - no lock bits required
- Request combining
  - no state saved in interconnect
- Efficient barriers
- Call-back and other optimizations
- Finish 1996 (This is good CS, parts getting urgent…)
P1596.3 Low-Voltage Differential Signals

- Started by Gary Murdock, National Semiconductor
- Chaired by Stephen Kempainen, National Semiconductor
- Fast signals need to be small
- Small signals need to be differential
- Compatible with CMOS, GaAs, 5, 3.3, 2 Volt processes
- 0.25 V centered on +1.2 V
- Define 4, 8, 32, 64, & 128-bit-wide(!) SCI link signals
- Passed sponsor ballot, finished '95. EIA/TIA using at 622Mb/s.
- Used in QuickRing, IBM 1 GByte/s BiCMOS, Vitesse/Unisys 1 GByte/s GaAs, Motorola Optobus, LSI Logic cells

Low Voltage Differential Signaling

Transmitter Design model

P1596.4 RamLink high speed RAM

- Chaired by Hans Wiggers, Hewlett Packard
- Gain access to high internal bandwidth
  - SCI point-to-point link technology
  - Simplified SCI protocols (one master)
- Reduce number of chips in basic memory system
- Reduce number of chips in each expansion option
- Increase bandwidth, keep latency low
- Support EEPROM, cached disks, etc.
- Excellent processor interface, much better than buses
- Passed sponsor ballot, finished 1995
- MicroUnity MediaLink is a descendant of this
P1596.7 SyncLink, SLDRAM

- Apply RamLink protocols to RAM array
- Chaired by Bill Vogley, Texas Instruments
- Starting 2Q95, finishing 4Q96?
- Define SyncLink signal layer, JEDEC SSTL
- Dual bus, 8 bit 2 ns from controller
- 16 bit 4 ns from RAMs to controller
- Optimized protocols for RAM applications
- High bandwidth to reduce parts count

IEEE Std 1596.5 Data Interchange Formats

- Chaired by David V. James, Apple Computer
- Define formats for use in heterogeneous multiprocessors
  - Data types
  - Alignment
- Define names for future compiler support
- Specify subset of formats to be used for interchange
- Ballot passed, recirculation early 1993
- Approved by IEEE Stds board June 1993
- Published

P1596.6 SCI-RT RealTime

- Compatible with base SCI via bridge
- Give up SCI’s forward progress guarantees for unknown computing load
- Gain more-predictable behavior (latency) given known realtime load
- Reinterpret or add packet priority bits
- Discard or bypass low-priority packets when they block high-priority packets
- Chaired by Ralph Lachenmaier, succeeding LCdr. Brad Stewart
- Main issue remaining is how to enforce priority
SCI/RT issues

• Scheduling is harder in systems with:
  – Cache memories
    • task performance depends on previous tasks
  – Multiple processors
  – Distributed components
    • speed of light limits information currency
    • no perfect view of system’s priorities

SCI/RT issues

• SCI’s forward progress mechanisms prevent starvation, but may increase the width of the latency distribution.
• Priority-based scheduling relies on a scheduler to understand the application and assign priorities to ensure that every important task gets done.
  – Priority-based protocols should reduce the width of the latency distribution.

Rate Monotonic Scheduling

• Rigorous theory, can determine schedulability of given tasks on a given system —
• but satisfying RMS assumptions isn’t always easy,
• and scheduling real systems is nontrivial.

RMS support

• Minimize priority inversions, where low-priority packets delay higher priority ones.
• Support priority for software tasks, and for everything the software task needs (e.g. the communication interconnect).
RMS support considerations

- Completely eliminating priority inversions is impossible.
  - Critical sections, mutex’s, interrupt handlers
  - Low priority packets entering a fiber or already in flight
- Reducing inversions can be expensive
  - Hardware priority-sorting queues
  - Large separate queues for each priority

RMS support tradeoffs

- Since some inversion is inevitable, how much shall we pay for small improvements?
- Some applications are compute-bound with little communication on the interconnect.
- Some applications are interconnect-limited.

SCI/RT alternatives

- Enforce priority everywhere in interconnect.
  - Hardware priority-ordering queues
  - Truncate and overwrite low-priority packets that are in flight?
  - Allow interconnect to store (queue) low-priority data, deliver it with high latency.

SCI/RT alternatives

- Support priority minimally in interconnect
  - Reduce interconnect storage
  - Minimize worst-case latency
  - Transport priority information, but
  - only distinguish fair vs. unfair (priority) packets.
  - Duplicate queues, one set for each priority
SCI/RT alternatives

- Some hybrid, intermediate between the previous two extremes.
- Cost/performance must be attractive for commercial users of realtime systems.
- The design should support schedulability.

SCI/RT Design Studies

- Use simulation to study behavior of alternative designs.
- Plot latency vs. throughput
- Look for high performance with narrow distribution of latencies.
- Estimate costs vs. RT usability
- Work in progress.

SCI/RT early results

- 2-bit proposal minimizes latency for “unfair”
- Guarantees minimum bandwidth for “fair”
- High priority emergency messages
- Keeps forward-progress features of SCI!!
- Latency short for “unfair”, all the long-latency traffic is “fair”!
- Sufficient for commercial use, multimedia
- Minimal change to SCI, might become mainstream SCI in future (Serial Express uses it)

SCI/RT early results, plots

Transaction latency(ns) at load 800 + 6400
File RT2modeTrans500TL-Version 8.92
Showing unfair+fair, and fair, latency distributions
 Fault tolerant systems

- Hardware fault-retry
  - Encapsulate packets to add larger CRC
  - Add duplicate-suppression mechanism
- Spares
  - alternate signals in cables
  - alternate scrubber nodes
  - alternate clock masters

Fault tolerant systems

- Redundant rings
- Alternate routes
- Hot replacement
- Checkpoint-restart

Other Projects useful to SCI

- ANSI/IEEE Std 1212 Control and Status Register Architecture
  - David V. James, Apple Computer
- P1212.1 DMA Architecture
  - Sam Duncan, DEC
- IEEE Std 1301, 1301.1 Metric Mechanical
  - Hans Karlsson, Ericsson (deceased ’92)
- P1394 SerialBus (FireWire™)
  - Jerry Marazas, IBM
  - Mike Teener, Apple
  - 10 – 20 Mbytes/s, low cost I/O
- Serial Express (Gbit extendable SerialBus and same protocols as next-generation SCI!)
  - Intel, Apple, Sun, Compaq etc.?

Conclusions

- Chips: GaAs 500 MByte/s received 2Q93 (Vitesse/Dolphin) (later but faster, Fujitsu)
  CMOS 125 MByte/s delivered 1Q94
  CMOS 200 MByte/s 1Q95 (LSI Logic/Dolphin)
  GaAs 1000 MByte/s expected 1Q95 (Vitesse/Unisys)
  BiCMOS 1000 MByte/s demo’d 2Q94 (IBM)
  CMOS dual ring 1200MBytes/s 4Q95 (Cray)
- OK for 100’s of processors in 1995 H-P(Convex)
- OK for 1000’s in 1996 (guess!) Intel/DOE ASCI?
- SCI will adapt and scale to new needs in future
Communication link from A to B with flow control

Symmetric links between A and B with flow control

Duplicate networks for requests and responses

Shared network for requests and responses
Shared network redrawn

Shared network with link bandwidth shared by other nodes: daisy chain

Loop can cause request dependency deadlock

Loop can be broken by using different routing