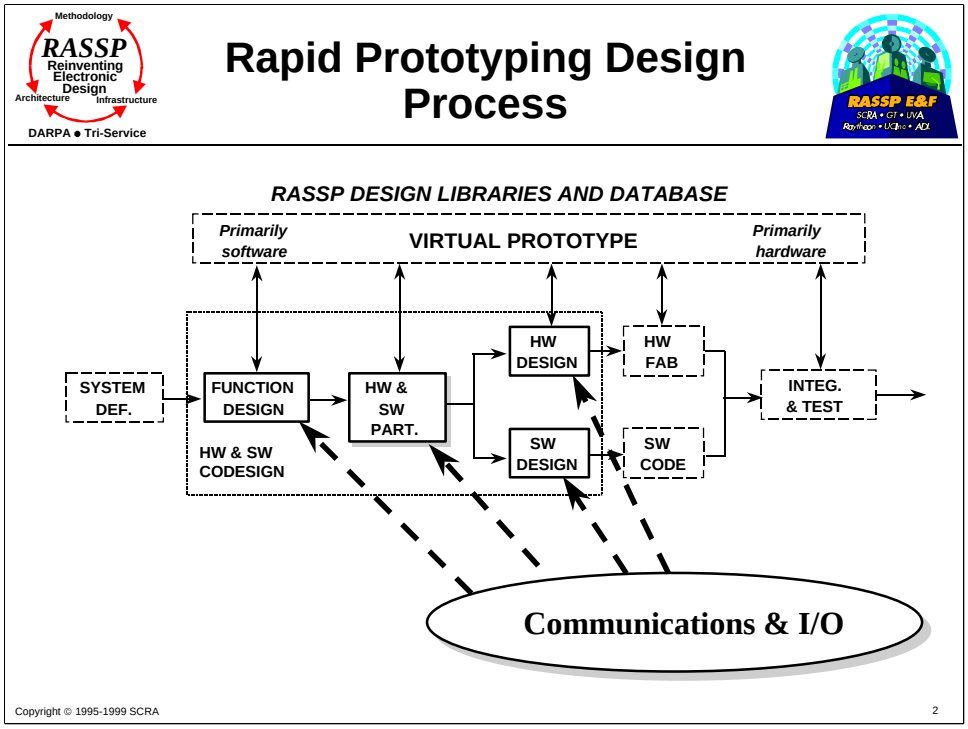
Methodology
RASSP
Reinventing Electronic Design
Architecture
Infrastructure
DARPA ● Tri-Service

RASSP E&F
SCRA • GT • UVA
Raytheon • UCDavis • ADX

Communication and I/O Protocols

RASSP Education & Facilitation Program
Module 25

Version 3.00



- Architecture design is based on the functional computational and communications requirements of the algorithm or algorithms selected to meet the specification. These trade-offs fall under the functional design and partitioning sections of the RASSP process.

RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA • Tri-Service

Module Goals

RASSP E&F
SCRA • GT • UVA
Raytheon • UCDavis • ADI

- Introduce communication and I/O protocols
- Describe trends and state-of-art
- Describe the impact of RASSP
- Understand the selected communication protocols used by RASSP systems

- Since RASSP requires scalable and upgradeable systems -- communication interconnects are just as important as computations

Module Outline

- Introduction
- State of the Art
- Impact of RASSP
- Advanced Applications
  - SCI
    - SCI overview
    - Communication protocols
    - Cache coherence
    - Physical layer
    - An Example: 2-D FFT
    - Conclusion
  - RACEway
  - FutureBus+
  - PI-Bus
  - Conclusion
- Summary

Module Outline

- **Introduction**
  - **State of the Art**
  - **Impact of RASSP**
  - **Advanced Applications**
  - **Summary**

RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA • Tri-Service

Motivation for Protocol Design

RASSP E&F
SCRA • GT • UVA
Raytheon • UCDavis • ADI

- **Current digital signal processors have been optimized for DSP applications**
- **The data rate is usually very high for the real-time signal processing**
- **The computational requirements are extremely demanded**

- DSP chips have been optimized for DSP applications by some features such as **single cycle multiply-accumulators(MAC)**, **modulo data addressing**, **operand preloading**, and so on.
- Because computational requirements are extremely demanded, the popular solution is to use **a multiprocessor system**. So, **a high speed inter-processor communication** is needed.

Requirements of High Bandwidth Data Networks

- **High scalability**
  - **The bandwidth scales well**
- **Efficient cache coherence protocols**
  - **The inter-processor communications are reduced**
- **High throughput rate**
  - **It implies high bandwidth**
- **No starvation**
  - **The bandwidth can be shared fairly**

Communication Elements

- **Shared Multi-drop Buses**
- **Switch Networks**
- **Point-to-point Links**

- Data can be transferred on the links in serial format or parallel format.

Communication Buses

- **Control Buses**
  - **FutureBus+, PI-bus, VME**
- **Data Buses**
  - **SCI, HIC, RACEway**
- **Test and Maintenance buses**
  - **Serial Bus, TM-Bus, MTM-Bus, JTAG**
- **Input/Output Buses**
  - **FC, SCSI**

- The buses can be grouped into the above categories by their functions.
- Control buses are used to exchange commands and some data among the processors and perform the inter-operations in a system.
- Data buses offer higher throughput for the transfer of data to augment the power of control buses.
- Test and maintenance buses provide a path to every hardware module to debug failures.
- Input/Output buses collect raw input data and output processed data.

RASSP Reinventing Electronic Design
Methodology · Architecture · Infrastructure
DARPA ● Tri-Service

Control Buses

RASSP E&F
SCRA • GT • UVA
Raytheon • Lockheed • ADX

NAME	STATUS	PERFORMANCE	INTENDED APPLICATION
FB+ IEEE 896.x ISO/IEC 10857:1994 "FutureBus+"	Released 1994	3200 MBytes/sec - 256 parallel 100 MBytes/sec - 32 parallel	Required by NGCR as backplane control bus Migration path for VMEbus
PI-bus JIAWG J89-N1A	Being revised	50 MBytes/sec - 32 parallel	Required by JIAWG as backplane control bus
VME64 IEEE P1014 Rev D	Recent revision	80 MBytes/sec - 64 parallel	Upgrade for VME bus
VME IEEE 1014-1987 "VersaModule Europa"	Released 1987 Widely used	40 MBytes/sec - 32 parallel	Commercial backplane control bus for high-performance systems

[Lockheed95]

10

- Control buses are typically used to allow multiple processors to inter-operate in a system through the exchange of commands and some data. In small systems with low throughput requirements, this may be the only bus required. Some of the bus choices are listed above.

RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA ● Tri-Service

Data Buses

RASSP E&F
SCRA • GT • UVA
Raytheon • LCSC • ADI

NAME	STATUS	PERFORMANCE	INTENDED APPLICATION
SCI IEEE 1596-1992 "Scalable Coherent Interface"	Released 1992	1000 MBytes/sec - 16 parallel 250 MBytes/sec - serial	Heterogeneous parallel processor
HIC IEEE P1355 "Heterogeneous interconnect"	Under development	250 MBytes/sec - serial	Low-cost parallel processor
RACEway VITA	Proposed by Mercury	160 MBytes/sec - 32 parallel	Real-time multicomputer interconnect

[Lockheed95]

11

- Data buses are used to augment control buses for systems with a higher throughput. To achieve the higher speed, the data bus is usually implemented point-to-point with unidirectional links. Some choices are listed above.

RASSP Methodology — Reinventing Electronic Design — Architecture — Infrastructure
DARPA ● Tri-Service

Test and Maintenance Buses

RASSP E&F
SCRA • GT • UVA
Raytheon • LMTec • ADX

NAME	STATUS	PERFORMANCE	INTENDED APPLICATION
Serial Bus IEEE P1394 "High Performance Serial Bus" "FireWire"	Under development	6 MBytes/sec - backplane 50 MBytes/sec - cable	Required by NGCR as T & M bus Used with FutureBus+
TM-bus JIAWG J89-N1B	Being revised	0.8 MBytes/sec - serial	Required by JIAWG as T & M bus Used with PI-bus
MTM Bus IEEE P1149.5 "Module Test and Maintenance Bus"	Under development	1.2 MBytes/sec - serial	Interconnect JTAG modules Based on TM-bus
JTAG IEEE 1149.1-1990	Release 1990 Widely used	3 MBytes/sec - serial	Interconnect JTAG modules (in hierarchical structures)

[Lockheed95]

12

- Test and maintenance buses are typically used to provide a minimally intrusive path to every hardware module in the system to isolate and debug failures. It is typically serial and low speed. It can also be implemented in redundant form for mission-critical fault tolerant systems.

RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA ● Tri-Service

Input/Output Buses

RASSP E&F
SCRA • GT • UVA
Raytheon • UCDavis • ADX

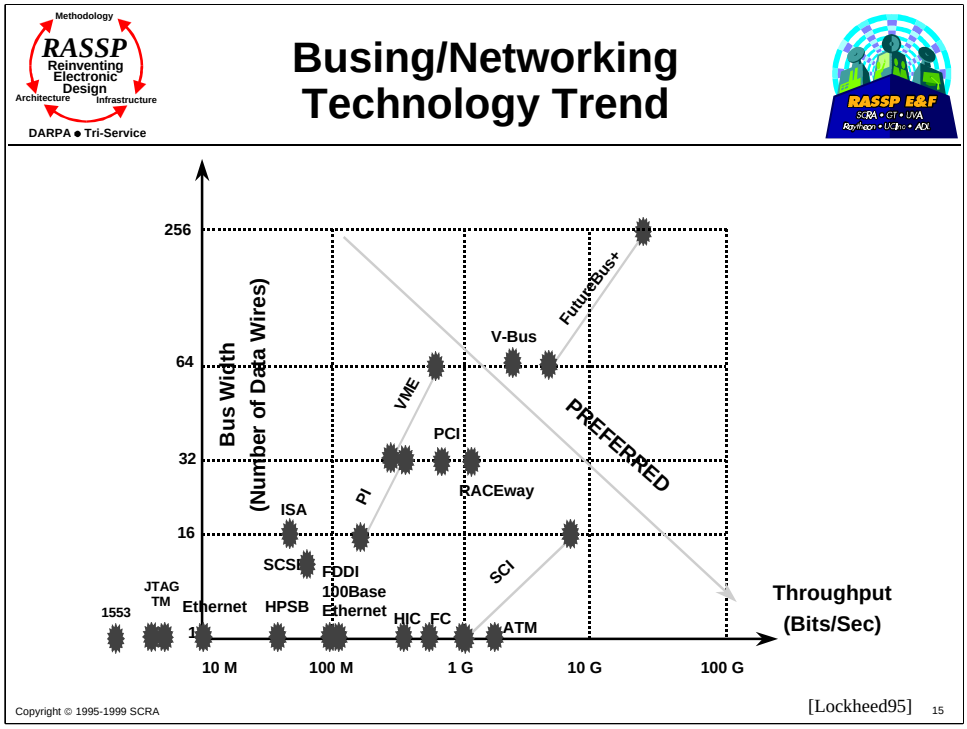
NAME	STATUS	PERFORMANCE	INTENDED APPLICATION
FC ANSI X3T9.3 "Fiber Channel"	Under development	100 MBytes/sec - serial	Proposed by NGCR for data channel (sensor input and video output)
SCSI "Small Computer System Interface"	Released Widely used	10 MBytes/sec - 8 parallel	Interconnect workstation peripherals
1553B MIL-STD-1553B	Released Military use	0.1 MBytes/sec - serial	Interconnect separate boxes in a military system

[Lockheed95]

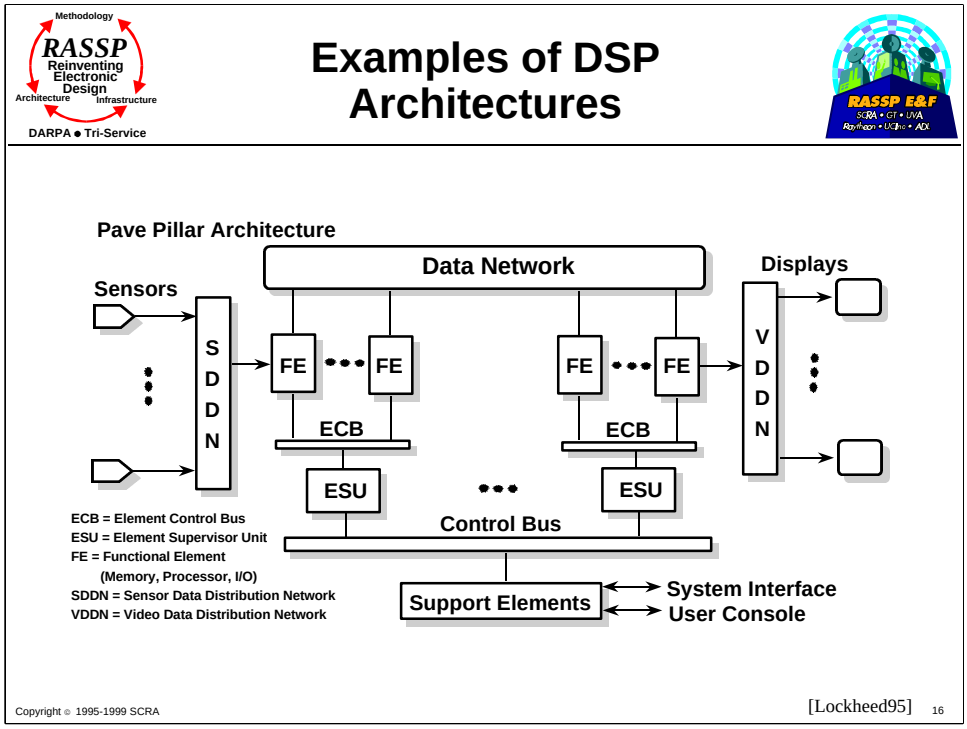
- I/O buses are used to collect raw data from the sensors and distribute it to the processors or to the displays of the system. They are optimized to transfer large blocks of data with minimal concerns for error checking and flow control. Some choices are listed above.

Module Outline

- Introduction
- **State of the Art**
- Impact of RASSP
- Advanced Applications
- Summary



- Communication between processors is accomplished by using buses of various types. These include direct point-to-point links, shared multi-drop buses, or networks made of links, buses, or switches.
- They can be implemented in serial or parallel form. Some choices of open systems bus standards available today are shown in the chart above with their respective throughputs.
- These buses can perform different functions such as control, data transfer, maintenance, I/O and area networking.



- The Pave Pillar Architecture was put together by the Air Force in the early 1980s. It was created with the intent of being modular, open, fault tolerant, and highly flexible.
- The control bus carries interprocessor messages and is usually implemented in redundant form for fault tolerance. It has an associated BIT bus for maintenance and debugging.
- The data network is usually implemented using a non-blocking crossbar network and transfers large blocks of data. This helps support multiprocessor dataflow.
- The architecture can be partitioned into core modules (for local control and communication), functional element modules (for high-performance processing, I/O, and storage), and miscellaneous modules (for power regulation and other support functions).

Application's I/O Profiles

- **The data type**
  - **The nature of the data to be transferred**
- **The overhead**
  - **A function of how much control data the interlink requires in the data stream**
- **The data distribution**
  - **The proximity of the data to its processing point**

- If transfers are short, then the most important factor of I/O performance may be latency. If the data tends to travel in long, sustained blocks, then throughput is probably more important.
- Overhead is how long it takes to process the transfer request. The more control data, the higher the latency and the lower the throughput. Also, the more negotiations that occur, the less extensible the architecture.
- Badly distributed data can degrade latency performance as well as throughput performance.

Performance Metrics

- **Function metrics**
  - **Real-time constraints**
- **Processor metrics**
  - **Computational throughput and latency**
- **Interconnect metrics**
  - **Communication speed**
- **Scalability metrics**
  - **Future upgradeability**

- Function metrics include real-time constraints such as accuracy, throughput, and latency. Real-time system is determinism. So, the emphasis is not on average throughput or average latency, but on worst-case throughput or worst-case latency. Being able to guarantee deterministic performance is more important than peak ratings.
- Processor metrics concern computational throughput determined by type of microprocessor, number of processors, memory size, and memory organization.
- Interconnect metrics measure the communication speed determined by clock speed, link width, link protocol, message length, link encoding, topology, flow control, routing, and access protocol.

Scalable Networks

- **A scalable network is not sensitive to the details of the underlying technology and is broadly applicable to a very wide range of application size, speed, cost, etc.**

- A scalable network is thus an important criteria for use in model year architectures that can easily be upgraded.

RASSP Reinventing Electronic Design
Methodology · Architecture · Infrastructure
DARPA ● Tri-Service

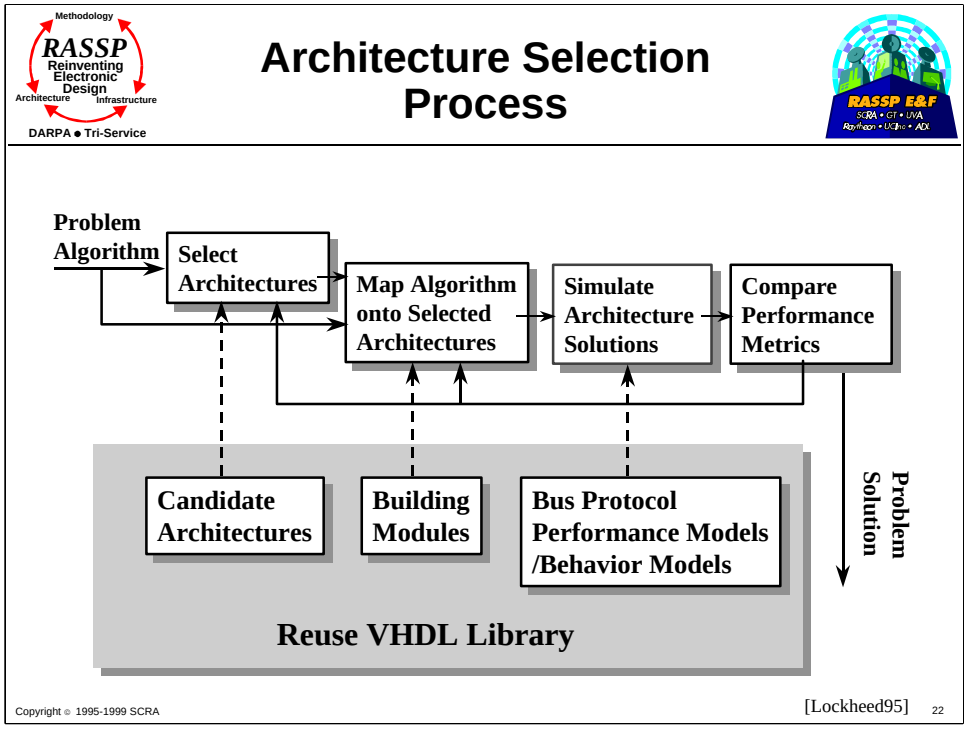
Scalability

- **Programmability**
- **Reconfigurability**
- **Upgrade cost**
- **Scalability limits**

- Programmability includes software and hardware.
- Reconfigurability includes switches and jumpers.

Module Outline

- Introduction
- State of the Art
- **Impact of RASSP**
- Advanced Applications
- Summary



- The first step in selecting an architecture is to assemble the signal processing requirements provided by the customer via the system specification, the statement of work, and other documents.
- The next step is to use the weighted metrics together with a knowledge of capabilities represented in the Reuse Library to make initial assessments that rule out all but one or a few of the candidate architectures.

RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA • Tri-Service

Rules of Scalability

RASSP E&F
SCRA • GT • UVA
Raytheon • UCSC • ADI

- Use building blocks
- Separate configuration control from application source code
- Provide for deterministic I/O
- Provide self-routing data packets
- Use transparent buffering of slave transfers to memory from external I/O
- Provide I/O that fits a broadcast capability and shared-memory regions
- No centralized arbitration

- Defining consistent system building blocks means knowing the functional boundaries that consistently exist within systems.
- By removing some of the process segmentation burden from the source code, features listed in the figure can reduce development complexity.
- System designers must decide what features to include within the I/O services and then optimize them for system-wide implementation.
- By implementing configuration control services and deterministic I/O, designers can direct which path particular data will take among several that are available.

RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA • Tri-Service

Why Broadcast Capability?

RASSP E&F
SCRA • GT • UVA
Raytheon • UCDavis • ADI

- **The proportion of messages that need to be received by *a large number of nodes* at the same time**
- **There is lots of *process synchronization* and a resulting *exchange of control flags***

- If real-time data will be received by many computer elements simultaneously, a broadcast capability is virtually mandatory.

Module Outline

- Introduction
- State of the Art
- Impact of RASSP
- **Advanced Applications**
- Summary

Section Outline

- **Advanced Applications**
  - **SCI**
    - **SCI overview**
      - **Communication protocols**
      - **Cache coherence**
      - **Physical layer**
      - **An Example: 2-D FFT**
      - **Conclusion**
  - **RACEway**
  - **FutureBus+**
  - **PI-Bus**
  - **Conclusion**

The Objective of SCI

- **Define an interconnect system that**
  - ***Scales*** **well as the number of attached processors increases**
  - **Provides a *coherent* memory system**
  - **Defines a simple *interface* between nodes**

- SCI provides very high-performance and bus-like functionality to a large number of processors by transmitting packets on a collection of point-to-point unidirectional links.
- The network bandwidth of SCI is scalable because SCI nodes are able to transmit packets concurrently and the bandwidth of point-to-point links depends on the transmission medium.

Levels in SCI Protocols

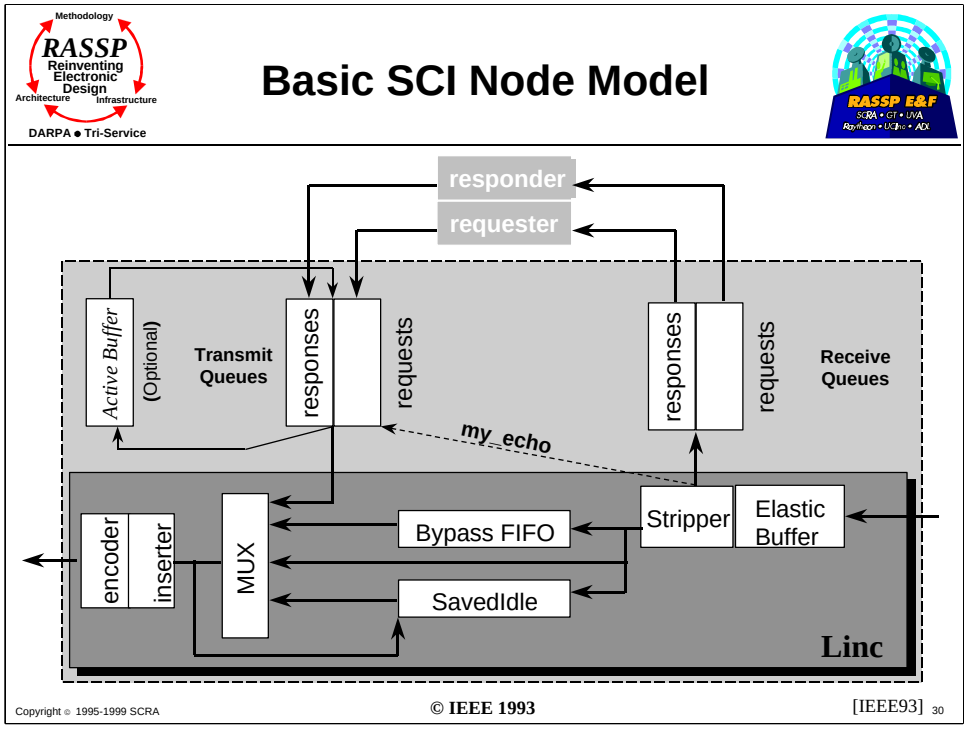
- **Logical level describes**
  - **Initialization and address space**
  - **Communication protocols**
  - **Error recovery facilities**
- **Coherence level provides**
  - **Optional cache coherence mechanisms**
- **Physical level specifies**
  - **Electrical, mechanical, and thermal characteristics of connectors and cards**

- SCI addresses issues at a number of levels. We'll focus on the logical level of communications in this module.

Basic SCI Nodes

- **Use unidirectional input/output links**
  - **The speed-of-light barrier to system growth is removed**
- **Handshake on a *packet basis* rather than a *bus cycle basis***
  - **No cycle-by-cycle handshakes**
- **Are always chained into ring networks**
  - **Sender can get back the status information by flowing around a ring**

- For the unidirectional links, there is no way for a receiver to send the status information back to the sender directly. To eliminate this problem, SCI nodes are always chained into ring networks.



- One of the nodes on each ring is assigned certain housekeeping tasks such as initializing the ring, maintaining certain timers, and discarding damaged packets. This node is called the scrubber.

Components of SCI Nodes

- **Elastic Buffer:** ***synchronizes input data***
- **Stripper:** ***strips incoming packets***
- **Bypass FIFO:** ***delays pass-through packets***
- **SavedIdle Buffer:** ***saves the needed information from incoming idle symbols***
- **Inserter:** ***inserts initialization packets***
- **Encoder:** ***re-inserts the flag symbols***
- **Dual queues:** ***keeps responses independent of requests***

- The elastic buffer is responsible for inserting and deleting elasticity symbols. Because the SCI standard allows each node to generate its own clock and because the clocks on separate nodes will drift in phase over time, elasticity symbols will sometimes need to be deleted or inserted.
- The stripper is also responsible for creating echo packets and replacing the selected non-idle symbol by an idle symbol and hence removing it from transmission on a ring.
- Without dual queues, excessive requests could prevent sending responses, resulting in deadlocks.

RASSP Reinventing Electronic Design Methodology Architecture Infrastructure
DARPA • Tri-Service

Basic Packet Types

Group	Description
Request-send	*request subaction* content
Request-echo	*request subaction* local acknowledge
Response-send	*response subaction* content
Response-echo	*response subaction* local acknowledge

[IEEE93] 32

- Transactions consist of two subactions: request subactions and response subactions.
- Each subaction consists of an echo packet transmission and a send packet transmission.

Send Packet Formats

targetId
command
sourceId
control
AddressOffset{00..15}
AddressOffset{16..31}
AddressOffset{32..47}
ext(0 or 16 bytes)
data (0, 16, 64, or 256 bytes)
CRC

Request-send Packet Format

targetId
command
sourceId
control
status
forwId
backId
ext(0 or 16 bytes)
data (0, 16, 64, or 256 bytes)
CRC

Response-send Packet Format

coherent transaction status

[IEEE93]

33

- A packet consists of an unbroken sequence of 16-bit symbols and has a 16-byte header that contains address, command, transaction identifier, and status information.
- Send packets are multiples of 16 bytes to simplify storage management at very high speeds.

Echo Packets and Special Packets

targetId
command
sourceId
control

Echo Packet

targetId
distanceId
stableId
uniqueId0
uniqueId1
uniqueId2
uniqueId3
CRC

Initialization Packet

CRC or idle
FFFF
0000
0000
0000
0000
0000
0000
0000

Sync Packet

any
FFFB
FFFB
FFFB
FFFB
FFFB
FFFB
0000
0000

Abort Packet

[IEEE93]

- The echo packet type is an 8-byte subset of the header.
- Initialization packets are only used during the system initialization process.
- Sync packets are generated by each node during the initialization sequence and from time to time during the normal operation so the downstream neighbor can deskew its receiver's data paths.
- The abort packet are generated by any node that wishes to initiate a reset to cleanly abort an arbitrary symbol stream transmitted by the node.

Idle Symbols

- **Format**

ipr(2)	ac	cc	hg	lg	old	lt	ipr*(2)	ac*	cc*	hg*	lg*	old*	lt*

- **The first idle symbol between packets is called non-*consumable idle symbol***
- **Idle symbols are generated when**
  - **There is no packet to transmit**
  - **Send or echo packets are stripped from the ring**
  - **A send packet is produced**

- Idle symbols are transmitted between packets to maintain synchronization and transfer flow-control information.
- The SCI protocol includes a flow control mechanism that uses go bits in the idle symbols to enforce an approximate round-robin ordering under heavy loads.

Section Outline

- **Advanced Applications**
  - **SCI**
    - **SCI overview**
    - **Communication protocols**
    - **Cache coherence**
    - **Physical layer**
    - **An Example: 2-D FFT**
    - **Conclusion**
  - **RACEway**
  - **FutureBus+**
  - **PI-Bus**
  - **Conclusion**

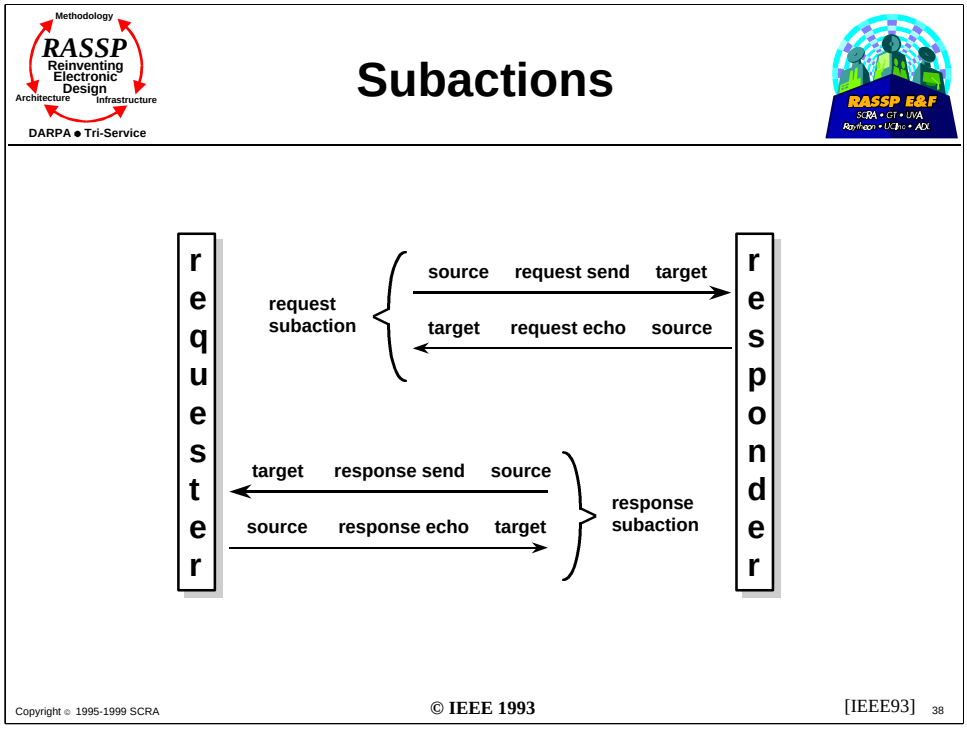
RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA • Tri-Service

Logical Protocols (1)

RASSP E&F
SCRA • GT • UVA
Raytheon • UCDavis • ADI

- **Transactions are initiated by a *requester* and completed by a *responder***
- ***Send packets* are transmitted from *source* nodes to *destination* nodes**
- ***Echo packets* are generated by destination nodes and notify source nodes if the send packets were accepted**

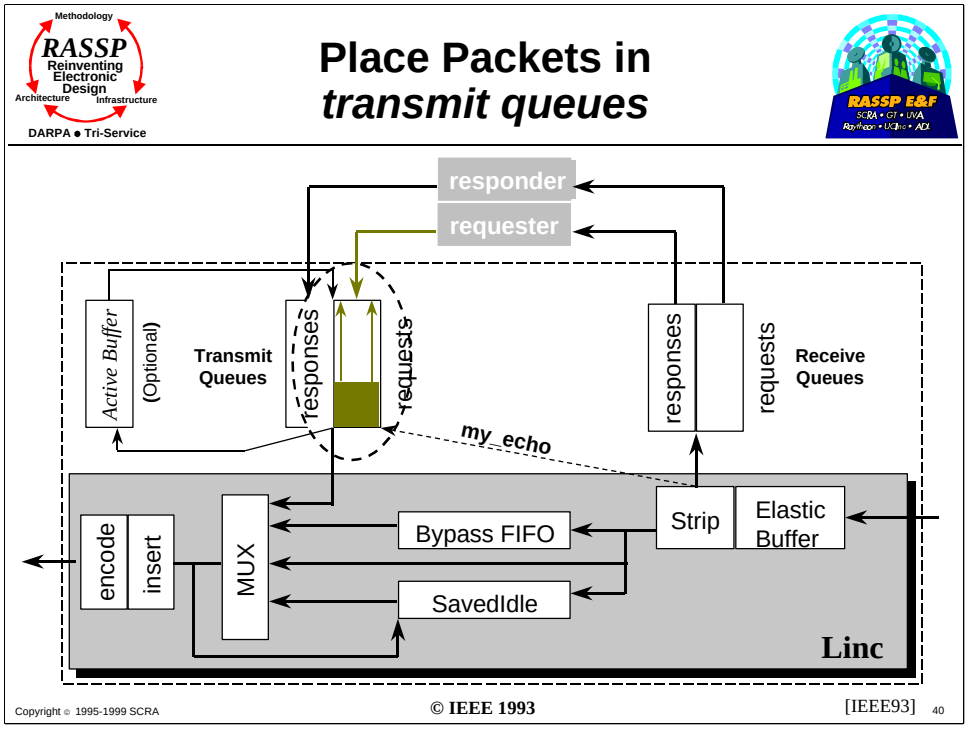
- The target node strips send packets, and it returns echo packets around the remainder of the ring.
- Echo packets notify source nodes whether send packets were accepted by destination nodes. If packets were not accepted, the source node has to resend it.



- A subaction consists of two packet transmissions, one sent on the output link and the other received on the input link.
- A subaction is initiated by a source, which generates a send packet. The subaction is completed by the destination, which returns an echo packet.

Logical Protocols (2)

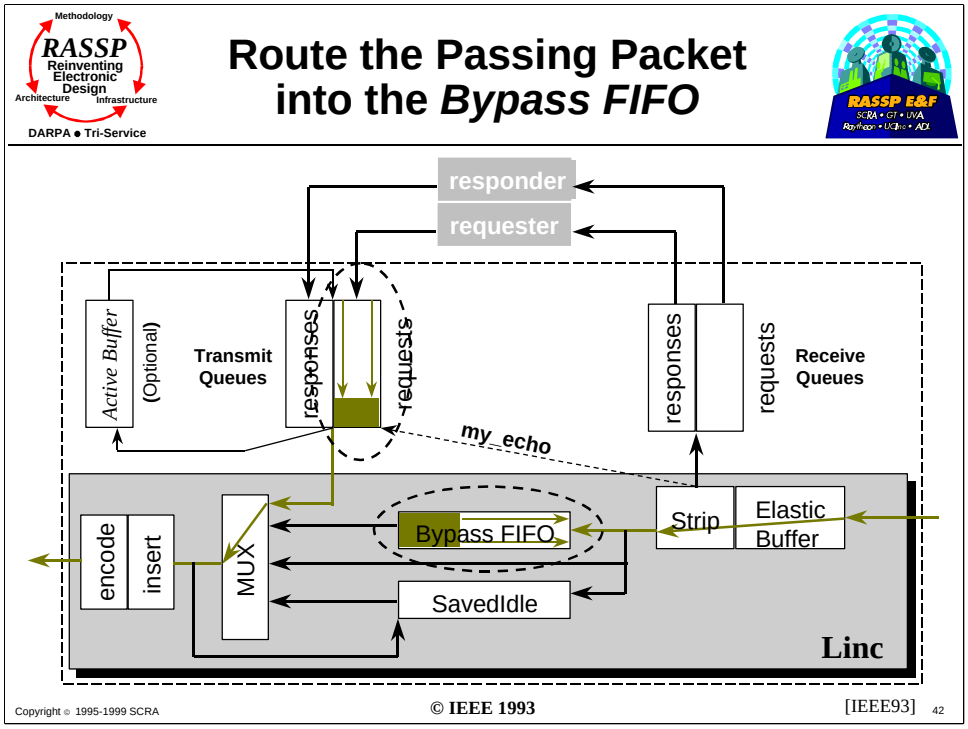
- **Sending a packet**
  - **If the *bypass FIFO* is empty and the node is not transmitting a packet**
    - **Transmit the packet and save its copy either at the head of *transmit queues* or at an optional *active buffer***
  - **Otherwise, place the packet in *transmit queues***



- When a source desires to send a packet over the ring, it places the packet in its transmit queue.

Logical Protocols (3)

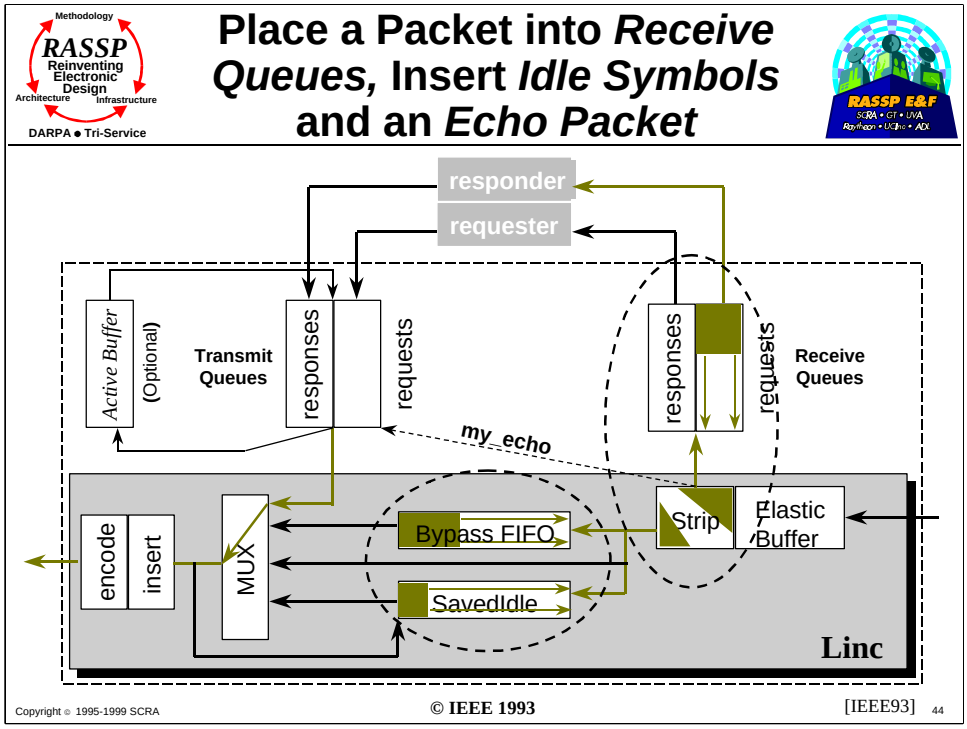
- **Upon arrival the downstream node**
  - **Parse the send packet and either pass it along the ring or strip it**
  - **If the *bypass FIFO* is not empty and the node is sending a packet**
    - **Route the passing packet into the *bypass FIFO***
  - **If a passing packet and a transmitting packet are ready to send at the same time, *transmit queues* are given priority**



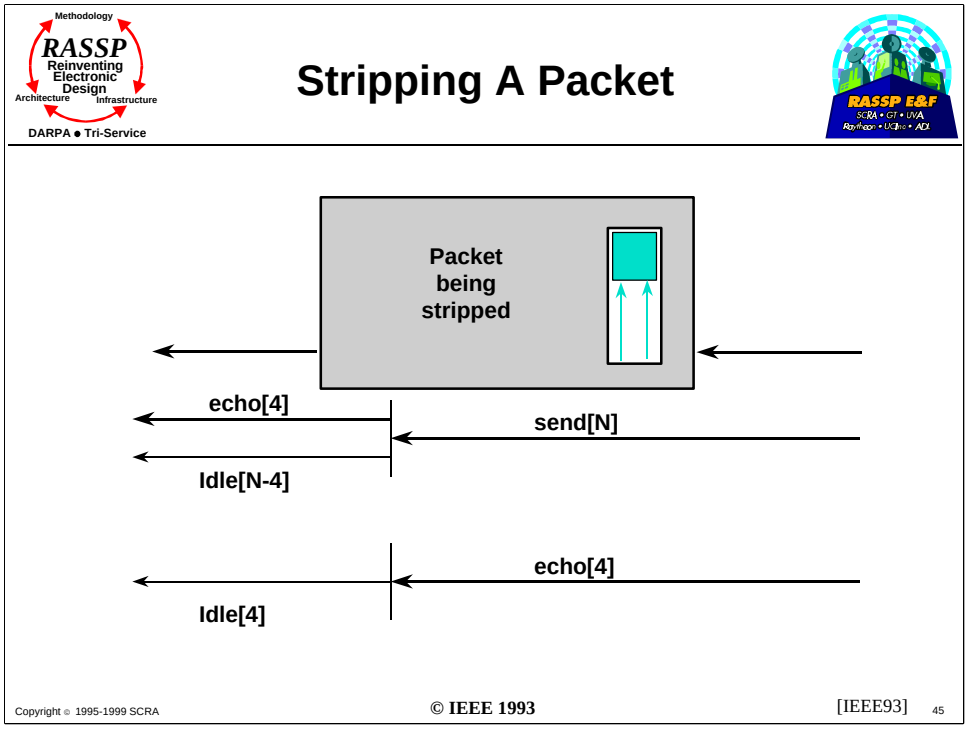
- Hold portions of packets that arrive during the transmission.

Logical Protocols (4)

- **Stripping a send packet**
  - **If *receive queues* are not full**
    - **Place the packet into *receive queues***
  - **Otherwise, discard the send packet**
  - **Insert *idle symbols* and replace the last four symbols of the received send packet with an echo packet**



- Consumable idle symbols are not inserted into the bypass FIFO, but its idle.lg, idle.hg, and idle.old bits are merged into the savedIdle buffer.



- If an echo packet is stripped, stripper creates four idle symbols.
- When stripper is stripping a send packet, it inserts idle symbols and replaces the last four symbols of the received send packet with an echo packet.

Logical Protocols (5)

- **Receiving an echo packet, either**
  - **Pass it along the ring**

    or
  - **Discard or retransmit the copy of the send packet in *transmit queues* or the *active buffer***

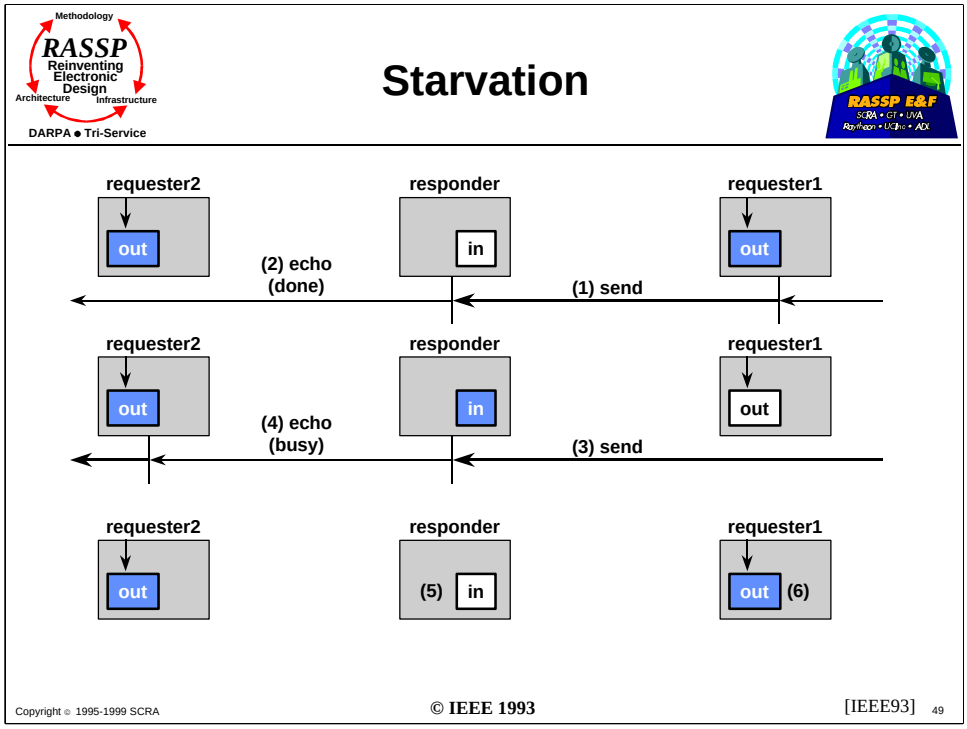
Bypass FIFO

- **Hold portions of packets that arrive during the transmission**
- **When *transmit queues* is done transmitting a packet, the *bypass FIFO* output resumes**
- ***Consumable idle symbols* are not inserted into the *bypass FIFO*, but its *idle.lg*, *idle.hg*, and *idle.old* bits are merged into the *savedIdle buffer***

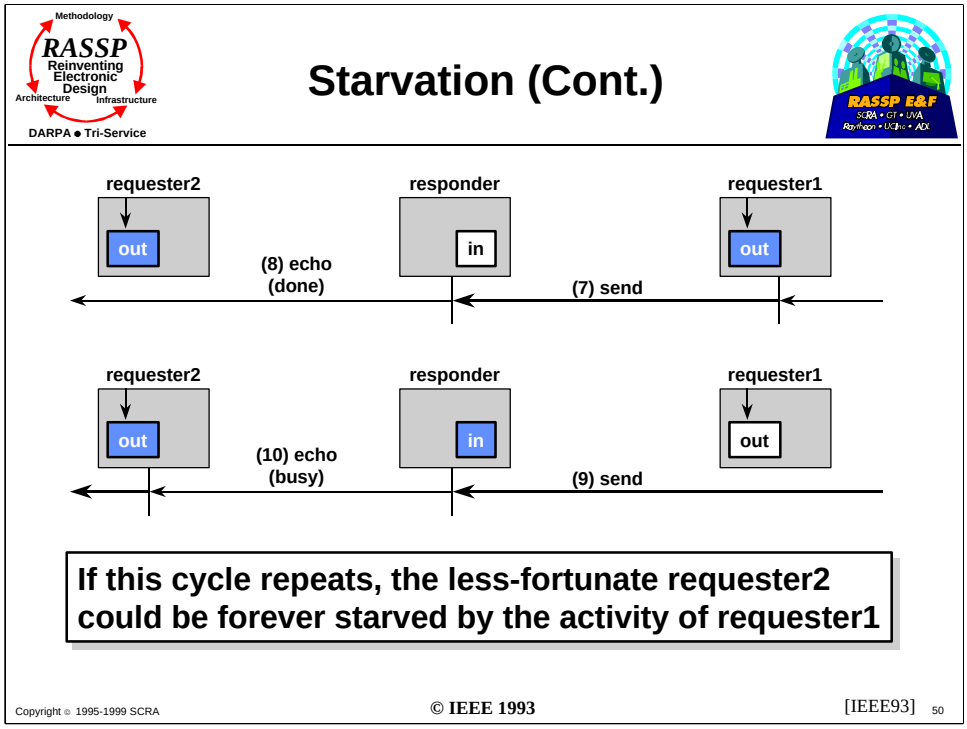
Starvation and Deadlock

- **Forward progress is guaranteed**
  - **Nothing prevents packets from being forwarded before they have arrived completely**
- **Queue reservation protocol ensures the fair use of part of the bandwidth**
  - **Avoids starvation**
- **Maintaining *dual queues* keeps responses independent of requests**
  - **Avoids deadlock**

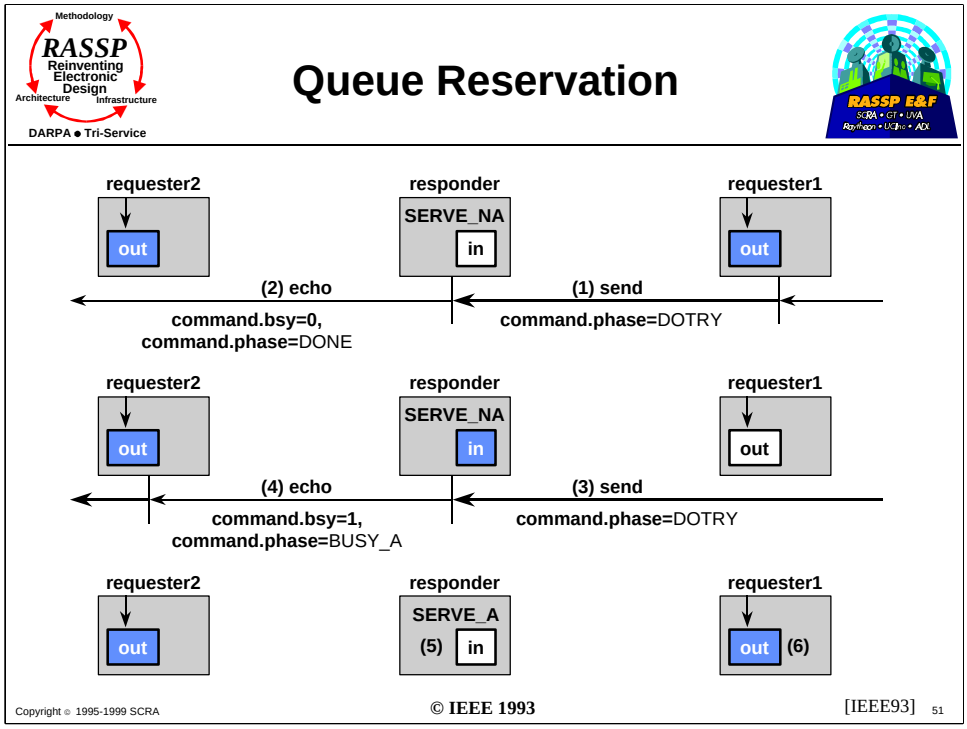
- Starvation, the failure to ever get a needed resource that is shared with others, may result in one or more processors not getting any work done until the rest have finished, effectively serializing what should have been a parallel computation.
- Deadlocks can occur when two processors request the same two resources in the opposite order.



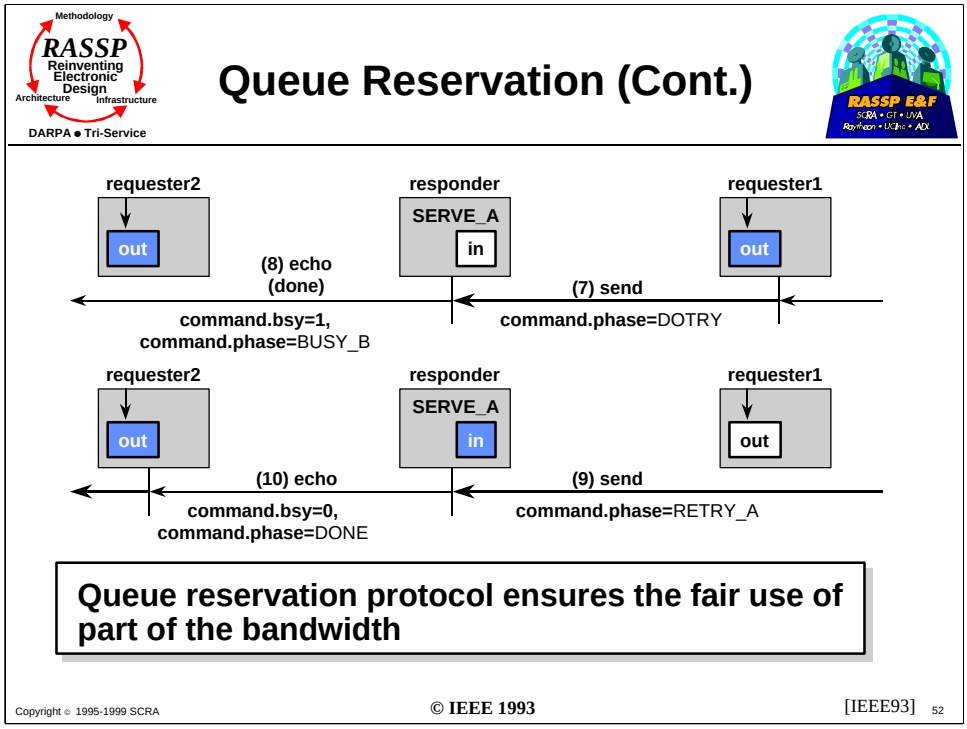
- Requester1 initially sends (1) a request-send packet to the responder; because the responder's queue is empty, the packet is accepted.
- The returned request-echo packet indicates (2) the request send was accepted without error.
- Before the responder has processed its input-request queue, another request-send packet is sent (3) from requester2; because the responder's queue is full, the packet is rejected.
- The returned request-echo packet indicates (4) the subaction was busied and should be quickly retried.
- Soon thereafter, the responder's input-request queue is emptied (5) and another request-send packet is generated (6) within requester1.



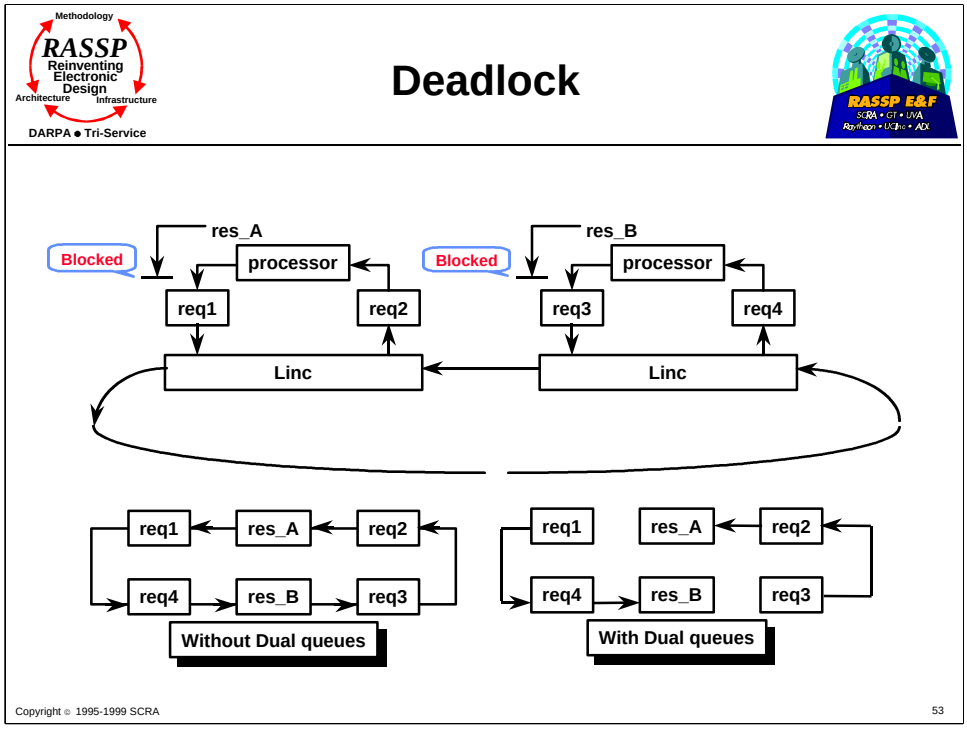
- The new request subaction is sent (7) from requester1; because the responder's queue is empty, the packet is accepted.
- The returned request-echo packet indicates (8) the request send was accepted without error.
- Then requester2 resends (9) its previously busied request-send packet, but because the responder's queue is once again full, the packet is rejected.
- The returned request-echo packet indicates (10) the subaction was busied and should be quickly retried.



- Requester1 initially sends (1) a request-send packet to the responder; because the responder's queue is empty, the packet is accepted.
- The returned request-echo packet indicates (2) the request send was accepted (command.bsy is 0) without error (command.phase is DONE).
- Before the responder has processed its input-request queue, another request-send packet is sent (3) from requester2; because the responder's queue is full, the packet is rejected.
- The returned request-echo packet indicates (4) the subaction was busied (command.bsy is 1) and should be retried with a RETRY_A command phase (command.phase is BUSY_A) . The responder's state is also changed from SERVE_NA to SERVE_A.
- Soon thereafter, the responder's input-request queue is emptied (5) and another request-send packet is generated (6) within requester1.

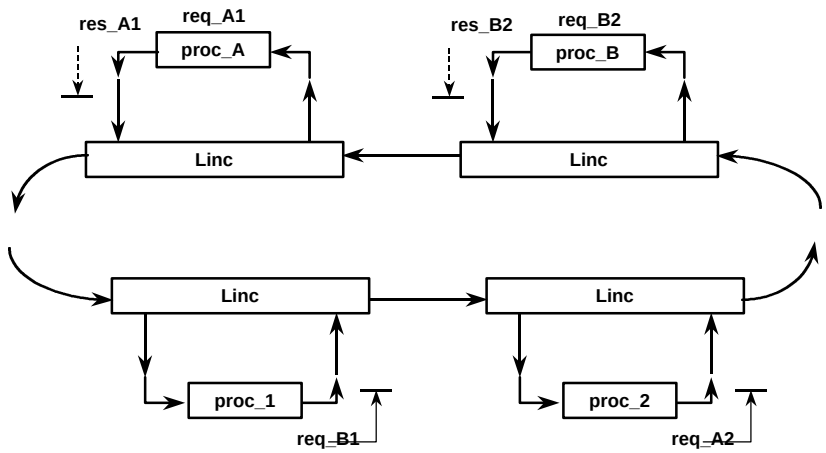


- When the new request subaction is sent (7) from requester1; the returned request-echo packet indicates (8) the request send was not accepted (command.bsy is 1) and should be retried with RETRY_B phase (command.phase is BUSY_B).
- Then requester2 resends (9) its previously busied request-send packet, using a RETRY_A command phase. The responder's state (SERVE_A) allows it to accept this re-send packets. When all previously busied RETRY_A requests have been accepted, the responder's state is changed to SERVE_NB; new or RETRY_B requests will be accepted next.

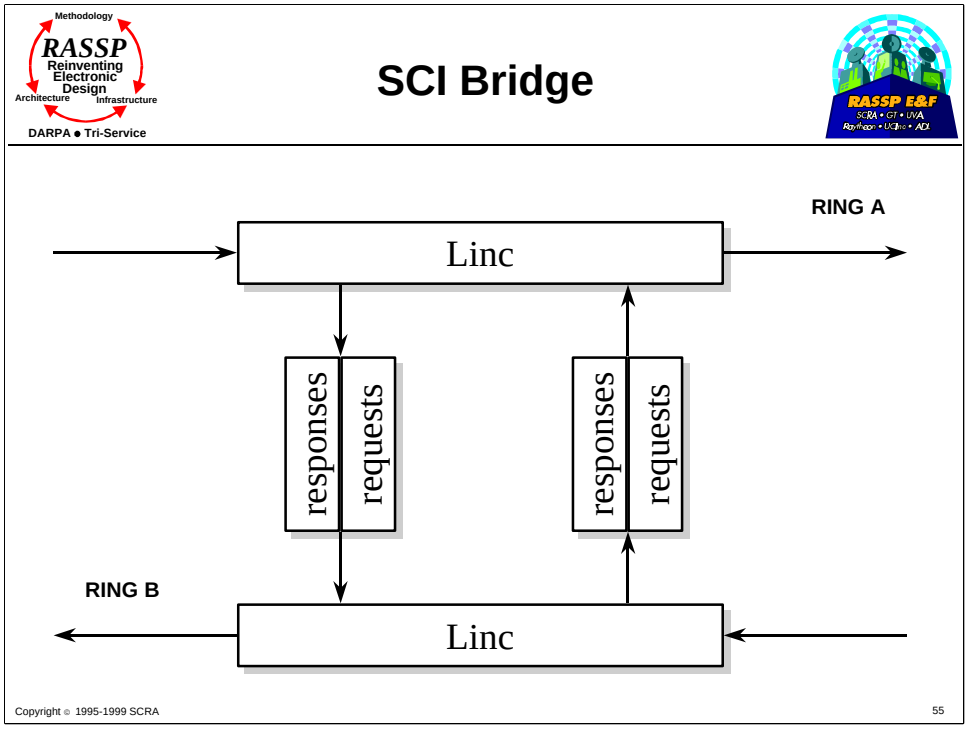


- With combined queues, all entries could be filled with requests, making responses impossible.
- Dual queues are used to keep responses independent of requests to avoid deadlocks. (i.e. The dependency graph doesn't form a cycle.)

Deadlock (Cont.)

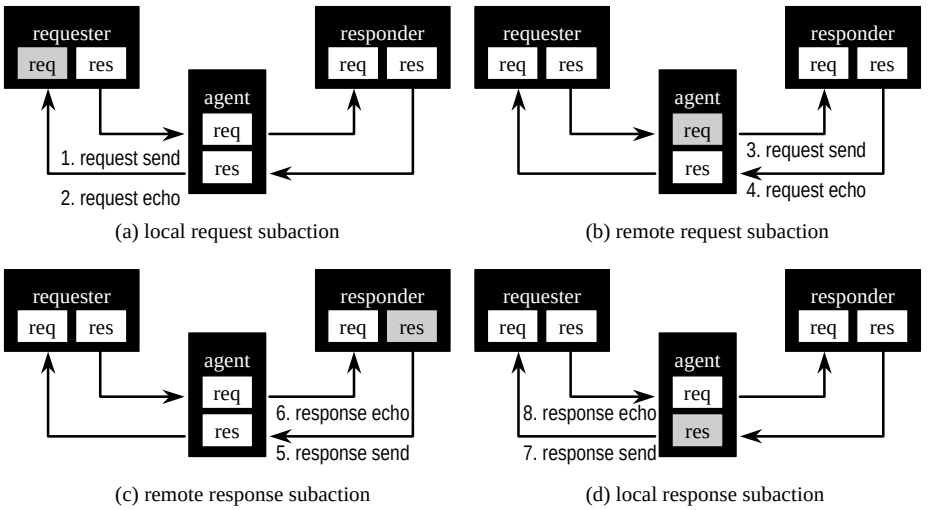


- Without queues, deadlocks can occur when two processors request the same two resources in the opposite order.



- SCI bridge is constructed from two SCI node interfaces connected back to back.
- The queues in the bridge are used to store a complete packet before data can be forwarded.
- The advantage of the SCI bridge is that the basic SCI node can be re-used without changing its structure.

Remote Transaction Components



[IEEE93]

56

- In the multiple ring network, packets are accepted by the switch queues in the intermediate agents while packets are switching to the other rings.
- Subactions do not care whether they are local or remote; only agents need know that the subaction is not local, and queues in agents take responsibility for further transmission.

Deadlocks in Multiple Rings

- **If routing is not carefully thought out, circular wait and deadlock might occur**
- **The deadlock problem is classical for store-and-forward or virtual cut-through networks**

- When packets switch rings, the entire send packets will be stored in switch queues.
- Routing in SCI is less restricted than in simple store-and-forward or virtual cut-through networks because packets are only stored at nodes where packets change rings.

Routing Mechanisms

- **Store-and-forward routing**
- **Virtual cut-through routing**
- **Wormhole routing**

Ù **Virtual cut-through routing can be used in SCI networks**

- Store-and-forward is also called packet switching. When a packet reaches an intermediate node, the entire packet is stored in a packet buffer. The packet is then forwarded to a selected neighbor when the next channel is available and the neighbor has an available packet buffer.
- Virtual cut-through: The packet header is examined upon arrival at an intermediate node. The packet is stored at the intermediate node only if the next channel is busy. That is, blocked packets be buffered.
- A bad packet might have a bad address, and thus get routed to the wrong place, but that should not cause any harm as long as it is checked before it is used.
- A packet starts out the other side of a bridge or switch before it has been entirely received or checked, so we argue that virtual cut-through routing can be used in SCI networks.
- Furthermore, SCI reserves sufficient space to store entire packets, so deadlock avoidance is easier to be solved by viewing SCI as a virtual cut-through network.
- Wormhole routing is not suitable because forward progress is guaranteed in SCI rings but wormhole routing will result in blocking rings in SCI. It is not a kind of store-and-forward routing policy.

Routing Mechanisms(Cont.)

- **Virtual cut-through routing can be used in SCI networks for the following reasons**
  - **A bad packet might have a bad address, and thus get routed to the wrong place, but that should not cause any harm as long as it is checked before it is used**
  - **A packet starts out the other side of a switch before it has been entirely received or checked**
  - **SCI reserves sufficient space to store entire packets**

- Check [IEEE93] for the first reason.

Section Outline

- **Advanced Applications**
  - **SCI**
    - **SCI overview**
    - **Communication protocols**
    - **Cache coherence**
    - **Physical layer**
    - **An Example: 2-D FFT**
    - **Conclusion**
  - **RACEway**
  - **FutureBus+**
  - **PI-Bus**
  - **Conclusion**

Cache Coherence

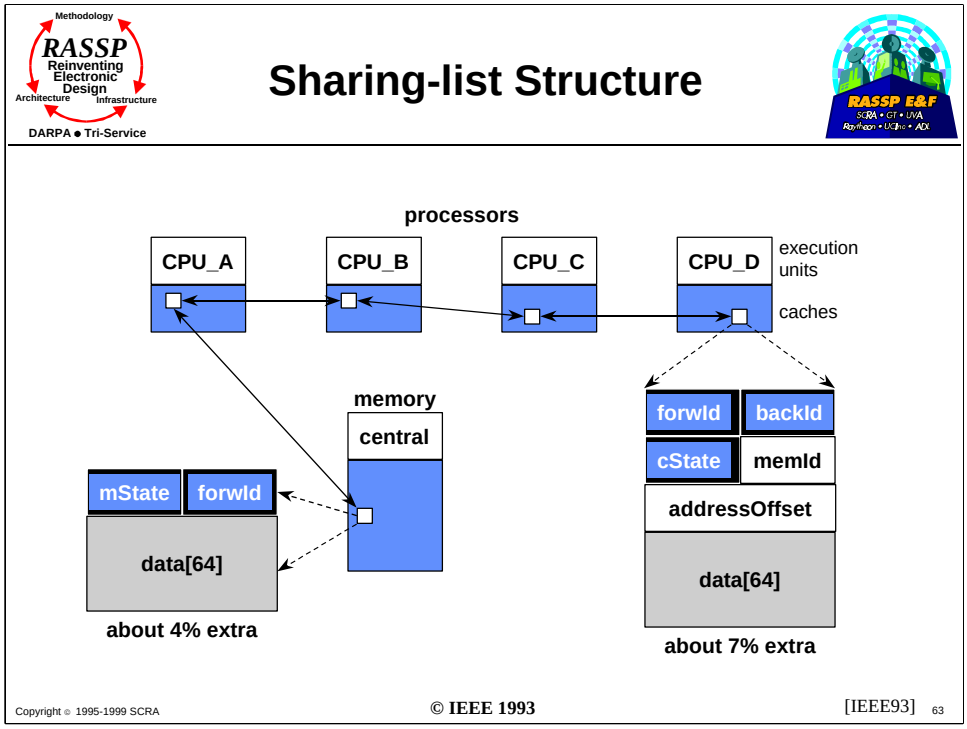
- **SCI does not use broadcast and eavesdropping mechanisms**
  - **Distributed directory-based cache coherence protocol**
- **SCI uses a *sharing list structure***
  - **Scale well**
- **SCI supports both *weak* and *strong sequential consistency***

- Broadcast transactions are inherent in a bus-based system, but are not feasible for large high-speed distributed systems.

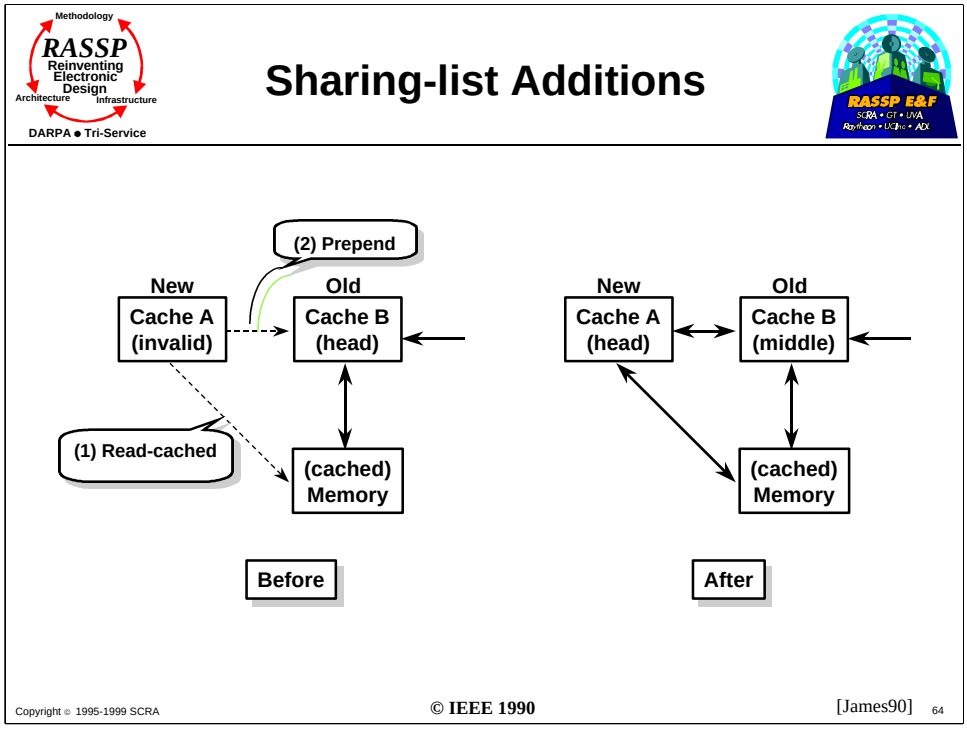
Standard Optimizations

- **Fresh copy optimization**
- **DMA transfer optimization**
  - ***DMA controller can often fetch its data without joining the sharing list***
- **Pairwise sharing optimization**
  - **Data are directly transferred from one cache to the other**
  - **The directory pointers need not be changed**

- The fresh memory state indicates that all shared copies are read-only; the data can be returned from memory when a new processor is attaching to the head of the previous sharing list.
- DMA data can be read directly from the sharing-list head without changing the directory state.
- When data is shared by a producer and a consumer, it is directly transferred from one cache to the other. The directory pointers need not be changed, and memory is not involved in the cache-to-cache transfer.

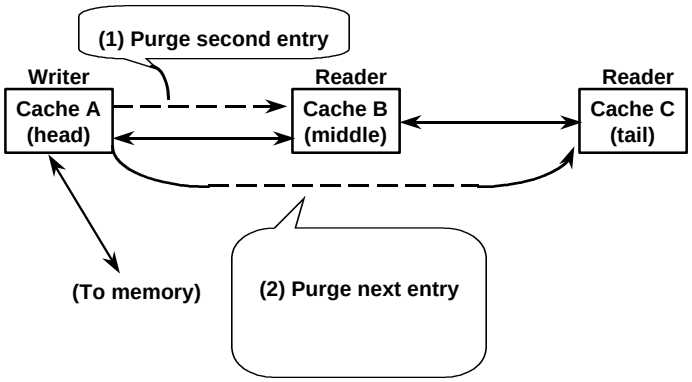


- Every memory line that supports coherent caching has an associated directory entry that includes a pointer to the processor at the head of the list.
- Each processor cache-line tag includes pointers to the next and previous nodes in the sharing list for that cache line.



- A new requester (Cache A) directs its **read-cached transaction** to memory, but receives a pointer to Cache B instead of the requested data.
- A second cache-to-cache transaction, called **prepend**, is directed from Cache A to Cache B.
- On receiving the request, Cache B sets its backward pointer to point to Cache A and returns the requested data.

Sharing-list Removals



[James90]

65

- The initial transaction to the second sharing-list entry purges that entry from the sharing list and returns its forward pointer (1).
- The forward pointer (2) is used to purge the next (previously the third) sharing-list entry.
- The process continues until the tail entry is reached.

RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA ● Tri-Service

Coherence Tags

- **Memory tags**
  - **A lock bit**
  - **A 2-bit memory state field**
  - **A 16-bit *forwId* field**
- **Cache tags**
  - **7-bit cache state**
  - **Two 16-bit pointer fields, *forwId* and *backId***
  - **A 16-bit *memId***
  - **48 bits of *addressOffset***
- **64-Byte cache line**
  - **Small tag overhead**
  - **Reasonable efficiency**
  - **Uniformity**

- The sizes of memory-directory and processor-entry tags are significantly less than the size of a line of data.
- The 64-byte SCI transaction is relatively efficient; approximately two thirds of the consumed bandwidth is used for data.
- The 64-byte size is shared by other bus standards (FutureBus+).

RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA • Tri-Service

The Options of the Cache-coherence Protocols

RASSP E&F
SCRA • GT • UVA
Raytheon • UCDavis • ADX

- **Minimal set**
  - **Maintain cache coherence in a trivial but correct way**
- **Typical set**
  - **Has provisions for read sharing, robust recovery from errors, efficient read-only data accesses, efficient DMA transfers, and local data caching**
  - **Full set**
  - **Implement all of the defined options. In addition to the provisions of the typical set, the full set supports clean cache-line states, cleansing and washing of dirty cache-line states, pairwise sharing, and QOLB**

- The minimal set can be used to maintain cache coherence in a trivial but correct way that has no provision for read sharing. This model could be useful for small multiprocessors where applications infrequently share data and manage coherence of shared instruction pages via software.
- This option set is likely to be implemented even in the first SCI systems.
- The typical set has provisions for read sharing, robust recovery from errors, efficient read-only data accesses, efficient DMA transfers, and local data caching.
- The full set implements all of the defined options. In addition to the provisions of the typical set, the full set supports clean cache-line states, cleansing and washing of dirty cache-line states, pairwise sharing, and QOLB.

Constraints of Optional Subsets

- **The options can be implemented without significantly increasing the size of tags in the *memory directory* or *caches***
- **The options work with the *basic SCI transaction-set* definitions**
- **The options should not affect the correctness of the *basic SCI cache-coherence* specification**

- The options are available to improve the performance of frequently used forms of cache sharing between entries in relatively short sharing lists.

Section Outline

- **Advanced Applications**
  - **SCI**
    - **SCI overview**
    - **Communication protocols**
    - **Cache coherence**
    - **Physical layer**
    - **An Example: 2-D FFT**
    - **Conclusion**
  - **RACEway**
  - **FutureBus+**
  - **PI-Bus**
  - **Conclusion**

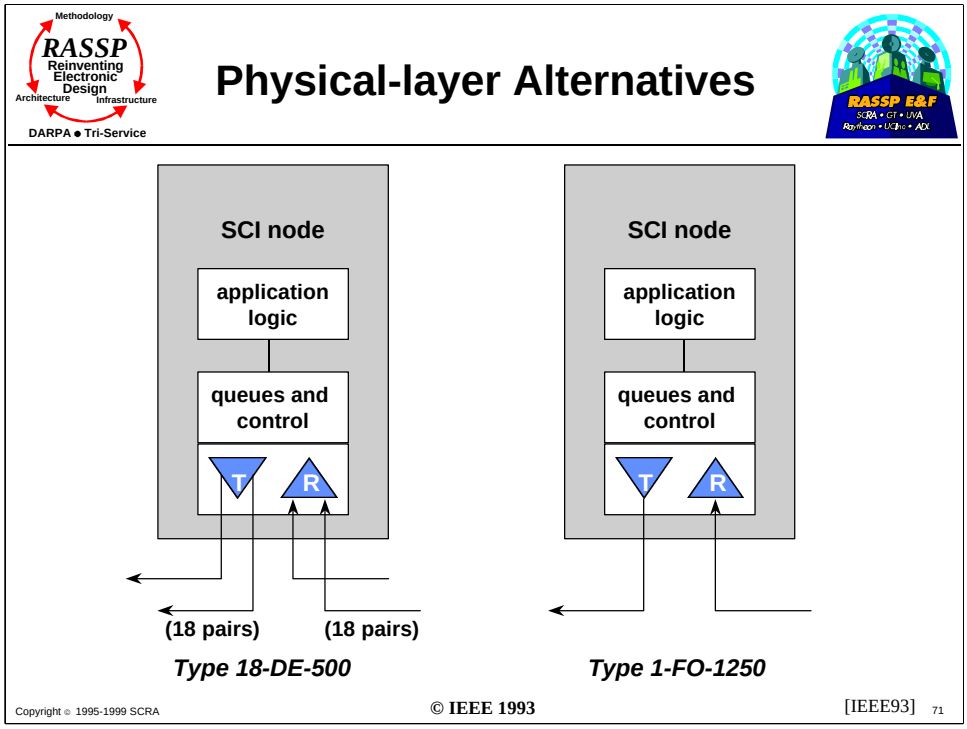
RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA ● Tri-Service

Physical Layer of SCI

RASSP E&F
SCRA • GT • UVA
Raytheon • UCDnc • ADX

- **SCI links transmit symbols that contain**
  - **A 16-bit data**
  - **A 1-bit packet-delimiter (flag signal)**
  - **A clock**
- **Notation of link types**
  - **Type <numbers of signals>-<kind of signals>-<bit rate per signal in Mbit/s>**
- **All signals are unidirectional differential signals**

- SCI links continually transmit symbols that contain 16 data bits plus packet-delimiter (called flag signal) and clock information.
- The signals are pairs consisting of “signal” and “signal*”. A “1”-bit has “signal” at its most positive voltage level and “signal*” at its most negative voltage level.
- Differential signaling results in constant current flow between connected modules, enormously simplifying the ground distribution problem compared to normal buses.



- Type 18-DE-500 signals support high-performance boards plugged into a system backplane or cable links connecting proprietary physical packages.
- The Fiber-Optic Physical Layer Type 1-FO-1250 is intended to support longer-distance local communications.

Section Outline

- **Advanced Applications**
  - **SCI**
    - **SCI overview**
    - **Communication protocols**
    - **Cache coherence**
    - **Physical layer**
    - **An Example: 2-D FFT**
    - **Conclusion**
  - **RACEway**
  - **FutureBus+**
  - **PI-Bus**
  - **Conclusion**

RASSP Reinventing Electronic Design — Methodology, Architecture, Infrastructure

DARPA ● Tri-Service

2-D FFT

	C0			C1			C2		
	00	01	02	03	04	05	06	07	08
R0	10	11	12	13	14	15	16	17	18
	20	21	22	23	24	25	26	27	28
	30	31	32	33	34	35	36	37	38
R1	40	41	42	43	44	45	46	47	48
	50	51	52	53	54	55	56	57	58
	60	61	62	63	64	65	66	67	68
R2	70	71	72	73	74	75	76	77	78
	80	81	82	83	84	85	86	87	88

(a) Natural Order

	C0			C1			C2		
	00	01	02	13	14	15	26	27	28
R0	10	11	12	23	24	25	06	07	08
	20	21	22	03	04	05	16	17	18
	30	31	32	43	44	45	56	57	58
R1	40	41	42	53	54	55	36	37	38
	50	51	52	33	34	35	46	47	48
	60	61	62	73	74	75	86	87	88
R2	70	71	72	83	84	85	66	67	68
	80	81	82	63	64	65	76	77	78

(b) Column-dependent Vertical Roll

Within each node:
roll rows up zero places in C0
roll rows up one places in C1
roll rows up two places in C2

- Initially, the input data is arranged in “natural order.” The data is first rearranged so that each row of data is contained in a single mesh multiprocessor.
- Then, a one-dimensional FFT transforms n^2 rows of data at once.

2-D FFT (Cont.)

	C0			C1			C2			
R0	00	01	02	13	14	15	26	27	28	i
	06	07	08	10	11	12	23	24	15	ii
	03	04	05	16	17	18	20	21	22	iii
R1	30	31	32	43	44	45	56	57	58	i
	36	37	38	40	41	42	53	54	55	ii
	33	34	35	46	47	48	50	51	52	iii
R2	60	61	62	73	74	75	86	87	88	i
	66	67	68	70	71	72	83	84	85	ii
	63	64	65	76	77	78	80	81	82	iii

(c) Lockstep Shift

Between nodes:
i. right shift 0 node
ii. right shift 1 node
iii .right shift 2 nodes

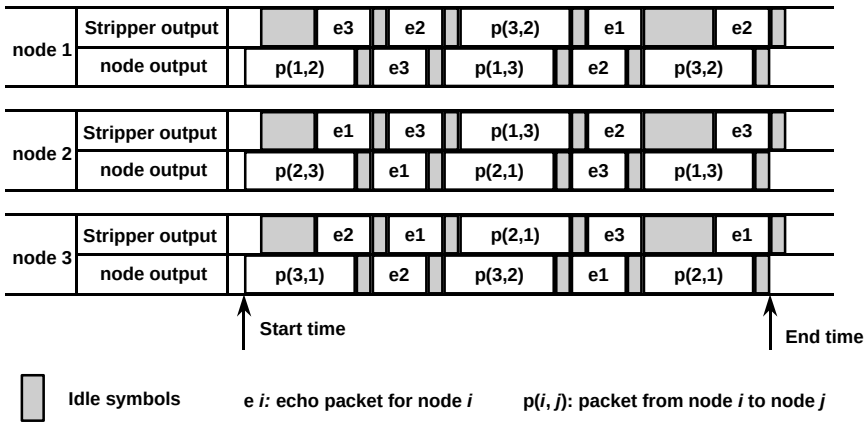
	C0			C1			C2		
R0	00	01	02	10	11	12	20	21	22
	03	04	05	13	14	15	23	24	25
	06	07	08	16	17	18	26	27	28
R1	30	31	32	40	41	42	50	51	52
	33	34	35	43	44	45	53	54	55
	36	37	38	46	47	48	56	57	58
R2	60	61	62	70	71	72	80	81	82
	63	64	65	73	74	75	83	84	85
	66	67	68	76	77	78	86	87	88

(d) Column-dependent Row Interchange

Within each node:
interchange row 2&3 in C0
interchange row 1&2 in C1
interchange row 1&3 in C2

- Next, the data is rearranged so that each column of results is contained within a single mesh multiprocessor.
- Finally, the results of the column transformation are restored in natural row-column order.

2-D FFT (Cont.)



Lockstep Timing Diagram

- Assume that the column-dependent vertical roll and the inversion can be implemented with address pointers.
- The interprocessor communication time is only determined by four lockstep shifts.

2-D FFT (Cont.)

Lockstep	Row FFT	Lockstep	Lockstep	Column FFT	Lockstep

Time

1 Lockstep Time = P * L + (P-1) * 4 + (2P-1)

N = The number of nodes per dimension
P = The number of transmitted packet per node, [1+2+...+(N-1)]
L = Packet Length (Unit: Block = 2 Byte)

Inter-communication Time = 4 * (Lockstep Time)

2-D FFT Timing Diagram

- For the SCI network, 9x9 torus topology is used, and the flow control mechanism is employed to get better performance.

Section Outline

- **Advanced Applications**
  - **SCI**
    - **SCI overview**
    - **Communication protocols**
    - **Cache coherence**
    - **Physical layer**
    - **An Example: 2-D FFT**
    - **Conclusion**
  - **RACEway**
  - **FutureBus+**
  - **PI-Bus**
  - **Conclusion**

SCI - A Suitable Protocol for High Bandwidth Data Networks

- **SCI networks provide a high throughput rate of 1GByte per second per node**
- **The bandwidth of SCI networks is scalable**
- **A point-to-point SCI network provides a high degree of parallelism**

- With a **16-bit** data bus and **2ns** cycle time, SCI provides a throughput rate of **1Gbyte per second per node**.
- SCI defines point-to-point connections between nodes so that packets can **be transmitted concurrently** in SCI networks and the **bandwidth is scalable**.
- A high degree of parallelism implies a high bandwidth data network.

Why Is SCI Suitable for RASSP?

- **A point-to-point SCI network is useful for building a local communication scheme for highly parallel real-time signal processing**
- **SCI can meet the requirements of parallelism and communication overhead**

- Traditionally, highly parallel real-time signal processing is dominated by dedicated parallel architectures such as SIMD array processors. Most SIMD systems use local connectivity between processors and distributed memory rather than global communication.
- Fine-grain computation often appears in DSP applications such as FFT, and it leads to a much high degree of parallelism and also to higher communication overhead.

Why Is SCI Suitable for RASSP? (Cont.)

- **Mesh architectures often provide very large speedups after an image is loaded. SCI networks can promote the performance of image processing tasks because mesh architecture is one of the network topologies which SCI favor**
- **SCI can be a good candidate for a large RASSP application because SCI can support a large system with up to 64K nodes and SCI interface is simple and inexpensive**

- Parallel processing systems such as mesh computers or pipeline processors have been successfully applied to some image processing tasks.
- Commercial DSP chips are becoming inexpensive, and they provide high computation performance for DSP tasks. It is a good opportunity to build powerful DSP systems by using a large number of DSP chips.

Section Outline

- **Advanced Applications**
  - **SCI**
    - **SCI overview**
    - **Communication protocols**
    - **Cache coherence**
    - **Physical layer**
    - **An Example: 2-D FFT**
    - **Conclusion**
  - **RACEway**
  - **FutureBus+**
  - **PI-Bus**
  - **Conclusion**

RACEway Overview

- Has a 32-Bit data path
- Operates at a 40 MHz clock rate
- Is designed for embedded applications where footprint and power are a concern
- The peak throughput is 160 MBytes per second per data path
- The basic element is a crossbar chip with six input/output channels

- Current commercial BGA package at about 1 square inch, 1 Watt.
- Routing information is contained in 32-bit words strobed onto a data bus on a crossbar port.

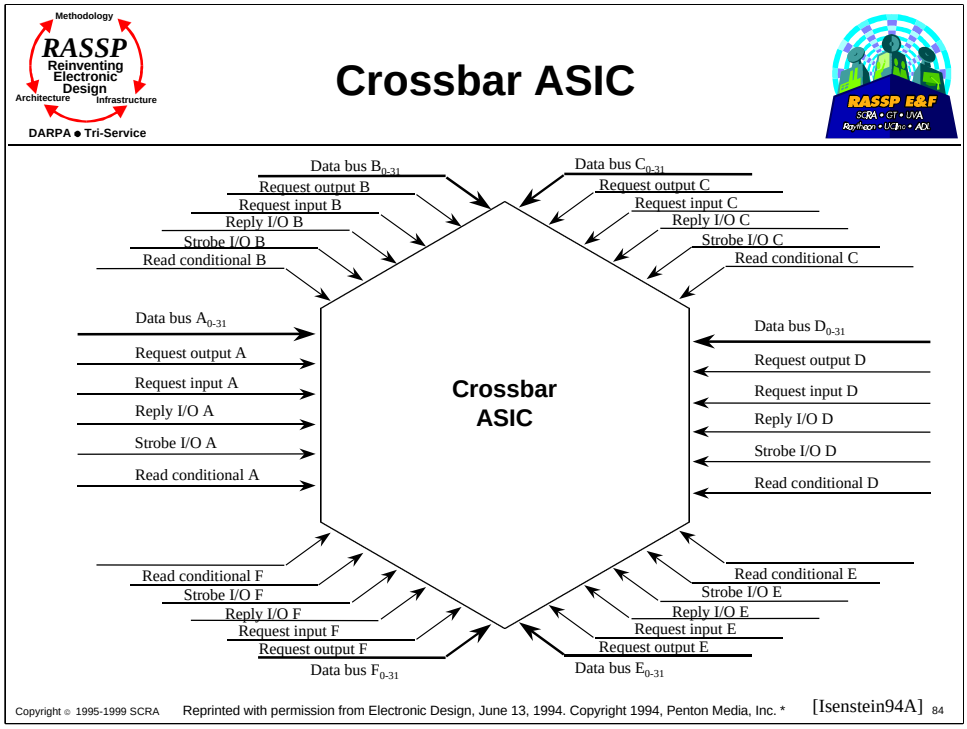
Methodology
RASSP
Reinventing Electronic Design
Architecture
Infrastructure
DARPA ● Tri-Service

RACEway Crossbar

RASSP E&F
SCRA • GT • UVA
Raytheon • UCDavis • ADX

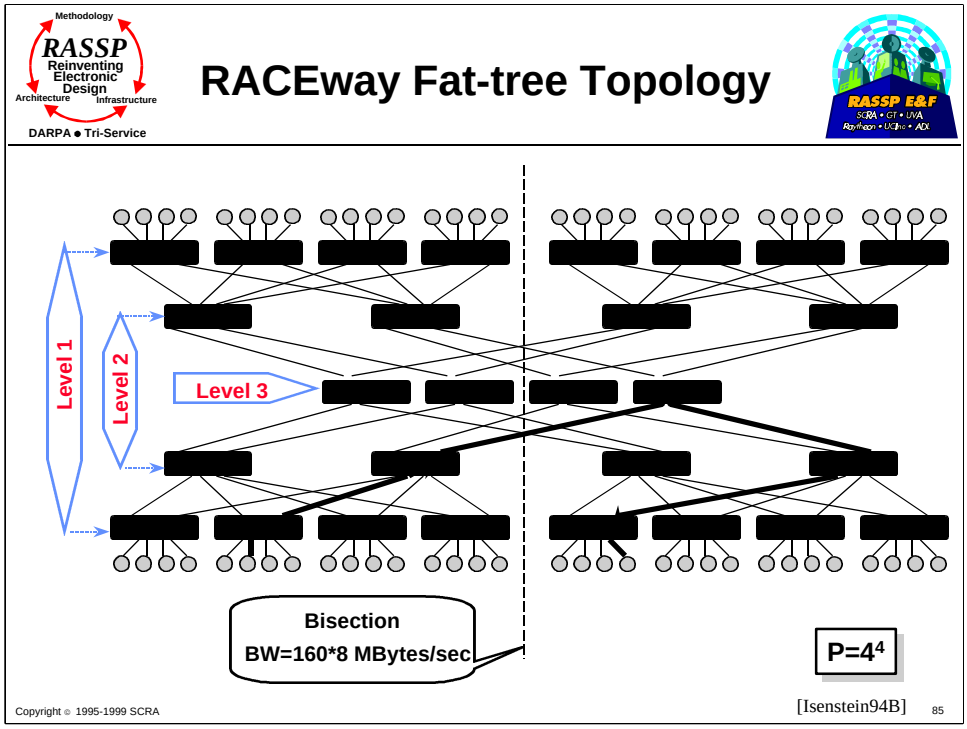
- **Each port is either a transaction initiator, recipient, or intermediary**
- **Data packet priorities allow for decentralized arbitration and preemption**
- **Data packet flow through the crossbar network is based on routing data contained in the packet header**
- **The data flow is *bi-directional*, but can only go in *one direction at a time***

- The packet contains 32-bit words ahead of the address word and the data bits.
- The address word contains a lock bit, up to 31 address bits, and 4 width-alignment bits. This lets the master access up to 16 Gbytes of slave address memory.



- In addition to the 32-bit data bus, five other signal wires connect to each crossbar port to supply port-to-port handshaking for such functions as ready-to-send address, ready-to-send data, ready-to-receive address, and ready-to-receive data.

* Address
Penton Media, Inc
611 Route 46 W
Hasbrouck Heights, NJ 07604 (U.S.A.)



- A highly scalable RACEway interconnect topology is a fat-tree.
- Each chip has two parents and four children.

RACEway Fat-tree Topology (Cont.)

- In a P(=4^i) processor machine
  - Network diameter is 2[$\log_4 P$] hops
  - The bandwidth across any bisection is

    $160\sqrt{P}$ MBytes/s in one direction
  - Each RACE crossbar port is capable of 160 MBytes/s providing an aggregate 480 MBytes/s for a single chip

Software Architecture

- **Application Programming Interface (API)**
- **Tools interface**
- **Hardware device driver interface**
- **Kernel service interface**

- API interface allows various standard and proprietary APIs to be implemented on RACE systems.
- Tools interface provides a method by which new tools can be adapted.
- Hardware device driver interface is used to enhance the ease with which devices can interface to the RACE crossbar.
- Kernel service interface specifies how the software backplane has the required access to kernel services provided by an operating system.

Section Outline

- **Advanced Applications**
  - **SCI**
    - **SCI overview**
    - **Communication protocols**
    - **Cache coherence**
    - **Physical layer**
    - **An Example: 2-D FFT**
    - **Conclusion**
  - **RACEway**
  - **FutureBus+**
  - **PI-Bus**
  - **Conclusion**

Objectives of FutureBus

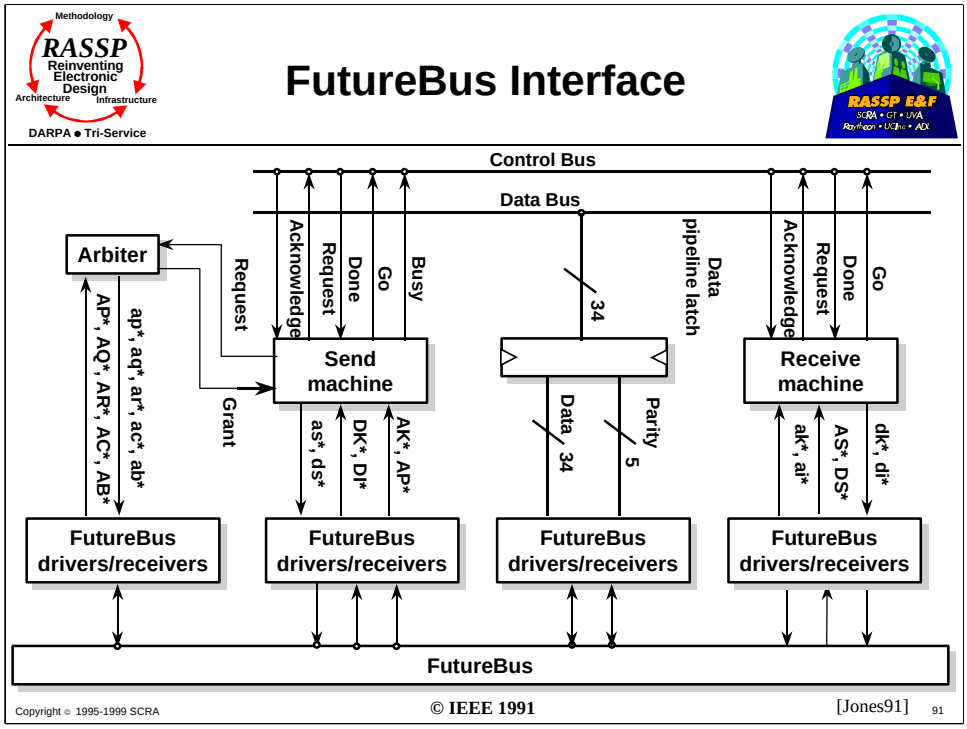
- System architecture independent
- Maximum interoperability of boards and systems
- Fault-tolerant system support
- Use IEC mechanical standards and connectors
- System configuration support
- Cost effective

RASSP Reinventing Electronic Design
Methodology Architecture Infrastructure
DARPA • Tri-Service

FutureBus

- **An asynchronous bus, using 32-bit multiplexed address and data path**
- **A fair distributed bus arbiter for allocating bus access**
- **A two-cycle handshake for transferring multiple data items**
- **Consists of: the address/data bus, the command bus, the status bus, and the arbitration bus**

- A fully loaded bus will be capable of a transfer cycle time of 60-80 ns. Most transactions would transfer data as well as address, thus taking at least two cycles.
- The bus consists of:
  - Address/data bus: 34 bits (32 bits for data, 2 bits for tags)
  - Command bus: 5 bits (Master to Slave)
  - Status bus: 3 bits (Slave to Master)
  - Arbitration bus: 11 bits (7-bit device number, 4-bit control)



- FutureBus uses uppercase signal names (such as AK*) to refer to the bus signals, while the corresponding lowercase name(ak*) identifies the signal used by the board.
- The asterisk (*) after signal names indicates its active-low state.

Arbitration

- **Fully distributed**
  - **No central control**
- **Asynchronous**
  - **Fully handshake**
- **Fairness mode normal**
- **Priority mode access**
- **Up to 32 logical units**
- **Parity error control**

- The bus master must contain a complete arbitration controller.
- The protocol is based on an asynchronous three-wire handshake that allows the modules to step through the various phases of arbitration in unison and adapt the speed of the protocol to the slowest participating module.
- The design of the lowest-cost board can affect the performance of the entire backplane. Fortunately, a careful design can avoid the worst pitfalls.

Compete for Mastership

- **Each competing board applies its unique arbitration number *an*[7 ... 0]* to the bus signals *AB*[7 ... 0]* by asserting *ab[n]* if and only if**
  - ***an*[n]* is asserted, and**
  - **For all *m>n* when *an*[n]* is released, so is *AB*[m]***
- **After a certain time delay, the signals on *AB*[7 ... 0]* will be the largest arbitration number of any competing board**

- The second condition fails when some higher priority board drives the bus, thereby asserting *AB[m]* for some *m*. In this case, all lower priority boards release ab[m-1, ... ,0] so that they do not interfere with the lower-order bits of the high-priority board's arbitration number.

Parallel Bus Protocol

- **Totally asynchronous and technology-independent handshake**
- **Totally transparent communication with memory using individual byte buses**
- **Mechanism for Broadcast on all sequences**
- **Multiple levels of locking for mutual exclusion**
- **Built-in *hooks* to extend the protocol**

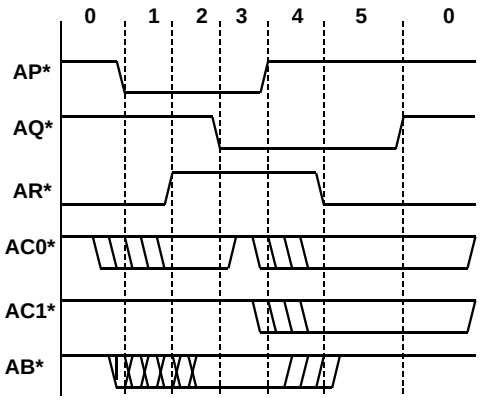
- Basic communication primitives include address cycle, read cycle, and write cycle. These may combined to form more powerful sequences such as address-only sequence, single-transfer sequence, mixed sequence, and block-transfer sequence.

Data Transmission Protocol

- The block length can be zero to 4Gbytes
- The protocol is asynchronous and uses a two-edged handshake
- A data transfer rate of up to 117 MBytes/sec can be achieved
- Broadcast data transfers over the full 4Gbyte address space are supported

- The data transmission protocol has been optimized for the transmission of data in blocks. In practice, blocks will be kept short, say 256 bytes, to ensure bus access within a given time period.
- Two-edged handshake enables data to be transferred on both the rising and falling edges of a strobe line and hence saves the time required to return the strobe signal to its released state.

Three-Wire Synchronization Protocol



1) The arbiter asserts AR* and release AP* and AQ*

2) When an arbiter is ready to move to the next state, it asserts ap* and release ar*

3) Only when all arbiters are ready to move will all boards release ar*, thus releasing the AR* bus signal. All arbiters register the release of AR* and then move to the next state

4) When an arbiter is ready to move on again, it asserts aq* and releases ap*

5) As before, only when all arbiters are ready to move will all boards release ap* and then bus signal AP*

6) Subsequent state transitions repeat a similar process

© IEEE 1991

[Jones91]

96

1) **Any arbiter requiring mastership initiates the protocol by asserting ap* and releasing ar*.** All other arbiters accept the assertion of AP* and signal their willingness to proceed.

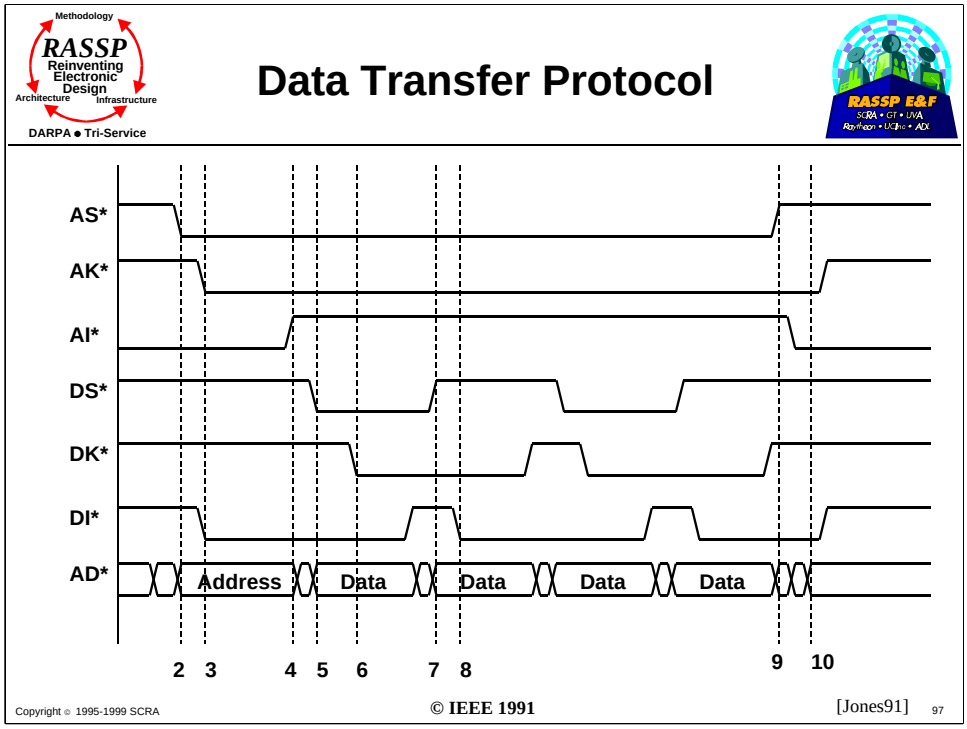
2) **The release of AR* by the last module must propagate across the bus and pass through the glitch filters of all other modules.**

3) **When one competitor's timer completes with its "win" signal asserted, it asserts aq* to end the competition.**

4) **The assertion of AQ* propagates across the bus to the remaining competitors.** If the current master is finishing its transaction, they can release ap*.

5) **The release of AP* by the last module must propagate across the bus.** If the current bus master has completed its tenure, it finishes disconnecting from the bus and asserts ar*.

6) **The assertion of AR* propagates across the bus to the other modules.** After verifying that the competition has not been canceled, the winner's local arbiter can issue a bus grant to its internal logic. All modules release aq*, bringing the bus back into the idle state.



- This shows the interaction between master and slave during the transfer of a 5-word packet.
- Such a packet transfers in five beats: an address beat to send the first word, and four data beats to send the subsequent words.

Data Transfer Protocol (Cont.)

1. **The sender gains control of the bus**
2. **The master places the address of the recipient on the address/data signals**
3. **When AS* is asserted, *each and every* board decodes the address**
4. **When AI* releases, the master can safely remove the address from the bus**
5. **Next, the master places the first word to be transmitted on the bus**
6. **When DS* asserts, the recipient reads the data from the bus.**
7. **When DI* releases, the transfer of the first data word completes. The master then places the next data word**
7. **When DS* releases, the recipient grabs the next data**
9. **Data transfer now continues in this way until the master has no more to send**
A. **The recipient, and all other slaves, acknowledge the release of AS* by releasing ak* and asserting ai***
B. **When AK* releases, the transaction completes**

 98

- The sender gains control of the bus, using the arbitration protocol. The sender thereby becomes the bus master.
- The master places the address of the recipient on the address/data signals AD*[0-31] and then asserts the control strobe as*. The strobe, in turn, causes the bus signal AS* to be asserted.
- When AS* is asserted, *each and every* board decodes the address. When decoding is complete, the board asserts the acknowledge signal ak* and releases a complementary signal ai*.
- When AI* releases, the master can safely remove the address from the bus.
- Next, the master places the first word to be transmitted on the AD[0-31] bus and asserts the data strobe ds*.
- When DS* asserts, the recipient reads the data from the bus. When it has captured the data, it asserts dk* and releases di*.
- When DI* releases, the transfer of the first data word completes. The master then places the next data word and releases ds*.
- When DS* releases, the recipient grabs the next data and then asserts di* and release dk*.
- Data transfer now continues in this way until the master has no more to send. Then the master releases as*.
- The recipient, and all other slaves, acknowledge the release of AS* by releasing ak* and asserting ai*.

Electrical Specifications

- **Single +5V power rail**
- **Total open-collector system in a transmission line environment**
- **Driver specification to solve the *bus driving problem***
- **Trapezoidal driver to reduce crosstalk**
- **Receiver filters to solve the *wired-OR glitch problem***

- Achieving a sufficiently large signal at the far end of the backplane therefore requires, optimistically, at least 100mA drivers.

RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA ● Tri-Service

Major Features

- **A bus driver can switch all receivers in one bus propagation**
  - **FutureBus does not require *settling time***
- **The protocol provides a fast, two-edged block-transfer mode**
- **Arbitration may take place concurrently with data transfer**
- **A truly technology-independent handshake**

- The protocols permit any board, no matter how fast, to interoperate with any other board, no matter how slow.
- Several drivers may drive a single bus signal simultaneously without damage.

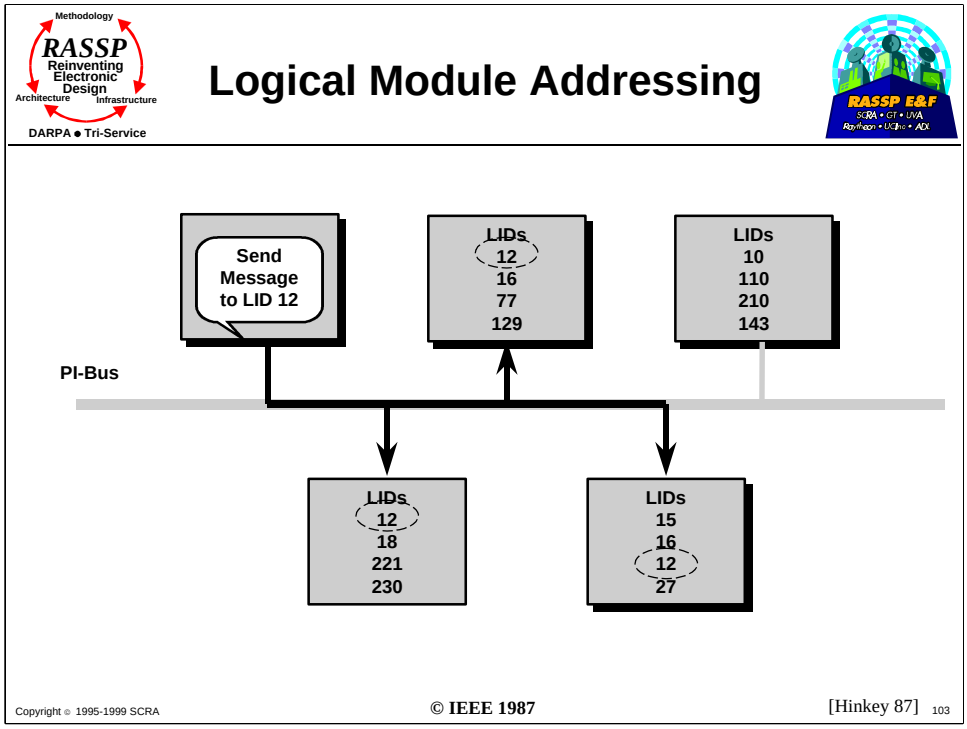
Section Outline

- **Advanced Applications**
  - **SCI**
    - **SCI overview**
    - **Communication protocols**
    - **Cache coherence**
    - **Physical layer**
    - **An Example: 2-D FFT**
    - **Conclusion**
  - **RACEway**
  - **FutureBus+**
  - **PI-Bus**
  - **Conclusion**

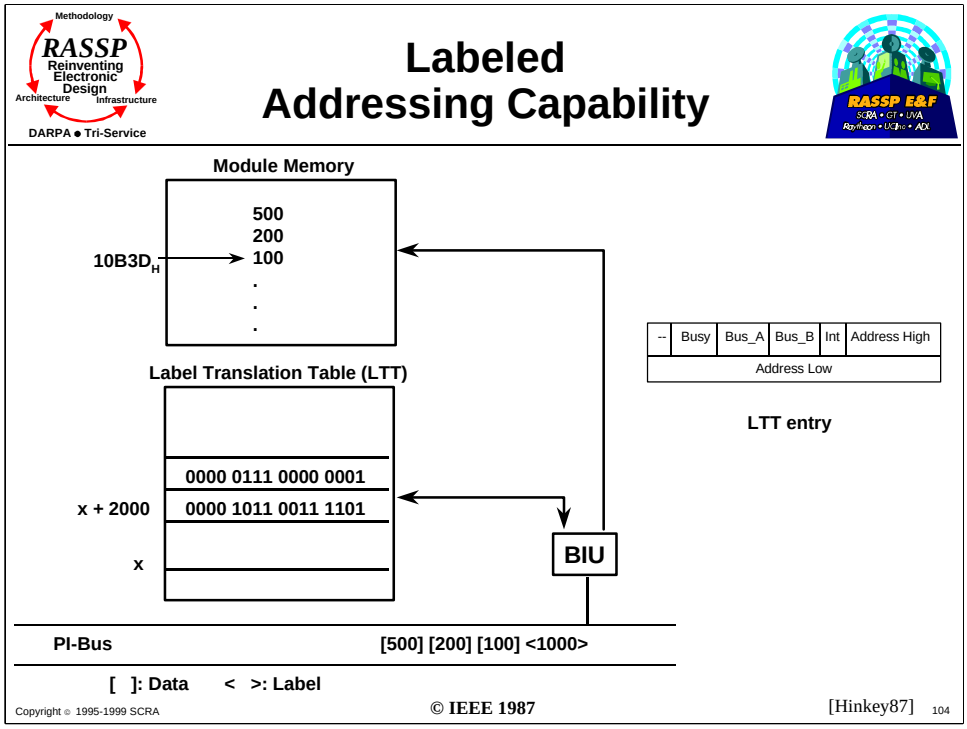
PI-Bus Overview

- **The PI-Bus is a multidrop parallel (16-bit or 32-bit) bus which uses its DMA capability to transfer data between the modules**
- **The PI-Bus has two levels of logical addressing capabilities**
- **Logical module Identifiers (LIDs) allow a message to be sent to a "logical" module**
- **The logical addressing capability provides efficient fault recovery**

- The PI-Bus has bus addresses 33-255 allocated for logical identifiers.
- The bus interface unit (BIU) on each module in the system can activate any number of these 223 logical identifiers.
- The logical module identifiers are mechanized through the configuration image which is loaded into the BIU from the host module**.**



- The labeled addressing capability of the PI-Bus is an enhancement that allows data to be read from a remote module without the reader knowing the physical address of the data he wants to read.



- The LTT entry contains a busy bit, an active bit for each of two buses, and an interrupt bit, as well as a 24-bit physical address.

Guidelines for Developing the Software

- **Each process or group of processes that will always be on the same processor will be assigned an LID**
- **Each data object in a process that can be referenced by another process has a label assigned to it**

- When sending data or requesting data from a process, all the application software will refer to that process by its logical identifier.
- Any data a process receives via an output from another process, or any data that another process will retrieve from this process via an input, must have a label assigned to it.

RASSP Reinventing Electronic Design
Methodology
Architecture
Infrastructure
DARPA • Tri-Service

Features of PI-Bus

- **Provides for addressing modules on the bus in a logical manner**
- **The memory locations in each of the processors on the bus can be referenced logically**
- **The application software developer need not be at all concerned about a processor failing**

- These logical reference capabilities allow a system to be made fault tolerant and to recover from processor failures in an efficient and straightforward manner.
- The operation system and hardware will handle the change in the configuration of the system, and the application software continues execution as if nothing had happened.

Section Outline

- **Advanced Applications**
  - **SCI**
    - **SCI overview**
    - **Communication protocols**
    - **Cache coherence**
    - **Physical layer**
    - **An Example: 2-D FFT**
    - **Conclusion**
  - **RACEway**
  - **FutureBus+**
  - **PI-Bus**
  - **Conclusion**

Pros and Cons of Shared Bus Architectures

- **Bus systems have a broadcast capability**
- **Buses have reached practical and inherent limits, such as**
  - **The speed of light**
  - **The capacitance of transceivers and connectors**
  - **The one-talker-at-a-time bottleneck**
- **Shared buses are limited in the degree of scalability by the fixed interconnect bandwidth**

- Broadcast and multicast capabilities are highly desired features in computer architectures.
- The speed of light limits the propagation velocity of signals.
- Bus connectors and transceivers load the transmission line. The ideal transmission line model is a very poor approximation indeed.
- A bus system can be used by only one transmitter at a time.
- For example, doubling the width of a bus does not double its speed because there are fixed overheads associated with arbitration and addressing.
- Multiple buses: it results in a complex bus-bridge mechanism to maintain cache consistency in shared-memory systems.

Loading Problems of Shared Buses

- **The fundamental physics limits - the speed of light**
  - **Solution: Unidirectional links**
- **Crosstalk between adjacent signal**
  - **Solution: Differential signaling**

Pros and Cons of RACEway

- **High degree of scalability**
  - **The I/O cost per processor remains the same**
- **High composite bandwidth**
  - **Each crossbar can transfer three messages simultaneously**
- **Topology independence**
  - **RACE system can be configured in many different kinds of networks**
- **Easy and compact interfacing**
  - **RACEway use CMOS signaling levels at 40 MHz with no requirement for exotic design or manufacturing techniques to overcome wire density or crosstalk problems.**

- Each node can transfer three messages simultaneously; thus, RACEway provides high composite bandwidth.
- RACE systems can be configured in many different kinds of networks, such as fat-tree, ring, mesh, and Clos. Topology independence allows a developer of real-time computing solutions to fit the topology to the problem instead of trying to fit the problem to the topology.

Pros and Cons of SCI

- **Pros**
  - **SCI networks scale well**
  - **SCI solves loading problems**
  - **SCI makes signaling speed independent of the size of the system**
  - **Independent transfers can take place concurrently**
- **Cons**
  - **Lacks a broadcast capability**
  - **A long sharing list turns out heavy traffic loads**

- Scalability is useful because the same mechanisms can be used in high-volume single-processor systems found in desktop machines, as well as in large highly parallel multiprocessors. Future technological improvements will bring new link standards; however, SCI protocols will still work.
- SCI solves the loading problems of the conventional buses by eliminating the multiple connectors or stubs, allowing only one driver and one receiver.
- Because SCI makes signaling speed independent of the size of the system, it increases the throughput rate of data networks.

Comparison of Scalability

	Shared Bus System	SCI	RACEway
Building blocks	No	Yes	Yes
Separate configuration control	No	No	Yes
Deterministic I/O ports	No	Yes	Yes
Self-routing data packets	No	Yes	Yes
Transparent buffering	No	Yes	Yes
Broadcast capability	Yes	No	Yes
Centralized arbitration	Yes	No	No

- The reader may wish to study other protocols such as Myrinet and MPI to add to the material covered in this module.

Applicability in High Bandwidth Data Network Design

	Scalability	High throughput	Efficient cache protocols	Starvations
Shared Buses	bad	bad	excellent	sometimes occur
RACEway	excellent	good	good	No
SCI	good	excellent	good	No

Module Outline

- Introduction
- State of the Art
- Impact of RASSP
- Advanced Applications
- **Summary**

Summary

- High bandwidth data network is demanded for RASSP applications
- High throughput, high scalability, and forward progress are required in high bandwidth data network design
- SCI and RACEway are suitable for high performance RASSP systems

References

[Analog94] Analog Device, ADSP2106x SHARC: Preliminary User's Manual, March, 1994.

[Hinkey 87] Michael Hinkey and Drew Clarke, "A Fault Recovery Mechanism Using Logical Bus Addressing", *IEEE*, 1987; © IEEE 1987

[IEEE] All referenced IEEE material is used with permission.

[IEEE93] "IEEE Standard for Scalable Coherent Interface (SCI)" , *IEEE Computer Society,* Aug 2, 1993 ; © IEEE 1993

[Isenstein94A] Barry Isenstein, "Scaling I/O Bandwidth With Multiprocessors," *Electronic Design*, June 13, 1994 also also http://www.mc.com.

[Isenstein94B] Barry S. Isenstein and Bradley C. Kuszmaul, "Overview of the RACE Hardware and Software Architecture," 1st RASSP Annual Conference, 1994 also http://www.mc.com.

[James90] David V. James, et al., "Distributed-Directory Scheme: Scalable Coherent Interface," *Computer*, June, 1990 ; © IEEE 1990

[Jones91] Simon L. Peyton Jones and Mark S. Hardle, "A FutureBus Interface from Off-the-Shelf Parts", *IEEE Micro*, Feb, 1991 ; © IEEE 1991

[Lockheed95] Shirley F., et. al., RASSP Architecture Guide , Lockheed-Sanders, Revision B, Jan 5, 1995. This work was performed by Sanders, a Lockheed Martin Company, as a part of the Sanders RASSP program under contract N00014-93-C-2172 to the Naval Research Laboratory, 4555 Overlook Avenue, SW, Washington, DC 20375-5326. The Sponsoring Agency is: Defense Advanced Research Projects Agency, Electronic System Technology Office, 3701 North Fairfax Drive, Arlington, VA 22203-1714. The Sanders RASSP team consists of Sanders, Motorola, Hughes, and ISX.

[Richards97] Richards, M., Gadient, A., Frank, G., eds. *Rapid Prototyping of Application Specific Signal Processors*, Kluwer Academic Publishers, Norwell, MA, 1997