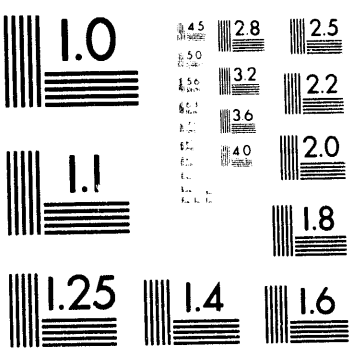

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

1 of 1

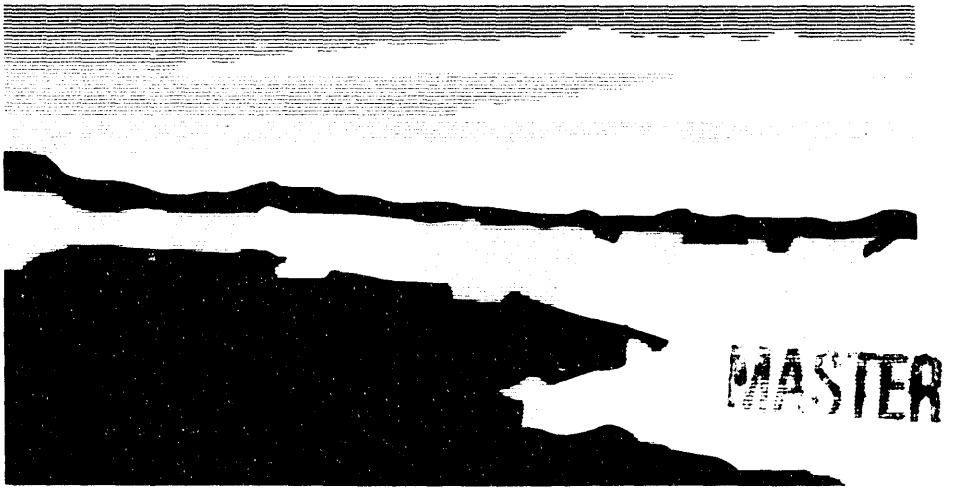
Conf-931220--2

LA-UR- 93-3137

Title: SIMULATED BEHAVIOUR OF LARGE SCALE SCI RINGS AND TORI

Author(s): R. Daniel, Jr., H. Cha, and A. Knowles

Submitted to: IEEE Symp./Parrallel and Distributed Processing
Dallas, TX
December 1-4, 1993



Los Alamos
NATIONAL LABORATORY

Form No. 836 R5
ST 2629 10/91

Simulated Behaviour of Large Scale SCI Rings and Tori

Hojung Cha[†1], Ron Daniel Jr.[‡2], and Alan Knowles[†]

[†]Department of Computer Science
University of Manchester

[‡]Cambridge University Engineering Department

Abstract

SCI (Scalable Coherent Interface) is a new IEEE standard for a high speed interconnect in parallel processors. It is attracting interest because of its high bandwidth (1 GB/sec/link) and low latency. The default SCI topology is a ring, which does not scale well to large numbers of processors. This paper uses stochastic and trace-driven simulations to compare the performance of SCI-based parallel computers with a ring topology to those based on a torus topology. We also look at the effects of varying some of the internals of the SCI components.

1 Introduction

The goal of the *ex*-static project is to simulate the sorts of massively parallel architectures that can provide a basis for implementing large scale Neural Networks (NNs). The simulations allow us to compare the performance of different architectures on various NN tasks [1, 2, 3]. This paper presents results from our simulations of members of the ClusterNode architectural family. The machines we model use clusters of TMS 320C40 DSP microprocessors [4] interconnected by the recently adopted standard IEEE 1596-1992 Scalable Coherent Interface (SCI) [5, 6]. The DSP chips are well suited to the intensive numeric operations that characterise NN applications, and their on-board communication links make it easy to implement the intra-cluster communication network. SCI provides high bandwidth, low latency communications. This paper compares ClusterNode machines based on SCI networks with ring and torus topologies. We also examine cluster size tradeoffs and the effects of varying some of the internals of the SCI interfaces. Sections 2 and 3 describe SCI and the 2x2 switch that we use in the torus networks. Section 4 describes the ClusterNode architecture, section 5 describes our simulators, section 6 presents and interprets our results, and section 7 gives our conclusions.

[1]Current address: Dept. of Computer Science, Kwangwoon University, Weolgyedong, Nowongu, Seoul, Korea

[2]Current address: Advanced Computing Lab, MS B287, Los Alamos Natl. Lab, Los Alamos, NM, USA, 87545, rdaniel@acl.lanl.gov

2 SCI

SCI began as an attempt to define an ultra-high-performance bus to follow FutureBus+. However, it is impractical to push backplane bus technologies significantly past the capabilities of FutureBus+ because of the "ringing" associated with multi-tapped transmission lines. Therefore, SCI abandoned the backplane model and adopted unidirectional point-to-point links. The SCI standard [5] is organised in three levels; physical, logical, and cache-coherence. The physical level defines 2 implementations. The first provides a bandwidth of 1 GB/sec/link over 18 differential pairs. The second provides 1 Gb/sec/link over medium length runs of coax or long runs of fibre.

The logical level defines a packet-based protocol that provides bus-like functions to the nodes. The difficulty of unidirectional links is, of course, that status information has no way to go from the receiver back to the sender. SCI nodes can be chained into ring topologies so that status information can get back to the sender by flowing around the ring. To reduce latencies, SCI packets are small and have very small headers. Interfaces can make routing decisions in nanoseconds, allowing the use of through-routing techniques to reduce latencies further.

The cache-coherence level allows large parallel machines to be built with a coherent system-wide memory. Machines are not required to utilise this level, and we only use SCI as the communications fabric for message-passing machines.

To get a feel for the operation of SCI, we will discuss the operation of the simplified interface shown in Figure 1.

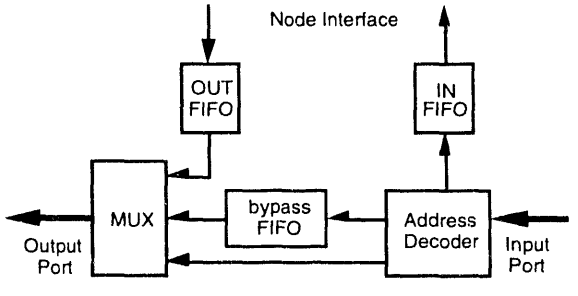


Figure 1: SCI Node Interface

A packet's detination ID is its first symbol. The address decoder looks at that ID and decides if the packet is to be accepted at this node. If not, and the node is not currently

transmitting another packet, the packet is routed to the output multiplexer for immediate transmission on the output link. This can make the per-hop latency very small (≈ 10 ns). If the node is transmitting, the incoming packet accumulates in the bypass FIFO until it can continue on its way. As noted in [7], the bypass FIFO allows multiple nodes on a ring to transmit simultaneously, unlike token-based schemes. It also allows the system to degrade gracefully from very low latency virtual cut-through routing when lightly loaded to store-and-forward when heavily loaded [3].

If the packet is addressed to this node, it is sent to the input FIFO. An echo packet is generated and sent back to the source node to let it know the packet has been safely accepted. If the input FIFO is full, or if a CRC error is detected, the echo packet carries error information back to the source node so that it can retransmit the packet. Once a packet is safely in the input FIFO, its processing is up to the node interface. Typically a DMA cycle will occur to place the packet into node memory.

When packets are sent by a node, they are first loaded into the output FIFO. As soon as some free space in the symbol stream is detected, the packet is sent. A copy of the packet is retained in the FIFO until reception is acknowledged, then it is dropped from the FIFO to make room for another. Note that a node can easily consume all its FIFO space, preventing it from doing any more communication until it has processed its current entries. To prevent deadlock, the SCI standard requires that the FIFOs have separate sections for requests and responses, so that outstanding requests do not block their responses in the FIFOs.

3 Simple SCI switch

While the default SCI topology is a ring, others are possible. The simplest possible SCI switch would transfer packets between two rings. Many useful topologies, such as tori, meshes, and multistage networks, can be developed from such 2x2 switching elements. The design we model in our simulations is shown in Figure 2.

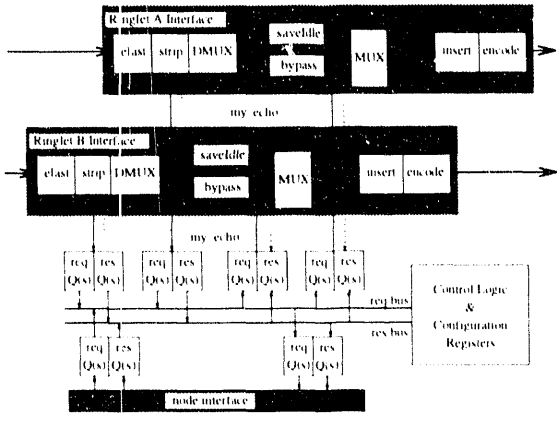


Figure 2: 2x2 SCI Switch

This illustration is more detailed than the node interface in Figure 1, but its principles are similar. The main differences are that the address decoder, labelled "strip", must detect more than one destination ID. Accepted packets not destined for this node are then emitted onto the other ring. Due to the more complex logic in the strip box, we assume that packets will be delayed by 4 ns more than for the simple interface, and packets that cross ringlets are delayed by a further 4 ns.

4 ClusterNode architecture

One of the architectures we are considering is the ClusterNode. This is a conceptually simple but interesting architecture with a hierarchical interconnection network. The basic structure is shown below in Figure 3A. The set of processing nodes is divided into clusters. For simplicity, this paper assumes that each cluster has an equal number of nodes. The intra-cluster interconnection networks, denoted by IN_0, are at the lowest level of the hierarchy. The clusters are connected by the next higher level interconnection network, IN_1. Extending the architecture to have more levels conceptually straightforward, as shown in Figure 3B.

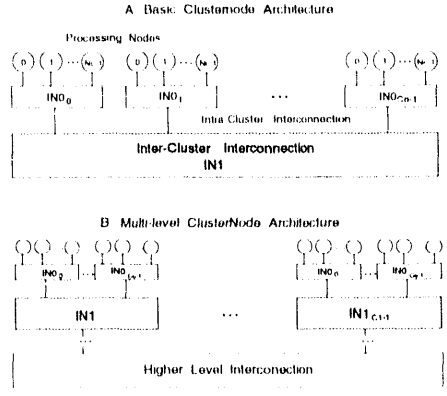


Figure 3: ClusterNode Architecture

This architecture is motivated by several considerations. First, IN_1 should be a higher-bandwidth network than IN_0. Under reasonable assumptions, this makes the cost per connection to it greater. Second, if there is some locality in communications, clustering the processors could take advantage of that and reduce the number of messages that must go over IN_1.

This paper examines the performance of 2-level ClusterNode architectures built around the TMS320C40 DSP processor (C40) and an SCI network. The TI device has 6 communications links, each rated at 20 MB/sec. For IN_0 we will use those links to form completely connected networks. For IN_1 we will compare a single SCI ring with a 2D SCI torus. As shown in Figure 4, one C40 serves as the interface between the SCI network and the rest of the processors in the cluster. This is known as the SCI-IP (SCI Interface Processor). It performs the packetisation and de-

packetisation to convert between SCI and C40 message formats. Since the C40 has 6 communication links and we will be simulating completely connected clusters, clusters may have from 1 to 6 C40 processors, and will always have an SCI-IP.

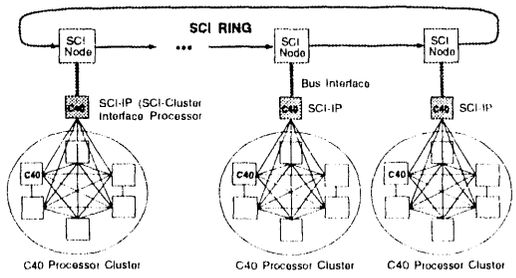


Figure 4: 1-D SCI-C40 Architecture

During simulations, we will be interested in three major timing parameters; *SCI Delay*, *IP Delay*, and *Node Delay*. The first is the time a message actually spends in transit in the SCI network, as well as any time in the queues in the SCI node. The *IP Delay*, is the *SCI Delay* plus the time that the message spends in the SCI-IP at both the source and destination clusters. Finally, the *Node Delay* will usually be the *IP Delay* plus the time spent in message queues at the source C40. However, note that intra-cluster messages do not go through the SCI-IP.

5 Simulation overview

The results presented in this paper were obtained from two simulators. The first is a stochastic simulator purpose-built for modelling aspects of the SCI protocol and its interaction with the node processors. The second is a trace-driven simulator which reads coarse-grained trace files produced during the execution of parallel NN benchmarks. The first simulator allows us to explore a wide variety of system configurations much faster than the trace-driven simulator. It also allows us to see the behaviour of the architectures over a wide range of workloads. The second simulator lets us examine the behaviour of the machines under the irregular workloads that are characteristic of real applications.

Note that our simulators implement a subset of the standard SCI logical protocol. Cache coherence is not investigated in the study. Also, message passing is assumed to use SCI's DMOVE64 command, rather than its more general READ and WRITE transactions.

5.1 Stochastic simulator

Our stochastic simulator uses the process model:

```
while (simulation_continue) do
   CpuWork(meanCPUtime);
   Send(msgType,msgTarg,msgSize...);
   BlockingReceive(msgType...);
end
```

where the *meanCPUtime* and *msgSize* parameters are exponentially distributed and the *msgTarget* is uniformly distributed. This is a closed system, where a process only sends a new message after receiving one. We believe this is a more realistic model for most parallel programs than the open model used in [7]. Our work also differs from that in [7] because we model the interaction with the node processor, while they concentrated solely on the SCI protocol.

The stochastic simulator allows us to model the three timing parameters (*SCI Delay*, *IP Delay*, and *Node Delay*) when we vary design choices of the candidate architecture, such as the number of nodes per cluster, and/or the number of clusters. It would be possible to vary other factors, such as the speed of the processor or the size of the SCI packet. However, we have tried to fix these values at reasonable settings in order to keep the problem space tractable. The most important values are that the SCI ring bandwidth is 1 GByte/sec, the node processor's memory bandwidth is 100 MByte/sec, and the C40 links have a bidirectional bandwidth of 20 MBytes/sec.

5.2 Trace-driven simulator

Figure 5 shows the structure of our trace-driven simulator. The two broad classes of activity in the system are reflected in the structure of the figure. The top half of the figure shows the process of preparing the benchmark NN programs for the system, the bottom half depicts the process of modelling a particular parallel architecture.

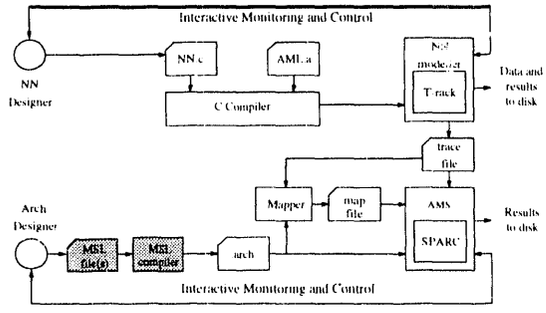


Figure 5: Organisation of the Trace Driven Simulator

A NN designer writes parallel NN simulations to serve as benchmarks. These programs make calls to an Abstract Machine Library (AML) which contains functions for matrix arithmetic, I/O, communications, process control, and memory allocation. Once compiled, the benchmarks are run on the ParSiFal T-Rack [8], an array of 64 T800 transputers. When an AML function is called it emits an entry to a trace file, logging the function called and its arguments. This trace file is used for modelling the parallel architectures.

The bottom half of the figure shows how a particular architecture is modelled. The *.arch* file specifies the number of nodes, their processing power, memory size, etc. It also specifies the characteristics of the interconnection network, such as the topology, bandwidth, latencies, routing

scheme, packet size, etc. At the start of the simulation the *.arch* file is read by the Abstract Machine Simulator (AMS), which configures itself to model the specified machine. The trace file is scanned to determine the number of processes involved in the benchmark, and the mapping tool assigns the processes to the nodes. The simulator then reads trace file entries and determines how long they will take to execute. Note that the AMS simulation does not carry out the operations, it only estimates the time required to do them.

The trace-driven simulator was validated by modelling an existing DM-MIMD machine and comparing the predicted times with measured times. The validation process is described in [9], which reports prediction errors of less than 10%. For our project this level of accuracy is more than sufficient.

6 Simulation results and interpretation

This section presents the results from simulations of possible implementations of the SCI-C40 ClusterNode architecture. We use the stochastic simulator to characterise the performance of SCI networks with ring and torus topologies, then to compare various realisations of the ClusterNode architecture. Finally, we present initial results from trace-driven simulations of neural networks.

6.1 SCI network behaviour

In large measure, SCI's high speed is due to its use of unidirectional links. However, in ring topologies, packet acknowledgements must flow all the way around the ring. This limits the number of nodes that can effectively use a SCI ring before it becomes saturated. There are two reasons for this. First, the more nodes on a ring the larger its diameter, delaying all messsages and their acknowledgements. Second, if each node generates a message at a fixed rate, increasing the number of nodes increases the traffic on the ring which can lead to saturation.

The torus topology should exhibit better performance than the rings for three reasons. First, it has a much smaller diameter ($O(\sqrt{N})$ instead of $O(N)$). Second, it partitions the traffic, reducing the shared traffic problem noted above. Finally, it provides twice as many links per node, doubling the aggregate bandwidth of the network.

6.1.1 Experiment 1: message generation rate: Our first experiment tests the behaviour of SCI rings and tori as they are driven into saturation by generating messages more and more rapidly[1]. Four configurations were tested; 64 and 256 node rings and 8×8 and 16×16 tori. Each node sends 64 byte messages to a random destination. The *meanCPUdelay* was varied from 100 ns, which saturates the network, to 1 ms, which is a very light load. Figure 6A shows the mean *SCI Delay* for a single 64 byte packet while Figure 6B shows the mean *IP Delay* for a 64 byte message. Note that a 64 byte message requires only a single SCI packet. When the network is lightly loaded all configurations have very low latencies. This is because the per-hop latency (≈ 10 ns) is small compared to the time for the packet to transit a node (≈ 80 ns). As the message generation rate increases, the virtual cut-through routing effects diminish as packets increasingly accumulate in the bypass FIFO while outgoing packets complete their transmission. Finally, the latency stabilises when the network is fully saturated since a process must get a packet before it can emit another one. When saturated, the routing has degenerated to store-and-forward, emphasizing the $O(\sqrt{N})$ advantage of the tori over the rings. Also note that the large ring saturates sooner because of the shared traffic problem.

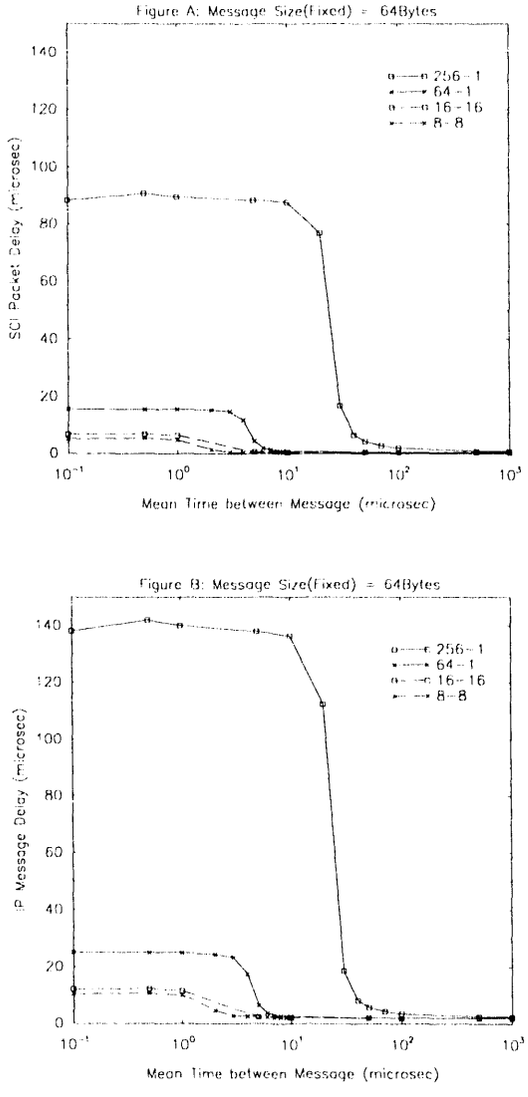


Figure 6: Packet and Message Latencies vs. Message Rate

[1] In order to more thoroughly saturate the network., in this experiment the application model does not call blocking_receive(), instead it relies on the SCI protocol to limit the number of messages.

6.1.2 Experiment 2: message length: Our second experiment was designed to show the behaviour of the SCI networks as the average message size is increased, a situation that is very important for message-passing machines. The 64 node ring and 8×8 torus configurations were used, and the *meanCPUdelay* was varied from 10 ns to 1 ms. The mean message size was varied from 64 to 1024 bytes, using an exponential distribution. Figure 7A shows the effects of message length on the latency for SCI packets, while Figure 7B shows the effect on the total message transmission time.

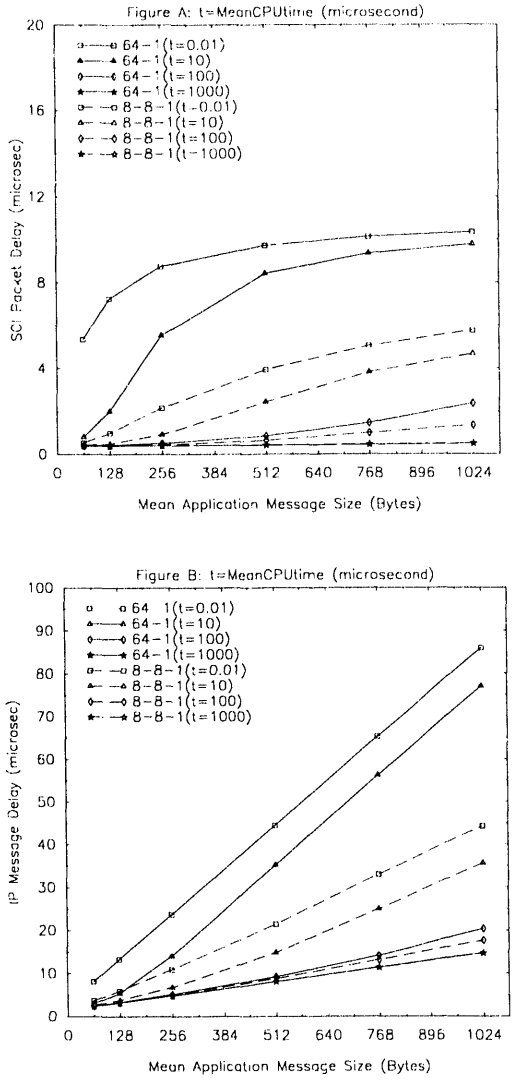


Figure 7: Packet and Message Latencies vs. Message Length

Note on Figure 7A how the curves for the lightly loaded ring topology start with a low packet latency, but the packet latency approaches that for a saturated network as the message length increases. Long messages give bursts of packets which can drive the network into saturation.. The effect this has on message transmission time can be seen in Figure 7B, where the increasing slope of the lines indicates increasing congestion.

6.1.3 Experiment 3: broadcast protocol: SCI provides a broadcasting protocol which we expect to be useful for NN models. A small message is sent around the ring so that nodes who will receive the broadcast message reserve queue space for it. If a node has space already, it allows the reservation message to go on to the next node on the ring. If it does not, the reservation message is changed into an echo packet and the originating node tries again. Once all the destination nodes have indicated that they have reserved queue space, the actual data packet is sent. While the SCI standard expects this to work well on lightly loaded networks, it comments that for a saturated network performance should degrade to simply sending a separate copy of the message to each destination processor.

This experiment was designed to compare the performance of the SCI broadcast protocol with that simple alternative, which we call multiple singlecasting. The two 64-node topologies were used. All nodes broadcast a 64 byte message to all other nodes. This was done using the broadcast protocol, then again using multiple singlecast. The *meanCPUdelay* was again varied from 10 ns to 1 ms to simulate different levels of network traffic. Figure 8 shows an interesting result. The broadcast protocol always performed better than the multiple singlecast, and the performance gap increased as the network became saturated.

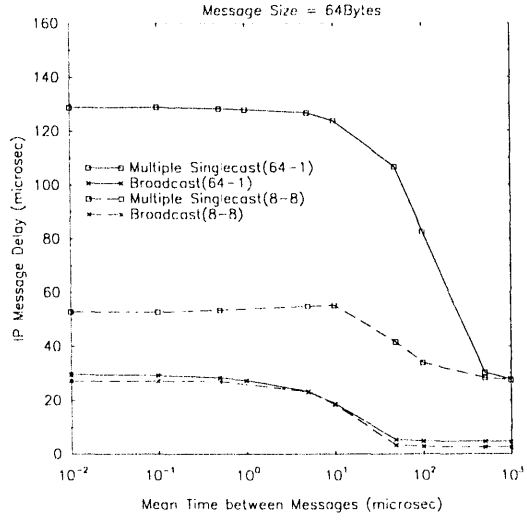


Figure 8: Broadcast Protocol Performance

The explanation for the better-than-expected performance in saturated networks is because, unlike the standards working group, we also consider the DMA interface between the SCI node and the memory system of the attached processor. For each message, the multiple singlecasting version has to contend for a free output queue space, which is at a premium in a saturated network. It

then copies the packet data, which encounters the limited bandwidth of the DMA interface to the node's memory system. The SCI broadcasting protocol only suffers these penalties once.

6.1.4 Experiment 4: multiple queue sections: We have mentioned queue contention several times. We were interested to know if it could be reduced by adding more space to the queues. To answer that question we ran experiments measuring *IP Delay* vs. *meanCPUtime* and *msgLen*, this time using SCI nodes with space for 1, 2, 4, and 8 data packets in each queue. The results for the 64 node ring and the 8×8 torus are shown in Figure 9.

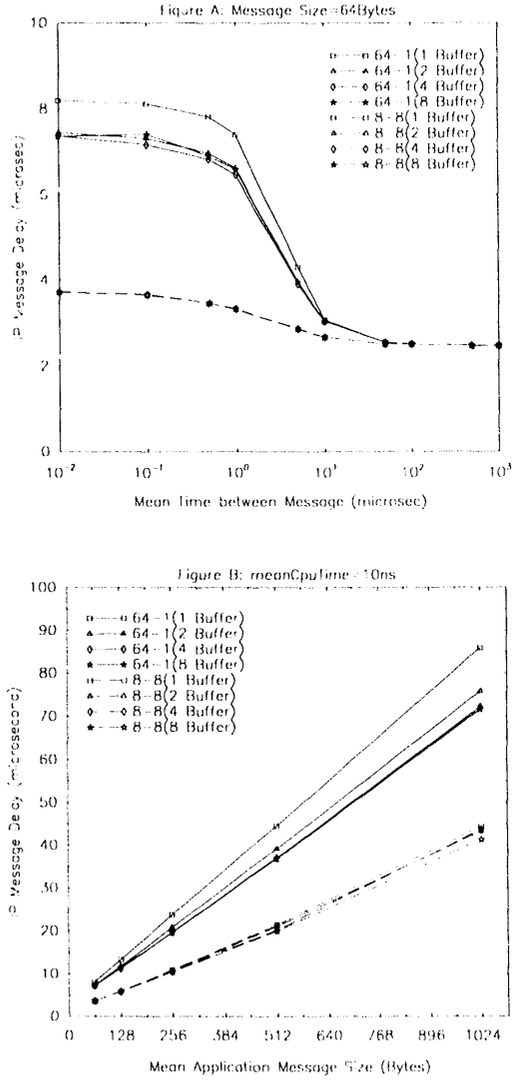


Figure 9: Effects of Multiple Buffers

The figures reveal that the extra queue sections are of use in large ring topologies, improving saturated network performance somewhat, but they are not much help to the tori. This was contrary to our intuition. We thought that extra queue spaces would be of special benefit to the switches since they have to have space for their own packets, as well as those packets crossing to the other ringlet. A partial explanation is that the switches already have twice as many queues as the simple ring interfaces. We are conducting additional experiments to understand this effect completely.

6.1.5 Experiment 5: effect of DMA bandwidth: Up to now we have been assuming that memory system has a bandwidth of 100 MB/sec. This experiment modelled the two 64 node topologies using DMA bandwidths from 100 MB/sec to 1 GB/sec, while the network was loaded by generating messages at intervals ranging from 10 ns to 1 ms. Figure 10A shows the results for the ring, while the torus results are in Figure 10B.

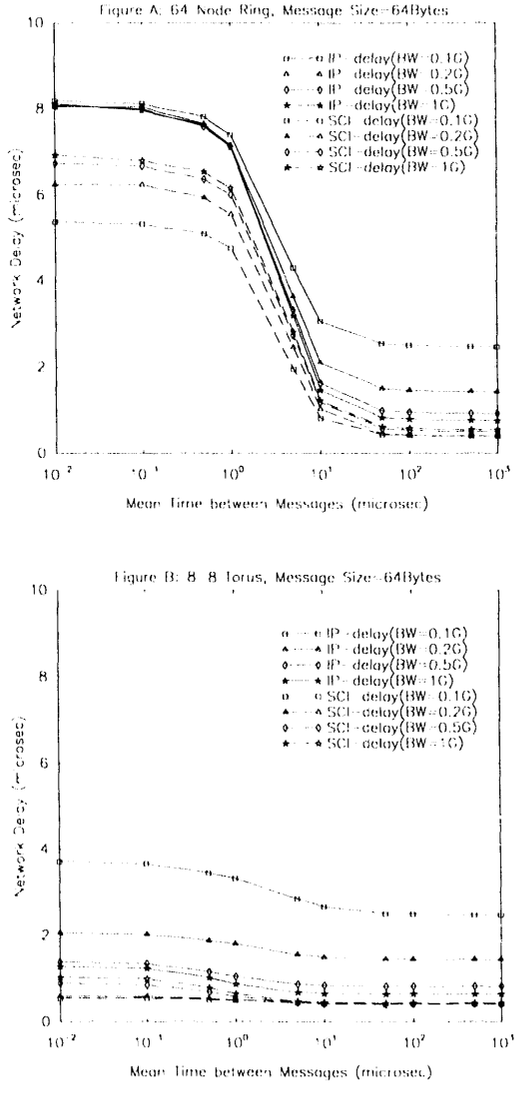


Figure 10: Ring and Torus Latencies vs. DMA Bandwidth

Note that for the saturated ring topology, the interface bandwidth has no impact on the *IP Delay*. However it is not too difficult to explain this phenomena. On the lightly loaded ring, the *SCI Delay* is essentially independent of the interface bandwidth, as it should be. Therefore, the *IP Delay* improves in line with the interface bandwidth, since packets can be sourced and sunk faster. However, as the network saturates, *SCI Delay* degrades as the interface bandwidth increases! Since filling and draining the queues takes less time, the % time they are full increases, increasing the number of rejected packets. It also causes more symbols to accumulate in the bypass FIFO. The degradation in *SCI Delay* happens to balance the improvement in *IP Delay*, thus we see the lack of effect when the ring network is saturated. For the torus the *SCI Delay* does not degrade as much, so not all of the benefit to *IP Delay* is lost.

6.2 ClusterNode behaviour: stochastic simula-tion results

Now that we have characterised the communication behaviour of the SCI ring and torus topologies, we wish to examine the communication performance of various ClusterNode machines.

6.2.1 Experiment 6: cluster size and number tradeoffs: Our first experiment is to examine, for a fixed number of nodes, the tradeoff between a small number of larger clusters and a large number of smaller clusters. In this experiment we varied the *meanCPUtime* in order to test the network under a variety of loads. Both the ring and torus topologies were tested. The ring results are shown in Figure 11A, while the torus results are in Figure 11B. For the ring we see that the performance differences are not very large when the network is lightly loaded. As the network saturates, the larger clusters benefit, since they send less traffic over the congested SCI network. They also benefit from the reduced diameter of the network. The tori do not benefit as much since their diameter is already small. They are penalized because the small SCI network is easily saturated due to the dimension-crossing demands, the smaller aggregate bandwidth, and the emergence of the shared-traffic problem.

6.2.2 Experiment 7: effect of communication locality: In the preceding simulations each processor had an equal probability of being the destination of a message. However, the ClusterNode architecture is intended to take advantage of locality in the message traffic. Our next set of experiments introduces local communication to compare clustersizes. Figures 12 A and B show the results of those experiments on 256 node topologies for saturated and lightly loaded network conditions, respectively. The Y-axis is the mean *Node Delay*. Note that the scale differs between the two charts. The X-axis represents the range of locality. For example, if the range is (-8..8), processor *j* only sends messages to processors whose IDs are in the range (*j*-8..*j*+8). Thus the left end of the chart is nearest neighbour communication and the right end is random communication.Larger clusters do exploit communication locality, but the effect is limited. Most of the benefit is not because of locality, but because of the diameter reduction explained in the previous experiment. Although the large clusters only show significant benefits with highly localised communication, we believe there are real world applications which exhibit such behaviour.

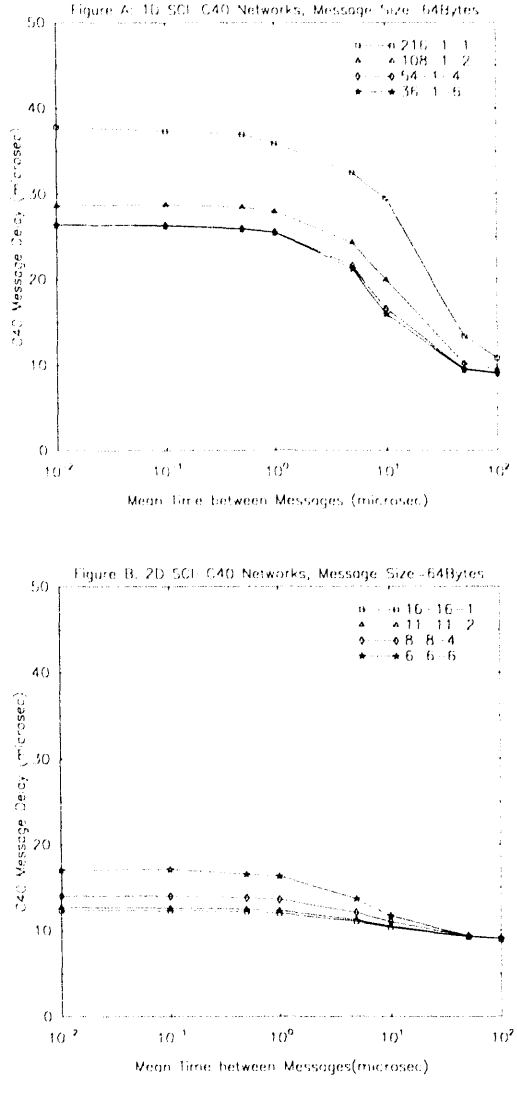


Figure 11: Cluster Size Tradeoffs

6.3 ClusterNode behaviour: trace-driven results

Finally, we wish to compare ClusterNode machines using the traces from real neural network programs. This work is underway, but we can report our results for one application on the ring topologies. The benchmark which generated the traces was an implementation of the widely used backpropagation (BP) learning algorithm [10].

The BP algorithm exhibits two kinds of parallelism. The first, *unit-parallelism*, is because all the units in one layer can compute their outputs simultaneously. The second, *training-set parallelism* is because different patterns in the training set can be processed independently, then the weight changes pooled at the end of a pass through the data [11]. Our program takes advantage of both types of parallelism. We make 6 copies of the network in order to take advantage of training-set parallelism. Each copy is implemented on 6 processors using a unit-parallel decomposition. We consider two ClusterNode configurations in this experiment: 36-1-1 and 6-1-6.

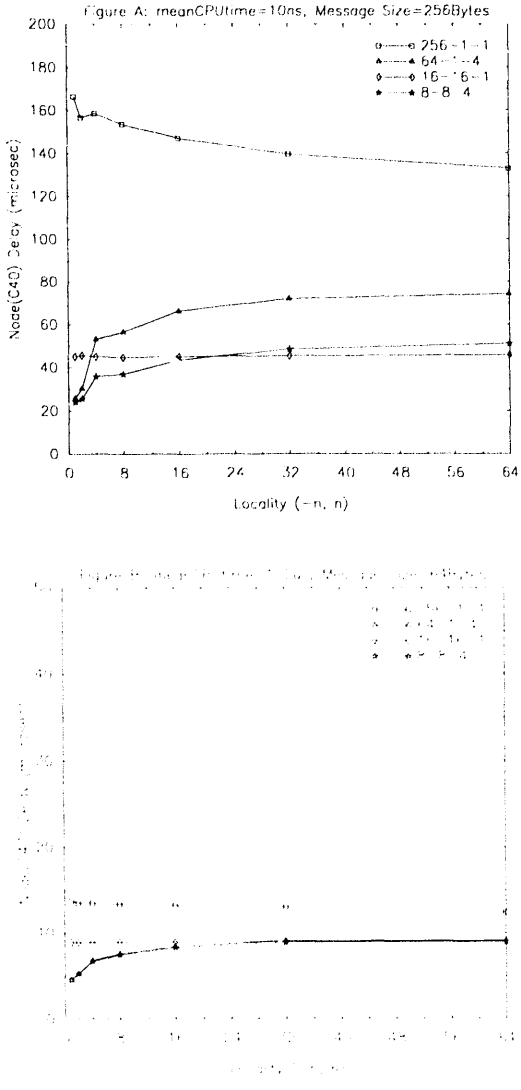


Figure 12: Communication Locality Effect

An important factor is the mapping of processes to processors. Up to now we have not had to consider this since the communication was so uniform. This is not the case in the benchmark programs. For the 36-1-1 topology we use a simple mapping where the 6 processes making up a network copy are assigned to adjacent processors. This is denoted as mapping A. For the 6-1-6 topology we use 2 mappings. Mapping B assigns all the processes for a network copy to the same cluster. This is the intuitive mapping to make since we expect many more messages within a network copy than we do between copies. Mapping C breaks this locality, and puts the corresponding process of all the copies into the same cluster. This makes the weight pooling communication local to a cluster, but at the cost of global communication during the forward and backward passes.

The results are shown in Figure 13. We see the surprising result that mapping C outperforms mapping B, with mapping A falling between the two. The reason for this is that while there are many messages sent during the forward and backward passes, they are small. The messages needed to pool the weights are much larger, and saturate the SCI network in mapping B. This behaviour is also dependent on the small size of the training dataset.

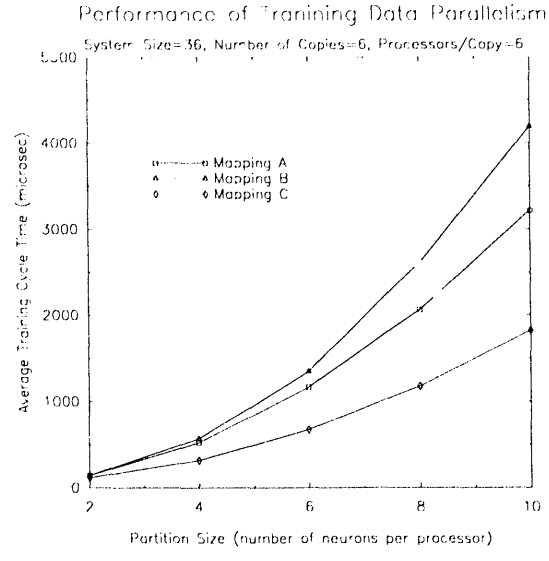


Figure 13: Backpropagation Network Results

7 Summary and conclusions

This paper set forth the goal of the *ex*-static project and described the simulators we developed to achieve it. The new high speed interconnect standard, SCI, was described along with our simple design for an SCI switch. We also described the ClusterNode architectural concept and the specific machines we are considering based on DSP chips and SCI. The bulk of the paper reports our results from modelling SCI rings and tori using stochastic simulation, and our initial results using trace-driven simulation.

As we would expect, a torus uniformly outperformed a ring with an equal number of nodes. This is due to its $O(\sqrt{N})$ advantage in diameter, its greater number of links

and internal queues, and the reduction in shared traffic. However, this does not mean that SCI rings perform poorly. For moderate numbers of nodes, a simple SCI ring still has impressive message passing performance compared to current interconnection schemes. This is due to its high bandwidth and virtual cut-through routing when not saturated. We should also note that the ring interfaces will have a lower cost than the switches since they are simpler and will be more common. We believe that multi-dimensional SCI networks are appropriate for large machines, while the simple ring is appropriate for small to medium-sized machines.

In our other experiments we found that SCI's broadcasting protocol performs significantly better than simply sending multiple copies of a message, especially when the network is saturated. We attribute this to lower contention for output queue space. Increasing the number of queue sections can help large rings, although it seems to have no significant effect on the tori. Even for large rings there seems little point in having more than 2, or at most 4, sections. Increasing the bandwidth of the DMA interface helps the tori and lightly-loaded rings, but contributes to saturation on large rings so has no net benefit there.

For ClusterNode machines, there is a definite benefit in a larger cluster size on 1D networks, most of which is obtained in the transition from 1 to 2 nodes/cluster. We attribute this to reducing the diameter of the SCI ring, and to delaying SCI saturation by reducing the number of packets sent over the ring. The benefits of increasing the cluster size are eventually balanced by the SCI-IP becoming busy doing the conversion between C40 message formats and SCI packets. In contrast, larger clusters can degrade the performance of the tori. For a fixed number of nodes, the aggregate SCI network bandwidth of a machine with small clusters is much larger than a machine with large clusters due to the increased number of SCI rings. This helps the saturated network traffic. It is important to note that this performance benefit is not without cost. Although the 16-16-1 network, for instance, performs slightly better than the 6-6-6 network, it requires 256 switches instead of 36. This suggests that even for 2D SCI networks the ClusterNode concept is much more cost-effective and perhaps a practical approach towards building massively parallel machines. ClusterNodes reap additional benefits when there is a very high degree of locality to the communications, which is often true in real world applications.

In a quest for greater accuracy, our stochastic simulator uses the closed system model instead of the analytically tractable open model. An analytical model for a single SCI ring, using the open process model, was developed in an earlier paper [7]. The authors of that paper acknowledged the limitations of such a model when considering networks near saturation, but assumed that saturation would be a rare occurance. For cache-coherent machines with a good hit rate, where messages are small (64 byte cache lines) and randomly generated, the assumption is reasonable. However, our results with long messages make us question the validity of this assumption for message passing machines. The "bursty" nature of long messages can drive the network to saturation, even though the average load is small.

Acknowledgements

The authors would like to thank Richard Prager and Jon Shapiro for their help, and the UK Science and Engineering Research Council for their financial support (SERC Grant number GR/F98758). Finally, we would like to dedicate this paper to the memories of Peter Jones and Frank Fallside, who both passed away during the course of this research.

References

[1] Daniel, R., Cha, H., Prager, R., Jones, P., Knowles, A., Shapiro, J., Fallside, F. and Marsland, T., `Simulation of Parallel Architectures for Neural Networks: the *ex*-Static Project', Workshop on Abstract Machine Models for Highly Parallel Computers, Leeds, March 1991

[2] Shapiro, J., Cha, H. and Daniel, R., `Parallel Machine Simulation for the Design of Architectures for Neural Networks', Proceedings on the Third Workshop on Parallel and Distributed Processing, April 1991, Sofia, Bulgaria

[3] Daniel, R., Cha, H., Prager, R., Knowles, A., Shapiro, J., and Fallside, F., `High Level Modelling of Parallel Computers for Conducting Neural Network Simulations: Interim Results of the *ex*-static Project', Joint Framework for Information Technology Technical Conference Digest, Univ. of Keele, March 1993

[4] Texas Instruments, TMS320C4x User's Guide, May 1991

[5] IEEE, "SCI: Scalable Coherent Interface", IEEE Approved Standard IEEE 1596-1992, April 1992

[6] Gustavson, D.B., `The Scalable Coherent Interface and Related Standards Projects', IEEE Micro, Vol.12, No.1, February 1992, pp.10-22

[7] Scott, S.L., Goodman, J.R. and Vernon, M.K., `Performance of the SCI Ring', Proceedings of the 19th Annual International Symposium on Computer Architecture, Australia, 1992, pp.403-414

[8] Knowles, A.E., `Parsifal - a parallel simulation facility based on the transputer', Proceedings of the Workshop on Parallel Distributed Processing, Sofia, Bulgaria, 1989. Bulgarian Academy of Sciences.

[9] Cha, H., Daniel, R., Knowles, A., Prager, R., Shapiro, J., and Fallside, F., `The *ex*-Static Project: Month 24 Status Report', Internal Research Report, June, 1992

[10] Rumelhart, D.E., Hinton G.E. and Williams, R.J., `Learning internal representations by error propagation', Parallel Distributed Processing, Vol.1, MIT Press, 1986, pp.318-362

[11] Singer, A., `Implementation of artificial neural networks on the Connection Machine' Parallel Computing, Vol.14, 1990, pp.305-315

DATE
FILMED
11/10/93
END