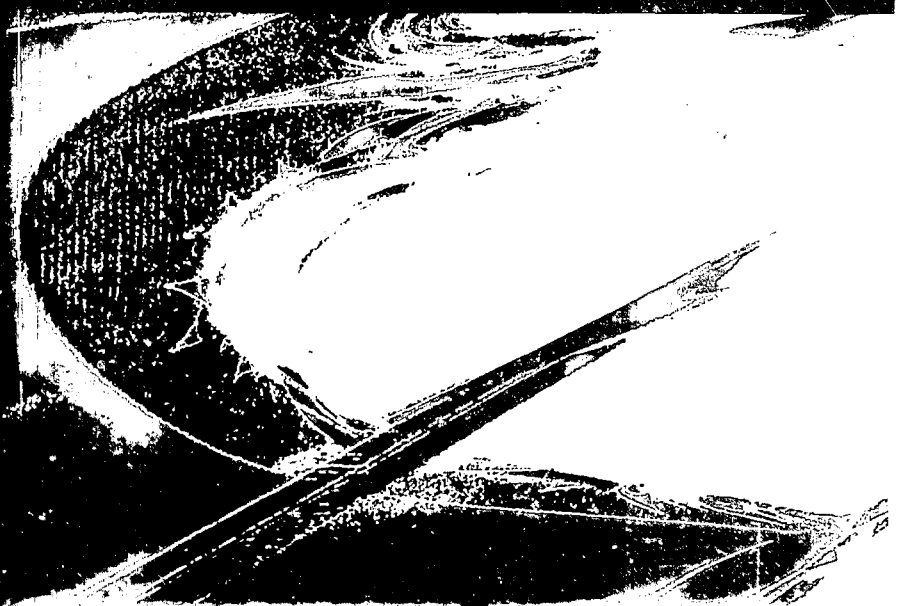
UNIVERSITY OF OSLO

DEPARTMENT OF PHYSICS

REPORT SERIES



A Study of Routing Algorithms for SCI-based Multistage Networks

Bin Wu[1], Andre Bogaerts[2],
Ernst Kristiansen[1,3], Bernhard Skaali[1]

[1]Department of Physics
University of Oslo, Box 1048
N-0316 Oslo 3, Norway

[2]CERN
1211 Geneva 23, Switzerland

[3]SINTEF/SI
Oslo, Norway

UIO/PHYS/94-06
ISSN-0332-5571

Received: 1994-03-07

A Study of Routing Algorithms for SCI-based Multistage Networks

Bin Wu[1], Andre Bogaerts[2], Ernst Kristiansen[1, 3], Bernhard Skaali[1]

[1] Dept. of Physics, Univ. of Oslo
P.O.Box 1048, Blindern
0316 Oslo, Norway

[2]CERN
1211 Geneva 23, Switzerland

[3] SINTEF/SI
Oslo, Norway

Abstract

The IEEE std. 1596-1992 the Scalable Coherent Interface (SCI) specifies a topology-independent communication protocol with the possibility of connecting up to 64 K nodes altogether. Among many choices of topologies, multistage SCI switching network is a candidate for large data acquisition systems in High Energy Physics. There are several possible ways of interconnection. Effectively connecting a multistage system with a large number of switch elements must focus on real hardware implementations with cost and performance as trade-offs. The routing algorithm is essential for transmitting SCI packets to their destinations. The routing decision must be fast and deadlock-free. In presence of contention, it may be possible to implement adaptive routing. There are different optimal choices of routing algorithms for different topologies. This paper will focus on a particular class of multistage network systems and two important routing algorithms, namely self-routing and table-lookup routing. The effect of routing delay on system performance is investigated by simulations. Adaptive routing and deadlock-free routing are studied.

Keywords: SCI, multistage system, performance/cost, self-routing, table-lookup routing, adaptive routing and deadlock-free routing

1. Introduction

Routing algorithms are essential for switching SCI packets in large SCI-based networks. Optimal routing algorithms may differ for different topologies. Choosing the optimal routing algorithm must consider many aspects, such as performance, cost and reliability, etc. Multistage SCI-based networks are considered for large data acquisition systems in High Energy Physics. They are similar to large crossbar switches and are relatively cheap to build. It is therefore important to study its routing algorithms.

SCI [Gust-92][SCI-92] provides a bus-like service using point-to-point links and a packet-transmission protocol. The parallel version of SCI can transfer up to 1 Gbyte/s over short distances. Different topologies are investigated to interconnect a large number of SCI nodes [JoGo-92][Wu1-93][Wu2-93]. To route an SCI packet from source to target requires a routing algorithm. SCI does not specify the routing protocols for SCI-based networks. There are many routing protocols for a class of conventional multistage networks, but few of them have been investigated to comply with SCI protocols. The congestion and deadlock issues in SCI-based multistage networks have not been addressed yet.

Many routing methods are self-routing or based on table-lookup routing. Self-routing is most suitable for symmetric, multistage interconnection networks, such as banyan and a class of topologically equivalent multistage networks. The routing is only determined by successive bits of its target address. It is fast and simple. Compared to self-routing, table-lookup routing is more general and can be used for irregular topologies. Every time a packet passes the address decoder, it consults the routing table and obtains the path. We will discuss in detail these two methods and their implementations in SCI-based multistage systems. However, none of these two algorithms are deadlock-free, so we will provide some of the potential solutions.

In the present study, self-routing and table-lookup are both implemented in an SCI system simulation environment -- SCILab [BoWu-93]. The hardware implementation is being considered.

In the following section we give a short introduction on SCI to ease understanding of rest of the paper. We also present multistage networks using SCI in this section. Section 3 illustrates how self-routing and table-lookup can be implemented in SCI-based multistage networks. In this paper, we will focus on the banyan-like SCI networks, though the methods and discussions are not restricted to that kind of networks only. Simulation results are shown to see the effect of routing delay on system performance. Adaptive routing is discussed in section 4. Section 5 describes the deadlock problem and provides several solutions. Finally, section 6 contains a short summary and conclusions.

2. SCI and SCI-based multistage networks

The basic building block of an SCI network is a ring of two or more nodes connected by unidirectional links. An SCI node would have the block diagram as shown in figure 1. A two-node SCI ring is also shown in the figure.

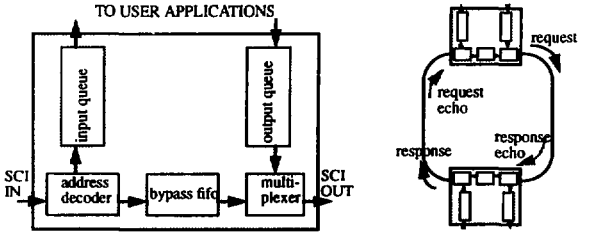


FIGURE 1. A block diagram of an SCI node and a two-node SCI ring

SCI uses packet-transmission protocol. Most SCI transactions are split in two subactions, a request and a response packet. The exception is the move packet which is without a response subaction. A packet contains target address (16 bits of nodeId which results in maximum 64 K nodes in an SCI system, plus 48 bits of internal address), command, source address (16 bits) and control bits followed by 0, 16, 64 or 256 bytes of data and a CRC checksum. New send packets are inserted by a sending node from its output queue subject to the handwidth

allocation protocol in SCI. The send packets are stripped by the target node and an echo is returned to tell the source node whether the send packet was accepted by target's input queue, or it was rejected due to queue overflow or other errors. A retransmission will happen if the sending was unsuccessful.

A single ring system with many SCI nodes will have limited performance. It is therefore necessary to connect many rings through SCI switches to achieve high throughput and low latency. Designs of SCI switch elements are discussed in [Wu1-93][Wu2-93]. The block diagram of one of the switch ports is shown in figure 2. We expect that the SCI switch ports would have minor difference from an SCI node. But no cache coherent logic is needed. Several switch ports are connected back-to-back with possible interconnections as bus, ring, or crossbar. Echoes are local to its corresponding ring. The routing info decoder would be more complicated than the address decoder in a normal node since it has to decode a range of address, and probably uses other informations other than packet's target address.

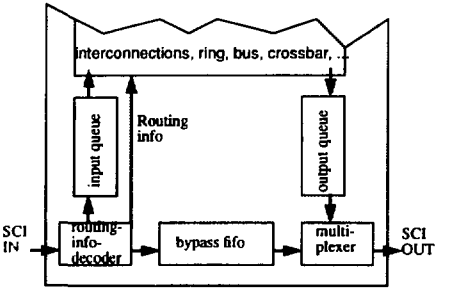


FIGURE 2. A block diagram of an SCI switch port

Multistage networks are extensively used network topologies. It has recently been studied for use in SCI-based large data acquisition systems [Wu2-93]. A multistage network built of SCI switches must have rings as basic elements as mentioned before. It is imposed by SCI protocol as SCI requires request-response-loops for transmitting and receiving packets. It is possible to construct both bidirectional (figure 3.a) and unidirectional (figure 3.b) multistage systems. The

bidirectional structure provides richer topologies and is therefore easier for adaptive routing. When the traffic is totally random, it doubles the throughput. It has lower latency too, because for unidirectional networks, echoes for the send packets will normally have to go through a longer path before being routed back to recognize the success of the sending. However, it requires larger switching elements. When the traffic is bidirectional "move" (responseless) packets, it does not efficiently use all the links.

Both these interconnection topologies support self-routing. We will investigate in the bidirectional multistage network in this paper.

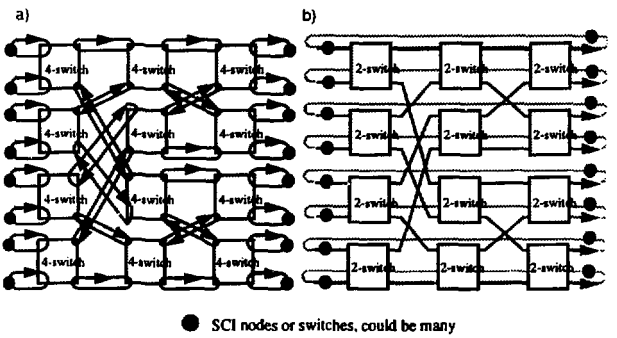


FIGURE 3. a) A $8_R x 8_R$ bidirectional multistage system interconnected by 4-switches; b) A 8x8 unidirectional multistage system connected by 2-switches.

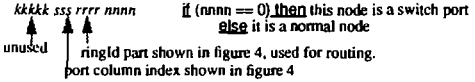
3. Self-routing and table lookup routing

3.1 Using self-routing in SCI-based multistage networks

Self-routing [WuFe-80][NaSa-81], also called digitally controlled routing, is most suitable for symmetric, multistage interconnection networks, such as banyan and a class of topologically equivalent multistage networks.

The main features of the self-routing algorithm are its small routing latency and simplicity of hardware implementation. The key property of the self-routing algorithm is that the routing is only determined by a bit sequence in its target address. An example of a multistage network built of 4-switch[1] elements is shown in figure 4. Here the routing is only determined by a single bit in the target address. When implementing this simple self-routing scheme, one must guarantee forward progress of packets being transmitted and pay attention to the assignment of nodeIds during the initialization. The first condition implies that the packet will flow from one side of network to the other. The packet cannot go back traverse the same stage twice. The routing between nodes on the same side is not so trivial but is possible. We will discuss it later.

The example in figure 4 shows how the self-routing algorithm works. One has to initialize the nodes in a special way as marked in the figure to comply with the address partition below. The address shown in the figures is only a portion of the 16-bit SCI nodeId. If we limit the number of nodes and switch ports on each edge ring to 16, the nodeIds have the following format:



The switch ports that are not on the edge rings can be assigned any nodeIds since they are irrelevant for the routing decision, but they do influence the decision by guarantee the forwardness of routing. They should also be chosen to ease management of the system. In the example we use 3 bits $(sss)_2$ for this purpose, but also potentially waste some address range.

Each bit of ringId $(rrrr)_2$ decides how to route a packet in the corresponding stage. Suppose a processor node $(00000\ 000\ 0011\ 0001)_2$ sends a packet to target $(00000\ 111\ 1001\ 0010)_2$, the packet will be accepted by the switch port $(00000\ 000\ 0011\ 0000)_2$ and routed to the lower port $(00000\ 001\ 0011\ 0000)_2$ on the other side of the $2_R x 2_R$ switch, because the target's ringId $(1001)_2$ starts with "1" from left. A "1" will cause packets go to the lower port and a "0" will cause packets go to the upper port. No packet will be routed to the ports on the same side in

1. A 4-switch is also called $2_R x 2_R$. It is different from conventional multistage nxn, which is unidirectional. The small R in subscript indicates the links are bidirectional, more precisely, in a ring.

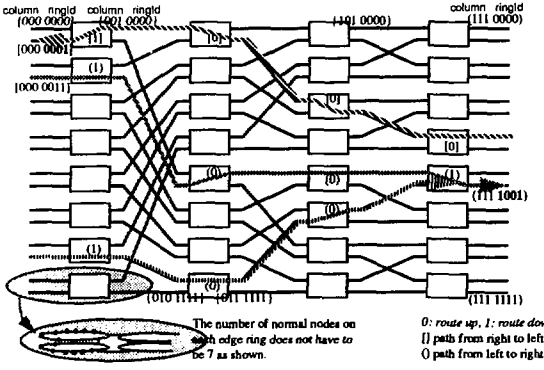


FIGURE 4. Using self-routing in a $16_R x 16_R$ bidirectional multistage SCI network

this example. In the second stage, packets will be routed upper based on the "0" of $(1001)_2$ and third upper, fourth down. At last it reaches $(00000\ 111\ 1001\ 0000)_2$ and continues on the target ring to its final destination $(00000\ 111\ 1001\ 0010)_2$. Echoes will be generated locally in each stage. The response packet is generated once the request is in its target node. The way of routing the response packet is the same. In this example, only one bit corresponds to the current stage of a packet determines how to route.

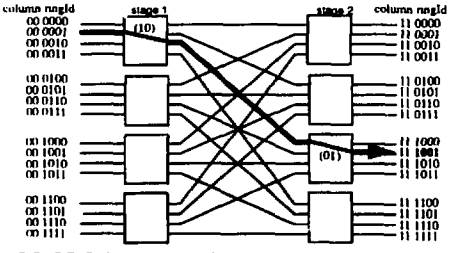


FIGURE 5 A binary $16_R x 16_R$ network formed by $4_R x 4_R$ switches

A $16_R x 16_R$ SCI-based multistage network can be also constructed by 8-switch ($4_R x 4_R$) elements in 2 stages as in shown figure 5. The assignment of nodeIds will be slightly different. Since only 2 stages are needed, we use 2 bits for column index. Choosing one of the 4 ports on the opposite side of a switch element is decided by two corresponding bits in ringId. If a packet from the left side of the system has its target's ringId as $(1001)_2$, the left two bits $(10)_2$ is used as routing information for the first stage and the remaining $(01)_2$ is used in the second stage.

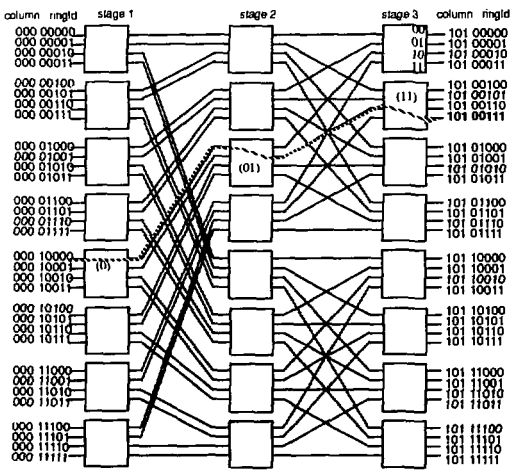


FIGURE 6. A binary $32_R x 32_R$ network formed by $4_R x 4_R$ switches

A $32_R x 32_R$ multistage network interconnected by $4_R x 4_R$ switches will have redundant links[1]. Such a network is not an optimized *multistage network*. We will discuss the optimized *multistage network* later. A proposal for connection is to group every two links as shown in figure 6. The routing in stage one is decided by only one bit. Two parallel links instead of one reduces contention on the output ports in stage one. Dynamic adaptive routing can also be

1. A link is understood to be a ring providing a bidirectional connection

implemented easily. A random or round-robin scheme could be used to choose one of the two links. The grouping can also be done in the links between 2nd and 3rd stage, but the routing will be different.

Building switches with 16, 32 or even more ports is a challenge to chip manufactures. Higher integration makes it obvious easily to construct large systems.

3.2 Table-lookup routing

While self-routing is very easy to implement on multistage-based networks, it is not true for some other networks, in particular, irregular ones. Self-routing is also inflexible because it is topology-dependent. Table-lookup routing provides another approach for routing. Every time a packet passes the routing-info-decoder in a switch port, its target address is checked against a routing table to determine the path. This algorithm is complete general.

In reality, to avoid contention and to get fast access, routing tables need to be distributed over each switch port, or at least distributed over each switch element. One implementation is to have a lookup table at each switch port with one entry per nodeId. Once an incoming packet's target address is decoded, the corresponding entry in the table will indicate which port the packet should be routed to. This method requires large tables. This can be reduced by grouping a range of nodeIds into a simple table entry. For instance, one could assume that nodeIds are grouped into rings and base the routing decision on the ringId.

A routing table should be able to perform a translation from a given address to: 1) an "is mine" decision to indicate that the packet is to stay on the same ring or be routed to far ports; 2) a bit pattern output to determine which of the port the packet should be route to. Both decisions have to be fast. For an echo packet, routing to a far port in the first decision means it reaches its destination. The second decision is not necessary since echoes are always local to a ring.

As an example, we give a routing table with one entry per ring, and we assume that 1) 4-switches are used; 2) There are 256 nodes in the system; 3) Each ring will have a maximum of 16 nodes (switch ports or normal nodes) so the least significant 4 address bits are dedicated to the address inside the ring and the 4 upper address bits are the index of the routing table.

Table 1. An Example of Table-lookup Routing [Wu1-93]

4 most significant address	4 least significant address	Output of Address Decoder: "is mine" decision (one bit)	Output of Address Decoder: Which of the 4 four ports to go (four bits)
0000	xxxx	0 (keep on the same ring)	xxxx (don't care)
0001	xxxx	1 (far ports)	0001 (to far port 1)
0010	xxxx	1 (far ports)	0010 (to far port 2)
0011	xxxx	1 (far ports)	0100 (to far port 3)
0100	xxxx(except 0101)	0 (keep on the same ring)	xxxx (don't care)
0101	xxxx	1 (far ports)	0001 (to far port 1)
...	...	...	...
1111	xxxx	1 (far ports)	0100 (to far port 3)
0100	0101	1 (local CSR access)	1000 (to local port)

The example in table 1 shows the routing table for an SCI-switch port with nodeId = $(0100\ 0101)_2$. By using the 4 most significant bits as the index for entry and not considering the least 4 bits, we make a table of 16x5 bits in this case, which is quite small. This example does not intend to show how small the table is, which is not always true, but to show how we can implement a table. In order to save one bit in the table, we can use some logic in detecting the accessing of the local Control and Status Registers (CSR), so that the corresponding bit for accessing local CSR is saved, under the condition that the delay introduced by this logic is tolerable. Another way to reduce the table size is to use 2 bits instead of 4, to determine which of the output ports the packet should be routed to.

3.3 Optimized multistage network

A multistage network is called optimized formed when one uses minimum number of switches. Obviously, there are no redundant links.

Only bidirectional multistage networks are considered here. For the size of the basic SCI switches, one would normally choose a different number of ports in powers of two, such as 4-switch, 8-switch, 16-switch, or 32-switch. It is totally technology dependent. Other port numbers would theoretically be possible, but their characteristics are less studied before. 2-switch is only considered for building of unidirectional multistage networks like the one in figure 3.b

The optimized network size depends on the size of switch elements. For an optimized $M_R x M_R$ multistage network built of $N_R x N_R$ elementary switch chips, $S = \log_N M$, where S is the number of stages and it must be an integer.

The example in figure 6 is not an optimized configuration. In the example, one wants to use 8-switches ($4_R x 4_R$) to form a $32_R x 32_R$ multistage network, and $\log_4 32 = 2.5$ is not an integer.

An optimized multistage network makes full utilization of switches and thus minimizes the cost.

3.4 Full Communication

A network is said to provide "Full Communication" if every node can communicate with every other node in the network. This indicates that the normal nodes on the same side of the network can communicate with each other as well as with those on the opposite sides. The SCI switch itself is capable of doing so since it is a crossbar switch. In order to route packets between the nodes on the same side, one has to decide the "turning-point". A packet will be routed towards the opposite side of the network till it reaches the "turning-point" where it turns back to its target, as shown in the two examples in figure 7. The routing of a packet before it reaches its "turning-point" is irrelevant. Only after the packet has reached the "turning-point", the packet's target address will be used as routing information, and self-routing can then be used. The following theorem specifies how to find such a "turning-point".

THEOREM: Assume that the two nodes S = s(m)s(m-1)...s(0) and T = t(m)t(m-1)...t(0) are on the same side. T stands for target and S for source. Define C = c(m)c(m-1)...c(0), where c(k) = s(k) XOR t(k) for all m >= k >= 0; Exactly 2^i possible shortest paths are available to connect S and T, and the number of links in the shortest path is exactly 2*(i+1), where i is obtained from C = c(m)c(m-1)...c(i)...c(j)...c(0), c(i)=1 and c(j)=0 for all j < i. i is also the number of stages a packet should pass before it reaches its "turning-point"

Doing such a calculation in hardware might not be easy. Another solution is to have fixed "turning-points" at the other side of the network, so that no calculation is needed. This scheme

will sacrifice bandwidth and cause longer latency. Allowing "Full Communication" will cause potential deadlock, see more discussion in section 5.

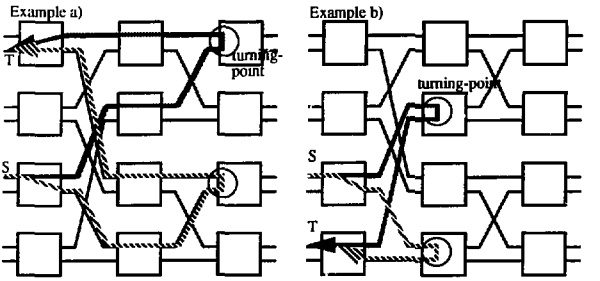


FIGURE 7. Packets route between nodes on the same side of the network. *More than one path could be find, adaptive routing is possible.*

3.5 The effect of routing delay on the system performance

Routing delay is the time that is consumed by the routing-info-decoder in a switch port to make a routing decision. Different routing algorithms will introduce various routing delays. For example, using self-routing algorithm will normally not need more than 10 cycles whilst table-lookup might take several hundred cycles, depends on technology. We did some simulations by SCILab to show the effect of routing delay on the system performance. The throughput and latency of the performance matrics are of primary concern in this paper. The throughput shown in figure 8 is the total system bandwidth of packets that sources send to targets. The latency of a send packet is measured from the entry of the packet into source's output queue until it is accepted in its target's input queue. The average latency of all packets is shown in figure 9. The simulations are based on figure 3.a.

Each normal node has both a sender model (i.e. processor) and a receiver model (i.e. memory, cache, etc.). Though traffic is bidirectional dmove64 (move 64 bytes) packets, it is not allowed to send packets between nodes on the same side. The processor model keeps sending packets to randomly chosen targets at certain rate. The average interval between two successive

packets being generated are 100, 150, 200, 300, 400 and 800 ns respectively, as shown in the figures. It is the rate each processor tries to issue, not the rate it succeeds to send.

The simulation parameters are very optimistic. The SCI links run at 1 Gbyte/s, the time needed to go through a bypass fifo plus the cable delay is 15 ns, and the transmission through the internal crossbar-like interconnection takes 10 ns. Packets are routed in virtual cut-through technique. Four-packet-deep queues are used for both input and output queues. We simulate *only one "ultra" fast normal node on each "edge"-ring instead of many slow nodes.* This saved us CPU resources.

We varied the routing delay on switch ports to see its effect on the system throughput and latency. We found that for a lightly loaded system, system throughput is insensitive to routing delay and for a heavily load system, system throughput decreases sharply when routing delay is longer than ca. 100 ns; the higher the load the more effect the routing delay has on the system throughput. The reason is that it takes longer time for an echo to come back and acknowledge the sending when the routing delay is longer. The consequence of such a late echo makes the copy of the send packet stay longer in the queue and thus blocks other packets.

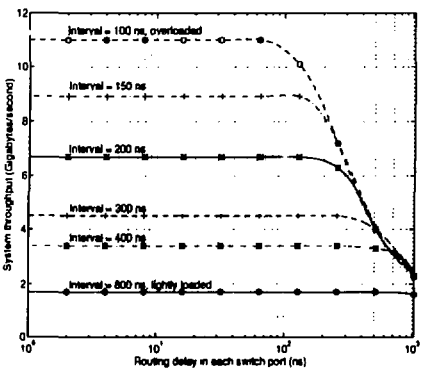


FIGURE 8. The effect of routing delay on system throughput

Latency increases linearly for lightly loaded systems. This is a good feature of multistage networks. When system becomes heavily loaded, latency starts to increase nonlinearly for long routing delay. This indicates that long routing delay will make congestion in heavily loaded systems even worse, mainly due to the retry packets also have to suffer a long routing delay. We also found that overloaded system suffers a much higher latency due to the congestion in the system.

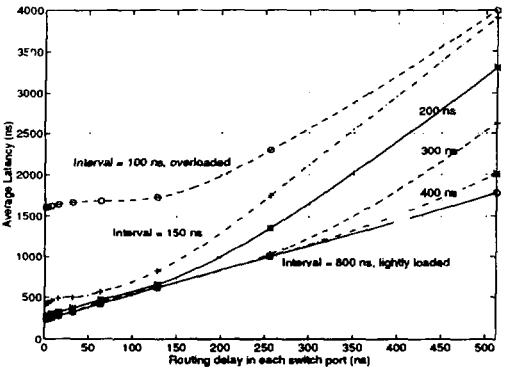


FIGURE 9. The effect of routing delay on system average latency

4. Adaptive routing

Deterministic routing results in path uniqueness, i.e. a unique closed loop is set up for each requester and responder pair. While simplifying routing decisions, it also implies a potential problem, switch blocking. Switch blocking is caused by different packets which are simultaneously routed to the same switch ports. It is obvious that, when the congestion is serious, even adding output buffers at the switch ports will not alleviate the situation, since due to the heavy oversubscription at the bottleneck, the buffers will eventually be filled up.

Adaptive routing, in contrast to deterministic routing, can route packets according to dynamic network conditions, such as congestion. As a result, if there is contention for an output channel, some of the packets are re-routed on purpose and have to travel an additional distance to the destination. Thus, in adaptive routing, some bandwidth is traded off for fast switching capabilities and storage requirements at the ports. However, one sometimes will be in a lucky situation where most of the bandwidth being traded is "supposed" to be wasted.

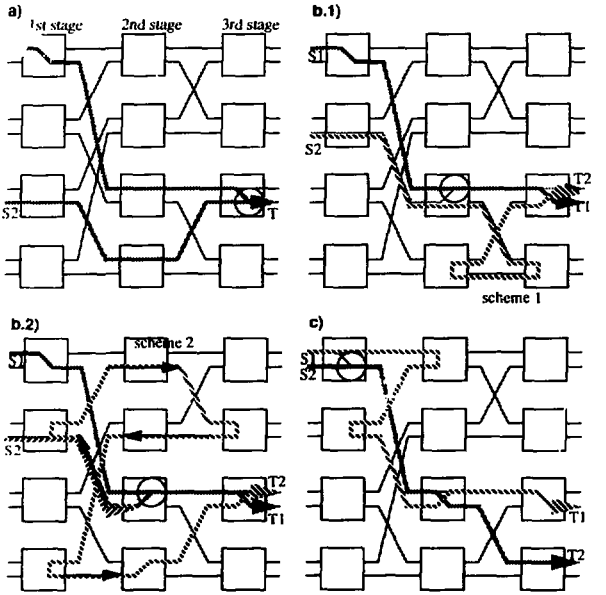
An adaptive routing scheme for a 4-switch-based multistage network is "forth-routing": when two packets from two different ports are routed to a third port simultaneously, one of them (maybe the one with lower priority) is supposed to go one stage further through the fourth port and turn one stage back to try with another path, see figure 10.b.1 and 10.c.

The second adaptive routing scheme is "back-routing": when two packets from two different ports are routed to a third port simultaneously, one of them (maybe the one with lower priority) is supposed to route back to the last stage through the same link (other part of the ring) it comes from and tries the other port, see figure 10.b.2. "back-routing" is more flexible than "forth-routing" but also has longer latency. It is more complex for each switch port to make a routing decision, especially on how to avoid livelock. For "forth-routing" this is not a problem.

If the blocking happens at the first stage as in figure 10.c, "forth-routing" is the only scheme we could use; if the blocking happens at the last stage, no adaptive routing is necessary since the only path to the target is blocked; For those packets that are blocked in the middle stages (figure 10.b.1 and 10.b.2), both "forth-routing" and "back-routing" are feasible.

One of the issues must be considered in adaptive routing is that, if right after a packet performed an adaptive routing decision, the minimal path it should go to is free, so the packet did a non-optimal choice. How often does this happen and how can we avoid this is not a trivial task. One possible way is to use registers to note how many conflicts happened and decide under which bunch to do adaptive routing. Weight scheme could be another choice.

Which of the two conflict packets should be misrouted is another important issue. Round-robin, random or priority-based algorithms can be used.



a) No alternative possibility here, when contention happens at last stage;
b) When contention is in second stage, packet can be routed to the third stage and then turns back to the second to continue as b.1, or goes first->third->first stage to continue, as in b.2;
c) When contention is in the first stage, the minimum adaptive routing is to go to the uncongested second stage and turn back to the first stage to continue.

FIGURE 10. Several examples of congestion and adaptive routing algorithm

In general, adaptive routing is useful when the traffic in the system is not well balanced or/and when the system is defected. The defection could be a link breaks, or a node collapses. However, adaptive routing is very suspicious to deadlock and when the system becomes large, the situation will be difficult to predict.

5. Deadlock free routing algorithm

Deadlock, without considering adaptive routing, is itself a serious issue and must be addressed even for static (deterministic) routing. Deadlock occurs when one packet holds some resources while requesting others. The resources here could be SCI queues. In SCI when it is a single ring system, the deadlock is solved by using two separate queues, one for request packets and one for response packets. The deadlock may also occur when a packet holds a resource and waits for a signal to release the resource, while the signal never comes. The reason could be the packet is lost or destroyed somewhere or it is unpredictably misrouted. In SCI, time-out scheme is used to solve such deadlocks [SCI-92].

However, SCI does not guarantee deadlock-free when an SCI-based system is large and interconnected in complex topologies. The multistage SCI network is not deadlock-free if traffic among the nodes on the same side of the system is allowed. Suppose 4 processor nodes send four packets marked with A, B, C, D respectively as in figure 11. When the situation is like.

packet A holds port2's output queue and requires port8's output queue[1];
packet B holds port8's output queue and requires port3's output queue;
packet C holds port3's output queue and requires port5's output queue;
packet D holds port5's output queue and requires port2's output queue.

A deadlock circle is created. Each packet sits in an output queue and requires another output queue that is occupied by another packet. Using deeper queues may reduce the probability of deadlock to approximately zero, but it cannot eliminate deadlock.

One way to solve this deadlock problem is to allow preemption of packets that are involved in the deadlock situation, and the preempted packets can be either re-routed or discarded. Discarding a packet needs an algorithm of retransmitting the packet again. This is difficult in SCI-based systems. Because once a packet is accepted by an output queue, an echo will be send back to the sender immediately to acknowledge the success of sending. The sender will

1. Indeed, packet A requires port7's input queue, but packets in port7's input queue can not be emptied due to port8's output queue is busy. Same for packet B, C and D.

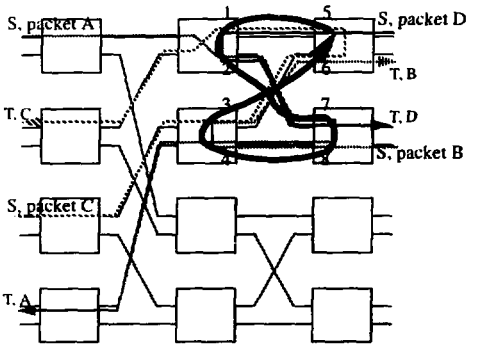


FIGURE 11. Deadlock issue in multistage SCI network

then discard the copy of the packet. To keep the copy longer in the sender or to send the acknowledgment later will need a modified protocol. This preemption protocol requires a time-out-like logic to detect the happening of deadlock. In principle, preempty one packet will solve the deadlock, so preempty all packets that involve in the deadlock situation is not needed, but how to synchronize and preempty only one packet is an issue need to be taken care of.

More realistically and effectively, deadlock could be solved by choosing suitable deadlock-free routing algorithms. A suitable routing algorithm could eliminate the circle that causes deadlock. The deadlock in figure 11 could be avoided by disabling the path along 3-6-5-1. This is the path that was used for packet C. So packet C would need to go via another path. We could easily find one and verify that the path is still minimal. The disadvantage is that we artificially unbalance the system and create potential hot spots.

When communication between nodes on the same side of an SCI-based multistage network is disallowed, there is no deadlock circle.

6. Conclusions

In this paper, connecting elementary SCI switches into a large multistage network has been introduced. The possibilities of using self-routing and table-lookup routing were investigated. The self-routing algorithm is simple and fast. It makes chip-design easy. The table-lookup routing, on the other hand, provides more flexibility in sacrifice of hardware complexity and cost. The simulation results showed clearly that for such a network with the parameters we specified in the simulations, routing delay up to 100 ns will not have much effect on system performance. It is therefore no need to make the routing decision very fast. But routing delay longer than several hundreds nanoseconds will reduce the system throughput significantly. Simulation also showed that for an heavily loaded system, the latency will not increase linearly as a function of routing delay because the contention problem gets worse when routing delay is long. For an overloaded system, latency will be much longer than expected, due to congestion in the system.

Adaptive routing can route packets according to the dynamic network conditions, such as congestion. It is especially useful when the traffic in the system is not well balanced or/and when the system is defected. Two schemes for adaptive routing in the SCI-based multistage networks were presented. The "forth-routing" is easy to manage and implement while "back-routing" is more flexible and has relatively long latency. Deadlock-free could be achieved either by preemption or more effectively, by choosing suitable deadlock-free routing algorithm that eliminates the circle that causes deadlock. Some examples were given to illustrate the routing methods.

The way the "standard" routing algorithm of SCI is not defined yet, and probably will never be. For different kinds of switches and different network architectures, one may find different solutions. We choose multistage SCI-based system as our start point and we hope that this paper could function as a trigger for a general understanding of routing algorithm in SCI networks.

7. Acknowledgment

The authors gratefully acknowledge the help from Stein Gjessing, Håkon Bryhni and Håkon Bugge.

Bin Wu is supported by a fellowship from the Norwegian Research Council. This work was also partly supported by Thomson-TCS Semiconducteurs Specifiques, France.

8. References

[BoWu-93] *A. Bogaerts and B. Wu*, "The SCILab Cook Book", internal note, CERN, Geneva, Switzerland, July, 1993

[Gust-92] *D B Gustavson*, "The Scalable Coherent Interface and related Standards Projects", IEEE Micro, February 1992, pp. 10-22

[JoGo-92] *R.E. Johnson and J.R. Goodman*, "Synthesizing General Topologies from Rings", Proceedings of the ICPP, August, 1992

[NaSa-81] *D Nassimi, and S Sahni*, "A Self-Routing Benes Network and Parallel Permutation Algorithms", IEEE Transactions on Computers, C-30, May 1981, pp. 332-340

[SCI-92] "SCI, Scalable Coherent Interface", IEEE Std.1596-1992,

[Wu1-93] *B Wu, A Bogaerts, R Divia, E. Kristiansen, H.Muller, B. Skaali*, "Constructing Large Scale SCI-based Processing Systems by Switch Elements", technical report PHYS/93-12, Univ. of Oslo, May, 1993

[Wu2-93] *B Wu, A Bogaerts, R Divia, E Kristiansen, H Muller, E Perea, B Skaali*, "Distributed SCI-based Data Acquisition Systems constructed from SCI bridges and SCI switches", Presented the 10th International Symposium of Problems of Modular Information Systems and Networks, St.Petersburg, 14-17 Sept. 1993. Also available as technical report PHYS/94-02, Univ. of Oslo, February 1994

[WuFe-80] *C Wu and T Feng*, "On a Class of Multistage Interconnection Networks", IEEE Trans. Computers, Vol. C-29, No. 8, Aug.1980, pp. 694-702

7. Acknowledgment

The authors gratefully acknowledge the help from Stein Gjessing, Håkon Bryhni and Håkon Bugge.

Bin Wu is supported by a fellowship from the Norwegian Research Council. This work was also partly supported by Thomson-TCS Semiconducteurs Specifiques, France.

8. References

[BoWu-93] *A. Bogaerts and B. Wu*, "The SCILab Cook Book", internal note, CERN, Geneva, Switzerland, July, 1993

[Gust-92] *D.B. Gustavson*, "The Scalable Coherent Interface and related Standards Projects", IEEE Micro, February 1992, pp. 10-22

[JoGo-92] *K.E. Johnson and J.R. Goodman*, "Synthesizing General Topologies from Rings", Proceedings of the ICPP, August, 1992

[NaSa-81] *D. Nassimi and S. Sahni*, "A Self-Routing Benes Network and Parallel Permutation Algorithms", IEEE Transactions on Computers, C-30, May 1981, pp. 332-340

[SCI-92] "SCI, Scalable Coherent Interface", IEEE Std.1596-1992.

[Wu1-93] *B. Wu, A. Bogaerts, R. Divia, E. Kristiansen, H.Muller, B. Skaali*, "Constructing Large Scale SCI-based Processing Systems by Switch Elements", technical report PHYS/93-12, Univ. of Oslo, May, 1993

[Wu2-93] *B. Wu, A. Bogaerts, R. Divia, E. Kristiansen, H.Muller, E. Perea, B. Skaali*, "Distributed SCI-based Data Acquisition Systems constructed from SCI bridges and SCI switches". Presented the 10th International Symposium of Problems of Modular Information Systems and Networks, St.Petersburg, 14-17 Sept. 1993. Also available as technical report PHYS/94-02, Univ. of Oslo, February 1994

[WuFe-80] *C. Wu and T. Feng*, "On a Class of Multistage Interconnection Networks", IEEE Trans. Computers, Vol. C-29, No. 8, Aug 1980, pp. 694-702

FYSISK INSTITUTT
FORSKNINGS-GRUPPER

Biofysikk
Elektronikk
Elementærpartikkelfysikk

Faste stoffers fysikk
Kjerne- og energifysikk
Plasma- og romfysikk
Strukturfysikk
Teoretisk fysikk

DEPARTMENT OF PHYSICS
RESEARCH SECTIONS

Biophysics
Electronics
Experimental Elementary Particle Physics

Condensed Matter Physics
Nuclear and Energy Physics
Plasma and Space Physics
Structural Physics
Theoretical Physics

ISSN 0332 55?1