The SCX channel:
A new, supercomputer-class system interconnect

Steve Scott
Cray Research, Inc.
sls@cray.com

Abstract

This paper describes the SCX system interconnect and I/O channel from Cray Research. The SCX channel provides high performance, scalable, reliable inter-system and system-to-peripheral communication. It is slated for use across all Cray platforms in 1996 and beyond, and will be made available as an open standard.

The SCX channel is ring-based, and borrows heavily from the logical layer of the IEEE standard Scalable Coherent Interface [2]. The channel has been enhanced, however, to address the reliability, flexibility and performance needs of distributed supercomputer environments. The SCX channel is capable of sustaining data payload bandwidths in excess of 900 MB/s half duplex, or 2×800 MB/s full duplex.

1 Introduction

Advances in computational performance and distributed computing techniques have created the need for a new supercomputer interconnect. This interconnect must of course provide high bandwidth, but also high reliability, flexibility, and scalability of both cost and performance. Cray Research has developed the SCX channel to satisfy these demands in the 1996 time frame and beyond.

SCX is a ring-based channel that connects multiple clients together with high-speed, point-to-point links. Each client includes a single ASIC – the SCX node chip – that implements the SCX channel functionality. The node chips are connected via cables to form a ring.

Cray Research is developing a new, scalable I/O system based upon the SCX channel [3]. This I/O system eliminates the expensive active switches of previous designs. Instead, all peripheral controllers and network bridges — referred to as I/O nodes (IONs) — attach directly to SCX channels. An I/O system is then simply a collection of I/O nodes threaded together onto one or more channels.

The ring topology provides great flexibility in configuring distributed systems. Large computers may connect to multiple SCX channels. Shared channels support shared peripherals and allow communication among multiple computers. Private channels allow a computer to access its own dedicated I/O system. Dedicated, point-to-point channels allow high-speed transfers between two computers. A sample configuration is shown in Figure 1.

Cost scalability is a natural result of this approach as well. Entry level systems need only include a single I/O node. Large systems can add channels and I/O nodes as necessary to match demand.

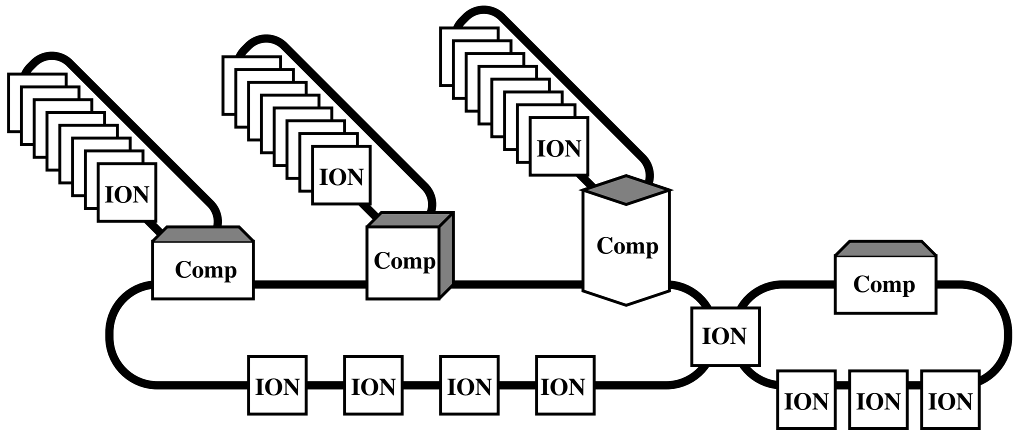
The SCX channel provides high performance through a combination of aggressive technology that provides high peak bandwidth and protocol features that allow high channel utilization with low latency. The channel provides a raw bandwidth of 1200 MB/s, and is capable of sustaining data payload bandwidths in excess of 900 MB/s half duplex, or 2×800 MB/s full duplex.

Availability and reliability are enhanced by a variety of features. Cyclic redundancy checkwords plus parity protect all transmissions, and channel hardware automatically logs error information. Mechanisms are provided to place firewalls between faulty clients and the channel or between logical node partitions on a single channel. In addition, the channel allows live node insertion, removal and swap, and can be reconfigured by software to map out faulty resources.

The SCX channel can be viewed as three layers: a physical layer, a logical layer, and a protocol layer. The physical layer provides the basic substrate for communication. The logical layer provides an efficient packet delivery service over that

An earlier version of this paper was presented at HOT Interconnects III, August 10-11, 1995.

FIGURE 1. Sample system configuration with four system nodes, three private channels and two shared channels connected by an SCX bridge



substrate. And the protocol layer defines the packet types and procedures used to communicate between clients. The physical and logical layers are implemented by the SCX node chip ASIC. The core of the logical layer is based directly on the IEEE Scalable Coherent Interface (SCI) standard [2], although we have culled what we do not need and added new features to meet our requirements.

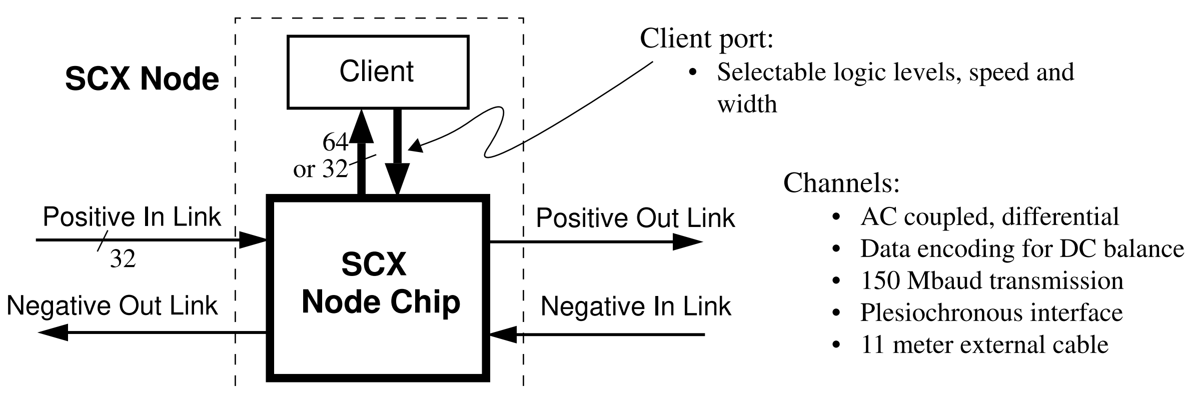
SCX supports three primary functions: DMA (direct memory access) transfers, messaging, and remote client maintenance and configuration. DMA transfers allow a client (the master) to directly read or write a block of memory in another client's memory (the slave). The master issues a series of read or write request packets containing slave client memory addresses, and receives a series of read or write response packets in return. Messages transfer data from one client to another without internal addressing information for the recipient. Remote client maintenance allows control and diagnosis of remote clients via a memory-mapped register interface.

The remainder of this paper is organized as follows. Sections 2, 3 and 4 describe the physical, logical and protocol layers of the SCX channel, respectively. Section 5 provides some brief performance information. Section 6 discusses the current status and future directions of the SCX project. Finally, a summary of the paper is provided in Section 7.

2 Physical Layer

The SCX channel actually contains a *pair* of unidirectional, counter-rotating rings, which provide resiliency and the ability to perform hot swaps. Each node on an SCX channel includes an SCX node chip ASIC, as shown in Figure 2. The node chip contains a 64-bit port for communicating with the local client (this can be configured as a 32-bit port for pin-constrained clients), and a pair of 32-bit channel inputs and outputs, for communicating with other nodes.

FIGURE 2. SCX node block diagram



The nodes on an SCX channel are cabled together as shown in Figure 3, to form the two counter-rotating rings. When the channel contains only two nodes, it functions logically as a single, point-to-point link. To tolerate a failed node or to perform live node insertion, removal or swap, the channel can be *folded* as illustrated in Figure 4 (a). To tolerate a broken link, a single ring can be *masked*, as shown in Figure 4 (b).

FIGURE 3. SCX channels

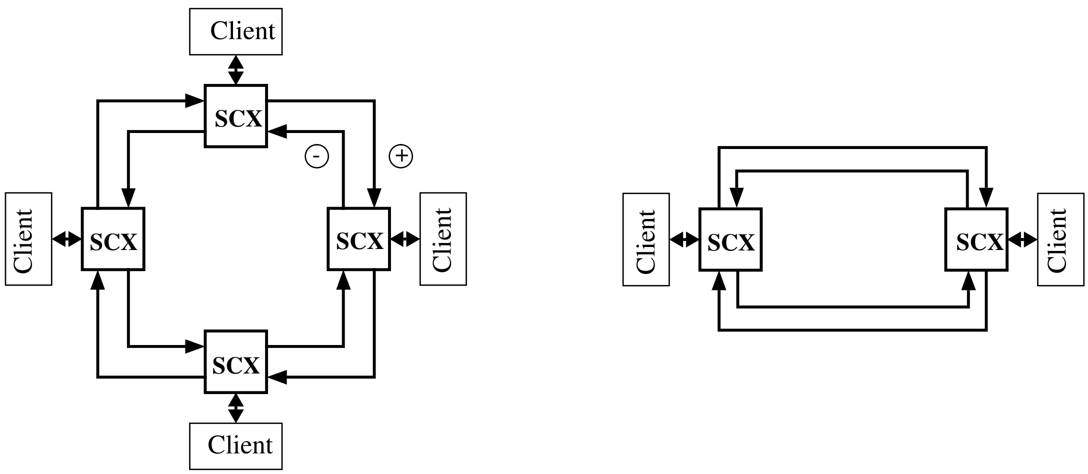
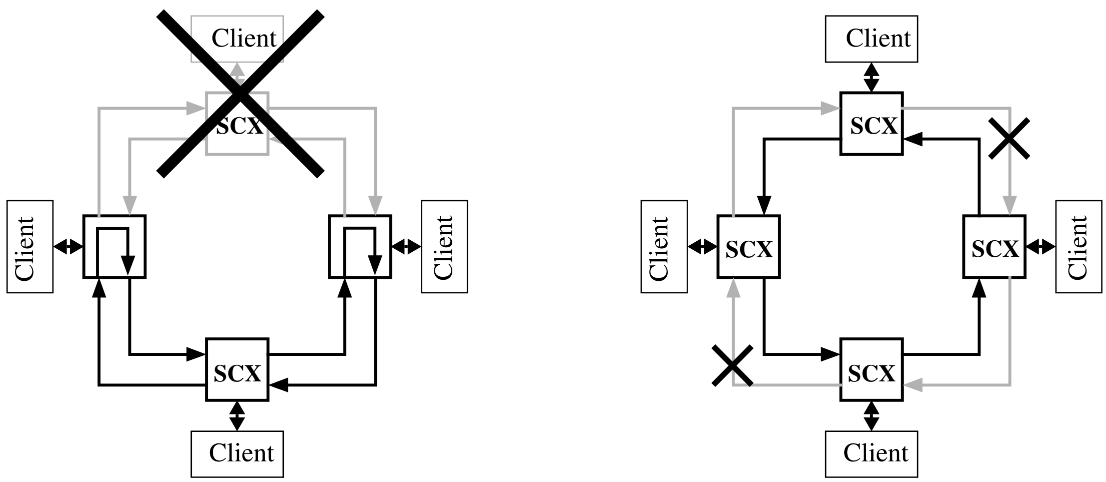


FIGURE 4. Reconfigured SCX channels



(a) folded channel (b) masked ring

The SCX node chip is a CMOS gate array, with approximately 300K logic gates and 100 Kbits of RAM. Estimated power consumption is 15-22 watts. The chip is packaged in a 50mm × 50mm, 840-pin ceramic column grid array (CCGA), with 536 signal pins.

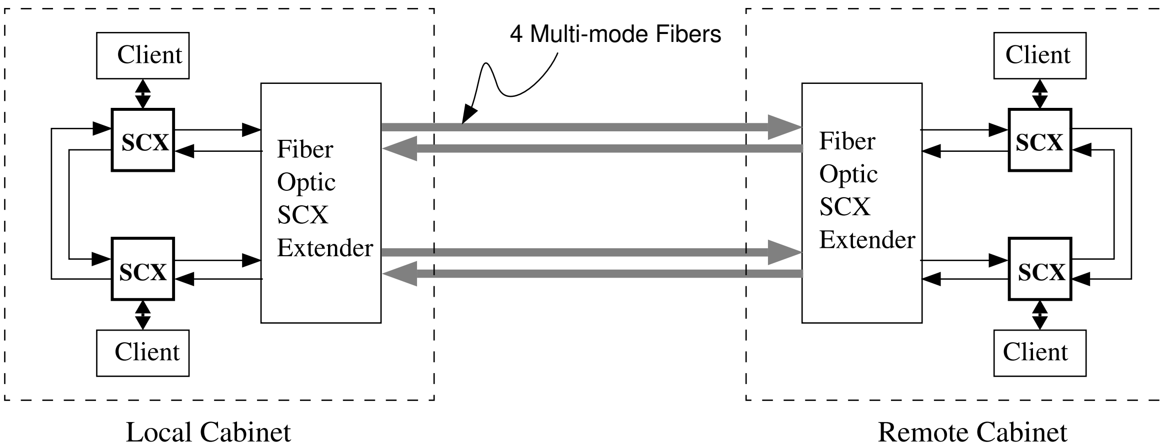
The inter-node channels are capacitively coupled to facilitate communication among electrically dissimilar machines. Data is encoded using one extra bit per 8 data bits in order to maintain DC balance on the channels. The channels are plesiochronous, allowing independently generated clocks at each node. Nodes compensate for slowly varying phase changes between the received and local clock by adding or deleting special slip symbols as necessary.

The client port clock speed is selectable, from 40 to 125 MHz. The channel is clocked at 75 MHz, with data transferred on both edges of the clock, resulting in a 150 Mbaud transmission rate. The node chip thus deals with four clock frequencies: the client port frequency, the internal/transmission frequency, and the incoming clock frequency on both the positive and negative rings.

The standard internode links can be up to 15 meters long, with internal drops of up to 2 meters, and external cables of up to 11 meters. The inter-cabinet cables use 37 individually-shielded twisted pairs and an 80-pin SCSI-2 connector. The primary limitation on distance is skew, rather than attenuation.

For longer distances, a fiber-optic extender can be used, as shown in Figure 5. The fiber extender is transparently inserted between pairs of node chips, and allows cabinet separations of up to 200 meters. The 37 150-Mbaud signals in each cable are converted into four 1.5 Gbaud fiber signals for transmission, and then converted back to copper at the other end. A full extender uses eight Hewlett Packard G-Link serializing chipsets and eight optical transceivers.

FIGURE 5. Fiber Optic Extender



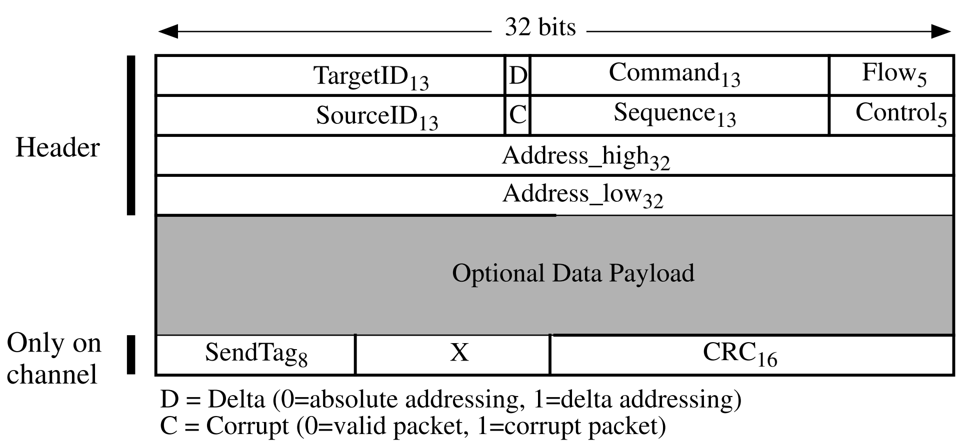
3 Logical Layer

The SCX node chip implements the logical layer of the SCX channel. Its responsibility is to efficiently deliver *packets* of information from one client to another.

3.1 Packet format

Packets are composed of some number of contiguous 32-bit *symbols*. The basic SCX packet format is shown in Figure 6.

FIGURE 6. Packet format



A packet contains a 16-byte header and an optional payload of up to 256 bytes. The *TargetID* and *SourceID* fields identify the target and source of the packet, respectively. Each node connected to an SCX channel (or interconnected network of SCX channels) is assigned a unique nodeID, supporting up to 8192 nodes. The *D* bit specifies delta (as opposed to absolute) addressing, which is useful for channel configuration and maintenance activities. If this bit is set, the *TargetID* represents a hop count rather than a nodeID.

The *Command* field specifies the packet type, payload length (where appropriate), and error status. Packets are divided into request and response packets. Most SCX traffic consists of *transactions*, in which a request packet is sent from one client to another and a response packet is returned in response.

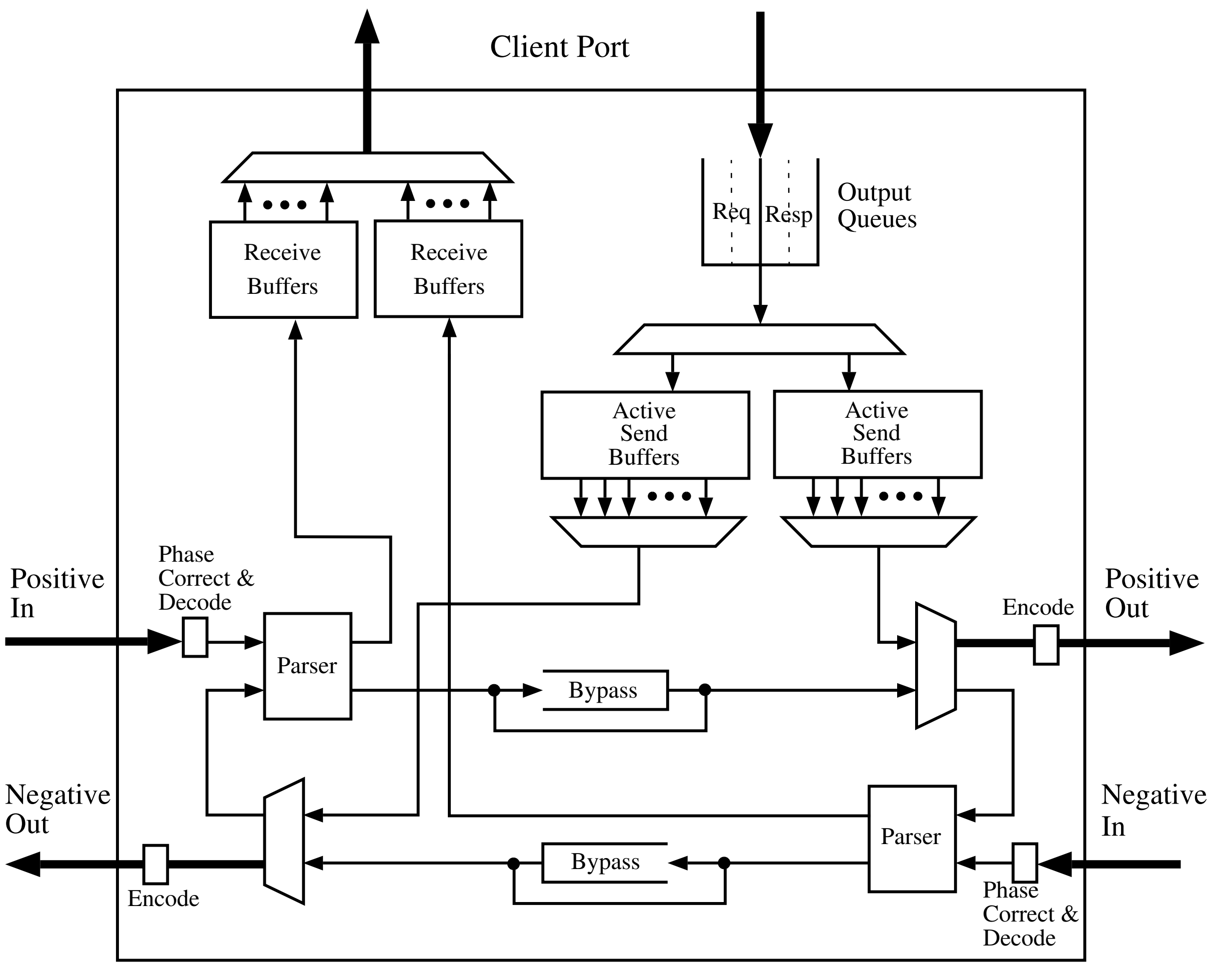
The *Flow* field is used internally by the SCX node chips. The *Sequence* field can be used by clients for identifying or ordering packets. The *Control* field allows the clients to specify routing direction (positive ring, negative ring, or “don’t care”), packet priority status, and packet ordering if desired. The *C* bit is used to mark corrupted packets, which can be delivered to the client for diagnostic purposes. Finally, the packet headers include a 64-bit internal (64-bit word) address.

When transmitting over the channel links, an additional symbol is appended to the packet containing a 16-bit cyclic redundancy checkword (CRC) and a tag that is used internally by the node chips. The CRC provides a very high level of fault detection, but is not guaranteed to catch all errors introduced by a single bad wire. For that reason, all packets are also protected by an extra parity bit per symbol.

3.2 Basic Node Chip Operation

Figure 7 presents a block diagram of the SCX node chip. Packets to be delivered are transmitted across the client port, through the output queues (used for flow control) and into the active send buffers for either the positive or negative ring. Unless explicitly specified (in the *Control* field), the choice of ring is made dynamically by the node chip.

FIGURE 7. SCX node chip block diagram



From the send buffers, packets are transmitted onto the ring, and pass through zero or more intermediate nodes until reaching their target. There the parser recognizes the *TargetID*, strips the packet from the ring, and routes the packet into a receive buffer where its CRC is checked. From there, it is delivered to its destination client across the client port.

Each ring of an SCX channel implements (with minor modifications) the ring access, bandwidth allocation and queue reservation protocols of the SCI standard [2].

A node cannot transmit a new packet onto a ring unless its *bypass buffer* is empty. Thus, during packet transmission, any packet symbols passing through the node can be temporarily buffered in the bypass buffer. When transmission completes, the node enters the *recovery phase*, and is inhibited from transmitting new packets until its bypass is drained.

The protocol is novel in that no flow control is used on the channel links whatsoever. Instead, feedback is provided via per-packet acknowledgements returned by the target nodes. Unlike the bypass buffers, the receive buffers can fill up (due to a slow client), leaving no room to store an incoming packet. Therefore, when the packet is stripped by the parser at its target, a two-symbol *echo* packet is routed the remainder of the way around the ring to tell the source of the packet whether the packet was accepted, or *busied*.

The source node retains a copy of the packet in its send buffers after transmission. When it receives the echo packet, it either frees the send buffer (if the packet was accepted) or retransmits the packet (if the packet was busied). The number of send buffers dictates the number of packets that a node can have outstanding on a ring. The SCX node chip implements 9 large packet send buffers and 9 small packet send buffers (for header only and control packets) per ring. This allows up to 4.5 KB of data to be streamed from a node before receiving the corresponding echo packets.

By avoiding back pressure on the channel links, the basic ring access protocol prevents a single congested target from blocking the entire channel. It also provides fair access to the channel under uniform traffic distributions. In the presence of non-uniform traffic, however, the basic protocol does not provide fair arbitration [4]. For example, if a node in the recovery phase encounters a steady stream of passing packets, it will never be able to empty its bypass buffer, and the node will be prevented from transmitting any new packets (*i.e.*: the node will be starved). The SCI/SCX bandwidth allocation protocol prevents this behavior by use of *go bits* (similar to tokens) contained in the idle flits on the channel.

Go bits are required to transmit new packets onto the channel. During the recovery phase, a node withholds the supply of go bits from its downstream neighbors, eventually inhibiting the transmission of new packets and allowing it to drain its bypass buffer. This protocol reduces the maximum sustainable bandwidth on a ring, but provides approximately equal shares of channel bandwidth to each client under contention [4].

The SCI/SCX queue reservation protocol guarantees forward progress for all packets. Packets that are busied are returned a retry status in their echoes. A four-state FSM associated with the receive buffers guarantees that, periodically, only the oldest set of busied packets are accepted while newer packets are refused. This prevents any packet from being busied indefinitely.

3.3 Virtual Channels

While the basic SCI protocol avoids congestion on the channel, it does little to avoid congestion at the *endpoints*. SCX has added two features to address this concern: multiple virtual channels over the client port and an adaptive congestion control mechanism.

SCI uses two virtual channels (VCs), one for requests and one for responses. The buffering for packets entering and leaving the ring is partitioned between the two VCs such that response packets can never be blocked behind request packets. This is a well-known technique to avoid deadlock: response packets can always (eventually) be sunk by their clients, thus new response packets can always be injected into the channel, thus request packets, which require the transmission of response packets, can eventually be sunk and processed by their clients, etc.

SCX uses four virtual channels, two for requests and two for responses. The VCs are roughly partitioned into read requests, read responses, write requests and write responses. By separating read and write packets, dissimilar data streams are decoupled. Consider, for example, a computer receiving a stream of read request and write request packets. It is likely the case that the write and read request packets can be serviced at different rates, as they require different types of handling. By using separate virtual channels, the two streams are decoupled such that they may “slip” with respect to each other.

Handshaking over the client port is performed separately for each VC. Upon power-on, both the client and the SCX node chip transmit a series of acknowledgment signals equal to their buffering capacity for each VC. These increment *transmission credits* on the other side. Each time the client sends a word (8 bytes) across the client port on a given VC it decrements the corresponding transmission credit. The transmission credit must be positive to transmit. As the node chip removes symbols from the output queue, placing them into send buffers, it returns acks to enable further transmissions.

The output queues are sufficiently deep to cover the round-trip turnaround time for acknowledgements, so that a client can transmit continuously on a single VC.

Flow control from the node chip to the client can be on a word *or* packet basis (or both). The node chip maintains word and packet transmission credits for each VC and requires positive credits on both counters to transmit data. A client wishing to use only one form of flow control (words or packets) simply ties the acknowledgement signals for the other form high.

The send and receive buffers on the SCX node chip can be flexibly partitioned among the four virtual channels under software control. Each buffer can be dedicated to a single virtual channel or dynamically shared among some set of virtual channels.[1] The optimal partitioning depends upon the type of client. System nodes, for example, due to the structure of I/O on Cray systems, tend to receive more request packets and send more response packets. I/O nodes, on the other hand, tend to send more request packets and receive more response packets. Different VCs tend to carry different mixes of large and small packets as well.

3.4 Adaptive Congestion Control

While the use of multiple virtual channels helps decouple different types of transfers, it does not eliminate congestion. A blocked target can cause excessive packet retransmissions for a particular DMA stream, wasting channel bandwidth[2] and blocking other traffic streams emanating from the same source. Client port flow control does not prevent the source node from sending additional packets until all the send buffers available to the blocked transfer have been filled. Thus, a congested stream will tend to hog send buffers that are shared among multiple virtual channels.

To prevent this type of congestion, a transmitting client must know the rate at which to send packets to a slower or temporarily congested target. In addition, a client with more than one stream to transmit must know the ratio by which to interleave packets for the different streams, so that a slow receiver for one stream does not unnecessarily slow down the other stream(s).

The SCX channel includes an adaptive congestion control mechanism that provides this information to clients. For each concurrent DMA stream mastered by a client, the client keeps track of and limits:

- the number of request packets *active on the channel*, and
- the number of request packets *active in the system*.

Requests "active on the channel" are those that have been transmitted from the client to its SCX node chip, but have not yet been cleared from the node chip via a positive echo packet. Each packet active on the channel consumes an active send buffer. The SCX node chip returns a short *ClientEcho* packet to the client each time a request packet is freed from its active send buffers. This enables the client to keep track of the current number of active request packets.

Requests "active in the system" are those requests that have been transmitted for which a corresponding response has not yet been received. This number will be larger, on average, than the number of requests active on the channel. The clients can keep track of this number by counting response packets.

Both *ClientEchoes* and response packets contain a bit informing the master whether the request or response packet was busied at least once before acceptance. The master uses this information to dynamically adjust the maximum number of requests allowed on the channel and in the system, respectively. The initial maximums are determined at transfer startup time using control registers in the master and slave hardware.

As request packets are busied, the congestion control mechanism reduces the allowable number of request packets on the channel. Limiting the number of requests on the channel avoids wasting send buffers and channel bandwidth by request packets. This mechanism works well for *WriteBlks*[3], in which the bulk of the data is carried in the request packets.

As response packets are busied, the congestion control mechanism reduces the allowable number of requests in the system. Limiting the number of requests in the system prevents requests or their responses from building up *after* being removed from the channel, and thus avoids wasting send buffers and channel bandwidth by response packets. This mechanism is well suited for *ReadBlks*[4], and for transfers involving bridges and multiple SCX channels.

1. At least one send and receive buffer must be reserved solely for response packets in order to prevent deadlock.
2. Note that a single node is limited by the bandwidth allocation protocol from wasting more than its "share" of the channel bandwidth.
3. *WriteBlks* are DMA transfers in which the master is writing data to the slave's memory (see Section 4).

Read packets are quite small compared to their responses, for example, and with no limit on the number of requests in the system, many *Read* requests could be quickly stripped from the channel by a slave causing a large backlog of *ReadResp* packets attempting to return to the master. These response packets could then over-consume send buffers in the slave and potentially waste channel bandwidth with their retransmissions if the slave's port was faster than the master's.

Similarly, for transfers involving bridges, the limit on active requests in the system addresses contention that may form after the requests have been stripped by the bridge. When the limit has been reached, new request packets will held back, even if the bridge could currently accept the requests.

Limiting the number of outstanding requests (both on the channel and in the system) provides negative feedback that naturally regulates the packet transmission rate (even if the limits were not adjusted). Request packets experiencing congestion will have longer delays, which will result in a decreased transmission rate. This relationship is characterized by Little's Law, N=XR, which states that in any finite system, the average number of "objects" in the system (N) is equal to the product of the rate at which the "objects" are entering the system (X) and the average time the objects spend in the system (R). By fixing N, as R grows, X decreases.

The SCX congestion control mechanism is completely tunable via software. Control registers on each client dictate both the operating points of the DMA transfers (what fraction of requests and responses should be delivered without being busied), and how fast the transfers should be adjusted to changing levels of contention. Different clients can be adjusted to be more or less "aggressive" in their use of the channel. Control registers used in transfer startup establish a speed match between clients, such that a high-BW client will not immediately swamp a low-BW client with packets it cannot process.

3.5 Maintenance and Control Features

The SCX channel provides a memory-mapped register (MMR) interface for controlling both the channel itself and remote clients. MMRs on the SCX node chips and remote clients are written and read via special SCX packets addressed to the appropriate entity.

SCX defines certain MMRs that must be implemented by the clients, such as those used by the congestion control mechanism. Clients can, in addition, use their SCX MMR space to implement arbitrary functionality. The Cray T3E MPP can, for example, be booted, diagnosed, reconfigured and otherwise manipulated via the SCX channel using client MMR packets.

Certain MMRs on the SCX node chip also support remote client maintenance. "Membranes" can be put up across client ports that allow client MMR response packets to get through, but prevent the client from injecting any other sort of traffic onto the channel. This allows remote diagnosis of a suspect client without allowing the client to corrupt the system. Various other levels of client privilege can be established, including normal user mode (allow regular SCX traffic, but no channel reconfiguration or remote client maintenance traffic) and complete isolation. Firewalls can also be placed between logical groups of nodes to prevent interference between groups.

Other MMRs on the SCX node chip control channel configuration and operation. The channel can be folded or rings masked, in order to tolerate faults or add, remove or swap nodes (see Figure 4). The send and receive buffers can be managed (refer to Section 3.3). The congestion control mechanism can be enabled or disabled. Various performance monitoring functions can be invoked.

A node chip can be configured to function as a bridge (see Figure 1), acting as the target for packets addressed to arbitrary sets of other channels. Packets for remote channels are stripped and relayed onto the other channel(s), and response packets are accepted from the other channel(s) and relayed back onto the original channel. Since the client port is symmetrical, a bridge can be constructed simply by connecting two SCX node chips back to back. Initialization requires simply writing a handful of MMRs on the node chips. Thereafter, traffic is automatically and transparently routed between the channels.

The node chips also include error monitors that count various types of errors and latch error state. These monitors can be read and cleared by maintenance programs and used to diagnose and isolate faults in the channel. Limited channel maintenance (such as detecting and scrubbing corrupted packets) is performed automatically by a node chip designated as the *scrubber* (as in SCI). The node chip itself can be tested via JTAG boundary scan [1].

4. *ReadBlks* are DMA transfers in which the master is reading data from the slave's memory

4 I/O Protocol

The protocol layer of the SCX channel provides two primary functions: peer-to-peer messaging and DMA. Peer-to-peer messages can be unsolicited, and require no intra-client addressing. Small messages (up to 256 payload bytes) can be sent in a single SCX packet (referred to as a *MsgPkt*). Larger messages can be sent either by breaking them up into multiple *MsgPkts* (which then require reassembly), or by prearranging buffer space and using DMA.

Incoming *MsgPkts* in Cray Research systems are placed into a predefined buffer area in memory and the operating system is notified. *MsgPkts* have no response counterpart in hardware. Thus, delivery of *MsgPkts* cannot be confirmed by the sender, save for receiving another *MsgPkt* or other indication in return. Client software is responsible for reliable delivery of messages.

DMA is performed through *ReadBlk* and *WriteBlk* operations, which allow a DMA *master* to directly access a *slave* clients memory without intervention from software on the slave. DMA transfers consist of a *BlkInit* packet exchange, followed by a series of pipelined *Read* or *Write* packets issued by the master, followed by a *BlkDone* packet exchange. The entire process is carried out in hardware. Software on the master client initiates a transfer and is interrupted when the transfer has completed

Table 1 lists the packets used by the SCX protocol layer. The last two groups of packet types are those used for channel and client control and maintenance, as discussed in Section 3.5.

TABLE 1 IO protocol packet types

Name	Description
MsgPkt	Send a self-contained message with a payload of 0-32 64-bit words to another client
ReadBlkInit	Set up transfer from target to source
ReadBlkInitResp	Acknowledge *ReadBlkInit* and negotiate speed or buffer requirements
Read	Request 1-32 64-bit words from target memory
ReadResp	Return *Read* data
ReadBlkDone	Notify completion of *ReadBlk*
ReadBlkDoneResp	Acknowledge *ReadBlkDone*
WriteBlkInit	Set up transfer from source to target
WriteBlkInitResp	Acknowledge *WriteBlkInit* and negotiate speed or buffer requirements
Write	Transfer 1-32 64-bit words to target memory
WriteResp	Acknowledge receipt of *Write* (but not necessarily completion).
WriteBlkDone	Request completion of *WriteBlk*
WriteBlkDoneResp	Acknowledge *WriteBlkDone*, signalling completion of *WriteBlk*
ClientEcho	Provide feedback to client for congestion control mechanism.
MMR_Read	Request read of 1 word from SCX node chip MMR.
MMR_ReadResp	Return value of SCX node chip MMR.
MMR_Write	Write 1 word to SCX node chip MMR.
MMR_WriteResp	Acknowledge write of SCX node chip MMR.
MaintRead	Request read of 1-32 words from client MMR.
MaintReadResp	Return data from client MMR.
MaintWrite	Write 1-32 words to client MMR.
MaintWriteResp	Acknowledge write of client MMR.

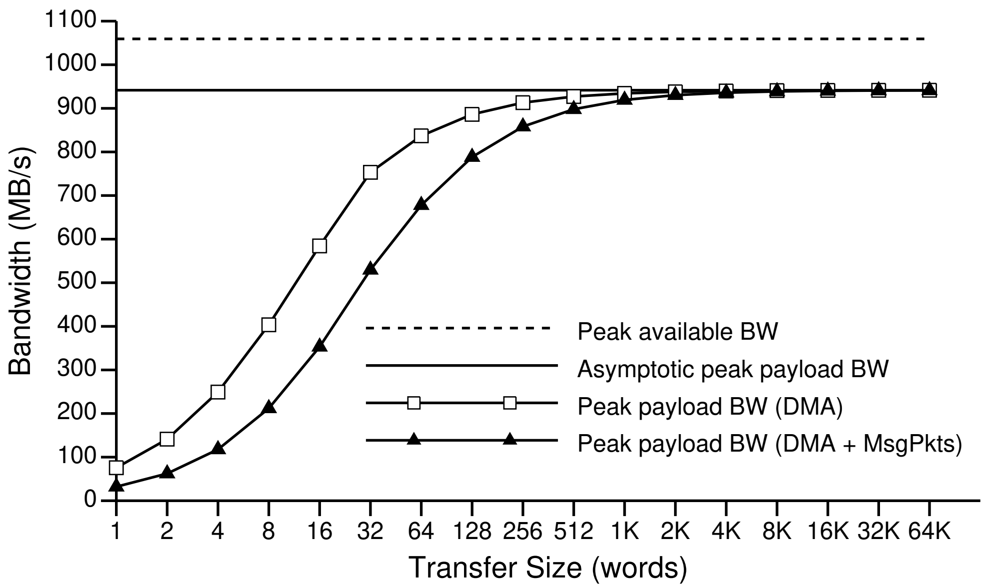
5 Performance

Calculation of SCX performance is somewhat complicated. The maximum payload bandwidth of a channel depends upon the traffic distribution, types of transfers, and even which clients are mastering the transfers (because the packet types influence the ratio of packet overhead to data). Different clients may have different port speeds as well, which may or may not limit the achievable bandwidth of a transfer, depending upon what other traffic exists on the channel.

The SCX channel itself has a peak raw rate of 1200 MB/s between neighboring clients (32 data bits in each direction with 150 Mbaud transmission). Of this, approximately 1059 MB/s remains after encoding the data for DC balance and the plesiochronous interface. Protocol overheads further diminish the maximum payload bandwidth. For a one-way transfer between two clients, the maximum payload rate is 941 MB/s. The maximum full duplex rate between two clients is 827 MB/s in each direction.[1]

For small transfers, the overhead of the DMA control packets (*ReadBlkInit*, etc.) reduce the maximum bandwidth, as well. Figure 8 shows the maximum data payload bandwidth for unidirectional DMA transfers across an SCX channel as a function of transfer length. Latency effects are ignored (*i.e.*: it is assumed that transfers are sufficiently pipelined such that latencies are hidden). The first curve includes the DMA transfers alone, including *BlkInit* and *BlkDone* packets. The second curve assumes in addition that two 16-word control packets are exchanged per transfer.

FIGURE 8. Effect of transfer size (in 64-bit words) on peak payload bandwidth, assuming pipelined transfers (no latency effects), half-duplex *WriteBlks*



Of course, in reality, latency will have an effect on realized bandwidths. Per-hop packet latency in the absence of contention is approximately 225 ns. Hardware processing at the endpoints and transmission across the client ports can add, perhaps, 1 to 2 μs additional latency, depending upon port speed and packet size. The *Read* or *Write* packets within a DMA transfer are fully pipelined, but the transfer as a whole involves three round trip delays. A one-word DMA transfer in a typical system might have a 10-15 μs hardware latency. Thus, from a hardware perspective, a channel or port can be kept fully utilized with an average of two outstanding transfers, each with a data transfer time of at least 10-15 μs.

The bandwidth of an SCX channel is sufficiently high, and the latency sufficiently low, that software latency and occupancy time is likely to be the limiting factor for short I/O transfers. Consider, for example, a processor with a 500 MB/s SCX port making 4KB transfers, with a hardware startup latency of 12 μs. The data transfer time is 8 μs, resulting in a total HW latency of 20 μs. Even a very low software overhead of 20 μs would double the latency of a single transfer. Moreover, software occupancy would limit the number of outstanding requests to an average of two, capping the sustainable data bandwidth to 200 MB/s. Were the processor to have a 900 MB/s SCX port, the maximum data rate would *still* be 200 MB/s. The moral, of course, is to use large transfers when possible and keep software latencies low.

6 Current Status and Future Directions

The SCX channel will be used in several Cray products shipping in 1996, including MPPs, parallel vector processors, and the new, scalable I/O systems.

1. The full duplex bandwidth is lower than the half duplex bandwidth because address-only packets and echo packets flowing in the opposite direction of the data packets steal bandwidth from the data packets for the *other* transfer.

Design and layout of the first SCX node chip was completed in July 1995. This chip will support system and subsystem testing in early October, but is not intended for production. As of this writing, a second revision with additional features and bug fixes is being readied for release to layout in late August. This chip will support SCX and system verification in late 1995. Work on the scalable I/O system is on a similar schedule, with initial customer shipments slated for early 1996.

Future work on the SCX channel will proceed in two primary thrusts: increasing speed and expanding functionality.

The initial SCX implementation achieves very high bandwidths via moderately aggressive circuit technology and a fairly wide datapath (64 bits). The second generation implementation of SCX will likely keep the same path width, and concentrate on increasing the signalling speed. At targeted inter-node distances (10-20 meters), the SCX clock rate is currently limited by signal skew among the parallel wires rather than by signal attenuation. This constraint will likely be relaxed via electronic de-skew circuitry in the second-generation SCX node chip.

There are several possible functional enhancements under consideration for second generation SCX hardware: automatic channel reconfiguration in the event of unanticipated node failure; support for "preferred" shortest path routing, which would allow software to specify routing direction, but adaptively make use of the other direction when congestion was encountered; expanded diagnostic capabilities and performance monitoring; etc. Experience using first generation hardware will likely suggest additional improvements.

Cray Research intends SCX to be an open standard, and is pursuing collaborations to that effect.

7 Summary

This paper presented the first public description of the SCX channel, a new system interconnect and I/O channel from Cray Research. Slated for introduction in 1996, this channel will provide the flexible, high performance communication mechanism required by next generation stand-alone and clustered supercomputer systems. We have augmented the basic communication protocol of the SCI standard with additional features for performance, reliability and functionality.

The SCX channel supports a peak payload bandwidth of over 900 MB/s half duplex (2 x 800 MB/s full duplex). Generous buffering, multiple virtual channels and an adaptive congestion control mechanism provide efficient support for many simultaneous streams of data. On-line diagnostics and maintenance are supported via a rich MMR interface, protective firewalls and live node insertion, removal and swap. Fault detection is provided via a 16-bit CRC plus parity and hardware error monitors. Hardware faults can be tolerated via ring masking and channel folding.

Acknowledgements

Many people have contributed to the definition and design of the SCX channel and we take this opportunity to thank a few of them. Steve Oberlin helped get SCX off the ground and contributed several key early ideas. The SCX task force that shaped the initial requirements for the channel and performed much of the early and later work includes Dan Hendrickson, Bob Halford, Pete Logghe, Rick Pribnow, Steve Scott and Paul Wells. The node chip logic designers are Jim Bedell, Dan Kunkel and Gerry Schwoerer.

The work represented in this paper was partially supported by the Advanced Research Projects Agency under Agreement No. MDA972-95-3-0032 dated 12 January, 1995.

References

[1] IEEE Standard 1149.1-1990, "Test Access Port and Boundary-Scan Architecture," 1990.

[2] IEEE Standard 1596-1992, "Scalable Coherent Interface (SCI)," 1992.

[3] Johnson, S. J. and S. L. Scott, "A Supercomputer System Interconnect and Distributed IOS," to appear in the 14th IEEE Symposium on Mass Storage Systems, September, 1995.

[4] Scott, S. L., J. R. Goodman and M. K. Vernon, "Performance Analysis of the SCI Ring," *Proc. 19th International Symposium on Computer Architecture,* pp 413-414, May 1992.