

# A Network Fabric for Scalable Multiprocessor Systems

Nitin Godiwala

Jud Leonard  
SiCortex, Inc.

Matthew Reilly

Three Clock Tower Place, Suite 210  
Maynard, MA 01754  
www.sicortex.com

## Abstract

Much of high performance technical computing has moved from shared memory architectures to message based cluster systems. The development and wide adoption of the MPI parallel programming model has hastened this transition. Parallel scaling, however, is frequently limited by the inefficient communication hardware commonly found in commodity based clusters. This paper describes a new communication network (the SiCortex fabric) employed in the SiCortex SC5832 integrated cluster system. The fabric switch and communications controller are integrated with a single-chip multiprocessor node and provides three point-to-point links per node chip. The resulting design provides low latency, high bandwidth, reliable communication between the 972 nodes of the SiCortex system.

## 1 Introduction

The SiCortex SC5832 system comprises a cluster of 972 six-processor computing nodes connected via a high speed parallel point-to-point network. The network (fabric) topology is a *Kautz graph* which guarantees that the longest path through the network is logarithmic in the number of network nodes. In the case of the SC5832, the network diameter is 6 hops, with each node connecting three input ports and three output ports into the fabric. [1], [2], [3]

Figure 1 shows the single-chip, six-processor cluster node. Twenty-seven cluster nodes are included in each of the thirty-six modules in the SC5832, for a total of 972 nodes or 5,832 processors in the system. Messages are originated (via MPI.SEND calls) by an application running on one of the node's six processors, packetized by the DMA engine, tagged with a network routing string, and inserted into the fabric via the DMA port on the fabric switch. The packets may pass through a number of other nodes (up to six in the SC5832 fabric) before arriving at the destination where the packet is passed through the destination node's

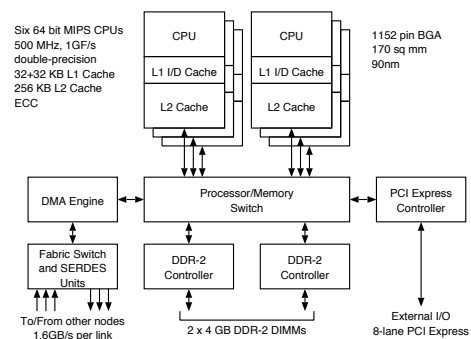


Figure 1. The SiCortex Six Processor Cluster Node

fabric switch to its DMA engine. The DMA engine then assembles the packets and delivers the message to a receiving application that has invoked the MPI.RECEIVE function.

The remainder of this paper will describe the SiCortex fabric's point-to-point links, switch components, and Kautz graph topology.

## 2 Engineering Motivation and Context

The design of the SiCortex fabric was guided by a small set of product requirements and features.

- The system would behave as a Linux/MPI cluster of approximately 1000 nodes.
- The system MTTF target was on the order of 1 year.
- The entire system would fit in a single cabinet of less than  $8 m^3$  volume.<sup>1</sup>

<sup>1</sup>We never expressed the actual size of the cabinet so explicitly, but we all had a notion of what constituted a "single cabinet."

During our initial research we found that MPI was the clearly dominant parallel programming model for most of our target customers. We also noticed that, while many we talked to were enthusiastic about regular, and especially cartesian topologies, there was little evidence for the assertion that such topologies were necessary to effective communication in most problem domains. In particular, the assertion was frequently made that the halo exchange properties of many codes all but mandated a mesh or torus interconnect pattern. While halo exchanges do appear in quite a few domains – ranging from seismic wave propagation codes to weather forecasting – many other models had no such physical correspondence. Even in cases where the problem formulation seemed to map to a cartesian space, we found that complexities of load balancing, job scheduling, and hardware reconfiguration in the presence of failures made an actual mapping of tasks to specific hardware nodes problematic where it was not outright impractical. Finally, we found evidence of many codes whose communication patterns were either unpredictable (*e.g.* global search and sorting) or far from nearest neighbor (*e.g.* three dimensional FFT, large system solvers, Ewald sum kernels). This indicated that little profit was to be had in asking the application developer to work at problem-to-topology mapping, and so we aimed for a network topology with minimum diameter.

Our surveys of over a dozen applications and application kernels (including the NAS Parallel Benchmarks, several weather codes, and one molecular modeling program) indicated a clear bimodal distribution of message sizes. The two dominant modes were on the order of 128 bytes and 100 kilobytes per message. The existence of the small-message cluster mandated a communication model that minimized the fixed overhead of message initiation and capture. This differs greatly from the criteria that motivate interconnects developed primarily for input/output applications.

A system with 1000 nodes and significant inter-node bandwidth would contain an enormous number of physical signal paths. This demanded link-level automatic retry for messages that were corrupted in transit. Forward error correction was discarded as a mechanism: few FEC schemes can tolerate persistent or long-lived errors that corrupt multiple symbols in a single message packet.

The cost of two-chip node would likely result in product margins that were insufficient to attract investors or ensure a lasting business, so the fabric switch, communication controller, and the processors would be integrated on a single die. We settled on a node chip size of approximately 1.7  $cm^2$  and 1100 external connections, of which fewer than 500 could be used for signaling. After budgeting (in round numbers) 120 pins for each of two DDR2 memory ports, 120 pins for an IO port (unspecified at the time, but we later chose PCI express), this left approximately 140 pins for the

fabric communications.<sup>2</sup> Inevitably long paths required uni-directional and fully differential link paths; this suggested a limit of about 35 output signals and 35 input signals. As we were not yet ready to commit all the remaining pins to the fabric (some would be required for ancillary signals) we chose a target of three input ports, and three output ports. Each port was eight bits wide and included a single bit reverse channel for flow control.

These considerations allowed us to refine the requirements for the fabric:

- The fabric must guarantee in-order delivery of all packets even in the presence of transient, but long-lived, errors in the physical channel.
- The fabric link must be capable of operating at 2GB/s over a path length of 180 cm that includes two high performance connectors.
- The fabric and message hardware must support very low latency (on the order of 1 microsecond) transfer for small messages and high bandwidth (greater than 100MB/s) for large messages.
- The fabric should exhibit low diameter and scale to larger configurations without changes to either the node chip or to its module.

### 3 Copper Paths and a Single Global Clock

Typically, a large cluster system consists of multiple, self contained server computer nodes. In such a system, the communication hardware on each node has its own time-base or reference oscillator that sets the data rate for its outgoing bitstream. These oscillators may vary from node to node by as much as a few hundreds parts per million (ppm). In a large cluster, when transferring data from one node to another, the communication interfaces at both ends of the link must account for the differences of two oscillators to ensure reliable data transfers. Receivers compensate for frequency differences by including elastic rate matching buffers that add to the path latency between nodes. The additional path latency is negligible when compared to the total time through most communication hardware, but the SiCortex communication fabric supports MPI transaction costs in the microsecond range.

In contrast to a typical cluster system, the SC5832 system implements a frequency locked (mesochronous) clock architecture. The single reference oscillator simplifies the node interface logic design and reduces node-to-node latency. The additional cost of the rate matching buffers was

<sup>2</sup>The author's reconstruction of this process is from memory. It is unlikely that the evolution of our thoughts were as orderly as all this would suggest, but schematically this is as accurate as we can be.

estimated to be greater than 50nS across a typical five hop path. This was 20% of the total expected transport delay.

In the SC5832 system, the master clock oscillator generates a 66.66 MHz low frequency system reference clock. Buffer amplifiers distribute the low frequency clock radially to each of the cluster's 36 modules. Each module radially distributes copies of the clock to its 27 SiCortex node chips. The total number of clock-distributing buffers from the source oscillator to any node is kept small and constant for all nodes.

Each node generates its high-frequency clocks for on-chip functions using on-chip phase-locked loops (PLL). The high-frequency output of each PLL is locked to the system reference clock. Thus, each high-frequency clock in a node is frequency locked with the corresponding high-frequency clock in every other node, though their phase relationship is indeterminate. The communication fabric described here makes use of two derived clocks: the high speed receive and transmit clocks (F-clock) at 2GHz to support a raw bit rate of 2Gb/S per serial channel, and an S-clock at 200MHz used to transfer parallel data between the SERDES units and the fabric switch and for the fabric switch's internal clock.

All of the system's inter-node paths are contained within a single cabinet and are purely electrical. The requirement to keep latency, cost, and power dissipation low argued against the available optical technologies. This presents challenges in producing 2Gb/S data rates over paths that may be longer than 1.5m. However, with the use of low loss PCB materials, high quality module connectors, careful layout, and significant pre-emphasis, signal integrity was maintained over even the longest signal paths.

### 3.1 The Fabric Links

Nodes in the SiCortex system are connected via unidirectional, point-to-point fabric links. Each fabric link comprises eight lanes of data (the data link) carried over eight differential wire pairs, and a single lane of flow control (the control link) traveling in the reverse direction on one differential pair.

Each fabric data path within a node chip is 64 bits (8 bytes) wide and is synchronous with the node's 200 MHz S-clock. Each byte of the eight byte data path is mapped onto a separate and individual serial lane. A serializer encodes the parallel data into an 8B/10B serial bit stream that embeds the transmit clock as well. The 8B/10B encoding ensures DC balance over each lane and sufficient bit transitions to allow the receiving deserializer to recover the phase of the transmit clock.

The raw data rate over each lane is 2.0Gbits/S, providing a fabric data rate of 1.6GBytes/S after accounting for the 8B/10B code overhead.

For each data link, there is a companion control link

flowing in the opposite direction (from data receiver to data transmitter). This control link provides an explicit ACK/NACK path from the receiving node, as well as control flow and virtual channel management information.

Each packet traveling on the data link carries a CRC (cyclic redundancy check) in its last word. "Bad packets" are rejected by the receiving node, which signals rejection of the packet over the control link. Control packets are also protected by a CRC word at the end of the packet. All payload bits in packets on the data link and control link are covered by the CRC. Control packets are designed so that a corrupted control packet can be safely ignored.

The combination of the per-packet error checking and control packet mechanisms provides per-hop retry of all communications. The SiCortex fabric guarantees delivery of all packets to the intended destination.<sup>3</sup>

### 3.2 The Data Link

Data packets comprise three components: the header, the payload, and the trailer. The payload carries the actual content of a message and may vary in size from 2 to 18 words of 64 bits each. The 64 bit header includes the start-of-packet marker, a 32-bit routing string, a virtual channel identifier, the packet length, a sequence number, and a buffer identifier. The trailer carries the end-of-packet marker, a 32-bit cyclic redundancy check, a packet type specifier, and 20 bits of information used by DMA engine microcode.

Four fields in the header and trailer deserve special attention.

**Sequence Number** Each packet is tagged with a sequence number in the range 0 to 15. Packet transmission is managed such that a sequence number is not reused until it has been acknowledged by the receiver at the end of the link. Packet sequence numbers are re-assigned at each transmitter.

**Buffer Identifier** The buffer identifier allows the transmitting node to control allocation of buffers in the receiving node.

**Cyclic Redundancy Check** Each packet ends with a 32 bit CRC to identify corrupted packets at the link receiver.

**Type Marker** The type marker in the trailer is used primarily by message handling microcode, but one type code is reserved to mark "poisoned" packets. Because

---

<sup>3</sup>"Guarantees" in the presence of transient errors: All packets are delivered to the appropriate destination under normal conditions. All links are protected against transient bit errors, either single or clustered. All RAM storage buffers in the network path are ECC protected. Physical damage to a path between nodes *will* cause failure delivery. This failure is signaled to the operating system and handled by system and application software. The topology of the network allows the fabric to route around a large number of permanent link failures.

the CRC code is contained at the end of the packet, a receiving node may have already forwarded the head of a packet over the next link in the path before it finds a CRC error. In this case, the forwarding node marks the tail of the packet as “poisoned.” Poisoned packets may be routed through the network or may be dropped along the way. If a poisoned packet arrives at an endpoint, it is ignored by message handling microcode.

### 3.3 The Control Link

Control packets are fifteen bytes long, starting with a start-of-packet marker, and ending with four bytes of CRC. The rest of the packet contains error control, buffer allocation, and out-of-band signaling information.

The error control portion of the packet includes the sequence number of the last packet to arrive with a good CRC. It also includes an error flag. The operation of these fields is described below in Section III.C.

The buffer allocation fields indicate which buffers in the receiving node are “currently” allocated to packets – that is, unavailable for allocation by the transmitting node. The allocation fields are a snapshot of some state in the past. They are accurate up to the receipt of the “last good packet” indicated by the error control portion of the control packet. Thus the transmitter is responsible for remembering which packet buffers it has allocated since it transmitted the acknowledged last good packet. Because the transmitter’s picture of busy buffers is a composite of its own record of allocations and information from the downstream node, any control packet can be safely dropped or ignored; the buffer usage state in the transmitter is pessimistic.

The out-of-band signaling information provides a low speed communication path to system maintenance software.

## 4 The Fabric Switch

The SiCortex node contains a four-by-four crosspoint switch with local buffering for packets traveling through each of the output ports, as shown in Figure 2. Each of the fifteen crosspoint buffers  $XB_{ij}$  holds packets arriving on port  $i$  and destined for port  $j$ . (Note that there is no  $XB_{33}$ , as the switch does not need to buffer transfers from the local node to itself.) Each of the crosspoint buffers has room for sixteen incoming packets. Typically, packets bypass the crosspoint buffers and pass directly, with minimal delay, from input port to output port. The crosspoint buffer entries are maintained to accommodate contention for output ports or downstream resources.

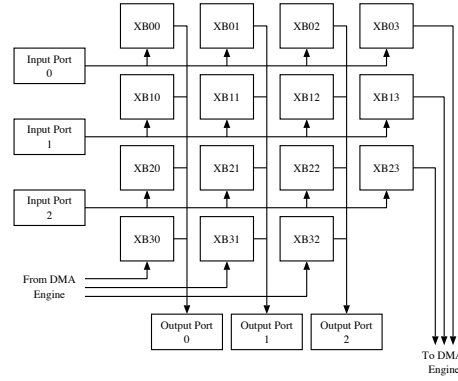


Figure 2. The Fabric Switch

### 4.1 The Routing Scheme

Though the SC5832 system uses a Kautz topology for its inter-node network, the node design is independent of the network graph. Each node contains three output ports to transmit packets into the network, and three input ports to receive packets from the network. Routing decisions at each stage in a packet’s path through the network are made by examining the low two bits of the packet’s routing instructions included in the packet header. The initial routing string is constructed by software. All packets are source routed.

As a packet arrives on the input port of a switch, the low two bits of the routing string are shifted off and used to direct the message to the appropriate output port and crosspoint buffer.

### 4.2 Virtual Channels and Deadlock Avoidance

The SiCortex fabric avoids topologically induced deadlocks by routing all packets over virtual channels. [4] The channel assignment scheme is independent of network topology. Each node in the network is assigned a number from 0 to  $N-1$  where  $N$  is the number of nodes in the graph. Each packet is assigned a starting virtual channel number that determines which crosspoint buffer entries it may occupy. (Each virtual channel is assigned at least one entry in each crosspoint buffer for its exclusive use.) When a message arrives at node  $Y$  over a link from node  $X$ , and bound for node  $Z$ , the fabric switch decrements the virtual channel number if  $Y$  is greater than  $X$  and  $Y$  is also greater than  $Z$ . It is easy to see that if the network diameter is  $D$ , then this scheme requires no more than  $D/2$  virtual channels. Since all packets are source routed, the originating node knows how many times the virtual channel will be decremented on

any given path, and can assign the packet to a virtual channel such that it arrives on channel 0 at its final destination. Alternatively, all packets can be assigned to virtual channel  $\text{floor}(D/2)$  at their insertion into the fabric.

It is possible for a packet to be corrupted by noise that affects the virtual channel assignment. In this case, the packet may arrive with a virtual channel assignment (VC) of 0 at a node that needs to decrement the VC. In this case, the packet is marked as poisoned and is ignored by the receiver. The recovery mechanism will ensure that the packet is automatically retransmitted. In the case of a corrupted VC that is not detected by a decrement operation, the packet will be labeled as poisoned (and perhaps even misrouted) by the CRC logic in the receiver. Again, the recovery mechanism will ensure a retransmission.

### 4.3 Errors and Recovery

The SiCortex 5832 fabric has over 26,000 differential signal pairs, each carrying 2.0Gbits/S. Even the most carefully designed communications mechanism will encounter single and multiple bit errors often enough to be significant. For example, if all 26,244 channels had a bit error rate of one bit in  $10^{15}$ , then an error will occur on *some* link in the system every 20 minutes. The fabric is designed to recover quickly and gracefully from packet corruption by ensuring automatic detection of corruption at the end of each fabric hop, and automatic retransmission at the link level.

As a packet arrives on a node's input port, it is routed to its destination and written into its assigned crosspoint buffer entry. (If an error has corrupted the buffer entry number in the packet header such that the corrupted number identifies an occupied buffer, the packet is not written to the crosspoint buffer.) When the trailer of the packet arrives, its CRC is compared with the accumulated CRC of the packet (including the contents of the rest of the trailer).

If the CRCs match, the receiver updates its "last known good sequence number received" to reflect the sequence number of the incoming packet. This (or some later sequence number) will be sent in the next control packet back to the transmitter. Control packets are sent out continuously.

If the CRCs do not match, the receiver sets the "error found" bit in the subsequent control packet. All subsequent data packets are dropped until the transmitting end of the link acknowledges receipt of the "error found" bit by sending a special short data packet. Once the receiver sees such a packet, it clears the error found bit in subsequent control packets, and the receiver resumes processing incoming data packets.

The transmitter, for its part, records each outgoing packet in a replay buffer indexed by the sequence number of the outgoing packet. When one or more packets are acknowledged by a change in the "last good sequence number"

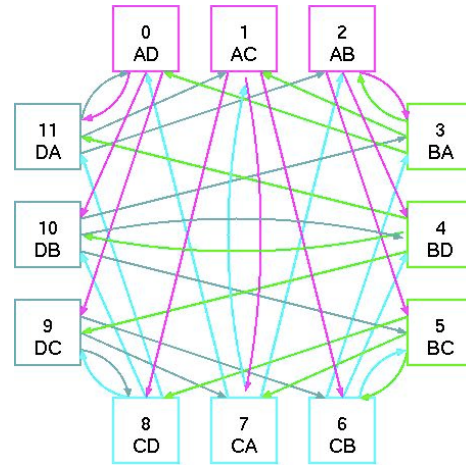


Figure 3. A Twelve Node Kautz Graph

field in a control packet, the entries in the replay buffer are released. When a control packet arrives with the "error found" bit set, the transmitter acknowledges the error, and retransmits all packets in the replay buffer beginning with the packet *after* the "last good sequence number."

There are, of course, dozens of possible error scenarios. Each is covered by the simple replay scheme. All packets are delivered in order to their destinations, even in the presence of errors and noise.

## 5 The Kautz Topology

Kautz digraphs have been discussed in the mathematics literature for almost 40 years; they have received attention because they are among the largest known graphs of a given degree and diameter. Figure 3 shows a twelve node Kautz graph of degree 3. Each node drives its three outputs to three other nodes, and receives its inputs from three other nodes. The longest path length from one node to any other node is 2 hops. The SC5832 expands this 12 node graph to 972 nodes, each no more than 6 hops from any other node. [2]

Unlike mesh or torus topologies, whose diameter is proportional to the square or cube root of the size of the network, the diameter of a Kautz graph grows as the logarithm of its size. For networks of fewer than a hundred nodes, the difference is not great, but with clusters growing well above that level, logarithmic diameter becomes increasingly important. Table 5 compares the diameter of a degree 3 Kautz graph with that of 2-D and 3-D tori for various network sizes. (The first number in each column is the network diameter, the second is the bisection width in number of links.)

Furthermore, unlike hypercubes (another logarithmic topology), the degree of a Kautz graph does not depend on

Topology	Switch Degree	Number of Nodes		
		108	324	972
<b>Kautz d=3</b>	3	4/40	5/97	6/243
<b>2-D Torus</b>	4	11/10	18/18	32/31
<b>3-D Torus</b>	6	7/23	10/47	15/98

**Table 1. Network diameters/bisection-widths for various topologies**

the size of the graph, and the base of the logarithm can be chosen to be larger than 2: the logarithmic base is equal to node degree. To put this in perspective: a degree 3 switch in a Kautz graph will connect 972 nodes in a 6 hop diameter network, where a degree 10 switch in a 1024 node hyper-cube produces a network diameter of 10.

Finally, in comparison to fat trees, the Kautz graph has the advantage of being a direct network (meaning that the switch is a part of each node) so its cost per node is very low in comparison, and (unlike a fat tree) does not increase with network size.

As a result of the very low redundancy in routing decisions, the Kautz digraph can use a significantly simpler switch design than other topologies; to a first order, the complexity of a switch increases with the square of the degree. This has a direct impact on the cut-through latency of the switch, multiplying the benefit of low diameter.

Furthermore, low switch degree means that larger numbers of pins can be dedicated to each link, improving the available bandwidth without resorting to exotic packaging and cabling technologies.

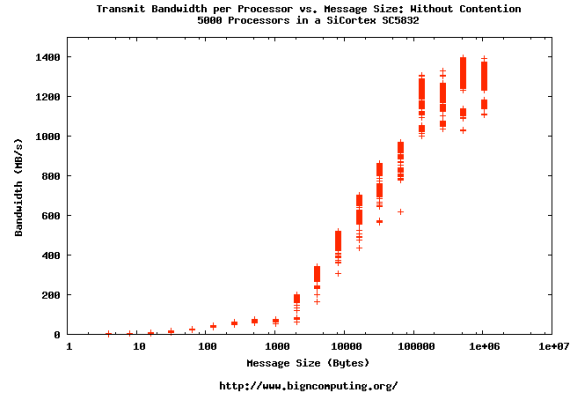
Given such low routing redundancy, it may be surprising to realize that even in the face of a link or node failure, every node in a degree  $d$  Kautz digraph has  $d$  independent routes to every other node [9]. Moreover, a link or node failure in a Kautz digraph increases the diameter by just 1.

An additional benefit of having multiple alternative paths is that the system can distribute traffic to minimize the impact of congestion and hot spots, without resorting to the complexity of adaptive routing.

In applications with irregular or broadly-distributed message traffic, communications performance is often limited by the number of available paths through the network’s narrowest bottleneck, called the bisection width. The best available lower bound [5] on the bisection width of a Kautz graph of degree  $d$  and diameter  $k$  is:

$$\frac{(d+1)d^k}{2k}$$

Table 5 compares the bisection width of a degree 3 Kautz graph, a 2-dimensional, and a 3-dimensional torus. The Kautz graph offers significantly higher bisection width.



**Figure 4. MPI.Send/MPI.Recv Delivered Bandwidth: no fabric contention**

This translates into better bisection *bandwidth* which is key to operations like matrix transpose and global sort.

## 6 Operating Results

SiCortex has had systems in the field for almost a year now. Our experience with these systems and with the engineering prototypes indicates that the architecture meets our initial reliability and robustness goals.

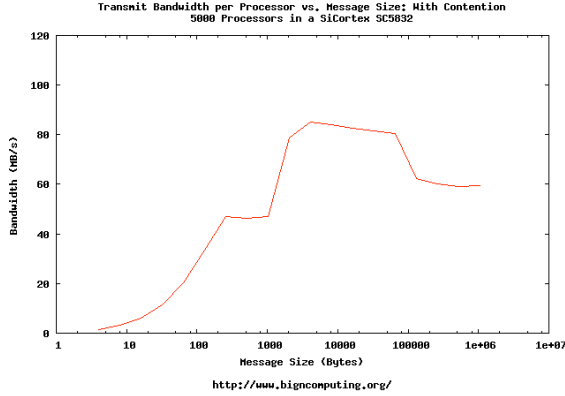
The fabric performance has been consistent with our initial models, though there have been a few surprises.

### 6.1 Send/Receive Bandwidth and Latency

The HPC Random Ring Bandwidth measure for the SC5832 is approximately 50MB. However, this characterizes a peculiar communication pattern at a single message size. In an effort to develop guidance for programmers, we have measured delivered bandwidth between processors across a range of message sizes and under conditions of high contention and no contention at all.

The available transmit bandwidth under no contention measures the average unidirectional message bandwidth across randomly selected pairs of processors among the 5832 processors in the system. The measurement is taken while all other processors are idle. Figure 6.1 shows a scatter plot of measured bandwidth vs. message size. Each point in the plot represents a trial for a randomly selected pair and a message size.

The peak of 1.4GB/s is determined by the throughput of the node’s DMA engine. The spread over samples at the same message size is caused by differences in the network



**Figure 5. MPI\_Send/MPI\_Recv Delivered Bandwidth: high fabric contention**

path length for each randomly chosen pair as it affects the latency of rendezvous response messages.

The available transmit bandwidth under high contention measures the average unidirectional message bandwidth across 2500 randomly assigned pairs of processors. The measurement is taken while all processors are exchanging messages: the network under these conditions is as busy as possible.

The first region of the curve, for messages ranging from 4 bytes to about 128 bytes, is entirely determined by the round-trip delay for a single message exchange. The payload for messages from 4 to 128 bytes in length fits entirely in one fabric level packet. The bandwidth curve reflects a constant initial overhead for message origination that is independent of message length. For message lengths from 129 to 1024, each message must be divided into multiple packets, each carrying up to 128 bytes of payload. Again, the round-trip time dominates, and since the cost per packet is relatively constant, the delivered bandwidth remains constant as well. As a first order approximation, all messages using the eager mode follow a simple model:

$$BW_{eager}(l) = \frac{l}{t_{ohd} \lceil \frac{l-128}{128} \rceil} : 128 < l < 1024$$

The measurements are fairly consistent with a value of  $t_{ohd}$  of about  $3 \mu S$ .

The message handling protocol converts from “eager” message delivery to a “rendezvous” discipline for messages longer than 1024 bytes. This causes the bandwidth vs. size curve to rise dramatically. The delivered bandwidth for a 4K message is 85MB/s, almost double the rate for 1K messages. The plateau of about 85MB/s and the sudden drop at 128K messages to 60MB/s is caused by contention in the network.

An intuitive argument would suggest that each processor shares its node’s out-bound links with the other five processors on the node. Each node has three out-bound links of 2GB/s each. Accounting for overhead from the 8:10 code and overhead bytes in each packet, the ultimate available bandwidth out of a node is about  $3 * 2GB/s * 0.8 * \frac{16}{19} = 4GB/s$ . This out-bound bandwidth is divided among the six processors, so that each share is about 670MB/s.

But the network diameter for the SC5832 is 6. The average message travels over about 5 hops. Consider a simple model. Each active process  $p$  originates a message stream that must at some point occupy a number of links equal to the average path length  $s$ . At the same time, the number of links in the system is fixed at  $3 \cdot N$  where  $N$  is the number of network nodes. So the average number of message streams contending for the same link is

$$C = \frac{ps}{3N}$$

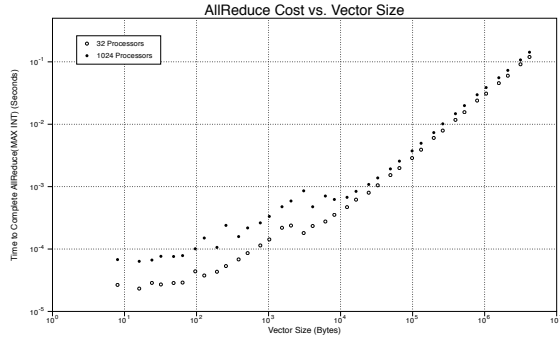
For the 5000 process experiment in Figure 6.1,  $C$  is approximately 8.6. Thus the model predicts an available bandwidth of about 80MB/s, a reasonable agreement with the collected data up to about 64KB long messages.

The drop in bandwidth at 128KB messages is caused by an optimization made in the protocol to reduce the incidence of “hot spots” in the network. Messages longer than 128K are split into three separate streams and sent “multi-rail” fashion over completely independent paths. This, however, increases the average path length  $s$  from 5 to 5.7. This changes the subscription rate  $C$  from 8.6 to 9.7 and the available bandwidth per processor to 70MB/s. The adverse performance impact may not be offset by the hot spot avoidance in low contention operation. We will be studying this in the near future.

## 6.2 AllReduce Cost

The SC5832 MPI library, based on MPICH, implements the AllReduce collectives using an algorithm requiring  $O(\log p)$  message exchanges between the  $p$  processes. The cost of the exchanges is  $O(l)$  where  $l$  is the number of elements in the vector. Figure 6.2 shows the cost of an integer MIN operation over a range of vector sizes. We do not yet have an explanation for the apparent “noisiness” of the measurement for vectors shorter than 10KB.

The cost of the allreduce operation for large vectors is fairly constant vs. communicator size from 16 to 1024 processors. The bimodal nature of the AllReduce algorithm [6] creates gives rise to a clear advantage for complements that are powers-of-two. Communicator sizes of  $2^k, 4 \leq k \leq 10$  result in AllReduce latencies of 120 to 138 milliseconds, while other sizes suffer an additional 40 millisecond cost.



**Figure 6. MPI AllReduce Execution Time vs. Vector Size**

### 6.3 Error Recovery

To date, we have not witnessed isolated bit errors in SiCortex systems either in the laboratory or the field. We have, however witnessed clustered bit errors caused by power supply misbehavior in some early debug units.

Power is supplied to each chip in the system through a hierarchical conversion network. Line current is converted to 48V (nominal) DC and distributed to each of the SC5832's 36 modules. Local DC to DC converters on the module step the 48V bus down to 9.7V. Point-of-load (POL) regulators convert the 9.7V bus to the various voltages needed by the processor cores, the IO drivers, the DRAMs, and the fabric IO transceivers.

In the link failures we have seen, transient errors in the POL regulator supplying current to a group of fabric drivers would cause the affected link to lose synchronization. Synchronization loss causes several packets in sequence to be corrupted which triggers the automatic retry logic. Because synchronization has been lost, all retry attempts fail until system software intervenes to resynchronize the broken link. Once synchronization has been achieved, the next replay attempt succeeds and all packets are delivered to their intended destination. No data is lost.

In actual operation, isolated single bit errors are so rare that we have not observed a sufficient number to derive a bit error rate with any degree of confidence. Our experience can only suggest that the bit error rate is very low, far below 1 in  $10^{18}$ . The extremely low bit error rate is consistent with (but not necessarily assured by) extremely clean eye-patterns we have captured at the end of several representative links.

On several occasions we have observed hard link failures, usually due to a mechanical defect in early engineering prototypes. In each case we were able to reconfigure the Kautz graph around the failed link. In operation, we have removed nodes from system configurations due to DIMM

failures. In these cases, too, the Kautz graph was reconfigured around the failed node with minimal impact to the operation of the system.

## 7 Conclusion

Our experience in actual operation indicate that the Kautz topology delivers good performance relative to its cost and complexity. We have also seen that the link and switch architecture have held up well in the presence of long-lived transient errors and permanent link failure. The contention in the network due to the long average path length was not anticipated in our earlier studies and has been as disappointing as it was inevitable. However the actual performance of the network under heavy load has clearly demonstrated the utility of the fabric design, as evidenced by delivered bandwidth on transpose operations (measured  $> 210\text{GB/sec}$  for HPCC PTRANS), random access benchmarks ( $> 5\text{ GUPS}$ , optimized), as well as more complex applications and operations,

## References

- [1] M. Reilly, L. C. Stewart, J. Leonard, and D. Gingold. (2006, Dec.) Sicortex technical summary. [Online]. Available: [http://www.sicortex.com/press/sicortex-tech\\_summary.pdf](http://www.sicortex.com/press/sicortex-tech_summary.pdf)
- [2] L. C. Stewart and D. Gingold. (2006, Dec.) A new generation of cluster interconnect. [Online]. Available: [http://www.sicortex.com/press/sicortex-cluster\\_interconnect.pdf](http://www.sicortex.com/press/sicortex-cluster_interconnect.pdf)
- [3] B. Elspas, W. H. Kautz, and J. Turner, "Theory of cellular logic networks and machines," Stanford Research Institute, Tech. Rep. AFCRL-68-0668, 1968.
- [4] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. 36, no. 5, May 1987.
- [5] Rolim, Tvrdik, Trdlicka, and Vrto, "Bisecting de bruijn and kautz graphs," *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, vol. 85, 1998. [Online]. Available: [citeseer.ist.psu.edu/rolim98bisecting.html](http://citeseer.ist.psu.edu/rolim98bisecting.html)
- [6] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in MPICH," *International Journal of High Performance Computer Applications*, vol. 19, no. 1, pp. 49–66, 2005.