The SP1 High-Performance Switch

Craig B. Stunkel Dennis G. Shea Don G. Grice[†]
Peter H. Hochschild Michael Tsao

IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

[†]IBM Highly Parallel Supercomputing Systems Laboratory
Neighborhood Road
Kingston, NY 12401

Abstract

The IBM Scalable POWERparallel Systems[1] 9076 SP1[1] connects RISC System/6000[1] processors via a communication network called the High-Performance Switch. This Switch—based upon the Vulcan parallel processor [1]—incorporates a number of unusual features to enhance reliability, diagnose faults, and simplify cabling. This paper examines the SP1 Switch architecture and implementation and overviews the Switch support software.

[1]A trademark or registered trademark of International Business Machines

1 Introduction

The IBM Scalable POWERparallel Systems 9076 SP1 connects RISC System/6000 processors via a communication network called the High-Performance Switch, or simply the "Switch." In the SP1 system, the Switch currently provides connectivity for 8 to 64 RISC System/6000 nodes, and future releases will scale to connect hundreds or thousands of processors, I/O nodes, and gateways. The Switch network is capable of scaling its aggregate bandwidth linearly with respect to the number of nodes, and provides low latency, fault-tolerant communication. The network can leverage the latest processor technology.

The Switch—based upon the Vulcan architecture and Vulcan prototype [1] developed in the Parallel Systems department at IBM Research—incorporates a number of unusual features to scale aggregate bandwidth, enhance reliability, diagnose faults, and simplify cabling. It is a bidirectional multistage network, and is a unified data and service network driven by a single oscillator. The attempt is made to detect all network errors via CRC checking and component shadowing. Switching elements contain a dynamically allocated shared buffer for storing blocked packet bytes from any input port.

The Switch hardware architecture is discussed in §2, and the implementation of the SP1 High-Performance Switch is found in §3. An overview of the Switch support software is given in §4.

2 Switch Network Architecture

This section examines the major attributes of the SP1 Switch, including topology, synchronization, communication protocols, error detection, and network service. First, we introduce some definitions.

A network comprises two device types—*switching elements* and *nodes*—interconnected by communication links. Nodes are non-switching devices such as processors, I/O nodes, and gateways. Switching elements contain a plurality of input ports and a plurality of output ports, and provide means of passing data arriving at an input port to an appropriate output port.

Nodes send messages to other nodes by breaking messages into *packets* and injecting these packets into the network. Each packet contains route information examined by switching elements to forward the packet correctly to its destination.

The smallest unit on which flow control is performed is called a flow-control digit, or *flit* [2], which is one byte in the SP1 Switch. The width of the data transmitted by an output port is also one byte.

2.1 Network topology

SP1 networks are bidirectional multistage interconnection networks (MIN's). In a bidirectional MIN [3, 4] each communication link comprises two channels which carry data in

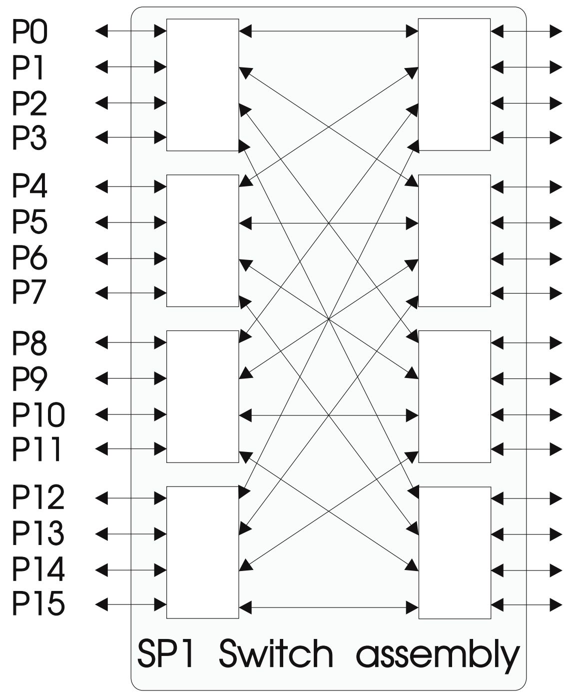

Figure 1: An SP1 frame—a 16 node bidirectional MIN

opposite directions. MIN's are capable of scaling *bisection bandwidth* [5] linearly with the number of nodes while maintaining a fixed number of communication ports per switching element. MIN's with hundreds or thousands of nodes are richly connected and require some long communication links to support large numbers of nodes. Thus, the channels employ a protocol that maintains bandwidth over long links (see §2.4.1).

Figure 1 illustrates a 16 node bidirectional MIN constructed from 4-way to 4-way bidirectional switch elements. The switching elements—physically 8 input, 8 output devices—are wired as bidirectional 4-way to 4-way elements. Each switching element can forward packets from any input port to any output port as directed by the packet's route flits. The bidirectionality of the Switch naturally incorporates increasing redundancy as the size of the network grows. Bidirectionality facilitates more modular systems—it is simpler to build systems in a wide range of sizes when each system is not required to wrap links from the final stage's output ports back to the node input ports (at differing stages dependent upon system size). Bidirectional links also simplify our service software (§4.1) by allowing the software to fully test a single link or device while relying solely on proven intermediate devices.

A system of up to 64 processors (4 frames) is currently supported by connecting the

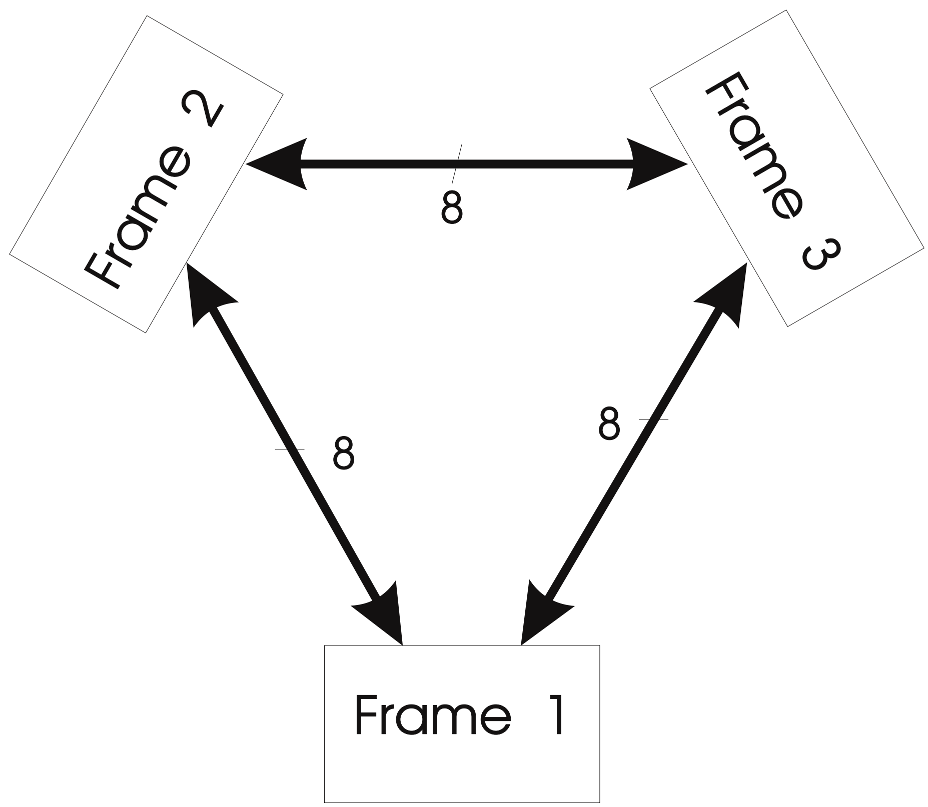

Figure 2: SP1 48-way system interconnection

frames via the links on the unused sides of their switch assemblies. Figures 2 and 3 illustrate 48-way and 64-way system topologies. For all SP1 configurations, there are at least 4 usable paths between each pair of nodes, except for node pairs that are directly attached to the same switching element. Node pairs are largely equidistant, simplifying the scheduler's assignment problem. Any shortest path route selection algorithm is deadlock-free. Thus, we can take full advantage of the redundant paths in the Switch.

SP releases supporting 128 nodes or more will scale Switch bandwidth by cascading switch assemblies analogous to an SW Banyan [6]. These larger SP1 Switch topologies are

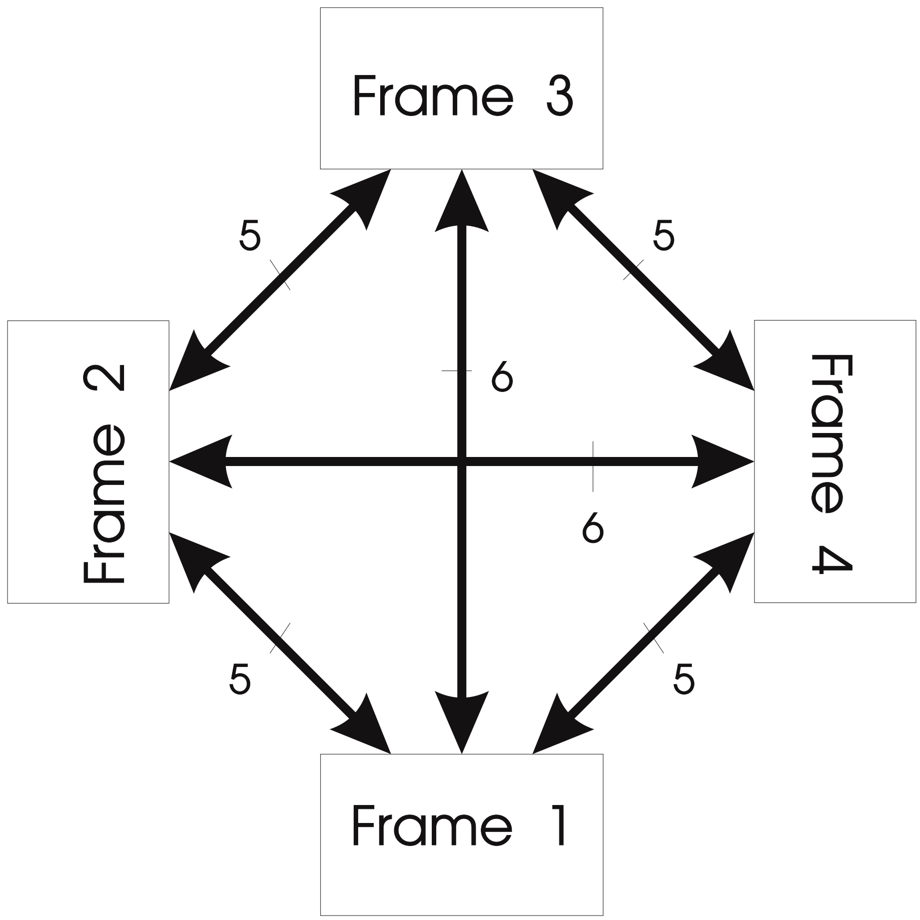

Figure 3: SP1 64-way system interconnection

also related to practical implementations of fat-tree networks [7] derived from Leiserson's idealized fat-trees [8].

2.2 Single global oscillator

The Switch operates synchronously—all network devices are driven by an image of a single oscillator. It is not necessary to adjust the phase of each device's oscillator image. Instead, each device adjusts the phase of *data* by a process we call *tuning* (see §4.1), using adjustable delay elements. This allows both the oscillator signal links and the data links to be of arbitrary length up to limits posed by reliability or protocol considerations.

Synchronous operation allows the Switch to conduct packet error checking on a time basis, rather than a packet basis. Thus, stuck or open faults are detected in a timely fashion. Synchronous operation enables the use of chip replication for the purpose of fault detection.

Synchronous operation, in conjunction with appropriate hardware mechanisms, facilitates the achievement of a closely synchronized and non-drifting *global time*, trivializing the well-known clock synchronization problem [9]. Each device in the Switch network accesses this time via its local copy: a loadable incrementer called the Absolute Time Counter (ATC). During the initialization of the network, the ATC within each device is synchronized to within one cycle of one neighbor device's ATC.

Accurate time stamps on activities occurring within a parallel partition can provide both functional and debug opportunities. For example, a global event trace from a parallel application can be correctly synthesized from the individual event traces.

2.3 Packet flow control

The SP1 method of packet transfer is related to *wormhole routing* [2]. In wormhole routing, each flit of a packet is advanced to the appropriate output port as soon is it arrives at a switching element input port. When the head of a packet is blocked, the flits are buffered in place. As soon as the output port is free, packet transfer resumes. The output port is chosen according to route information contained in one of the first flits of the packet. For SP1, the second flit of each packet holds the current route information.

The Switch flow control differs from canonical wormhole routing in that packet flits are not typically buffered in place when a packet is blocked. In addition to small buffers at each input port, each switching element contains a relatively large shared buffer called the central queue. The central queue is dynamically allocated to blocked packet flits from any input port. To highlight this departure from standard wormhole definition, we call the Switch flow control method *buffered wormhole routing*. Details of central queue allocation are covered in §3.1.4.

2.4 Unified data and service network

Most commercial multicomputers utilize one network for normal node-to-node (data) communication and another for network service. This adds to the complexity of the system design and reduces its reliability. The Switch incorporates a unified data and service network, permitting network service operations to utilize a high-speed network. The entire

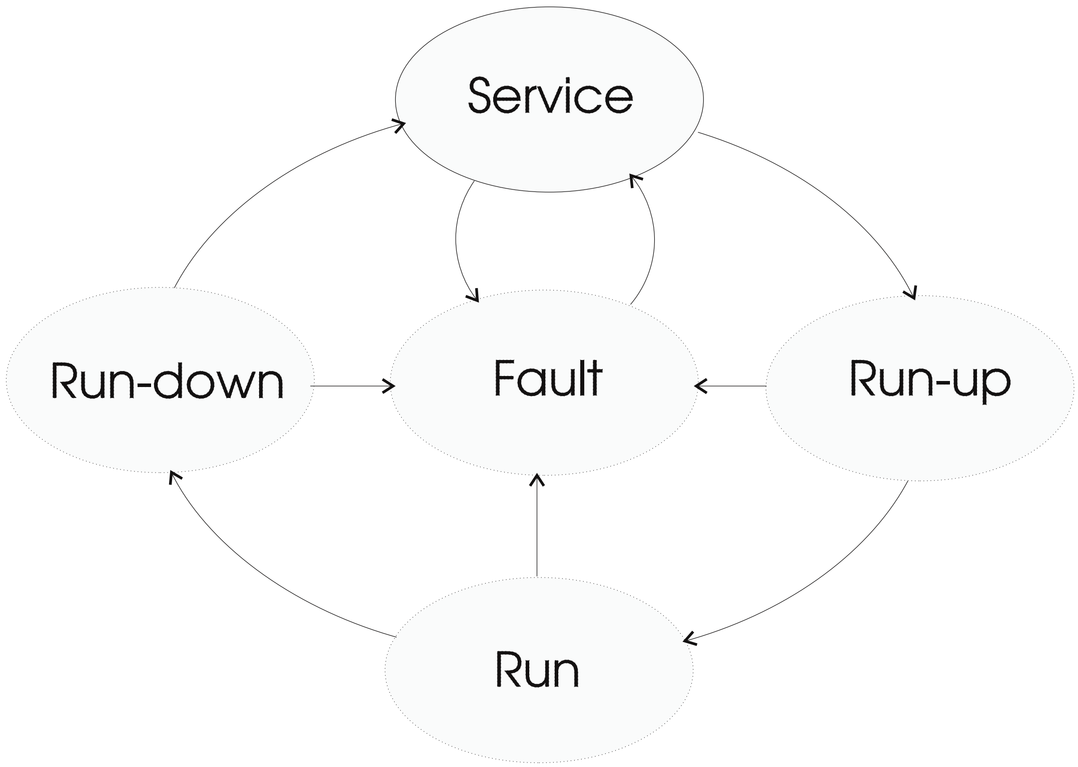

Figure 4: Switch mode transition diagram

network typically operates in one of two *modes*, each mode encompassing a unique communication protocol. These modes are *service* mode for servicing the network, and *run* mode for normal data transfer. Figure 4 displays the modes of the Switch.

2.4.1 Run mode protocol

The Switch run mode link protocol is token-based to accommodate link propagation delays of several cycles. As shown in Figure 5, each input port transmits tokens to the upstream output port. Each token represents the input port's readiness to accept one flit. An output port spends one token each time it transmits a flit. To maintain bandwidth for a link with a propagation delay of p cycles, the downstream input port must contain a buffer of at least size $2p$. The Switch implementation contains buffers of at least 31 flits at any input port, permitting full bandwidth communication for $p \leq 15$.

Packets vary in length up to 255 flits. An active tag indicates a valid packet flit. The first flit holds the packet length, as shown in Figure 6. The next flit or flits contain route

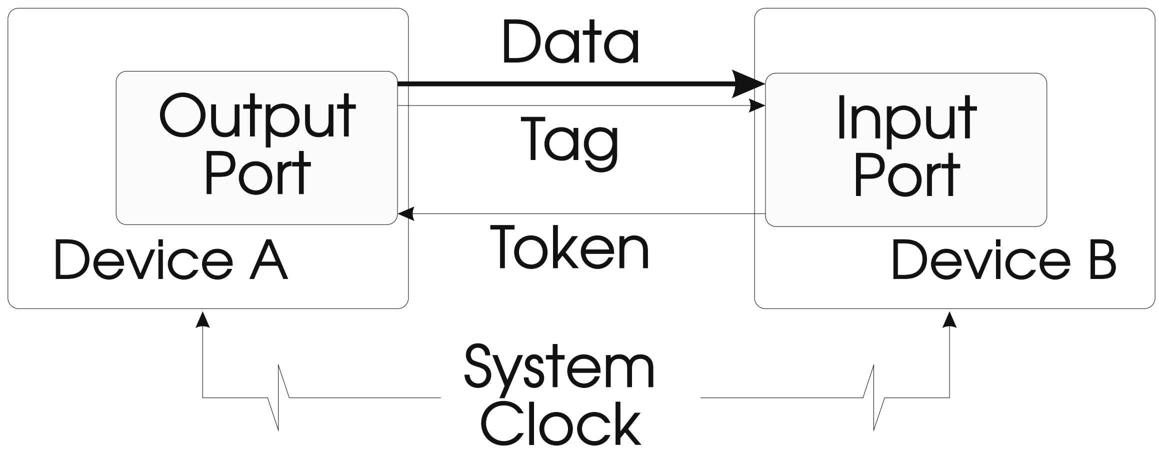

Figure 5: Switch ports connected by a channel

Figure 6: The SP1 Switch run mode packet. Length $= r + n + 1$

information. The rest of the packet is data.

Time is divided into 64-cycle frames. During the last 2 cycles of each frame, both tag and data carry CRC codes. The token signal is encoded in two-cycle frames. A frame sequence of 0-1 represents no tokens, and 1-0 represents two tokens. The 0-0 and 1-1 encodings are invalid to detect token errors.

Routing is source-based (i.e., the packet route is determined by the source node). Each switching element examines the first route flit to determine the appropriate output port for packet forwarding. Used route flits are discarded (the length flit is correspondingly decremented). A packet contains no route flits when it reaches its destination.

2.4.2 Service mode protocol

Service mode employs a simple circuit-switched protocol to reliably communicate with selected network devices. Devices are controlled by an arbitrarily elected service node.

Service nodes control the network via service messages of the format shown in Figure 7. The command field identifies the type of service message. Each device on the current circuit executes the service command if the MID field matches the device's Message Identifier

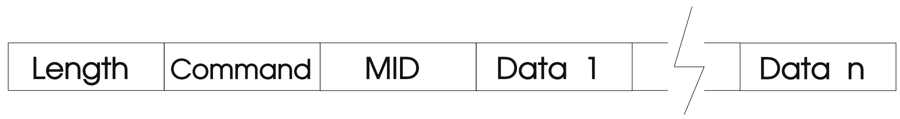

Figure 7: Service message format. Length = $n + 3$

Register (MIR). The MIR can be altered via a special service message. The execution of some service commands alters data flits (e.g., status query commands).

A subset of these messages provide circuit switching operations. For example, the `set-pass-thru` message instructs a switching element to relay succeeding service messages to a specified set of output ports after it conditionally executes the message. Broadcast or multicast trees may be constructed. Switching elements provide two pass-thru circuits (forward and backward pass-thru) for the formation of loop circuits. Loop circuits pass information back to the service node. The start time and duration of the next run mode are chosen during service mode and broadcast to all devices via service messages.

2.4.3 Mode transitions

During normal operation, devices cycle through the outer modes of Figure 4 in synchrony. The start time and duration of the next run mode are chosen during service mode and broadcast to all devices via service messages.

Run-down mode is a 16-cycle transitional mode that drains the links. Service mode always begins in an idle state. Run-up mode is a 16-cycle transitional mode that primes the links with the identical data present on the links at the end of the previous Run mode. To accomplish this, during run mode, each device continually buffers the previous 16 cycles of transmitted data. This limits the propagation delay for a link to 15 cycles, because valid signals must have been sampled by every link input before run mode begins.

Service mode may be used to initiate service actions such as adding or removing a device, or to collect monitoring or debug information. Monitoring information can thus be collected without perturbing run mode operation.

2.5 Error detection

The objective in the SP1 Switch is to detect and record the cause of every network error. Every switching element (switch chip) in the system is *shadowed*. Each output pin of a switch chip is implemented as a bidirectional I/O, and the identical output pins of two switch chips are electrically connected. One chip, designated the master, drives its signal outputs. The other (slave) checks the value received on its corresponding inputs against the value it would have driven, were it the master.

The signals transmitted between networks elements cannot be checked by this shadowing methodology. Instead, all transmitted signals are checked with CRC codes as described in §2.4.1. Furthermore, it is not straightforward or cost-effective to shadow the processor nodes; thus additional CRC codes are embedded into each packet at the source node, and are checked at the destination to ensure reliable packet transmission. When an error is detected by any device, that device transfers to fault mode (recall Figure 4) and notifies its neighbors. The service node then collects the error status and restarts normal operation.

3 SP1 Switch Implementation

An SP1 system is composed of frames containing up to 16 processors and one switch assembly that implements the two stage network building block shown in Figure 1. The Switch is built upon two custom ASIC chips: the Vulcan switch chip, and the Vulcan MSMU (Memory & Switch Management Unit) [1]. The 8×8 switch chip is the basic switching element of the network, while the MSMU functions as the interface between a switch chip and the RISC System/6000 processor. The Switch operates at 40 MHz, providing peak bandwidth of 40 megabytes per seconds over both byte-wide channels of each communication link. This section describes the major implemented components of the Switch, including the switching element, the switch assembly, the processor communication adapter and the link technology.

3.1 The Vulcan switch chip

The Vulcan switch chip, illustrated in Figure 8, contains 8 receiver modules and 8 transmitter modules, an unbuffered crossbar, and the central queue. All ports are one flit (one byte) wide. In the absence of contention, packet flits incur 5 cycles of latency cutting through the chip via the crossbar path (§3.1.2). The design of this chip was originally targeted for the TF-1 machine [10].

3.1.1 Receivers

The switch chip contains eight identical receiver modules, one associated with each of the eight input ports. The receiver module performs five major functions: (1) administering the link flow control protocol, (2) checking the link CRC codes, (3) buffering incoming data, (4) decoding packet route information, and (5) deserializing incoming packets into 8-flit *chunks* when the packet is blocked. Buffering is accomplished with a 31-flit first-in-first-out (FIFO) queue. When an incoming packet encounters no contention for the selected output port, packets are immediately forwarded via the switch chip's crossbar. When the packet is blocked, 8-flit packet chunks are sent to the central queue for temporary buffering.

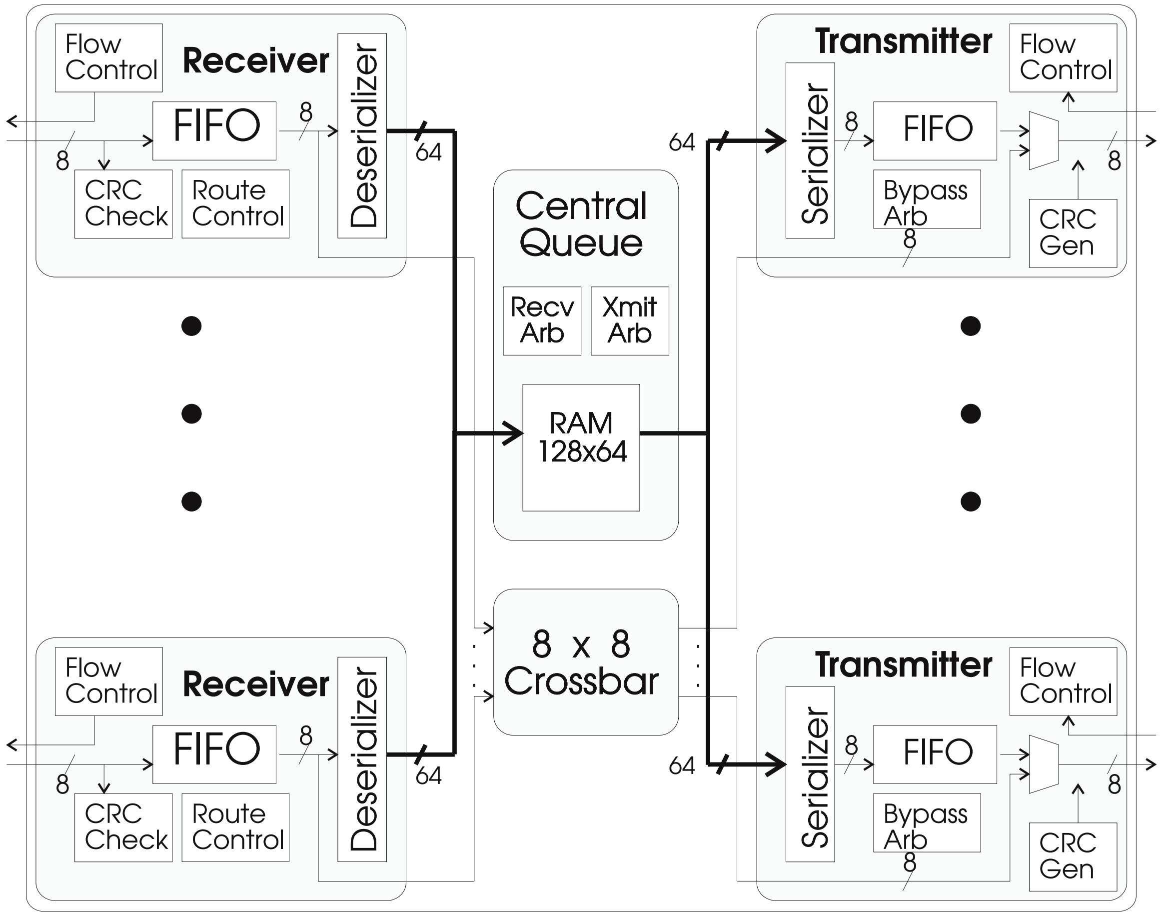

Figure 8: Vulcan switch chip organization

Route determination

In SP1, routing is source-based and deterministic: the switching elements do not make adaptive routing decisions. Each route flit (byte) contains two 3-bit route fields—residing in bits 6-4 and bits 2-0—for directing two successive switch chips. Bit 7 controls which field holds the active route. When bit 7 is cleared, bits 6-4 determine the output port, and the receiver subsequently sets bit 7 before forwarding the route flit. When bit 7 is set, bits 2-0 determine the output port, and the receiver subsequently discards that route flit and decrements the length flit of the packet.

3.1.2 Crossbar Routing

The switch chip incorporates an unbuffered logical crossbar that allows packets to pass directly from the receivers to the transmitters. These flit-serial crossbar paths permit packets to pass through the chip with low latency whenever there is no contention for the output port. As soon as a receiver decodes the route information carried by an incoming packet, it asserts a crossbar request to the appropriate transmitter. If the crossbar request is not granted by the time the entire first chunk of the packet has been received, the crossbar request is dropped (and hence the packet will go to the central queue). Each transmitter arbitrates crossbar requests on a least-recently-served basis. A transmitter will honor no crossbar request if it is already transmitting a packet or if it has packet chunks stored in the central queue.

3.1.3 Transmitters

There are eight transmitter modules, one corresponding to each output port. When the central queue contains packet chunks destined for a transmitter, that transmitter requests the next packet chunk. Transmitter modules are served by the central queue in a least-recently-served fashion. As long as data is available, one transmitter is served each clock cycle. The transmitter accepts packet chunks from the central queue, serializes them, buffers them in a 7-flit output FIFO, and transmits them to the link in accordance with the link flow control protocol. The transmitter is also responsible for computing and transmitting the CRC codes.

3.1.4 Central Queue

The central queue implements a buffered, time-multiplexed 8-way router. It accepts packet chunks from the receivers, stores them, and eventually passes them to the appropriate transmitters. The Vulcan switch chip central buffering scheme is similar to the centrally-buffered, dynamically-allocated switch described by Tamir and Frazier [11], but the Vulcan chip allocates central queue space on a chunk basis instead of a packet basis.

The central queue stores packets until they can be transmitted. The storage, a 128 by 64-bit dual-port RAM, holds up to 128 eight-flit packet chunks. The queue does not reserve a fixed amount of space for each output port; storage is allocated dynamically according to demand.

Stored packets are queued in FIFO order on eight linked lists, one list corresponding to each of the eight switch output ports. These lists are stored in a dual-port 128 by 7-bit RAM. Furthermore, for each packet, there is a linked list consisting of the constituent chunks. These lists are stored in another dual-port 128 by 7-bit RAM.

As long as queue space is available, one receiver can be served every cycle. As long as data is available, one transmitter is also served every cycle. Thus the bandwidth through the central queue matches the bandwidth into and out of the switch chip: eight flits per cycle.

To prevent starvation, receivers requesting service from the central queue are divided into two groups: those offering *critical* chunks and those offering *non-critical* chunks. A critical chunk is one that is ready for immediate forwarding to its destination transmitter. The first chunk of a new packet is critical if no other packets are queued for its transmitter. A continuation chunk is critical if all previous chunks of the packet have been forwarded to the transmitter. Chunks not immediately ready for forwarding are called non-critical.

One chunk of central queue storage is reserved for each transmitter. The remaining central queue storage is shared. Whenever a receiver has a non-critical chunk and space is available in the shared pool, the receiver requests service from the central queue. Whenever a receiver has a critical chunk, it requests service. Critical chunks are stored in the destination transmitter's reserved queue slot. Priority is assigned to the receivers on a least-recently-served basis.

3.2 Switch assembly

The switch assembly housed in the SP1 frame contains hardware to provide power, clock and supervisor functions for the Switch. The main component is the switch board that implements the interconnection shown in Figure 1. It provides any-to-any paths for 32 external connections. All switch chips are shadowed; thus the board contains 16 switch chips. The Vulcan switch chip pin assignments were chosen such that all the pins associated with ports 0-3 are on one half, and ports 4-7 are on the other. Taking advantage of this, ports 0-3 are used for inter-board links, and ports 4-7 for intra-board links. This minimizes the wire cross-overs on the board. All inter-board signals, both data and clock, are driven as 100K ECL differential pairs. Additionally, on-board system clock trees are also maintained in ECL.

3.3 Link technology

All communication in SP1 uses point-to-point bidirectional communication links that consist of two channels, each carrying data in opposite directions to their respective output port and input port pairs. A description of the link protocol was given in §2.4.1. For each channel, the actual transmission is via a set of 10 signal lines, 8 for data, one for tag and one for the token as was shown in Figure 5. For all cases where the link is driven off board, 100K ECL differential pairs are used. Logic level translation for data bits is provided by a specially designed low-skew 20 bit buffer with 10 bits in each direction.

3.4 Processor communication adapter

Each node of an SP1 incorporates a RISC System/6000 processor running at 62.5 MHz. The High-Performance Switch adapter card connects the processor's microchannel to output and input ports of the Switch network via the MSMU.

The MSMU provides communication services commonly found in the physical, datalink and network layers of most communication adapters. One output port and one input port implement the link level protocol. Memory-mapped FIFO's produce and consume switch packets to and from the processor and I/O nodes. Two FIFO pairs are utilized, one for run mode communication and one for service mode. The run mode FIFO's are 2K bytes in each direction.

The SP1's MPL/p message-passing software (also commonly known as EUI-h), currently achieves 8.3 megabytes per second with a latency of 28 microseconds. This communication library maps the MSMU registers into user memory space. SP1 node to node performance has been benchmarked and compared to several commercial machines by researchers at Argonne [12].

4 Support Software

The Switch network function is supported by service software known as the "Worm." This service software is coupled with the Route Table Generator, which examines the state of the Switch as determined by the Worm execution, and generates valid routes from every node to any other node. This section overviews the Worm and the Route Table Generator.

4.1 The Worm

The Worm package is the SP1 Switch network service software [13]. It executes on an arbitrarily chosen "service" or "primary" node and controls the network during service mode. The Worm package provides initialization, tuning, global time synchronization, fault determination, and diagnostic services to all devices within or attached to the Switch. It further supplies routines that facilitate direct control or interrogation of these devices.

The Worm is designed to service an arbitrary pre-specified network of Switch devices. The only inherent topology assumption is that each cable in the network is bidirectional. An input file describes the expected network topology. The Worm searches the network in a breadth-first manner, entering status information about each device and link in a database. This database is subsequently used by the Route Table Generator (§4.2) to generate valid node-to-node routes.

Tuning

A particularly crucial Worm function is tuning of link data arrival. In the SP1 Switch, all tuning is software-controlled, and utilizes delay logic at each network output port. As a contrasting example, the Monarch parallel processor incorporated a hardwired method for tuning a serial link [14]. Two service commands enable the service software to synchronize data arrival for a channel. One command (tuning-control) adds an identical amount of delay to each of the Data and Tag outputs, and selects a separate delay adder for the output port's Token input. A second command (cycle-counting) determines the number of cycles taken to: (a) send a pulse out of the output port on a Data or Tag signal, (b) receive the pulse at the downstream input port, and *reflect* it back onto the Token signal, (c) receive the reflected pulse back on the incoming Token signal. "Reflection" at the input port is an action utilized specifically during tuning that: (1) clocks the 9 input signals into a register R, and (2) OR's the output of R into its Token output flip-flop.

A number of cycle-counting experiments—for carefully chosen delays—are performed to determine the best delay adder for the output and input signals for synchronizing signal reception. The simplified principle: as the Worm code monotonically and incrementally increases the delay on an outgoing (or incoming) pulse, at some point the number of cycles recorded by a subsequent cycle-counting command will increase. At this point, the selected output port tuning delays are near the *worst* setting (near the set-up/hold region of receiving register logic.) Hence, the proper delay setting is $\approx \frac{1}{2}$ clock cycles away from the currently selected delay.

4.2 Route generation

System software obtains routes by table lookup. The table may contain multiple entries per source-destination pair to facilitate source-based random routing. The Route Table Generator, or Router, produces this route table. Just as for the Worm, no topology is assumed. Instead, the Router uses the set of links and devices found acceptable by the Worm, and builds breadth-first spanning trees [15] using the resulting network graph to find shortest-path routes. A simple load-balancing strategy is used to ensure that links are included in node-to-node paths in a balanced manner.

5 Conclusion

The SP1 High-Performance Switch provides a low latency, high bandwidth communication medium that can scale bisection bandwidth for systems containing up to hundreds or thousands of nodes. The Switch is a bidirectional MIN, and provides at least 4 usable redundant paths for most pairs of communicating nodes. Data and service operations share a single network. The network is synchronous, with mechanisms for achieving a closely synchronized global time visible to each node. The basic switching element is the Vulcan switch chip, which contains a unique central buffering scheme for reducing the impact of network contention.

References

[1] C. B. Stunkel, D. G. Shea, B. Abali, M. M. Denneau, P. H. Hochschild, D. J. Joseph, B. J. Nathanson, M. Tsao, and P. R. Varker, "Architecture and implementation of Vulcan," in *Proc. 8th Int. Parallel Processing Symp.*, 1994. (An extended version of

this paper is also available as an IBM Technical Report from the IBM T. J. Watson Research Center, 1994.).

[2] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, pp. 194–205, March 1992.

[3] G. J. Lipovski and M. Malek, *Parallel Computing: Theory and Comparisons.* New York, NY: Wiley & Sons, 1987.

[4] I. D. Scherson and C.-H. Chien, "Least common ancestor networks," in *Proc. 7th Int. Parallel Processing Symp.*, pp. 507–513, 1993.

[5] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. on Computers*, vol. 39, pp. 775–785, June 1990.

[6] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," in *Proc. 1st Ann. Symp. on Computer Architecture*, pp. 21–28, 1973.

[7] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. St. Pierre, D. S. Wells, M. C. Wong, S.-W. Yang, and R. Zak, "The network architecture of the Connection Machine CM-5," in *Proc. 1992 Symp. Parallel Algorithms and Architectures*, pp. 272–285, ACM, 1992.

[8] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Trans. on Computers*, vol. C-34, pp. 892–901, Oct. 1985.

[9] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, pp. 558–565, July 1978.

[10] M. M. Denneau, P. H. Hochschild, and G. Schichman, "The switching network of the TF-1 parallel supercomputer," *Supercomputing Magazine*, pp. 7–10, Winter 1988.

[11] Y. Tamir and G. L. Frazier, "Hardware support for high-priority traffic in VLSI communication switches," *J. of Parallel and Distributed Computing*, vol. 14, pp. 402–416, April 1992.

[12] W. Gropp, "Early experiences with the IBM SP1 and the high-performance switch," tech. rep. ANL-93/41, Argonne National Laboratory, 1993.

[13] C. B. Stunkel, "The Vulcan Worm," tech. rep., IBM T. J. Watson Research Center, 1994.

[14] R. D. Rettberg, W. R. Crowther, P. P. Carvey, and R. S. Tomlinson, "The Monarch parallel processor hardware design," *IEEE Computer*, vol. 23, pp. 18–30, April 1990.

[15] E. M. Reingold, J. Nievergelt, and N. Deo, *Algorithms and Complexity.* Englewood Cliffs, NJ: Prentice-Hall, 1977.