**LABORATORY FOR COMPUTER SCIENCE** — MASSACHUSETTS INSTITUTE OF TECHNOLOGY

# The Monsoon Interconnection Network

Computation Structures Group Memo 340
January, 1991

**Christopher F. Joerg**
**Andy Boughton**

Appeared in: *Procedings of the International Conference on Computer Design*, October 1991.

This report describes research done at the Laboratory for Computer Science of the Massachusetts Institute of Technology. Funding for the Laboratory is provided in part by the Advanced Research Projects Agency of the Department of Defense under the Office of Naval Research contract N00014-89-J-1988.

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

# The Monsoon Interconnection Network[†]

Christopher Joerg[‡] and Andy Boughton
Lab for Computer Science
Massachusetts Institute of Technology
Cambridge MA 02139

**Abstract**

*This paper describes the interconnection network for Monsoon, a parallel processing dataflow computer. This network provides reliable, high bandwidth, low latency communication between the nodes of Monsoon. The major components of this network are two gate arrays: PaRC and the DLC. PaRC (Packet switched Routing Chip) is a CMOS gate array that implements a 4x4 packet routing switch that provides a high raw bandwidth (800Mbits/sec) to each port. PaRC is designed so that much of this bandwidth can be utilized. The DLC (Data Link Chip) is a high speed ECL gate array that reliably transfers data between boards.*

## 1 Introduction

Monsoon [5],[6] is a dataflow multiprocessor currently being built by MIT and Motorola. There are two types of nodes in Monsoon: highly pipelined processing elements and memory modules that support I-structure storage. The nodes communicate solely by passing fixed sized messages known as tokens; this is done using a packet switched interconnection network. A node uses split-phase transactions to read memory locations on other nodes. This allows the first node to perform other useful work while waiting for the response.

This paper describes the Monsoon interconnection network.

### 1.1 Network Requirements

A major requirement of this network is that it provides high bandwidth to and from each node. We do not want the network to be a bottleneck on this machine, so the network must provide each node with more sustained bandwidth than most programs will need. It is also important that the bandwidth per node does not significantly decrease as the size of the machine is increased. In other words the bandwidth of the network must grow proportionally with the number of nodes in the machine. This problem is made more difficult by the fact that the communication a node performs is uniformly distributed around the machine. This means that we can not assume that much of a node's communication will be local.

[†]The research described in this paper was supported in part by the Defense Advanced Research Projects Agency under Office of Naval Research contract N00014-84-K-0099.

[‡]Supported in part by a graduate fellowship from the National Science Foundation.

It is also important that this network is very reliable. This has two meanings. First, since recovering from errors will be very expensive, errors must rarely occur. Second, it important that if an error does occur it will be detected. Otherwise a corrupted data value could cause a program to complete normally but give an incorrect answer.

Also, the latency of the network should be kept to a minimum. Most algorithms go through periods where the potential parallelism is large, and periods where it is small. During periods of high parallelism, latency will be masked by working on other portions of the computation. However, during periods of low parallelism a processor may have very little to do until a request is answered. Since the time spent waiting for a reply may directly increase the execution time of the program, we want the network latency to be as short as possible. Since during low parallelism traffic may be light, and since we are especially concerned with latency during periods of low parallelism, we want to pay particular attention to minimizing the latency during light traffic.

In order to make certain software easier to write, there are two additional requirements on the network. The network must ensure point-to-point FIFO ordering. This means that if a node sends two tokens to the same destination, then the tokens must arrive in the same order in which they were sent. Additionally, the network should be able to provide, for a small number of selected packets, a fast acknowledgment that the packet has been received.

Lastly, the number of nodes in Monsoon may vary from just a few nodes to many hundreds. The network hardware must be designed such that networks of various sizes can be built out of the same components. These components should be flexible enough to allow many different network topologies to be built.

### 1.2 Network Characteristics

The above requirements, along with other characteristics of Monsoon helped define some of the high level characteristics of the network. The most basic decision was to make the network a packet switched network (also called store-and-forward). Since all communication in Monsoon consists of sending fixed sized tokens, it was natural to design a network that only transmits fixed sized packets. Another important reason packet switching was chosen was the high bandwidth that was needed. A "link" is a connection which sends data to or from a switch. These links provide connections between boards or even between racks. The cost of the links is often the major cost in a network. So to minimize the cost of the network we want to make the best possible use of the available bandwidth. A packet switched network will typically make better use of its raw link bandwidth than would a similar circuit switched network; and this is especially true as the size of the network grows. An important goal in the design of the switching chip was to make the best possible use of the raw bandwidth.

One advantage that circuit switched networks have is a shorter latency if the path for a message is not blocked. In a circuit switched network the message can quickly pass through each switch, while in a packet switched network a switch may wait until the packet is fully received before it sends the packet to the next stage. This cause of latency can be lessened by allowing the switch to send out a packet before it has fully arrived. This is known as virtual cut-through. When this is done, and a packet's path is clear, the latency for a packet switched network will be comparable to that of a circuit switched network.

A large Monsoon system will be spread out over many boards in many different cages. Providing a synchronous clock over such a large machine would be a difficult task. Since the nodes of this system are independent and may operate asynchronously, we did not want to provide a synchronous clock just for the network. For this reason each switching node will operate asynchronously. The cost of this decision is that arriving packets must be synchronized to the local clock; this may slightly increase the latency of a packet.

An indirect butterfly topology is used in this implementation of the Monsoon Network. With this topology, as the size of the network increases the bandwidth available to each processor remains fairly constant. Also, messages can go between any two nodes while passing through only a small number of switches. (Log $N$ switches, where $N$ is the number of nodes being connected.) This is important since each node that a message passes through adds to its latency. The only disadvantage of this topology is that the amount of hardware in the network grows as $O(N \log N)$. Some other networks grow only as $O(N)$, but for these networks the per processor bandwidth decreases significantly as the machine grows.

The design of the network consisted of two major parts: the switch and the links. The next two sections will describe these components. The switch is implemented by a CMOS gate array called PaRC. It is a 4x4 packet routing switch that provides each port a high raw bandwidth (800Mbits/sec), much of which is usable. For the links a high speed ECL gate array, called DLC, and flex cables were designed to reliably transfer data between boards.

## 2 The Packet Routing Chip

PaRC [4] is a 4 by 4 packet switched routing chip. PaRC receives packets via one of its four input ports, stores the packet in an on-chip buffer, and eventually transmits the packet via one of its four output ports. Each input port operates asynchronously and has enough buffering to store four packets. Each output port contains a transmitter, which sends packets off the chip, and a scheduler, which determines which packet the transmitter should send next. There is also a control interface which allows the user to control the operation of PaRC.

Packets consist of twelve 16 bit words: 1 header word, 9 data words (which is the size of one Monsoon token), and 2 words of error checking. PaRC operates at 50 MHz, and each port has 16 data lines. This gives each port a peak bandwidth of 800 Mbits per second (or 4.17 million packets per second). The buffering and scheduling algorithms used in PaRC were designed to provide high utilization of the available bandwidth. This is done by reducing contention for on-chip resources. This also helps provide PaRC with a low latency, as low as 100 ns for non-blocked packets. The PaRC chip can also treat certain packets specially in order to provide the sender with acknowledgment when that packet reaches its destination. PaRC is implemented as a CMOS gate array using LSI Logic's 1.5 micron technology. Figure 1 shows a top level view of PaRC, and the rest of this section describes some of the important features of PaRC.

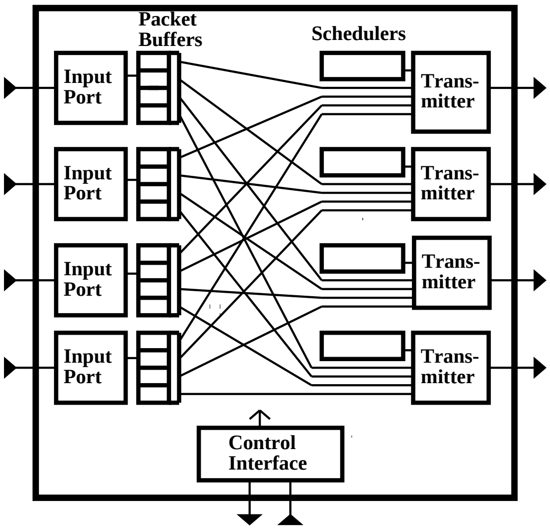


Figure 1: Top level view of PaRC

### 2.1 Packet Buffering

A design decision that greatly affects the performance of a packet switched network is how much buffering each switch has and how that buffering is used. Typically, store and forward networks store messages coming in over the same input in one long FIFO (first-in-first-out) buffer associated with that input. This lessens the throughput the network can sustain: If the packet on the top of the queue is blocked (*i.e.,* another packet is using the output this packet needs to use) then the second packet on the queue could not be sent even if the output it needs is sitting idle. Assuming that the top of each fifo has a packet which is heading for a random destination, then on average only 2.7 of those 4 packets would not be blocked. This means that even if the links entering a chip can supply a packet whenever room is available, the utilization rate for links leaving that chip would be only 68%.

To avoid this problem PaRC does not use one long FIFO. Instead it divides the memory associated with each input port into four separate buffers, each large enough to hold exactly one packet. This allows packets to be read out in any order, regardless of the order in which the packets arrived. In PaRC this is easy to do because all the packets are the same size. Similar methods can be used when this is not the case [2],[3].

Giving each packet a different buffer can be further used to our advantage. Each buffer is given its own output circuitry, thus allowing each buffer to be read independently. It then becomes possible to read out two (or more) buffers from the same input port at the same time. This permits each output port to read from any buffer that has a packet for it, regardless of which other buffers are currently being read. This eliminates blockage due to contention for on-chip resources. If an output port is ready to send a new packet it will be able to do so if there is a packet in any buffer that needs to use that port.

Another advantage of adding output circuitry to each buffer is that it makes it faster to begin to read out a packet. Since the buffer always knows which word will be read next, it can begin the read of that word on the cycle before it is needed. This helps to minimize the latency of packets. The packet buffers are also designed so that a packet can begin to be read out before the entire packet has arrived. This allows PaRC to provide the virtual cut through mechanism described earlier, which further decreases the latency of the network.

PaRC's buffering strategy greatly improves the chip's throughput and latency, but these improvements do come at a price. In addition to adding to the size of the packet buffers, this scheme makes the scheduling problem much more complex. PaRC must ensure that packets following the same path do not get out of order. This is automatically done if each input has a single fifo buffer. Since this is not the case in PaRC, a scheduling strategy is needed that ensures point-to-point fifo ordering.

To do this PaRC gives each scheduler its own FIFO. This queue does not buffer packets but instead buffers pointers to packets. When a new packet arrives, the input port makes a request to the appropriate scheduler, telling the scheduler where the packet is being stored. The scheduler simply adds a pointer to the requesting packet to the bottom of its scheduling fifo (s-fifo). Choosing the next packet to be transmitted is then as simple as reading the top of the s-fifo. Since this queue is FIFO, it guarantees that packets following the same path stay in order. This is also a very equitable strategy since even packets which arrive via different input ports will be transmitted in the virtually the same order they arrived. Lastly, this scheme is very fast; scheduling adds only one cycle (20 ns) to the latency of non-blocked packets.

## 2.2 Acknowledged Packets

The Monsoon software will sometimes need to know when a certain packet has been received. PaRC can provide, for selected packets, a fast acknowledgment that the packet has been received. These packets, called circuit switched packets are sent through the network in the same manner as normal packets. The difference is that as one of these packets passes through the network, the network keeps track of the packet's path. When a packet reaches its destination this path is used to send an acknowledgment to the packet's sender.

To implement this, each link between a transmitter and the input port it sends to contains a dedicated wire that is used for transmitting this acknowledgment signal. This wire transmits data in the opposite direction as the rest of the link. When PaRC transmits a circuit switched packet, PaRC's transmitter keeps a "back pointer" to where the packet came from. The transmitter then begins to watch the acknowledgment line associated with its link. When it sees an acknowledgment it passes that acknowledgment along to the link on which the packet arrived. After a circuit switched packet is transmitted, and before the acknowledgment arrives the only resource being used by the circuit switched packet is the acknowledgment line. The rest of the link is not blocked and other packets can still be transmitted normally. The only exception is if a circuit switched packet tries to use an output port which is already waiting for an acknowledgment. Then the port becomes blocked until the previous circuit switched packet receives its acknowledgment. PaRC is able to generate an acknowledgment when it transmits a circuit switched packet. Using this feature the Monsoon Network can safely generate the acknowledgment when the circuit switched packet leaves the next-to-last stage of PaRC chips.

## 2.3 Other Details

PaRC has been designed such that with very high probability it will be able to detect all link errors. This is done by checking for errors both when packets are being received and when they are not. To check packets for errors, at the end of each packet there is a 32 bit CRC (Cyclic Redundancy Code). As a packet enters PaRC, PaRC checks to see if the CRC is correct. (PaRC can also generate or ignore the CRC.) Using this code there is less than a 1 in $10^9$ chance of an incorrect packet being mistaken for a correct one. (In addition there are certain classes of errors, such as all errors on the same wire, which will always be detected.) This mechanism detects errors due to corrupted data within a packet, and errors due to a corrupted start-of-packet signal which causes a packet to appear where there should not be one.[1] (If a packet is erroneously created it will have a bad CRC, so the error will be detected.)

There is one type of error that the CRC does not detect, namely packets lost due to a corrupted start

[1] PaRC does not use an extra wire to mark a start of packet. Instead this information is included in one bit of the packet header and one bit of the 0 or more unused words between packets.

of packet signal. This is detected by checking for errors between packets. To do this PaRC defines two idle patterns. When not sending packets a transmitter will always send one of the two idle patterns. When waiting for the start-of-packet signal, input ports check to see that the data being received is one of the idle patterns. If not, an error is detected. In this way if corruption causes the start-of-packet signal to be changed, and a packet lost, the data will likely not be an idle pattern, so the error will be detected. These error checking mechanisms are also useful because they detect the error as soon as it occurs. This makes it easier to locate and repair any faulty connections.

When an input port receives a packet it must decide which output port to route that packet to. The header of each packet contains 15 bits which are used as routing information. The usual mechanism used by PaRC is to take two bits from this header and concatenate them to form a two bit number. This number identifies the port that the packet will be sent to. Which bits a PaRC uses is user controllable via the control interface. This mechanism allows butterfly and other networks to be built with as many as $2^{15}$ nodes.

PaRC is also able to do routing based on local congestion. In this mode only one bit of the routing data is looked at. This bit is used to determine if the packet should got to an upper port (ports 3 and 2) or to a lower port (ports 1 and 0). If it is determined that the packet should go to an upper (lower) port, PaRC will choose the upper (lower) port with the shortest queue of waiting packets. This mode will be useful in networks in which there are several paths that a packet can take to its destination.

## 3 The Data Link Chip and Cables

DLC[1] is a custom designed ECL gate array that is used to interface the inputs and outputs of PaRC to interboard cables. Each DLC contains two independent circuits: a transmitter and a receiver. The transmitter multiplexes 16 bit data from a PaRC output onto the 4 bit wide data path of a cable. The receiver demultiplexes data from a cable and presents it to a PaRC input port.

The primary design goal of DLC and of the interboard cables is high reliability. Low impedance differential signals are used on the cables to provide high noise immunity. A four bit wide data path is used in the cable to minimize the number of connections in each connector. Since the inputs and outputs of PaRC are 16 bits wide at 50MHz, the interboard signals are transmitted at 200MHz. DLC uses 100K ECL drivers and receivers. This assures that the signal levels are relatively constant over variations in temperature and supply voltage. 200MHz is well below the limits of the drivers and receivers and their quality at this speed allows cable attenuation to be tolerated without a loss of reliability.

The 50MHz clock for a PaRC and the 200MHz clocks for its attached DLC transmitters are derived from the same reference. However, because of process variation the delay of a PaRC's clock circuit can not be predicted accurately. Thus at initialization the DLC transmitter determines the phase of PaRC's output clock relative to the reference clock and adjusts DLC timing appropriately.

The DLC receiver is timed to the 200MHz clock that it receives from the interboard cable.

The interboard cables used in the current Monsoon systems were designed by Rogers Corporation under the direction of Tom Bousman of Motorola. Each cable is a stripline flex circuit. The stripline construction results in a signal layer sandwiched between two ground layers. This provides significant shielding of the signal layer. Each bit is carried by a differential pair of wires. The differential impedance of each pair is uniform over its length and is 100 ohms. Electrical connections between a PC board and a cable are accomplished using direct contact between board pads and cable pads. The geometry of the contact pads is such that the impedance of the connector matches the impedance of the cable. This approach requires a minimum amount of board space for the connector.

While the longest link used in the current Monsoon systems is roughly 10' long, 40' links, constructed with a different type of cable that has 26 gauge conductors, have also been tested. The tests have shown the links to be extremely reliable.

## 4 Conclusions

The above components allow a variety of reliable, high bandwidth interconnection networks to be built. One such network is used in the 16-node Monsoon system. The first 16-node system is currently in operation.

## References


[1] G. A. Boughton. Data Link Chip. Internal Memo, Computation Structures Group, MIT, Cambridge MA, March 1989.

[2] W. J. Dally. Virtual Channel Flow Control. In *Proceedings of the 17th International Symposium on Computer Architecture*, Seattle, Washington, May 1990.

[3] G. L. Frazier and Y. Tamir. The Design and Implementation of a Multi-Queue Buffer for VLSI Communication Switches. In *Proceedings of the IEEE Conference on Computer Design: VLSI in Computers and Processors*, Cambridge, MA, October 1989.

[4] C. F. Joerg. Design and Implementation of a Packet Switched Routing Chip. Technical Report LCS/TR-482, MIT, December 1990

[5] G. M. Papadopoulos. Implementation of a General Purpose Dataflow Multiprocessor. PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge MA, August 1988.

[6] G. M. Papadopoulos and D. E. Culler. Monsoon: An Explicit Token Store Architecture. In *Proceedings of the 17th International Symposium on Computer Architecture*, Seattle, Washington, May 1990.