88-HP

ISMM

MINI AND MICROCOMPUTERS

From Micros to Supercomputers

MIMI '88

Miami Beach, Florida, U.S.A.
December 14-16, 1988

Editor: M.H. Hamza

A Publication of
The International Society for
Mini and Microcompters - ISMM

ISBN 0-88986-107-2

ACTA PRESS

ANAHEIM * CALGARY * ZURICH

A Hybrid of Packet-switching and Circuit-switching Networks

Qiang Li and Naphtali Rishe

School of Computer Science
Florida International University–
The State University of Florida at Miami
University Park, Miami, FL 33199, USA

Abstract

An interprocessor communication network for a massively parallel multiprocessor system is presented. The network has the advantages of both packet-switching network and circuit-switching network, namely, fast response time for the short messages and high transfer rate for long messages, over a wide range of message sizes. Higher reliability is also achieved.

Keywords: Interprocessor network, Parallel architecture, Transputers, Database machine.

1 Introduction

Parallel processing and parallel architecture have become a major research trend in computer science. As massive number of independent components are used in a system, the data flow between the components becomes very heavy. As a result, the interconnection network becomes a critical part of parallel architecture design. Tremendous efforts have been put on this area and many results have been reported [9,10,11,19,20,2,7].

Our recent research on a parallel database machine LSDM [16] is an attempt to introduce massively parallel processing power into a database system. Database systems require heavily both computation power and I/O capacity. I/O operation is the bottleneck in most of the database machines. Many commercial and experimental parallel database machines have been built or proposed [21,15,3]. Based on the analysis on the data transfer pattern on the conventional database machines, we try to eliminate the I/O bottleneck by introducing massive number of independent secondary storage devices and contention-free I/O channels. The configuration of the proposed system is as follows. There are thousands of processors (called database processors) and thousands of secondary storage devices, e.g., 20 Megabyte disk drives. Each processor has its own local memory module and is exclusively connected to a secondary storage device. There is no hardware level memory or secondary memory sharing between processors. Each processor-memory-disk module forms a processing unit of the system. An inter-processor network is used to connect the units together. A processor which needs data held by another unit has to send request to and receive the data from the unit through the network. Analysis [19] has shown that the I/O operations should have much less impact on the system under this configuration than in the conventional systems.

Although the system consists of thousands of independent units, it is not considered a distributed system because of the frequent communication between the units and the requirements on the communication speed. Data is transfered very frequently, e.g., several transfers per query. Both bandwidth and response time of the network are critical. Due to the nature of the system, data flowing through the network varies greatly in size. The control or synchronization messages between processing units are a few bytes long while a large chunk of database data can be hundreds of Kbytes long. While the response time is more critical to the short control messages than to the large messages, the transfer rate is more important to the large messages. In other words, a network having fast response time for short messages and high transfer rate for large messages is required.

The two types of networks most widely used nowadays are the packet-switching network and the circuit-switching network [1,9,16,20,24].

The packet-switching network has the advantage of fast response time and flexibility of sending messages to any node at any time. It is suitable for the short messages in our system. But, the main disadvantage of the packet-switching network is that when a large amount of data is sent, the data has to be split into many packets in order to fit into the network. The packaging and depackaging time is long, and the store-and-forward delay can be very long when the packets have to travel through a number of nodes in the network. In addition, if there are M messages per second passing through the network and a message has to travel N nodes on the average, the total number of messages passing through all the channels is $N * M$, which effectively reduces the channel bandwidth by a factor of N.

On the other hand, the circuit-switching network has the advantage of sending a large amount of data to a node at the hardware speed without disturbing other nodes in the system. However, such networks suffer from the circuit set-up delay time, especially when short messages are sent. In the situation where not very frequent but large size messages are transfered, the circuit-switching network is a good candidate. In addition, a packet-switching network consisting of only passive switching components needs a control network to pass the configuration messages from the processing units to some "circuit controllers" which select the proper circuit and carry out the circuit set-up operations of the network.

In general, the packet-switching networks favor short messages and the circuit-switching networks favor large data sets. In our database system, both short messages and large sets of data have to be sent frequently. This calls for a compromise between the two types of networks.

This paper proposes a new network architecture. For short messages, the network works like a packet-switching network; and for long messages, the network works like a circuit-switching network.

This research has been supported in part by a grant from the Florida High Technology and Industry Council.

2 The proposed network

2.1 The hardware structure

The proposed network consists of two main parts: a circuit-switching network which can connect between every pair of processors upon demand; and a semi-tree-structured packet-switching network which statically connects all the processors. Figure 1 shows the block structure of the network. Each processor is connected to both networks with one link to each. The database processors are the leaves of the tree. The inner nodes of the tree are communication processors. To avoid a bottleneck, the "root" of the tree is not a single processor but a small hypercube. In other words, instead of a single tree, a group of smaller trees whose roots are connected into a hypercube is used. Several nodes of the hypercube are connected to the circuit controllers shown on the top of the tree.

At the first glance, the architecture is merely a superficial combination of a packet-switching network and a circuit-switching network and redundancy is introduced. But the fact is that the packet-switching network is just an extension of the control network of a passive circuit-switching network. The difference is that in addition to delivering circuit switching control messages to the circuit controllers, the network now also delivers short data messages from a database processor to another.

The circuit controllers are responsible for selecting an available route for setting up the circuit, keeping track of the current status of the network, maintaining a queue of the connecting requests not able to be satisfied for the time being, etc. The circuit controllers are also responsible for sending the hardware signal to actually set up the circuit. Although a group of processors can play the role of the circuit controller well, the associative memory is a perfect candidate for this job.

Since it is a tree shaped network, the nodes on the top part of the tree will have heavier traffic than the nodes at the lower part do, especially the roots of the subtrees. The communication capacities of the roots of the subtrees have direct impact on the capacity of the entire packet-switching network. We need to calculate the load for the node on the top. Assume that messages passing between database processors are uniformly distributed, i.e., the probability for a database processor to send a message to any other processor is the same for all processors pairs. Let N be the number of subtrees in the network; M be the number of messages passing through the network in a second; m_1 be the number of messages per second which are originated inside the subtree and flow out of the subtree through the root; m_2 be the number of messages per second which are originated outside the subtree and flow into the subtree through the root. Then, there are M/N messages originated in a subtree, and $1/N$ of those messages are sent to the nodes inside the subtree. Therefore, the total number of messages flowing out of the subtree is

$$m_1 = M/N - (M/N)/N = (1/N - 1/N^2) * M$$

Similarly, the nodes of a subtree receive in total M/N messages per second, and $1/N$ of those are originated from inside of the subtree. Thus, the total number of messages flowing each second into a subtree is

$$m_2 = M/N - (M/N)/N = (1/N - 1/N^2) * M$$

When N is relatively large, the right hand sides of the formulas are approximately M/N messages per second. In other words,

$$M = N * (m_1 + m_2)/2 = N * m$$

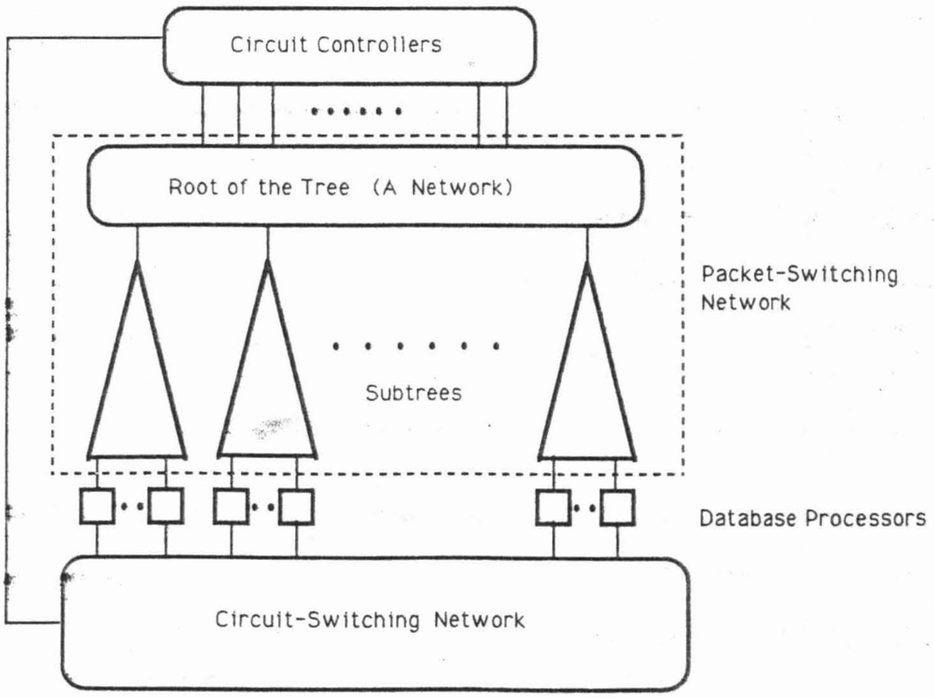

Figure 1: Network Structure

where $m = m_1 = m_2$. This gives the relationship between the capacity of the roots of the subtrees and the message rate of the system. Note that the data flow pattern is assumed to be uniform. In practice, it is often possible to arrange the data stored in the system so that the traffic will tend to be heavier within a subtree, than between the subtrees. In this case, the system message rate can be higher than $N * m$. Thus, in fact, the uniform distribution is a very pessimistic assumption.

Since the number of subtrees is not large, it is not difficult to select a topology to connect the roots of the subtrees together, e.g., completely connected graph or hypercube. We have chosen the latter. Although the formula above indicates that using more subtrees can give a higher system capacity on a given individual node capacity, the number of subtrees should not be too large. Otherwise, the connections between the roots of the subtrees will become a problem itself. A compromise must be made here.

The setup time of the circuit-switching network is the time between the moment a database processor sends a request to the circuit controllers and the moment it receives the acknowledgement. The time depends on the speed of the tree network and the speed of the controllers. It should be normally within a few milli-seconds for our system.

2.2 The Protocol

As mentioned above, two kinds of messages flow through the packet-switching network. The messages for setting up a circuit are called control messages, and the messages between the database processors are called data messages. The control messages and the data messages have different logical formats as shown in Figure 2. When a short message is to be sent, no control messages are exchanged; only the short message itself is sent to its destination through the tree network. The system is arranged in such a way that most short messages are between processors in the same sub-tree. In our database machine such an arrangement is possible since the primary pattern of message exchange is known in advance [16].

Message-type
Function
Destination
Originator

Control Message

Message-type
Destination
Originator
Message-length
Message

Data Message

Figure 2: Message formats of the packet-switching network

When a message is sent into the packet-switching network, each communication processor will check the type of the message. If the message is a control message, it will be sent to the communication processor's parent and eventually reach the top of the tree. If the message is a data message, the communication processor will check its destination. If the destination node of the message is in a subtree of the communication processor, the communication processor will send the message down to the subtree. Otherwise, the message will be sent to the parent of the communication node. Thus, a data message will climb up the tree until it reaches the first common ascendent of its originator and destination nodes, and then go down the tree until it reaches the destination.

There are three reasons why the tree network is satisfactory in terms of the communication speed. First, only short messages are sent through the tree network. Secondly, since the tree network is an acyclic graph, there is no deadlock in the network. Further, it has been proven that a hypercube with the so-called "fixed routing algorithm" is deadlock free [15]. Therefore, deadlock prevention overhead is eliminated. Thirdly, since the processors in the inner nodes of the tree network are dedicated communication processors, the messages are processed very quickly.

When a long message needs to be sent, the originating database processor first sends a short control message up the tree to a circuit-controller. Upon receipt of a request for a long message transmission, a set of circuit-controllers determine the connection path and send hardware signals to set up the circuit. Then, a circuit controller sends a "green light" control message to the long message's originator. Finally, the long message is sent through the dedicated circuit at the maximum speed of the hardware. The circuit is released when the circuit-controller receives an end-of-transmission message from the originator.

There are two types of messages sent through the circuit-switching network, namely, data messages and contingency messages. The data messages are the normal communication messages between the pair of processors which are currently connected by the circuit, and the contingency messages are those to be sent to a third processor under abnormal circumstances such as communication failure in the packet-switching network, or load balance between the nodes. The formats of data messages and the contingency messages are shown in Figure 3.

There is a process on each database processor constantly waiting for messages coming through the circuit-switching network. Upon receiving a message, it buffers the message, and then decides either to give the message to other processes or pass the message to other processors if the message is an contingency message.

Message-type
Originator
Message-length
Message

Data Message

Message-type
Originator
Destination
Message-length
Message

Contingency Message

Figure 3: Message formats of the circuit-switching network

2.3 Reliability

The network also provides a higher reliability. When a path of the circuit-switching network fails, i.e., a processor finds itself unable to send message through the circuit-switching network, long messages can be temporarily sent through the packet-switching network, or the messages can be sent to a nearby neighbor in the packet-switching network and relayed to the destination by the neighbor through the circuit-switching network. Although the performance under these circumstances will be worse than normal, the system can be kept alive until the problem is fixed. On the other hand, when a part of the packet-switching network fails, the group of processors affected can still respond to the requests coming from other processors through the circuit-switching network.

In summary, the network proposed here preserves the advantages of both the packet-switching networks and the circuit-switching networks, while avoiding their disadvantages. Furthermore, higher reliability is also achieved.

3 Experimental Data

The experimental system we are currently building consists of a number of INMOS transputers. The configuration under investigation has 1024 database processors and about 550 communication processors. The network is being implemented to support a Semantic Binary DBMS [14].

The circuit-switching network is a three stage Clos network [6,5,4] consisting of 96 INMOS C004 dynamic reconfigurable switches. All the communication links are serial links, including those in the circuit-switching network and have a speed of 20 Megabaud (effective rate of 0.7 Megabyte per second). The circuit-switching network can provide a link between any pair of database processors. Depending on the dynamic connection sequence, the number of pairs which can be linked simultaneously varies from 256 (512 processors) to the best base of 512 (all the 1024 processors). Translated into the data transfer rate, it is from 358 Megabytes to 716 Megabytes per second.

The capacity of the packet-switching network depends on the dynamic dataflow pattern. It is estimated that under normal operation, among the messages flowing through the network 20% are control messages and 80% are data messages. The top of the tree gives the limit of the message rate. Experiments have shown that under an average message size of 100 bytes, messages can flow through the top of the tree at a rate 3.6 Megabytes per second.

Acknowledgement

The authors gratefully acknowledge the advice of David Barton, Nagarajan Prabhakaran, Doron Tal, Neil Wiseman, and Wan-lai Chen.

References

[1] B.W. Arden and H. Lee. Analysis of chordal ring network. *IEEE Transactions on Computers*, C-30(4):291–295, 1981.

[2] B.W. Arden and H. Lee. A regular network for multicomputer systems. *IEEE Transactions on Computers*, C-31(1):60–69, Jan 1982.

[3] J. Banerjee, D.K. Hsiao, and K. Kannan. DBC - a database computer for very large database. *IEEE Transactions on Computers*, C-28(6):414–429, Jun 1979.

[4] V.E. Benes. *Mathematical Theory of Connecting Network and Telephone Traffic*. Academic Press, New York and London, 1965.

[5] V.E. Benes. On rearrangeable three-stage connecting networks. *The Bell System Technical Journal*, 1481–1492, Aug 1962.

[6] C. CLos. A study of nonblocking switching networks. *The Bell System Technical Journal*, 406–424, Mar 1953.

[7] K.W. Doty. New designs for dense processor interconnection networks. *IEEE Transactions on Computers*, C-33(5):447–450, May 1984.

[8] L.R. Goke and G.J. Lipovski. Banyan networks for partitioning multiprocessor systems. In *Proc. of the 1st Annual Symposium on Computer Architecture*, pages 21–28, 1973.

[9] J.R. Goodman and A.M. Despain. A study of interconnection of multiple processors in a data base environment. In *Proc. of the International Conference on Parallel Processing*, pages 269–278, IEEE, Harber Springs, MI, Aug 80.

[10] J.R. Goodman and C.H. Sequin. Hypertree: a multiprocessor interconnection topology. *IEEE Transactions on Computers*, C-30(12):923–933, 1981.

[11] K. Hwang and F. Briggs. *Computer Architecture and Parallel Processing*. McGraw-Hill, 1984.

[12] E. Ozkarahan. *Database Machines and Database Management*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1986.

[13] F.P. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Communications of the ACM*, 24(5):300–309, May 1981.

[14] N. Rishe. *Database Design Fundamentals: A Structured Introduction to Databases and a Structured Database Design Methodology*, chapter 1, pages 1–40. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1988.

[15] N. Rishe and Q. Li. Hypercube network with fixed routing algorithm is deadlock free. Sept 1988. In preparation.

[16] N. Rishe, D. Tal, and Q. Li. The architecture for a massively parallel database machine. *The Euromicro Journal*, Aug 1988. In press.

[17] H.S. Stone. Parallel processing with the perfect shuffle. *IEEE Transactions on Computers*, C-20(2):153–161, 1971.

[18] S. Su. *Database Computers*. McGraw-Hill Book Company, New York, N.Y., 1988.

[19] A.S. Tanenbaum and R.V. Renesse. Distributed operating systems. *ACM Computing Surveys*, 17(4):419–470, Dec 1985.

[20] L.G. Valiant and G.J. Brebner. Universal schemes for parallel communication. In *STOC, ACM Conference Proceedings*, Milwaukee, 1981.

[21] C.-l. Wu and T.-y. Feng. On a class of multistage interconnection networks. *IEEE Transactions on Computers*, C-29(8):694–702, Aug 1980.