Bandwidth Anaylsis of Message-Passing Networks

Technical Report No. CENG 89-24

Wing C. Lee

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California

October 10, 1989

Abstract

The design of effective multiprocessor systems is a difficult and complex task. In this paper, we examine one of the most critical issues in multiprocessor design: the design of the interconnection network. We built a simulator to study the characteristics of many interconnection networks to enable us to select a suitable network for a muliprocessor we are currently designing. Simulation results and evaluations for Bus, Ring, Grid, and Hypercube Networks are presented.

[0]This research has been funded by the National Science Foundation Grant No. MIP-89/02426

1 Introduction

The design of effective multiprocessor systems is a difficult and complex task. It involves the interaction of many diverse elements. They range from parallel algorithms to parallel processors, from programming languages to communication schemes.

In this paper, we will examine one of the most critical issues in multiprocessor design: the design of the interconnection network. An interconnection network enables the processors in a system to communicate with one another. A good interconnection network is vital for the stable operation of the multiprocessor. This requirement results from two basic tenets of parallel processing: effective cooperation of several processors on the same task and fast access to distributed data. Both required high communication bandwidths. If all of the processors are simply connected with the same bus, then this shared resource becomes a bottleneck preventing simultaneous communication between different pairs of processors. In this case, the effective throughput of the system actually goes down as the number of processors increase. A suitable interconnection network, which provides as much bandwidth as possible between any pair of processors, is thus needed.

This paper is based on work done for a multiprocessor, called SNAP [Moldovan 1989], currently under development at USC. We built a simulator to study the characteristics of many interconnection networks to enable us to select a suitable interconnection network for SNAP. The results we found are presented here.

The types of networks evaluated for SNAP were *static networks*. Static networks consist of active computing nodes connected by passive communication links. All active switching occurs at the nodes. Thus, we did not consider reconfigurable multistage switchiing networks, such as Banyan [Goke 1973], Omega [Lawrie 1975], shuffle-exchange [Lang 1976], and indirect binary N-cube [Pease 1977]. These methods have most often been sugested for data routing in SIMD machines, although some could be used to interconnect MIMD processors. There is a rich literature comparing these reconfigurable interconnection methods [Lee 1984] [Dias 1981] [Lawrie 1980] [Wu 1980] [Franklin 1986].

2 Simulator

Our simulator was built to model the performance of several types of communication schemes for SNAP.

2.1 SNAP

SNAP (Semantic Network Array Processor) is a multiprocessor targeted toward Artificial Intelligence applications. It consists of 256 custom-designed chips, packaged into four boards of sixty-four. For the purpose of our simulation, each chip was considered as a processing element capable of sourcing and sinking messages[1].

[1] In actuality each chip represents many nodes. However, only one message may enter or leave each of the data ports. Thus our assumption is valid.

SNAP communicates via messages. Each message consists of eight 8-bit packets. The format of the message is:

Byte 1: Destination
Byte 2: Message Type
Byte 3: Data
Byte 4: Data
Byte 5: Data
Byte 6: Data
Byte 7: Data
Byte 8: Data

The messages are sent together as one unit. Thus, if a chip gains access to a communication path, it keeps the path until all eight packets have been sent. Packets cannot become seperated nor get of order.

Each chip has upto four data ports for communication with other chips. Four data ports is an upper bound due to pin limitations on the chip.

Each chip maintains an output queue for holding messages waiting to be transmitted. A routing controller inside each chip determines the shortest path to the destination and places the message onto the path. Thus, SNAP uses *distributed control* to coordinate message traffic.

2.2 Operation of the Simulator

The simulator was written in C, and created an environment whereby we could monitor the performance of each network. The simulator operated by introducing messages into the network and recording the time the messages took to reach their destinations. The messages were placed into the network at frequencies varying from (1 new message)/clock to (32 new messages)/clock. Thirty-two thousands messages were sent, and the networks were graded according to the average time it took a message to reach its destination.

Each processor had its own routing controller and queue. If a processor could not transmit a message, it would keep the message in its queue until a network path was available. The routing controller operated by computing the shortest path to the destination. If more than one path was avalilable, then the routing controller would choose one. If that path was busy, then others would be tried. No routing information was transmitted over the network, and thus, the routing controller made no attempt to route messages around congested areas.

The simulator ran on a sequential machine, so compromises had to be made in order to simulate a parallel architecture. Since, all the networks were synchronous, the smallest interval of time was one clock. Within a clock, messages were transmitted serially, with PE #1 attempting to transmit first and PE #255 transmitting last. Thus, some processors were given more favorable treatment than others.

New messages were placed at the end of the processor queues when the transmit phase was complete.

3 Evaluation Criteria

The interconnection networks were simulated to study two aspects: physical and computational.

3.1 Physical Aspects

A serious constraint in the physical design of large networks is imposed by the limited number of signal pins available both at the chip and board levels.

Chip Pin Constraints

We are restricted in the number of pins we can dedicate to communication. A key requirement is that to be able to fit 64 chips and their associated support logic onto one 11 x 14 in board. Thus, each chip can only be around 100 pins. Subtracting pins dedicated for instructions and other uses, each chip can only support four data ports.

Networks that require more than four ports to connect 256 processors cannot be used. Networks that require less than four ports enable the chip to be smaller and thus provide more room for chip placement and margin. However, having less than four ports is not an important characteristic and was not given much weight.

Board Pin Constraints

Similar to the chips, we are restricted in the number of pins we can use to connect the processors on one board to the processors on another. Each board has connectors that plug into the backplane. The boards we are using have approximately 300 connections that can be used to comunicate with the processors on other boards. Networks that require more than 300 connections cannot be implemented and thus cannot be used.

3.2 Computational Aspects

The compuational aspects refer to the speed of computation and message flow within the network.

Average Communication Time

Average communication time is the most important consideration in the evaluation of a network. It is the average time required to send a message from one chip to another.

Average communication time is a value measured by simulator.

Network Diameter

The network *diameter* is related to the average communication time. It is the maximum number of communication links that must be traversed to transmit a message from any node along a shortest path. A network with a high diameter will have a higher communication time than a network with a small diameter.

Network diameter is an analytical value.

Mean Internode Distance

The *mean internode distance* is the expected number of communication links a "typical" message needs to traverse in order to reach its destination. The mean internode distance is a better indicator of the average communication time than the network diameter. However, it is a static value and does not take into account the queue delays that a message encounters at each node it visits.

Mean internode distance is an analytical value.

Message Capacity

The *message capacity* of a network is simply the maximum number of messages that can be transmitted in the network at one time. If we assume that only one message can be transmitted on each of the communication links, then message capacity is also a measure of the number of communication links in a network. Networks with a high message capacity are able to handle heavier message loads and degrade more slowly than networks with a small message capacity.

Message capacity is an analytical value.

Average Message Density

The *average message density* measures the average number of messages circulating in a network. It is the sum of the number of messages being transmitted and the number of messages sitting in the queues of the chips. If the average message density is larger than the message capacity than the network is in danger of forming infinite queues.

Average message density is a value measured by the simulator.

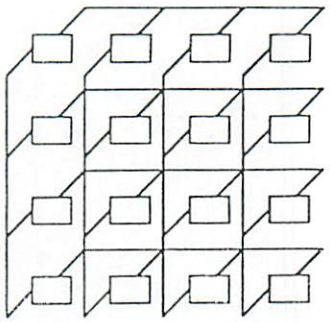
4 Communication Networks

We examined four different types of message passing networks: Bus, Ring, Mesh, and Hypercube.

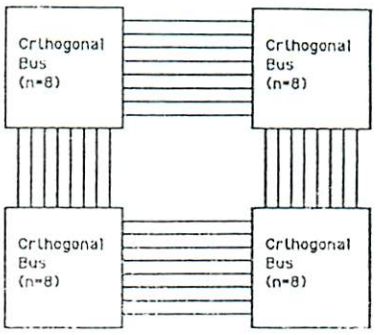
Figure 1 shows all of the interconnection networks simulatated. Figure 2 shows the average communication time as a function of the network message frequency. Figure 3 shows the message density of the networks as a function of message frequency.

4.1 Bus Networks

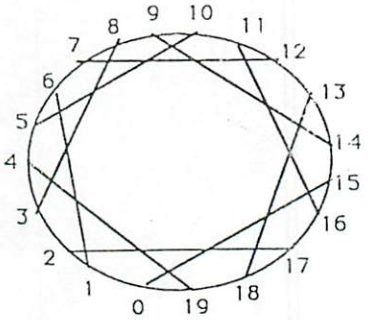
Single Bus



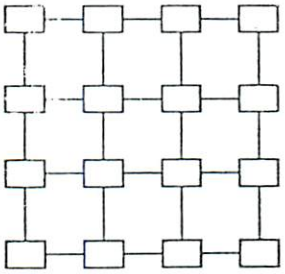
a) Orthogonal Bus (n=4)



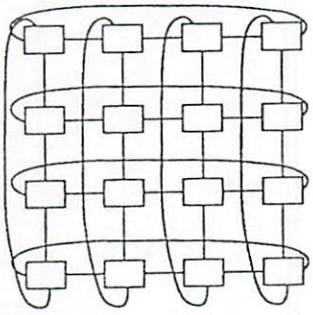
b) Quad Bus (256 PEs)



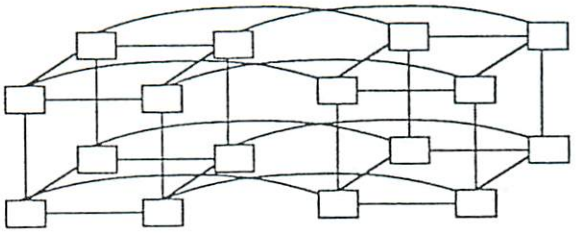
c) Chordal Ring (m=20, w=5)



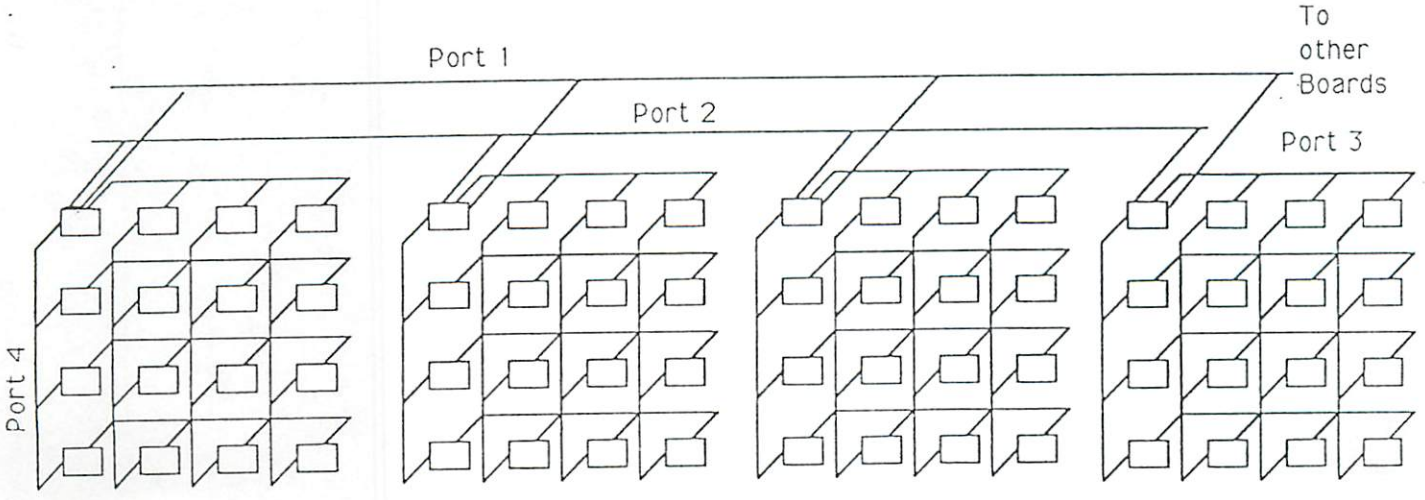
d) Grid (n=4)



e) Torus (n=4)

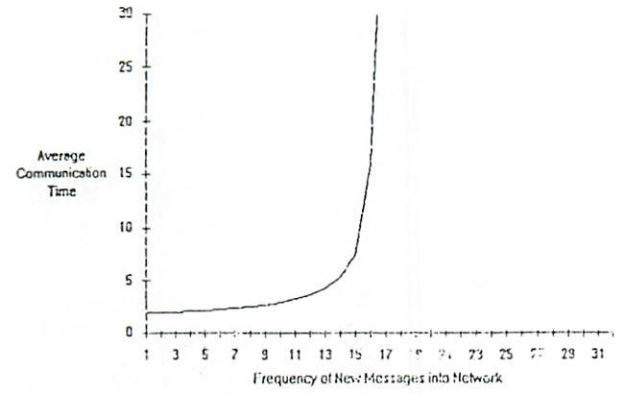


f) Hypercube(n=4)

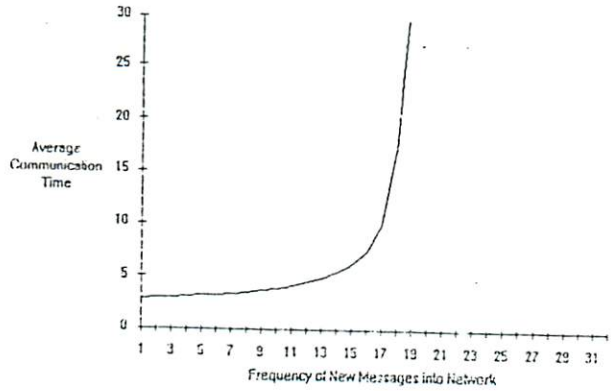


g) Bus Hypercube

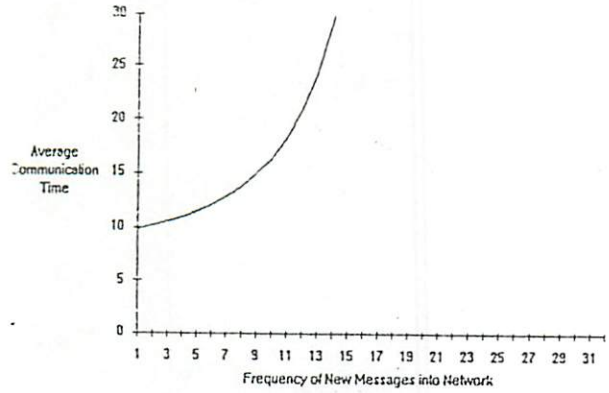
Figure 1: Examples of the Networks Evaluated



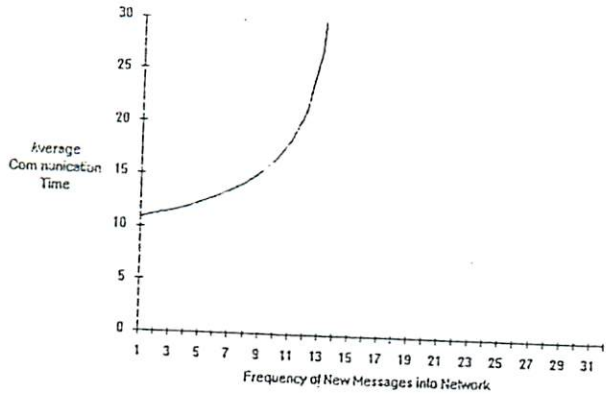
a) Orthogonal Bus



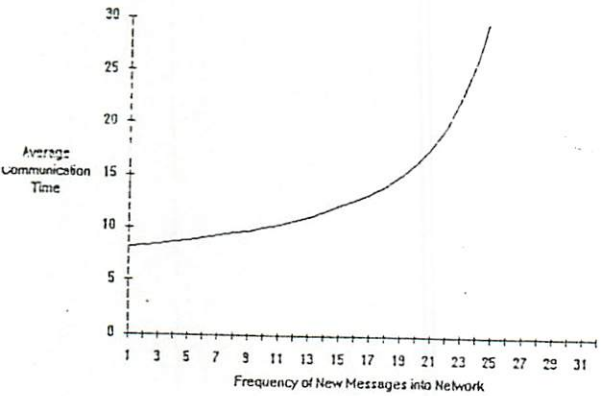
b) Quad Bus



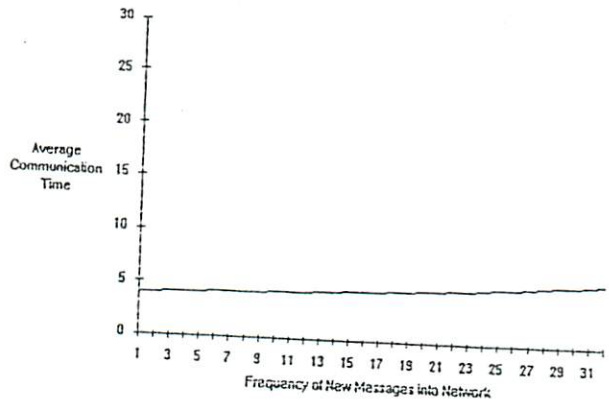
c) Chordal Ring



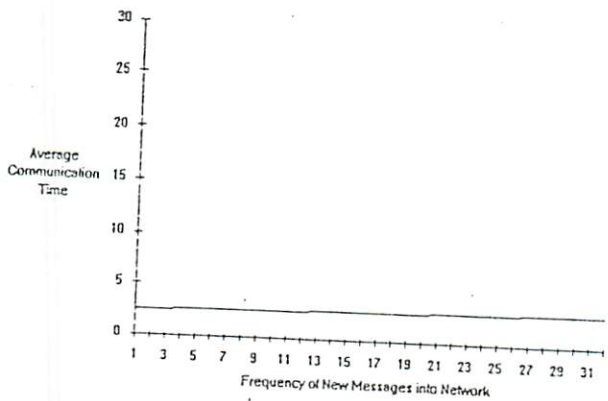
d) Grid



e) Torus

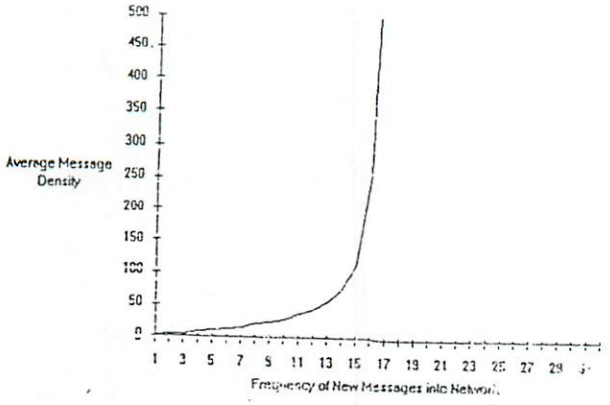


f) Hypercube

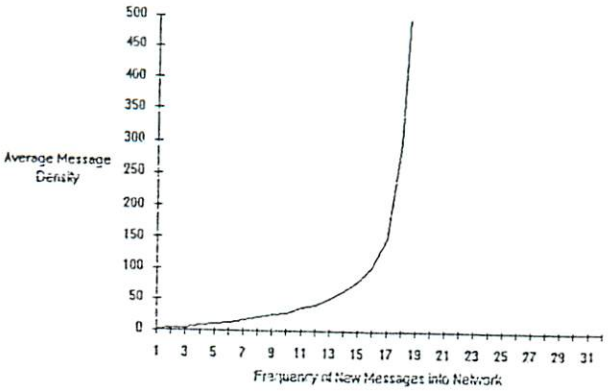


g) Bus Hypercube

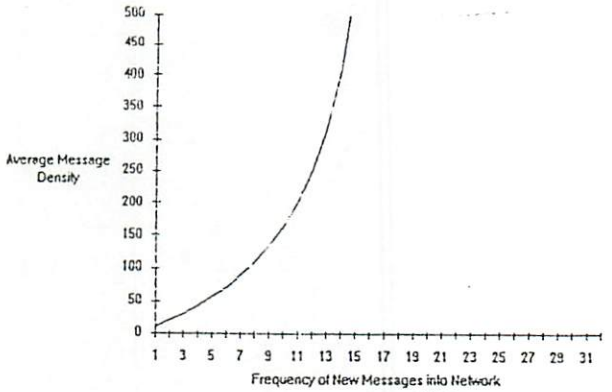
Figure 2: Average Communication Time of the Networks Simulated



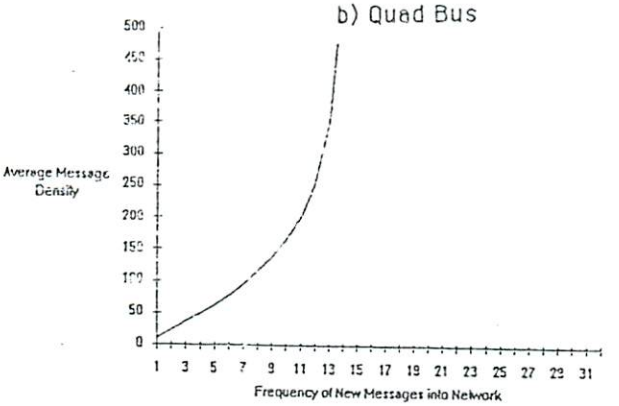
a) Orthogonal Bus



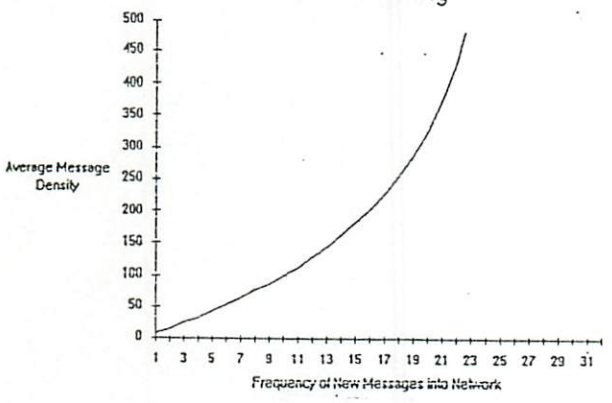
b) Quad Bus



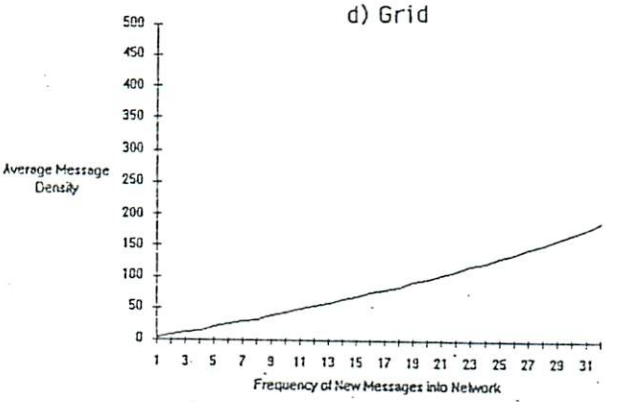
c) Chordal Ring



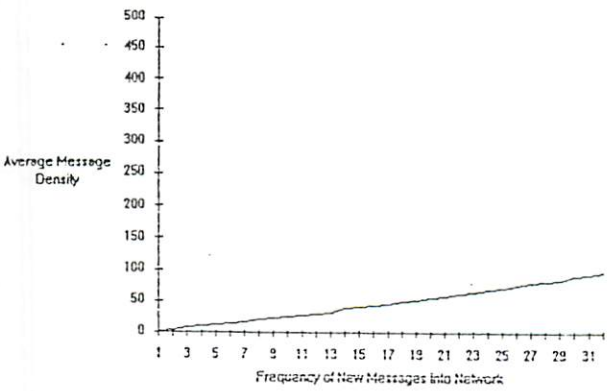
d) Grid



e) Torus



f) Hypercube



g) Bus Hypercube

Figure 3: Average Message Density of the Networks Simulated

The simplest interconnection network is a common bus connecting all of the processors. It has the characteristic of requiring only one clock cycle to transfer a message from one processor to another. However, a single bus can only send one message/clock and thus cannot support 256 processors.

Orthogonal Bus

An extension of a single bus is to have each processor connected to two buses. In an orthogonal arrangement, the processors are divided into rows and columns. Thus for 256 processors, the network consists of 16 rows buses and 16 column buses. The Orthogonal Bus network has the virtue of requiring a maximum of two message transfers to reach any destination. It has a message capacity of 32 messages/clock.

The simulation results show that the network works very well for lightly loaded networks. However, the network quickly degrades as the message load increases. At a message rate of eleven, the message density was greater than the message capacity, and at a rate of sixteen, the network became saturated and was unable to function:

The Orthogonal Bus suffers from two problems. The main problem is message capacity. As the message load increases, more processors try to send messages on the same bus at the same time. When this occurs, the simulator grants access of the bus to the processor with the higher priority. If the higher priority processors always have messages to send, then the messages in the other processors will wait indefinitely.

A related problem is routing. Messages have a most two routes to use. If either one of routes becomes congested, the messages have no recourse but to sit in a queue and wait. Plus, if the message at the head of a processor queue has a problem gaining access to the bus, then this prevents other messages in the queue from being transmitted, even if the other bus port is free.

Quad Bus

The Orthogonal Bus can be extended further into three or four bus networks. However, the backplane connectors can only support thirty-seven 8-bit buses. Thus, further extensions are not feasible.

Instead, the network can be divided into quadrants, with each quadrant corresponding to a board. Each quadrant has its own Orthogonal Bus network (8 row buses and 8 column buses). Backplane buses connect the quadrant row and column buses to their adjacent quadrant's buses. With this setup, the network can support nintey-six messages. The maximum transfer time of messages is four.

The simulation results show that the network does not perform as well as the Orthogonal Bus for lightly loaded networks, but does slightly better at higher loads. Messages require more intermediate nodes in the Quad Bus than they do in the Orthogonal Bus. Thus, intrinsically messages take longer. At light loads, the message collision rate is low and thus the primary factor in determining the average communication time is the mean internode distance.

At higher loads, the Quad Bus suffers from the same problems as the Orthogonal Bus. The overall message capacity is ninety-six, but each quadrant can only send sixteen messages at a time.

Since each processor has only eight processors, the collision frequency is inherently less. However, Quad Bus messages travel through more intermediate nodes, so the probability that a processor wants access to the bus is greater.

In terms of routing, messages still have only two routes to choose from. Plus, once a message travels on one of the paths, it can no longer use the other route. Thus for three of the four legs a message may have to travel, it has only one path to use. This coupled with the message capacity limits the usefulness of the Quad Bus.

4.2 Ring Networks

Assuming the processors are numbered 0, 1, 2 ,.., (N-1) ,N, (N+1),... each of the processors in a ring are connected to their (N-1) and (N+1) neighboors. The two end processors are connected together to form a complete circle.

Sinple Ring

The simple ring is a modest improvement over the simple bus, but it too can only support a small number of nodes. The ring is limited to a small number of processors because the average communication time and message traffic density increase linearly with the number of nodes in the ring.

Chordal Ring

The Chordal Ring is the same as the simple ring, except that each node has an additional link, called a chord, to some other node across the network. Each even-numbered node j (j=0,2,..,n-2) is connected to a node *(j-w) mod n*. The w is called the chord length and is assumed odd. For a given number of nodes n, a number of Chordal Rings can be obtained for different values of chord length w. [Arden 1981] analyzed Chordal Rings to find the optimal chord length for a given number of nodes. Using his results, we simulated a Chordal Ring with n=256 and w=19.

The diameter of the Chordal Ring that we simulated was fifteen and the message capacity was 384. The Chordal Ring is an improvement over the simple ring but suffers from routing and mean internode distance. There is only one optimal path a message can use to reach its destination. If one of the nodes in that path has a problem, then all of the messages that go through that node suffer as well. In addition, performace suffers because the mean internode distance is high. At each intermediate node, a message must go through a queue. Thus, a message may have to wait in as many as fourteen queues. These two factors make Chordal Rings unsuitable for systems with a large number of processors.

4.3 Mesh Networks

In a Mesh Network, processors are organized into a two dimensional array. Each processor is connected to its four neighboor processors.

Grid Network

The Grid is the simplest Mesh Network. PE(i, j) is connected to PE(i-1, j), PE(i+1, j), PE(i, j-1), and PE(i, j+1) The edge PEs have three connections and the corner PEs only two.

For 256 processors, the network is 16 x 16. The maximum communication time is thirty and results from sending a message from one corner PE to another. The network appears robust as it can handle 480 messages at one time. However, as the simulation results show, the network becomes overloaded at just thirteen new messages/clock.

Unlike the Bus Networks, the Grid Network suffers from neither lack of message capacity nor lack of routing options. All the connections are point to point, thus the probability of collision is very low. In addition, since each processor can output onto four ports, the probability of using a particular point is only 25%. With an ability to send almost 500 messages and a low probability of collision, message capacity is not an issue.

Similarly, routing is not a problem. At most of the twenty-nine intermediate nodes a message can visit, a message has a choice of two paths to use. Thus, a message from one corner PE to the other has a choice over 40,000 possible routes.

The problem with the Grid Network is due rather to the number of intermediate processors messages have to pass through. The processor queues grow very quickly. At a message rate of fourteen, there are an average of over 600 messages waiting in the processor queues. Thus, every time a message reaches an intermediate node, it must wait a long time to get to the head of the queue. With as many as twenty-nine intermediate nodes to visit, the communication time quickly becomes intolerable. At a frequency of ten, it could takes almost sixty clock cycles for a message to reach its destination.

Torus

An improvement of the basic Grid formation is to add wrap-around paths to the edge and corner PEs. Thus, every PE has four connections and the worst case transfer time is reduced to fifteen.

Like the Grid Network, the message capacity in the Torus is very good (512 messages/clock) and the collision rate is very low. The number of possible routes is half of the number in the Grid, but 20,000 paths is more than adequate.

The simulation results show that Torus is a much more stable network than the basic Grid. Degradation occurs at twenty-four, which is almost double the degradation rate of the Grid Network. The intermediate processor queues build up more slowly in the Torus. This is due to the fewer nodes a message must visit. The maximum queue size stabilizes at around twelve messages until the queues explode at a frequency of 24.

The Torus would be almost ideal for light to moderate message traffic, if the worst case communication time were better. But, at best a message could take almost twenty clock cycles to reach its destination. This is five times worst than the performance of the Orthogonal Bus network at light loads.

4.4 Hypercube Networks

In a Hypercube organization, the N processors are placed at the vertices of a hypercube of dimension k, where k=$log_2 N$. The PEs are numbered 0,1,....,(N-1), and PE(i) is connected to PE(j) such that the binary representation of j differs from that of i in exactly one position.

Basic Hypercube

A hypercube of dimension eight is required for 256 processors. Each processor has eight connections and the maximum number of intermediate nodes for a message is seven. The network is capable of transmiting 1024 messages simultaneously. Similar to the Grid, the Basic Hypercube has over 40,000 possible routes for a message to use. But, the advantage of the Basic Hypercube is that a message has to visit less than 25% of the number of nodes that a Grid Message does.

Thus, it is not suprising that the performance of the Basic Hypercube is excellent. The network was able to handle all the message frequencies with little variance in communication time. At the heavyest load, the average time a message required was still under six clocks. The worst case communication time remained very stable, averaging around ten clock cycles. The queues grew linerly, and at the end of simulation were only at 18.5% of message capacity.

However, the Basic Hypercube suffers from two main problems. The first problem is that the Hypercube is very expensive to implement. Over 8000 connections (wires) are required just for the data lines. The second problem arises from the physical constraint we placed on the network at the very beginning. Each of the four processor boards have only 300 connections, but the Hypercube requires 1024 connections between boards. Thus, the Basic Hypercube is not feasible.

Bus Hypercube

One modification to the Basic Hypercube is to use buses instead of point to point connections. In the Bus Hypercube, each processor has four ports. Assuming that the binary representation of the addresses of processors is (a7 a6 a5 a4 a3 a2 a1 a0), the configuration of the ports is:

Port 4 communication with processors that vary in (a1 a0)
Port 3 communication with processors that vary in (a3 a2)
Port 2 communication with processors that vary in (a5 a4)
Port 1 communication with processors that vary in (a7 a6 a5 a4)

Note that Port 1 is a superset of Port 2. Port 1 is used for communication between boards, so this arrangement requires only 16 buses between boards instead of 64. Ports 2-4 are buses that connect four processors. Port 1 connects sixteen processors. The network has 64 each of Ports 2-4 and 16 of Port 1. The maximum number of intermediate processors is three (two if Port 1 is used) and the message capacity is 208. The Bus Hypercube requires 1664 data connections, of which 128 are between boards.

Routing is not as good as in the Basic Hypercube. There are only twenty-four possible routes that a message may take. But with a maximum of only three processors to visit, the fewer routes is not a problem.

The simulation results show that the Bus Hypercube is the best network simulated. The average communication time was almost the same for lightly loaded networks (2.47 clocks) as it was for the heavyest loads (3.02 clocks).

The Bus Hypercube performs better than the Basic Hypercube because it takes half as many intermediate nodes to send a message. Even though the queues in the Basic Hypercube networks are small, they still require time to pass through. The limitation of the Bus Hypercube is that the message capacity is only a fifth of the Basic Hypercube. But since the maximum number of messages the network can source at a time is 256 (one message per processor), most of the bandwidth of the Basic Hypercube is wasted.

5 Conclusions

We studied several types of communication networks for SNAP: Grid, Bus, Ring, and Hypercube. Their characteristics are summarized in figure 4.

The Basic Hypercube had the largest message capacity, followed by the Mesh Networks. The Bus Networks required the fewest intermediate nodes, while the Mesh Networks required over ten times as much. The Basic Grid and both of the Bus Networks had the biggest variance in communication time, while the Hypercubes had the least.

The Bus Networks had problems with bus loading and are useless for all but lightly loaded systems. The Mesh Networks suffered from the delays that resulted from sending messages through too many intermediate processors. The Ring Networks lacked routing options and have a high mean internode distace. The Hypercube networks had the best performance. The Basic Hypercube network was too expensive to implement, but the Bus Hypercube stands out as the best network. The Bus Hypercube was the network chosen for SNAP.

6 Acknowledgement

The author would like to thank Dan Moldovan for his discussions and comments on this paper.

References

Arden, Bruce W and Lee, Hikyu [1981], "Analysis of Chordal Ring Network", IEEE Transactions on Computers, April 1981.

Diaz, Daniel M. and Jump, J. Robert [1981], "Analysis and Simulation of Buffered Delta Networks", IEEE Transactions on Computers, April 1981.

Doty, Karl W. [1984], "New Designs for Dense Processor Interconnection Networks", IEEE Transactions on Computers, May 1984.

Franklin, Mark A. and Dhar, Sanjay [1986], "On Designing Interconnection Networks for Multiproessors", Proceedings of the 1986 International Conference on Parallel Processing.

	Number of Data Ports	Number of Backplane Data Pins	Diameter	Mean Internode Distance	Message Capacity	Frequency when Avg Msg Density > Msg Capacity	Variance in Avg Comm Time
Orthogonal Bus	2	256	1	0.88	32	10	580
Quad Bus	2	128	3	1.64	96	16	504
Chordal Ring	3	320	15	7.5	384	14	166
Grid	4	128	29	14.5	480	14	849
Torus	4	256	14	7	512	23	174
Hypercube	8	1024	7	3	1024	——	0.32
Bus Hypercube	4	128	3	1.45	208	——	0.03

Figure 4 : Comparision of the Networks Simulated

Feng, Tse-yun [1981], "A Survey of Interconnection Networks",Computer, December 1981.

Goke, L.R. and Lipovski, G.J. [1973],"Banyan Networks for Partitioning Muliprocessor Systems", Proceedings of the 1st Annual Symposium on Computer Architecture.

Lang, T and Stone, H.S. [1976], "A Shuffle-Exchange Network with Simplified Control", IEEE Transcations on Computers, Jan 1976.

Lawrie, D. [1975], "Access and Alignment of Data in an Array Processor", IEEE Transactions on Computers, Dec 1975. ;

Lawrie, D. and Padua, D. A. [1980], Analysis of Message Switching with Shuffle-Exchanges in Multiprocessors", Proceedings of the Workshop on Interconnection Networks for Parallel and Distributed Processing, 1980.

Lee, Manjai and Wu, Chuan-lin [1984], "Performance Analysis of Circuit Switch Baseline Interconnection Network", Proceedings of the 11th Annual Symposium on Computer Architecture.

Mazumder, Pinaki [1986], "Evaluation of Three Interconnection Networks for CMOS VLSI Implementation", Proceedings of the 1986 International Conference on Parallel Procesing.

Moldovan, Dan, Lee, Wing, and Lin, Changhwa [1989], "SNAP: A Marker Processing Architecture for Knowledge Processing". Technical Report CENG-89-10. University of Southern California-Department of Electrical Engineering Systems.

Pease, M.C.[1977], "The Indirect Binary N-Cube Microprocessor Array", IEEE Transactions on Computers, May 1977.

Reed, Daniel A, and Grunwald, Dirk C. [1987], "The Performance of Multicomputer Interconnection Networks", Computer, June 1987.

Wittie, Larry D [1981], "Communication Structures for Large Networks of Microcomputers", IEEE Transactions on Computers, April 1981.

Wu, Chuan-lin and Feng, Tse-yun [1980]. "The Reverse-Exchange Interconnection Network", IEEE Transactions on Computers, Sept. 1980.

Bandwidth Analysis of Message-Passing Networks

Wing C. Lee
Technical Report CENG-89-24
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California

Abstract

The design of effective multiprocessor systems is a difficult and complex task. In this paper, we examine one of the most critical issues in multiprocessor design: the design of the interconnection network. We built a simulator to study the characteristics of many interconnection networks to enable us to select a suitable network for a muliprocessor we are currently designing. Simulation results and evaluations for Bus, Ring, Grid, and Hypercube Networks are presented.

1 Introduction

The design of effective multiprocessor systems is a difficult and complex task. It involves the interaction of many diverse elements. They range from parallel algorithms to parallel processors, from programming languages to communication schemes.

In this paper, we will examine one of the most critical issues in multiprocessor design: the design of the interconnection network. An interconnection network enables the processors in a system to communicate with one another. A good interconnection network is vital for the stable operation of the multiprocessor. This requirement results from two basic tenets of parallel processing: effective cooperation of several processors on the same task and fast access to distributed data. Both required high communication bandwidths. If all of the processors are simply connected with the same bus, then this shared resource becomes a bottleneck preventing simultaneous communication between different pairs of processors. In this case, the effective throughput of the system actually goes does as the number of processors increase. A suitable interconnection network, which provides as much bandwidth as possible between any pair of processors, is thus needed.

This paper is based on work done for a multiprocessor, called SNAP [Moldovan 1989], currently under development at USC. We built a simulator to study the characteristics of many interconnection networks to enable us to select a suitable interconnection network for SNAP. The results we found are presented here.

The types of networks evaluated for SNAP were *static networks*. Static networks consist of active computing nodes connected by passive communication links. All active switching occurs at the nodes. Thus, we did not consider reconfigurable multistage switchiing networks, such as Banyan [Goke 1973], Omega [Lawrie 1975], shuffle-exchange [Lang 1976], and indirect binary N-cube [Pease 1977]. These methods have most often been sugested for data routing in SIMD machines, although some could be used to interconnect MIMD processors. There is a rich literature comparing these reconfigurable interconnection methods [Lee 1984] [Dias 1981] [Lawrie 1980] [Wu 1980] [Franklin 1986].

2 Simulator

Our simulator was built to model the performance of several types of communication schemes for SNAP.

2.1 SNAP

SNAP (Semantic Network Array Processor) is a multiprocessor targeted toward Artificial Intelligence applications. It consists of 256 custom-designed chips, packaged into four boards of sixty-four. For the purpose of our simulation, each chip was considered as a processing element capable of sourcing and sinking messages[1].

[1] In actuality each chip represents many nodes. However, only one message may enter or leave each of the data ports. Thus are assumption is valid.

SNAP communicates via messages. Each message consists of eight 8-bit packets. The format of the message is:

Byte 1: Destination
Byte 2: Message Type
Byte 3: Data
Byte 4: Data
Byte 5: Data
Byte 6: Data
Byte 7: Data
Byte 8: Data

The messages are sent together as one unit. Thus, if a chip gains access to a communication path, it keeps the path until all eight packets have been sent. Packets cannot get of order or become seperated.

Each chip has upto four data ports for communication with other chips. The number of data ports needed is one of the key results we hoped to gain from this simulation. Four data ports is an upper bound due to pin limations on the chip.

Each chip maintains an output queue for holding messages waiting to be transmitted. A routing controller inside each chip determines the shortest path to the destination and places the message onto the path. Thus, SNAP uses *distributed control* to coordinate message traffic.

2.2 Operation of the Simulator

The simulator was written in C, and created an environment whereby we could monitor the performance of each network. The simulator operated by introducing messages into the network and recording the time the messages took to reach their destinations. The messages were placed into the network at frequencies varying from (1 new message)/clock to (32 new messages)/clock. Thirty-two thousands messages were sent, and the networks were graded according to the average time it took a message to reach its destination.

Each processor had its own routing controller and queue. If a processor could not transmit a message, it would keep the message in its queue until a network path was available. The routing controller operated by computing the shortest path to the destination. If more than one path was avalilable, then the routing controller would choose one. If that path was busy, then others would be tried. No routing information was transmitted over the network, and thus, the routing controller made no attempt to route messages around congested areas.

The simulator ran on a sequential machine, so compromises had to be made in order to simulate a parallel architecture. Since, all the networks were synchronous, the smallest interval of time was one clock. Within a clock, messages were transmitted serially, with PE #1 attempting to transmit first and PE #255 transmitting last. Thus, some processors were given more favorable treatment than others.

New messages were placed at the end of the processor queues when the transmit phase was complete.

3 Evaluation Criteria

The interconnection networks were simulated to study two aspects: physical and computational.

3.1 Physical Aspects

A serious constraint in the physical design of large networks is imposed by the limited number of signal pins available both at the chip and board levels.

Chip Pin Constraints

We are restricted in the number of pins we can dedicate to communication. A key requirement is that to be able to fit 64 chips and the associated support logic into one 11 x 14 in board. Thus, each chip can only be around 100 pins. Subtracting pins dedicated for instructions and other uses, each chip can only support at most four data ports.

Networks that require more than four ports to connect 256 processors cannot be used. Networks that require less than four ports enable the chip to be smaller and thus provide more room for chip placement and margin. However, having less than four ports is not an important characteristic and was not given much weight.

Board Pin Constraints

Similar to the chips, we are restricted in the number of pins we can use to connect the processors on one board to the processors on another. Each board has connectors that plug into the backplane. The boards we are using have approximately 300 connections that can be used to comunicate with the processors on other boards. Networks that require more than 300 connections cannot be implemented and thus cannot be used.

3.2 Computational Aspects

The compuational aspects refer to the speed of computation and message flow within the network.

Average Communication Time

Average communication time is the most important consideration in the evaluation of a network. It is the average time required to send a message from one chip to another.

Average communication time is a value measured by simulator.

Network Diameter

The network *diameter* is related to the average communication time. It is the maximum number of communication links that must be traversed to transmit a message from any node along a shortest path. A network with a high diameter will have a higer communication time than a network with a small diameter.

Network diameter is an analytical value.

Mean Internode Distance

The *mean internode distance* is the expected number of link a "typical" message needs to traverse in order to reach its destination. The mean internode distance is a better indicator of the average communication time than the network diameter. However, it is a static value and does not take into account the queue delays that a message encounters at each node it visits.

Mean internode distance is an analytical value.

Message Capacity

The *message capacity* of a network is simply the number of messages that can be transmitted in the network at one time. If we assume that only one message can be transmitted on each of the communication links, then message capacity is also a measure of the number of communication links in a network. Networks with a high message capacity are able to handle heavier message loads and degrade more slowly than networks with a small message capacity.

Message capacity is an analytical value.

Average Message Density

The *average message density* measures the average number of messages circulating in a network. It is the sum of the number of messages being transmitted and the number of messages sitting in the queues of the chips. If the average message density is larger than the message capacity than the network is in danger of forming infinite queues.

Average message density is a value measured by the simulator.

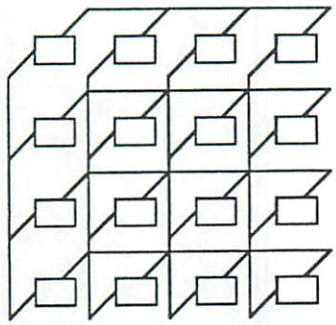
4 Communication Networks

We examined four different types of message passing networks: Bus, Ring, Mesh, and Hypercube.

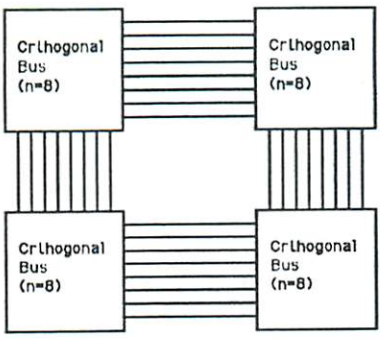
Figure 1 shows all of the interconnection networks simulatated. Figure 2 shows the average communication time as a function of the network message frequency. Figure 3 shows the message density of the networks as a function of message frequency.

4.1 Bus Networks

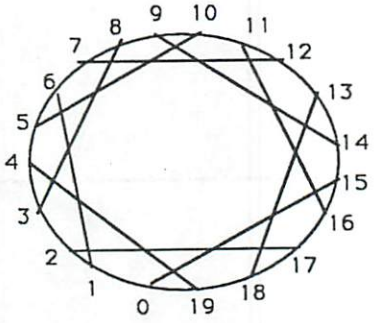
Single Bus



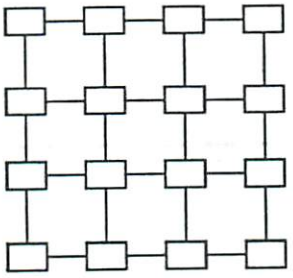
a) Orthogonal Bus (n=4)



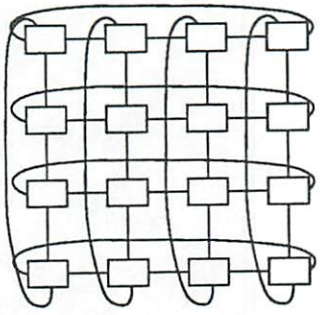
b) Quad Bus (256 PEs)



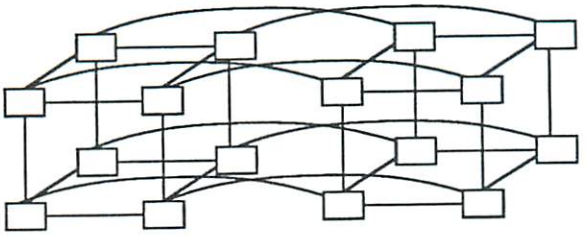
c) Chordal Ring (m=20, w=5)



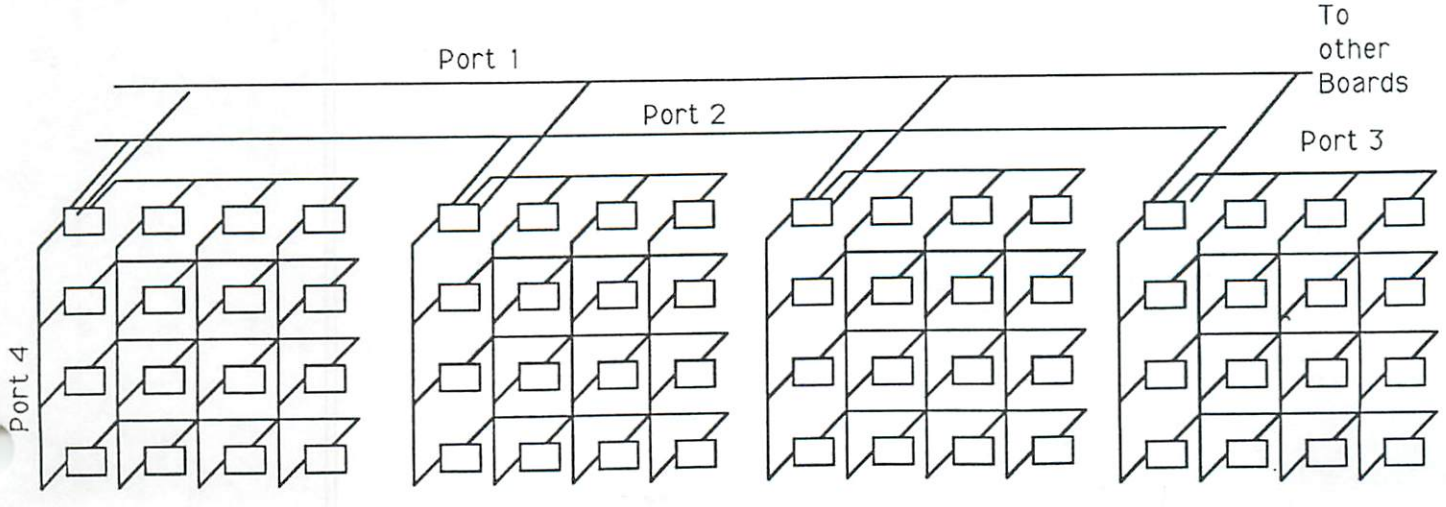
d) Grid (n=4)



e) Torus (n=4)

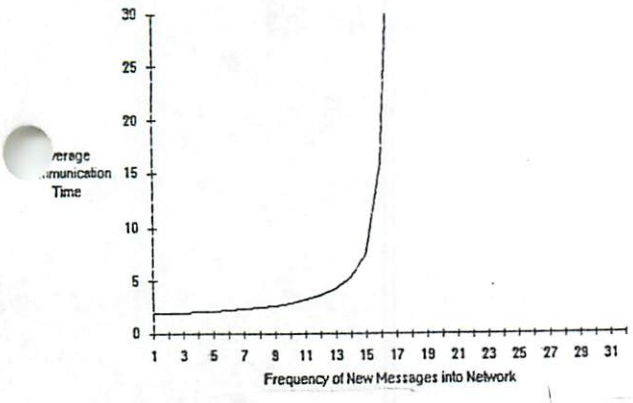


f) Hypercube(n=4)

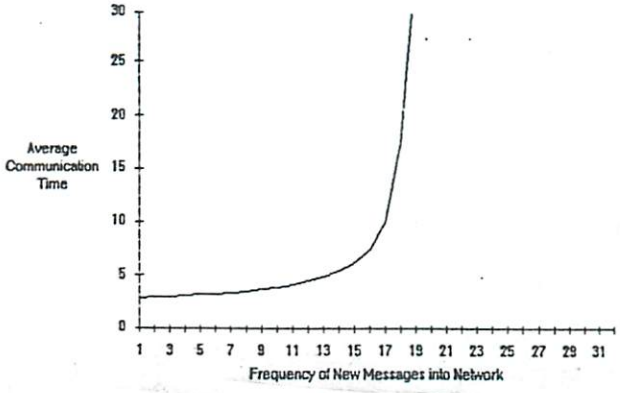


g) Bus Hypercube

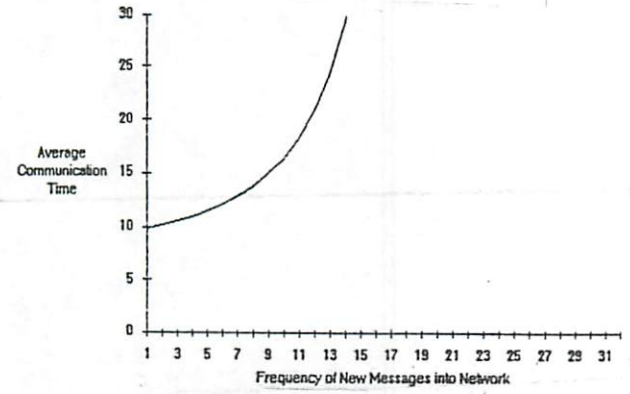
Figure 1: Examples of the Networks Evaluated



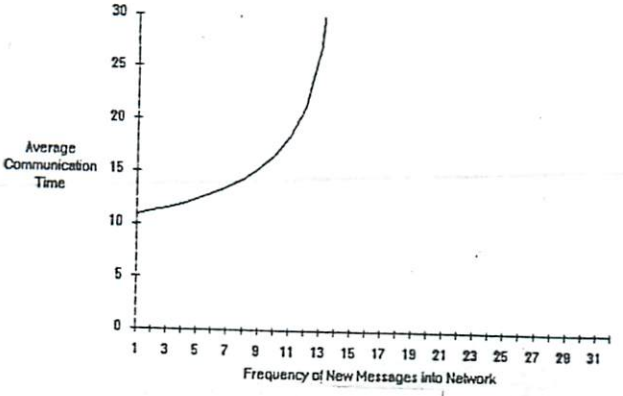
a) Orthogonal Bus



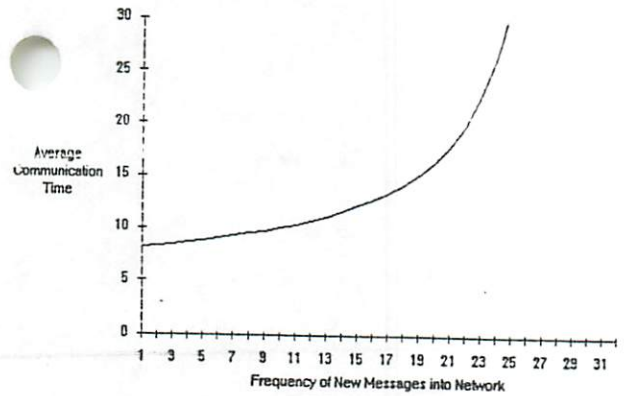
b) Quad Bus



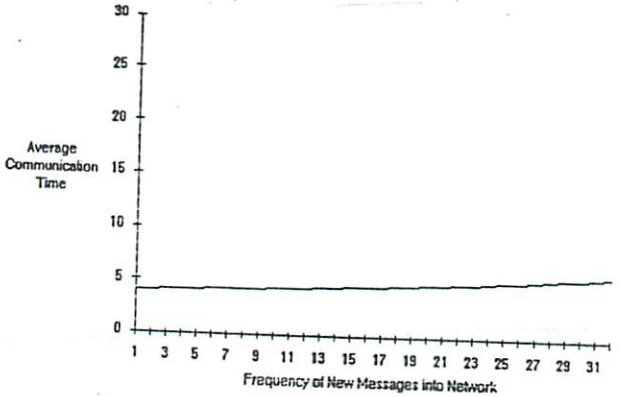
c) Chordal Ring



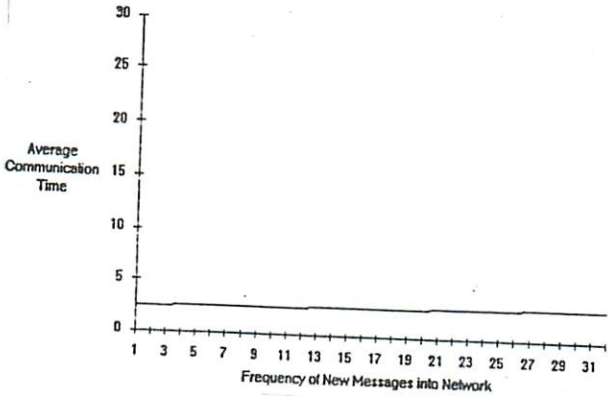
d) Grid



e) Torus

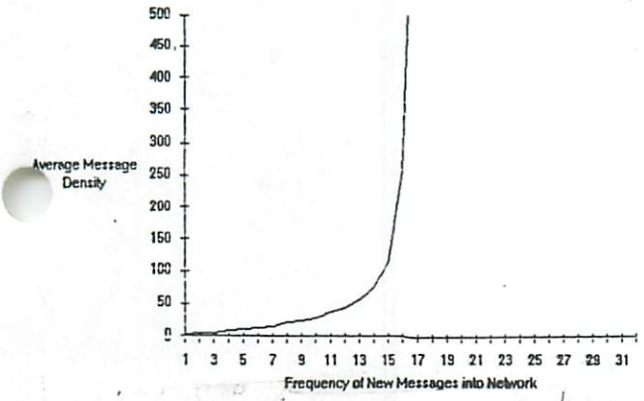


f) Hypercube

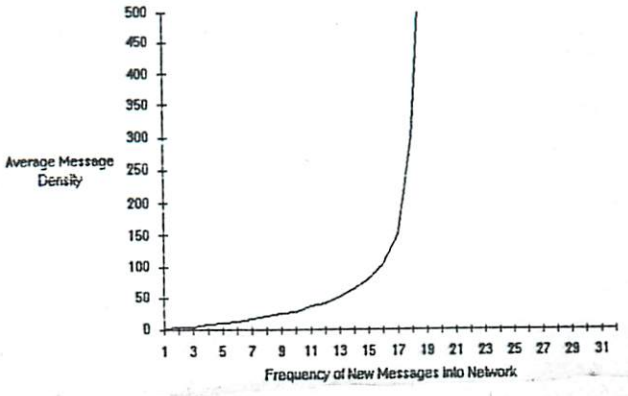


g) Bus Hypercube

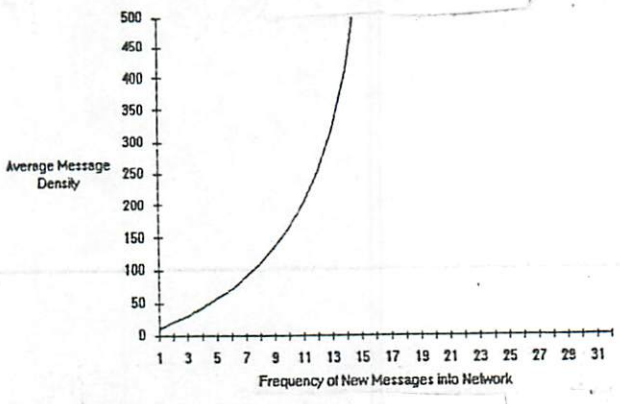
Figure 2: Average Communication Time of the Networks Simulated



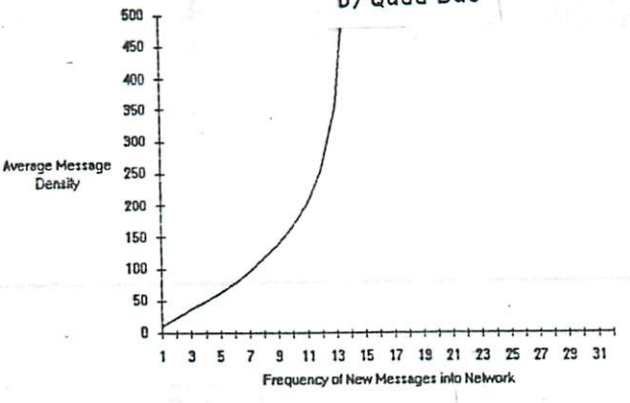
a) Orthogonal Bus



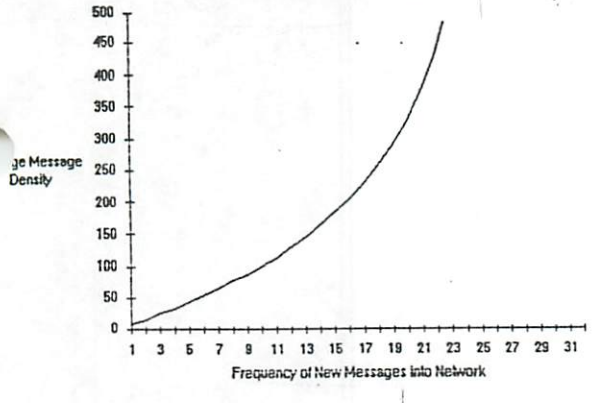
b) Quad Bus



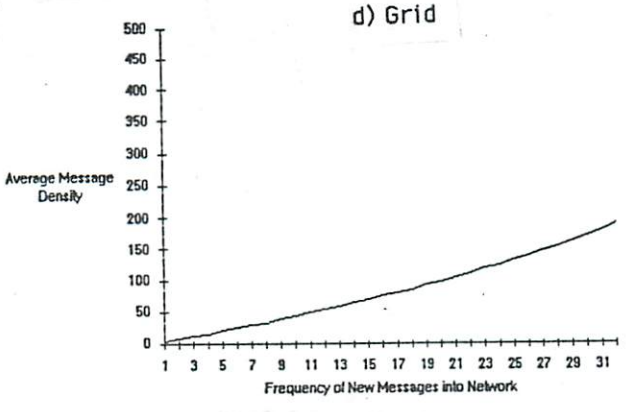
c) Chordal Ring



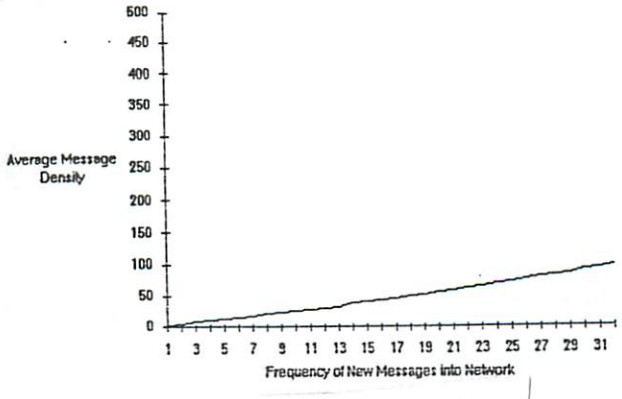
d) Grid



e) Torus



f) Hypercube



g) Bus Hypercube

Figure 3: Average Message Density of the Networks Simulated

The simplest interconnection network is a common bus connecting all of the processors. It has the characteristic of requiring only one clock cycle to transfer a message from one processor to another. However, a single bus can only send one message/clock and thus cannot support 256 processors.

Orthogonal Bus

An extension of a single bus is to have each processor connected to two buses. In an orthogonal arrangement, the processors are divided into rows and columns. Thus for 256 processors, the network consists of 16 rows buses and 16 column buses. The Orthogonal Bus network has the virtue of requiring a maximum of two message transfers to reach any destination. It has a message capacity of 32 messages/clock.

The simulation results show that the network works very well for lightly loaded networks. However, the network quickly degrades as the message load increases. At a message rate of eleven, the message density was greater than the message capacity, and at a rate of sixteen, the network became saturated and was unable to function.

The Orthogonal Bus suffers from two problems. The main problem is message capacity. As the message load increases, more processors try to send messages on the same bus at the same time. When this occurs, the simulator grants access of the bus to the processor with the higher priority. If the higher priority processors always have messages to send, then the messages in the other processors will wait indefinitely.

A related problem is routing. Messages have a most two routes to use. If either one of routes becomes congested, the messages have no recourse but to sit in a queue and wait. Plus, if the message at the head of a processor queue has a problem gaining access to the bus, then this prevents other messages in the queue from being transmitted, even if the other bus port is free.

Quad Bus

The Orthogonal Bus can be extended further into three or four bus networks. However, the backplane connectors can only support thirty-seven 8-bit buses. Thus, further extensions are not feasible.

Instead, the network can be divided into quadrants, with each quadrant corresponding to a board. Each quadrant has its own Orthogonal Bus network (8 row buses and 8 column buses). Backplane buses connect the quadrant row and column buses to their adjacent quadrant's buses. With this setup, the network can support nintey-six messages. The maximum transfer time of messages is four.

The simulation results in show that the network does not perform as well as the Orthogonal Bus for lightly loaded networks, but does slightly better at higher loads. Messages require more intermediate nodes in the Quad Bus than they do in the Orthogonal Bus. Thus, intrinsically messages take longer. At light loads, the message collision rate is low and thus the primary factor in determine the average communication time is the mean internode distance.

At higher loads, the Quad Bus suffers from the same problems as the Orthogonal Bus. The overall message capacity is ninety-six, but each quadrant can only send sixteen messages at a time.

Since each processor has only eight processors, the collision frequency is inherently less. However, Quad Bus messages travel through more intermediate nodes, so the probability that a processor wants access to the bus is greater.

In terms of routing, messages still have only two routes to choose from. Plus, once a message travels on one of the paths, it can no longer use the other route. Thus for three of the four legs a message may have to travel, it has only one path to use. This coupled with the message capacity limits the usefulness of the Quad Bus.

4.2 Ring Networks

Assuming the processors are numbered 0, 1, 2 ,.., (N-1) ,N, (N+1),... each of the processors in a ring are connected to their (N-1) and (N+1) neighboors. The two end processors are connected together to form a complete circle.

Sinple Ring

The simple ring is a modest improvement over the simple bus, but it too can only support a small number of nodes. The ring is limited to a small number of processors because the average communication time and message traffic density increase linearly with the number of nodes in the ring.

Chordal Ring

The Chordal Ring is the same as the simple ring, except each node has an additional link, called a chord, to some other node across the network. Each even-numbered node j (j=0,2,..,n-2) is connected to a node *(j-w) mod n*. The w is called the chord length and is assumed odd. For a given number of nodes n, a number of Chordal Rings can be obtained for different values of chord length w. [Arden 1981] analyzed Chordal Rings to find the optimal chord length for a given number of nodes. Using his results, we simulated a Chordal Ring with n=256 and w=19.

The diameter of the Chordal Ring that we simulated was fifteen and the message capacity was 384. The Chordal Ring is an improvement over the simple ring but suffers from routing and mean internode distance. There is only one optimal path a message can use to reach its destination. If one of the nodes in that path has a problem, then all of the messages that go through that node suffer as well. In additon performace suffers because the mean internode distance is high. At each intermediate node, a message must go through a queue. Thus, a message may have to wait in as many as fourteen queues. These two factors make Chordal Rings unisuitable for systems with a large number of processors.

4.3 Mesh Networks

In a Mesh Network, processors are organized into a two dimensional array. Each processor is connected to its four neighboor processors.

Grid Network

The Grid is the simplest Mesh Network. PE(i, j) is connected to PE(i-1, j), PE(i+1, j), PE(i, j-1), and PE(i, j+1) The edge PEs have three connections and the corner PEs only two.

For 256 processors, the network is 16 x 16. The maximum communication time is thirty and results from sending a message from one corner PE to another. The network appears robust as it can handle 480 messages at one time. However, as the simulation results show, the network becomes overloaded at just thirteen new messages/clock.

Unlike the Bus Networks, the Grid Network suffers from neither lack of message capacity nor lack of routing options. All the connections are point to point, thus the probability of collision is very low. In addition, since each processor can output onto four ports, the probability of using a particular point is only 25%. With an ability to send almost 500 messages and a low probability of collision, message capacity is not an issue.

Similarly, routing is not a problem. At most of the twenty-nine intermediate nodes a message can visit, a message has a choice of two paths to use. Thus, a message from one corner PE to the other has a choice over 40,000 possible routes.

The problem with the Grid Network is due rather to the number of intermediate processors messages have to pass through. The processor queues grow very quickly. At a message rate of fourteen, there are an average of over 600 messages waiting in the processor queues. Thus, every time a message reaches an intermediate node, it must wait a long time to get to the head of the queue. With as many as twenty-nine intermediate nodes to visit, the communication time quickly becomes intolerable. At a frequency of ten, it could takes almost sixty clock cycles for a message to reach its destination.

Torus

An improvement of the basic Grid formation is to add wrap-around paths to the edge and corner PEs. Thus, every PE has four connections and the worst case transfer time is reduced to fifteen.

Like the Grid Network, the message capacity in the Torus is very good (512 messages/clock) and the collision rate is very low. The number of possible routes is half of the number in the Grid, but 20,000 paths is more than adequate.

The simulation results show that Torus is a much more stable network than the basic Grid. Degradation occurs at twenty-four, which is almost double the degradation rate of the Grid Network. The intermediate processor queues build up more slowly in the Torus. This is due to the fewer nodes a message must visit. The maximum queue size stabilizes at around twelve messages until the queues explode at a frequency of 24.

The Torus would be almost ideal for light to moderate message traffic, if the worst case communication time were better. But, at best a message could take almost twenty clock cycles to reach its destination. This is five times worst than the performance of the Orthogonal Bus network at light loads.

4.4 Hypercube Networks

In a Hypercube organization, the N processors are placed at the vertices of a hypercube of dimension k, where k=$log_2 N$. The PEs are numbered 0,1,....,(N-1), and PE(i) is connected to PE(j) such that the binary representation of j differs from that of i in exactly one position.

Basic Hypercube

A hypercube of dimension eight is required for 256 processors. Each processor has eight connections and the maximum number of intermediate nodes for a message is seven. The network is capable of transmiting 1024 messages simultaneously. Similar to the Grid, the Basic Hypercube has over 40,000 possible routes for a message to use. But, the advantage of the Basic Hypercube is that a message has to visit less than 25% of the number of nodes that a Grid Message does.

Thus, it is not suprising that the performance of the Basic Hypercube is excellent. The network was able to handle all the message frequencies with little variance in communication time. At the heavyest load, the average time a message required was still under six clocks. The worst case communication time remained very stable, averaging around ten clock cycles. The queues grew linerly, and at the end of simulation were only at 18.5% of message capacity.

However, the Basic Hypercube suffers from two main problems. The first problem is that the Hypercube is very expensive to implement. Over 8000 connections (wires) are required just for the data lines. The second problem arises from the physical constraint we placed on the network at the very beginning. Each of the four processor boards have only 300 connections, but the Hypercube requires 1024 connections between boards. Thus, the Basic Hypercube is not feasible.

Bus Hypercube

One modification to the Basic Hypercube is to use buses instead of point to point connections. In the Bus Hypercube, each processor has four ports. Assuming that the binary representation of the addresses of processors is (a7 a6 a5 a4 a3 a2 a1 a0), the configuration of the ports is:

Port 4 communication with processors that vary in (a1 a0)
Port 3 communication with processors that vary in (a3 a2)
Port 2 communication with processors that vary in (a5 a4)
Port 1 communication with processors that vary in (a7 a6 a5 a4)

Note that Port 1 is a superset of Port 2. Port 1 is used for communication between boards, so this arrangement requires only 16 buses between boards instead of 64. Ports 2-4 are buses that connect four processors. Port 1 connects sixteen processors. The network has 64 each of Ports 2-4 and 16 of Port 1. The maximum number of intermediate processors is three (two if Port 1 is used) and the message capacity is 208. The Bus Hypercube requires 1664 data connections, of which 128 are between boards.

Routing is not as good as in the Basic Hypercube. There are only twenty-four possible routes that a message may take. But with a maximum of only three processors to visit, the fewer routes is not a problem.

The simulation results show that the Bus Hypercube is the best network simulated. The average communication time was almost the same for lightly loaded networks (2.47 clocks) as it was for the heavyest loads (3.02 clocks).

The Bus Hypercube performs better than the Basic Hypercube because it takes half as many intermediate nodes to send a message. Even though the queues in the Basic Hypercube networks are small, they still require time to pass through. The limitation of the Bus Hypercube is that the message capacity is only a fifth of the Basic Hypercube. But since the maximum number of messages the network can source at a time is 256 (one message per processor), most of the bandwidth of the Basic Hypercube is wasted.

5 Conclusions

We studied several types of communication networks for SNAP: Grid, Bus, Ring, and Hypercube. Their characteristics are summarized in figure 4.

The Basic Hypercube had the largest message capacity, followed by the Mesh Networks. The Bus Networks required the fewest intermediate nodes, while the Mesh Networks required over ten times as much. The Basic Grid and both of the Bus Networks had the biggest variance in communication time, while the Hypercubes had the least.

The Bus Networks had problems with bus loading and are useless for all but lightly loaded systems. The Mesh Networks suffered from the delays that resulted from sending messages through too many intermediate processors. The Ring Networks lacked routing options and have a high mean internode distace. The Hypercube networks had the best performance. The Basic Hypercube network was too expensive to implement, but the Bus Hypercube stands out as the best network. The Bus Hypercube was the network chosen for SNAP.

6 Acknowledgement

The author would like to thank Dan Moldovan for his discussions and comments on this paper.

References

Arden Bruce W and Lee, Hikyu [1981], "Analysis of Chordal Ring Network", IEEE Transactions on Computers, April 1981.

Diaz Daniel M. and Jump, J. Robert [1981], "Analysis and Simulation of Buffered Delta Networks", IEEE Transactions on Computers, April 1981.

Doty Karl W. [1984], "New Designs for Dense Processor Interconnection Networks", IEEE Transactions on Computers, May 1984.

Franklin Mark A. and Dhar, Sanjay [1986], "On Designing Interconnection Networks for Multiproessors", Proceedings of the 1986 International Conference on Parallel Processing.

	Number of Data Ports	Number of Backplane Data Pins	Diameter	Mean Internode Distance	Message Capacity	Frequency when Avg Msg Density > Msg Capacity	Variance in Avg Comm TIme
Orthogonal Bus	2	256	1	0.88	32	10	580
Quad Bus	2	128	3	1.64	96	16	504
Chordal Ring	3	320	15	7.5	384	14	166
Grid	4	128	29	14.5	480	14	849
Torus	4	256	14	7	512	23	174
Hypercube	8	1024	7	3	1024	——	0.32
Bus Hypercube	4	128	3	1.45	208	——	0.03

Figure 4 : Comparision of the Networks Simulated

Feng Tse-yun [1981], "A Survey of Interconnection Networks",Computer, December 1981.

Goke L.R. and Lipovski, G.J. [1973],"Banyan Networks for Partitioning Muliprocessor Systems", Proceedings of the 1st Annual Symposium on Computer Architecture.

Lang T and Stone, H.S. [1976], "A Shuffle-Exchange Network with Simplified Control", IEEE Transcations on Computers, Jan 1976.

Lawrie D. [1975], "Access and Alignment of Data in an Array Processor", IEEE Transactions on Computers, Dec 1975.

Lawrie D. and Padua, D. A. [1980], Analysis of Message Switching with Shuffle-Exchanges in Multiprocessors", Proceedings of the Workshop on Interconnection Networks for Parallel and Distributed Processing, 1980.

Lee Manjai and Wu, Chuan-lin [1984], "Performance Analysis of Circuit Switch Baseline Interconnection Network", Proceedings of the 11th Annual Symposium on Computer Architecture.

Mazumder Pinaki [1986], "Evaluation of Three Interconnection Networks for CMOS VLSI Implementation", Proceedings of the 1986 International Conference on Parallel Procesing.

Moldovan Dan, Lee, Wing, and Lin, Changhwa [1989], "SNAP: A Marker Processing Architecture for Knowledge Processing". Technical Report CENG-89-10. University of Southern California-Department of Electrical Engineering Systems.

Pease M.C.[1977], "The Indirect Binary N-Cube Microprocessor Array", IEEE Transactions on Computers, May 1977.

Reed Daniel A, and Grunwald, Dirk C. [1987], "The Performance of Multicomputer Interconnection Networks", Computer, June 1987.

Wittie Larry D [1981], "Communication Structures for Large Networks of Microcomputers", IEEE Transactions on Computers, April 1981.

Wu Chuan-lin and Feng, Tse-yun [1980]. "The Reverse-Exchange Interconnection Network", IEEE Transactions on Computers, Sept. 1980.