

AD-A104 747

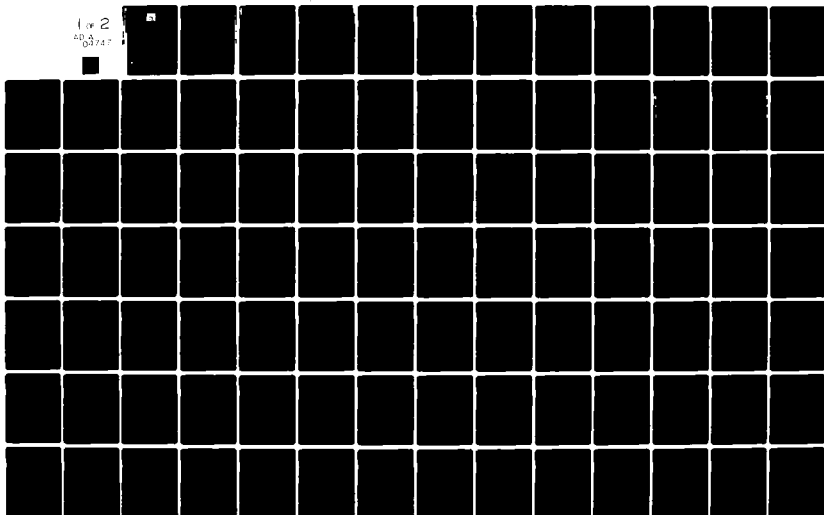
WASHINGTON UNIV ST LOUIS MO CENTER FOR COMPUTER SYST--ETC F/G 9/2
VLSI BASED MULTIPROCESSOR COMMUNICATIONS NETWORKS.(U)
SEP 81 M A FRANKLIN, D F WANN

N00014-80-C-0761

NL

UNCLASSIFIED

1 of 2
AD-A
03747



AD A104747



12

WASHINGTON UNIVERSITY

SCHOOL OF
ENGINEERING
AND
APPLIED SCIENCE

Annual Progress Report

for the

Office of Naval Research

NR#:375-033

CONTRACT #: N00014-80-C-0761

SEP 20 1981

A

VLSI BASED MULTIPROCESSOR COMMUNICATIONS NETWORKS

Submitted by

Washington University

Center for Computer Systems Design

Principal Investigators: Mark A. Franklin, Donald F. Wann

Date: September 1981

THE UNIVERSITY OF CHICAGO
LIBRARY
SEP 20 1981

WASHINGTON UNIVERSITY / ST. LOUIS / MISSOURI 63130

819 29 040

DTIC FILE COPY

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A104747	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) VLSI BASED MULTIPROCESSOR COMMUNICATIONS NETWORKS		5. TYPE OF REPORT & PERIOD COVERED Interim Progress Report Sept. 1, 1980-August 31, '81
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Mark A. Franklin Donald F. Wann		8. CONTRACT OR GRANT NUMBER(s) ONR-N00014-80-C-0761
9. PERFORMING ORGANIZATION NAME AND ADDRESS Washington University Center for Computer Systems Design - Box 1115 St. Louis, MO 63130		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61153N 021-05 01 375-033
11. CONTROLLING OFFICE NAME AND ADDRESS ONR-Office of Naval Research-Code 414 P. Papantoni-Kazakos Arlington, Virginia 22217		12. REPORT DATE September 1, 1981
		13. NUMBER OF PAGES 113
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution Unlimited		
17. DISTRIBUTION STATEMENT (of this abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Interconnection Networks, Connecting Networks, VLSI, Partitioning of Networks, Network Control		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) An analysis of the impact of VLSI technology on the design of multiprocessor communications networks is presented. Models for partitioning networks in terms of subnetworks which reside on a single VLSI chip, and in terms of path bit slicing are shown. Pin limitations, intra and inter-chip network type and various VLSI technology parameters are included in the model and results of simulation studies yielding optimum partitioning strategies under a variety of conditions are given. Investigations of procedures for handling network control are detailed and a promising design which solves the plane synchronization problem is illustrated.		

9 ANNUAL PROGRESS REPORT *7 Sep 80 - 31 Aug 81*
for

THE OFFICE OF NAVAL RESEARCH

NR#: 375-033

CONTRACT # 15 N00014-80-C-0761

Submitted

by

Washington University

Center for Computer Systems Design

St. Louis, Missouri 63130

Contract Title

6 VLSI BASED MULTIPROCESSOR COMMUNICATIONS NETWORKS

Principal Investigators

10 Mark A. Franklin Donald F. Wann
Professor: Electrical Engineering Professor: Electrical Engineering

Date

11 September 1981

412560

A

TABLE OF CONTENTS

1. Introduction
 2. Research Summary and Major Accomplishments
 - 2.1 Formal Models
 - 2.2 Space-Time Performance Bounds
 - 2.3 Partitioning and Pin Limitations
 - 2.4 Synchronization of Partitioned Networks
 3. Research Tasks: Year Two
 4. Conclusions
- Appendix I: Interconnection Networks - A Formal Graph Model and Survey
- Appendix II: VLSI Based Interconnection Networks
- Appendix III: Pin Limitations and VLSI Interconnection Networks
- Appendix IV: Word Inconsistency in Partitioned VLSI Interconnection Networks
- Appendix V: Digital Systems: Research Review and Perspectives on the Future

VLSI BASED MULTIPROCESSOR COMMUNICATIONS NETWORKS

Progress Report: Year 1

Mark A. Franklin and Donald F. Wann

1. Introduction

This document is the Annual Progress Report for the Office of Naval Research contract number N00014-80-C-0761, (NR#: 375-033) entitled "VLSI Based Multiprocessor Communications Networks". The contract began on September 1, 1980 and was approved on scientific/technical grounds for a duration of three years. Incremental funding was approved for year two of the research and this work has just begun. The bulk of this report documents research progress and major achievements during the first funding year of the contract. In addition, research plans for the coming year are reviewed.

The research undertaken has been principally concerned with high bandwidth communications networks suitable for use in multiprocessor computer systems. The goals have been to study the impact of VLSI technology on the design of such networks, and to advance the development of design methodologies oriented to this technology and application domain.

The need for such an effort and the rationale for the approach taken was discussed in the original proposal. Briefly, the research is motivated by three factors. The first relates to current computer architecture efforts at achieving very high computational power and reliability. As technology presses the physical limits of component performance (1,2) large increases in computational power will become achievable principally through the exploitation of parallel numerical methods (3-8) implemented on appropriate parallel (multiple processor) computer systems (9-15). Such tightly coupled computer systems (perhaps made up of large numbers of low cost microprocessors) inevitably require high bandwidth communications networks for exchange of information

and sharing of memory. Thus the need for such networks is present and growing (16-20). This is explored in a somewhat wider context in Appendix V. Note that the sort of high computational power referred to occurs in numerous advanced Navy applications ranging from complex pattern recognition and signal processing problems to on-line missile defense and tracking problems.

The second factor relates to the costs and performance associated with such networks when embedded in large (hundreds to thousands of microprocessors) computer systems. A poorly designed network can rapidly become a performance bottleneck as the number of processors and traffic in the system increases. Very high bandwidth networks, on the other hand, can be extremely costly, and may indeed have a cost which grows faster than the growth of processors in the system (e.g. the number of processors may grow as $O(N)$ while the network may grow as $N \log N$). Thus for large systems, the cost of the network may dominate the cost of the overall multiprocessor system, and the performance of the network may be the critical factor in overall systems performance.

This leads to the third and final point to be considered, the role of VLSI. VLSI technology opens up a new, and largely unexplored, design domain which possesses certain promising properties and interesting problems when applied to network design. There is great potential here for reducing the costs while maintaining or enhancing the performance of interconnection networks. The complexity and intelligence of the switching nodes, combined with the regular topologies associated with such networks seem to make them well-suited to requirements for successful VLSI design (i.e. large amounts of logic needed in a step and repeat manner). On the other hand the problems of network nonplanarity pin limitations, network partitioning, and synchronization represent problems which must be overcome before well-designed VLSI networks

can be achieved. This contract is concerned with the exploration of these questions.

The major accomplishments of the research thus far are summarized in Section 2 to follow. These accomplishments can be divided into four parts. The first relates to developing formal graph models which permit characterization of general interconnection networks. The second concerns application of these models, in conjunction with a general descriptive model of VLSI components, to determine space (area of the fabricated chip) and time (delay in communication through the chip) bounds of various networks when implemented in the VLSI technology so that pin constraints of the packaged chip can be satisfied. The third deals with how such VLSI based interconnection networks can be partitioned in an optimum manner. The fourth reports on an important synchronization problem which arises in partitioned networks and considers certain design techniques for overcoming this problem.

Section 3 summarizes our plans for year two of the contract. Basically the outline discussed in the original proposal will be maintained. This includes further work in the research areas discussed above. Section 4 concludes with a summary discussion of the research thus far.

A number of appendices follow the main body of the report. These include research papers which have been published or submitted for publication during the first contract year, and several working papers which discuss research in progress.

2. Research Summary and Major Accomplishments

2.1 Formal Models

One of the first research tasks undertaken concerned determining what effect differing interconnection topologies have on such VLSI oriented measures as chip area and delay. Note that traditional network measures of complexity have almost always centered on costs determined by switch counts, and delays based on aggregate mean path switch delays. Such measures indeed make sense in an environment where either discrete electrical or electromechanical devices are utilized. When considering placing networks on VLSI chips, however, the situation changes substantially. Within a chip networks often tend to be connection intensive rather than component intensive. That is, an extensive area on the chip is occupied with connections between components rather than with the components themselves. Furthermore, the delays associated with signal propagation along these connections can be an important component in the overall delay.

These lines of inquiry were initially pursued in a comparison study of Banyan and crossbar networks (21). In that work, a constructive approach was used to determine the area and delay requirements of these two networks when implemented on a single chip. The technical approach used associated a particular chip layout with each network. This layout was posed as being near optimal. The layouts reflected the topology of the networks, and various geometric and physical electronics arguments were presented in developing overall area and delay expressions. While this work was successful, it clearly demonstrated the need for a more comprehensive study of the problem. That is, a more general approach was needed which was not so tied to particular networks or layout assumptions.

Research was pursued in this area over the past year. First a general graphical approach to specifying the topological properties of interconnection schemes was developed and tested on a variety of networks. While reviewing the principal interconnection networks in terms of their graph specifications, a characterization of these networks from both blocking and traditional complexity viewpoints was undertaken. The blocking properties of a network relate to how connections and components in the network are shared by different input/output paths, and are important factors in determining the bandwidth of the network. This work is reported in Appendix I.

While a graph model may be used to specify network topology, there remain the problems of specifying the VLSI components onto which the network is mapped, and the procedure to be followed in the mapping process. With regard to the first problem, one would like a reasonably high level approach to specifying components which avoids the issues of detailed electronic design while retaining realistic performance properties. In the method adopted, components (i.e gates) are modelled in terms of parameters describing their active channel areas, fixed pullup to pulldown ratios, device sizes related to minimum feature sizes, and bounding boxes which can be readily mapped onto a chip area grid. Time in this model is associated with the time required for a minimum size transistor to drive the capacitance of a unit length line. A full description of the model is not pursued here; however, work is proceeding (22) and will be reported in a subsequent paper. The second problem of describing the VLSI mapping procedure will be considered in section 2.2.

2.2 Space-Time Performance Bounds

There are three principal approaches to the mapping problem discussed in the prior section. The first is to perform detailed chip designs and layouts for the networks of interest and from these designs determine the space and time performance measures in question. While this is clearly very time consuming and perhaps impossible when more than a few networks are involved, of more importance is that a clear comparison between networks is very difficult since uniform design standards are essentially nonexistent. In this situation differences in design details may have an important effect on results. The second approach considers plausible layouts of networks where individual network nodes are assumed to be rectangular regions of known area. Detailed logic design is avoided with this approach and order expressions (e.g. $O(N \log N)$) for the performance measures of interest can be derived (21). While this intermediate level of analysis can yield fairly realistic expressions for area and delay it still relies on individual design judgments to determine reasonable node and connection layouts.

The final approach considers networks as abstract computation graphs and attempts to develop lower bound expressions for arbitrary networks based on topological properties of their graphs and some knowledge of the information flow through the graphs. Thompson (23) pioneered in this area (now referred to as VLSI complexity theory) by combining certain graph theoretical results with a general VLSI component and time model. His main result is that the area occupied by the wires and nodes of a VLSI design that corresponds to a graph with minimum bisection width w is greater than $w^2/4$. Informally, the minimum bisection width of a graph is the smallest number of edges that must be removed to disconnect one half of the vertices of a graph from the

other half. While being asymptotically tight, this bound is in many cases weak. The second half of Thompson's model has to do with time, and though important, will not be pursued here.

There are two important drawbacks to Thompson's development. The first relates to the VLSI area model he used in formulating the area bounds. A key assumption made is that the area assigned to nodes is independent of the logic inside them and varies as the square of the input and output lines associated with the node. Thus a node with 4 input and output lines is assigned an area of 16 units irrespective of the internal logic of the node. The second drawback concerns the calculation of time delays. The model takes no account of delay within nodes and assumes unit delay across wires (by assuming logarithmically staged line drivers matched to each line). The result is that the time bounds established are very weak.

The goal of our research in this area has been to preserve the spirit of Thompson's approach while making the model more realistic. That is, to preserve the graph theoretic orientation which permits uniform application of the model over all graphically defined networks, but to provide for more appropriate area and time expressions and thus obtain tighter lower bounds on network area. To do this a number of techniques have been used to determine node area.

The first technique considers establishing node areas based on fan-in/fan-out considerations. These results are not significantly better than Thompson's bound but provide some insight into various node area arguments. In particular, the bounds can be applied where logic within a node is simple, but the node has a large degree (e.g. a parallel in/parallel out shift register). The second technique demonstrates how node areas can be defined in a recursive manner based on recursive

definitions of network computation graphs. The third technique applies in situations where the nodes are defined only in terms of their functional capabilities, not in terms of given logic implementations. In these cases a finite state machine model of a node has been proposed, and a clocked PLA (Programmable Logic Array) implementation developed. Lower bound expressions for the area required by the machine can be obtained and used as a lower bound on node area. Details of this work are now being prepared. The major results of our area studies applied to interconnection networks are summarized in the table below. In almost all cases the lower bounds obtained are significantly higher than those obtained using traditional component count measures.

Network	Blocking Category	Area (lower bound)	Control Included?
Crosspoint Switch	Nonblocking	N^2	Yes
Mesh Connected Crossbar	Nonblocking	N^2	Yes
Clos Network	Nonblocking	$N^{2.5}$	No
Delta Network	Blocking	N^2	Yes
ORAN	Rearrangeable	N^2	No
Batcher	Rearrangeable	N^2	Yes

Interconnection Networks in VLSI
(N=Number of Ports)

Research on establishing tighter time bounds has been pursued for those computation graphs with loop-free input-output paths. Most interconnection networks of interest fall in this category. The initial

assumption has been made that the system operates in a synchronous manner with a two phase clock where the active phases of the clock are used either for capturing the input data at nodes or for validating output data at nodes. Inter-clock periods are used either for computation within nodes or transfer of information between nodes. Based on the PLA structure of the finite state machine and the topology of the network, bounds on the node time and the data transfer time can be obtained. In conjunction with the area bounds discussed above, space-time bounds can be derived. The time bound derivations here are complex and are still being developed. A written report on this material will be forthcoming. In the sections to follow we move from the individual chip domain to the systems domain where problems which arise when large networks requiring collections of chips are investigated.

2.3 Partitioning and Pin Limitations

Once a fuller understanding of the impact of network topology, layout and general VLSI design constraints has been achieved at the single chip level, it is important to examine the problems associated with large networks requiring many chips. The problems of chip interconnections, pin limitations and network partitioning have often been omitted from the formal modelling process. These questions seem to fall in the domain of "packaging" problems and related esoterica, and as such have been traditionally neglected by the research communities interested in characterization of formal design processes and methodologies. Indeed the research discussed here is unique and the publication found in Appendix III is one of the only reported efforts in the area.

The principal problem can be seen from a simple example. Consider a network with N' input ports and M' output ports, with each port being B' bits in width. Pick N' , M' and B' to be 12, 12 and 16 so that the logic required for implementation will have little difficulty fitting on a single VLSI chip. To support this chip at least $B'(N'+M')$ or 384 pins would be required. This is much larger than common commercially available integrated circuit carriers. Given that pins are typically placed on 100 mil centers along the periphery of the package, the total number of pins is limited mainly by the increase in the physical length of the package, which in this example would require a 19.2 inch dual-in-line package.

Another way of looking at this is in terms of the number of pins, P_{in} , typically required when implementing a logic function requiring C circuits. This has been found to be reasonably approximated by (2):

$$P_{in} \approx KC^b \qquad b \approx .5$$

One would expect a function of this form since the number of circuits on a chip varies directly with the chip area, while the number of pins varies directly with the chip perimeter. A straightforward set of calculations indicates that for interconnection networks of the sort considered the value of K is more than an order of magnitude greater than would be expected from circuit number considerations alone.

Given the pin constraints imposed by standard packaging technologies, the problem is how does one optimally partition a large network so that pin constraints are satisfied, and a given performance measure (e.g. chip count, network delay, count-delay product) is minimized. This is the problem attacked in the paper given in Appendix III.

Two partitioning strategies are introduced, and expressions for obtaining the optimum chip size and chip data path width are developed for two different network types, and two standard network control schemes (e.g. synchronous and asynchronous). These expressions are parameterized on the basis of chip pin constraints and a variety of VLSI component parameters. They can be readily used as part of a larger design study.

One key conclusion is that while single bit per slice partitioning is often optimum when dealing with incremental or mesh connected cross-bar networks, it is generally not optimum when $N \log N$ networks (e.g. Banyan, Omega, Inverse Binary N -Cube) are used. For example when implementing a Banyan network of size $N'=M'=512$ and $B'=16$, if network delay is the performance measure, and the chip is limited to 90 pins, then selecting a single bit per slice rather than the optimum eight bits per slice will result in about a factor of four increase in delay.

A full discussion of this material is presented in Appendix III which was presented at the 1981 International Symposium on Parallel Processing and published in the symposium proceedings.

2.4 Synchronization and Partitioned Networks

All communication networks, whether partitioned or not, must have some type of control that a) allows the network paths to be established, and b) permits the orderly transfer of data from a source to a destination port. For very small networks that do not require partitioning and thus can be placed on a single VLSI chip, it is possible to design an efficient control mechanism that has a central physical location and which transmits its control information over special paths to the various switching and transmission elements. Once the network size increases (either by adding additional ports or by increasing the number of bits per port) to a point where partitioning is necessary, modularity of the design becomes very important and the use of a central control structure places severe limitations on the network designer and on the network performance. This is a result of two factors: 1) with modular networks the size of the control structure is usually unknown at design time, thus it may be necessary to include a large control structure even though it is not often fully utilized (e.g. control for 1000 ports, even if only 50 ports are used) and 2) the large size of the partitioned network may result in physically long paths from the central control to the individual switching elements, thus requiring excessively long times for path establishment and data transfer. In fact, the longest path, and thus the longest time, often dictates the overall network performance.

We have made a relatively thorough investigation of these issues and have explored how one might employ a modular switching element combined with a distributed self-timed control structure for such partitioned communication networks. The detailed results of this investigation are presented in Appendix IV, where we show the specification for a self-timed

switching module that uses local, distributed control and that can be used in a modular manner to create networks of any arbitrary size. Since these modules operate by responding to sequences of signals (i.e. signal occurrences must be ordered) rather than requiring signals to be separated by certain time intervals (i.e. signals must precede other signals by x nanoseconds) it is not necessary for the designer to redo or readjust the logical design (and hence the VLSI implementation) as the network size changes. Thus they offer some significant advantages if large, variable size networks are to be constructed.

One unexpected "synchronization" problem had to be overcome in the development of this class of network interconnection. The problem appears only when the bits in a word are partitioned into separate individually controlled bit planes. As mentioned in Section 2.3 in many situations the optimum partition occurs with small numbers of bits per plane and in certain cases is produced with one bit per plane. To understand the synchronization problem, consider a system having many ports, each with an 8 bit word where each source and destination bit interconnection is achieved on its own bit plane (e.g. 8 planes). Notice that if two sources, say S_i and S_j are concurrently trying to establish a path to the same destination, say D_k it is possible that paths from S_i to D_k will be established on some planes and paths from S_j to D_k will be established on other planes. This is a direct consequence of distribution of control - each switch element operates in an autonomous mode (desirable) but operates without a knowledge of how the other bits in the word are being treated on the other planes (undesirable). Thus it is possible for the destination to receive a mixture of bits from these two sources; we call this a nonhomogenous or inconsistent word. Notice that this only occurs during the path establishment phase; once the path is established, all bits are transmitted properly. Fortunately, we have been able to develop a simple procedure that allows us to

detect when an inconsistent word occurs during path establishment and this can be used to generate a path retry request for that port. We have also made a theoretical analysis to determine how often such a problem will arise (as a function of certain network parameters, such as switch element delay, request rates, etc.) and have found that the probability of generating such an inconsistent path is normally quite small (e.g. $P_{\text{retry}} < 0.07$). Finally we point out that central control does have one advantage over distributed control: it requires fewer pins. We currently are investigating this issue and are considering modules that are still locally controlled but that are "nearly" self-timed.

3. Research Tasks: Year Two

Research into quantifying the area and delay properties of various network types when layed out on a VLSI chip will continue. Some of this work, as described in sections 2.2 and 2.3, is near completion and documentation of the results is now underway. The models developed will be used in part to investigate the properties of different types of crossbar switch design.

Research on the effects of VLSI chip pin constraints on network partitioning will be extended to consider the impact of data pipelining and path blocking. Computer based modeling and simulation studies will be undertaken in this area with the goal of obtaining partitioning strategies which are near optimum over a wide range of network sizes and chip pin constraints. The possibilities of developing designs for a switch chip set which would be suited to a variety of network needs and sizes will be examined.

The problem of quantifying the impact of physical constraints (i.e. component, pin, power densities) at the chip, board and rack levels on the partitioning of VLSI oriented chip arrays will be explored. While the networks to be considered will primarily be interconnection networks, other networks more oriented towards special purpose computation will also be examined as time permits.

Studies on the synchronization of bit sliced, plane partitioned networks will be completed and preliminary research on the effects of centralized versus decentralized control of networks will be extended. The modularity, growth and reliability properties of centralized and decentralized control schemes will be explored. Based on the results of the above research, work

will begin on the physical implementation of a VLSI network chip. This will act as a testbed for a number of the research ideas developed.

4. Conclusions

This annual report has documented research progress and achievements which have occurred during year one of ONR contract N00014-80-C-0761 entitled "VLSI Based Multiprocessor Communications Networks". The work was performed at the Washington University Center for Computer Systems Design, St. Louis, Missouri. This work has been motivated by the potential for increased speed and high reliability associated with the implementation of multiple processor systems, recognition of the importance of the interconnection network over which the processors communicate, and by the availability of new design options afforded by the ongoing VLSI technology revolution.

A great deal of progress has been made on the basic research tasks outlined in the original proposals. These accomplishments span the development of formal network and VLSI based models; the establishment of space and time performance bounds for various networks when implemented on a single chip; the study and modelling of the partitioning of large networks requiring many chips and having severe pin limitation constraints; the analysis of an important synchronization problem which occurs in partitioned networks, and the preliminary functional design of a switch component which solves the synchronization problem while maintaining network modularity and distributed control.

Proposed research during year two is outlined in Section 3 of this report. Work will continue on tasks begun during the first year. Work

will be initiated on a number of new tasks including the general modelling of constraints such as pin limitations over several levels of physical design; the design of network chips which are optimum over a wide range of network sizes and chip pin constraints; the study of centralized versus decentralized design techniques; the effect of pipelining on various aspects of network design; the further study of synchronization problems which arise in VLSI network design; the preliminary design of a VLSI network chip or chip set.

It is our hope that the research momentum of the first year will continue and that significant progress will be made during year two of the contract.

REFERENCES

1. Keyes, R.W.
"Physical Limits in Digital Electronics"
Proc. of the IEEE 63, 5 (May 1975) 740-767.
2. Block, E. and Galage, D.J.
"Component Progress: Its Effect on High Speed Computer Architecture and Machine Organization"
Proc. Symp. on High Speed Computer and Algorithm Organization, Academic Press N.Y. (1977) 13-39.
3. Sameh, A.H.
"Numerical Parallel Algorithms - A Survey"
Proc. Symp on High Speed Computer and Algorithm Organization, Academic Press N.Y. (1977) 207-228.
4. Miranker, W.
"A Survey of Parallelism in Numerical Analysis"
SIAM Review, Vol. 13 (1971) 524-547.
5. Heller, D.
"A Survey of Parallel Algorithms in Numerical Linear Algebra"
Carnegie-Mellon Univ., Computer Science Dept. Tech. Rept. (Feb. 1976).
6. Franklin, M.
"Parallel Solution of Ordinary Differential Equations"
IEEE Trans. on Computers, C-27, 5 (May 1978).
7. Franklin, M. and Soong, N.
"One Dimensional Optimization on Multiprocessor Systems"
IEEE Trans. on Computers, C-30, 2(Feb. 1981).
8. Katz, I.; Franklin, M., and Mattione, R.
"A Partitioned Parallel Runge-Kutta Method for Weakly Coupled Ordinary Differential Equations"
Inter. Journal for Numerical Methods in Engineering, 12, 2(1978).
9. Enslow, P.H., Jr., "Multiprocessor Organization - a Survey", ACM Computing Surveys, 9,1 (March 1977) 103-129.
10. Keller, R.M., et. al., "A Loosely-Coupled Applicative Multi-Processing System", Proc. AFIPS Nat. Comp. Conf. (1979) 861-869.
11. Sejnowski, M.C., et. al., "An Overview of the Texas Reconfigurable Computer", Proc. AFIPS Nat. Comp. Conf. (1980).
12. Sullivan, H. and Baskow, T.R., "A Large Scale, Homogeneous, Fully Distributed Parallel Machine I", Proc. 4th Ann. Symp. on Computer Architecture (March 1977).
13. Widdoes, L.C., Jr., "The S-1 Project: Developing High-Performance Digital Computers," Proc. of the Spring Computer Conf. 80 (Feb. 1980) 282-291.

14. Wulf, W.A. and Bell, C.G.
"C.mmp-A Multi-Mini-Mini-Processor"
Proc. AFIPS Fall Joint Comp. Conf., Vol. 41 (Fall 1972) 765-777.
15. Dennis, J.B. and Miunas, D.P.
"A Preliminary Architecture for a Basic Data-Flow Processor"
Proc. 2nd Ann. Symp. on Comp. Arch. (Dec. 1974) 126-132.
16. Siegel, J.J., Mcmillen, R.J. and Mueller, P.T., Jr.
"A Survey of Interconnection Methods for Reconfigurable Parallel Processing Systems"
Proc. 1979 Nat. Comp. Conf. (June 1979) 529-542.
17. Thurber, J.J.
"Interconnection Networks - A Survey and Assessment"
Nat. Comp. Conf. (May 1974) 909-919.
18. Franklin, M.S., Kahn, S.A. and Stucki, M.J.
"Design Issues in the Development of a Modular Multiprocessor Communications Network"
Proc. of the Annual Symposium on Computer Architecture (April 1979) 182-187.
19. Goke, L.R. and Lipoviski, G.J.
"Banyan Networks for Partitioning Multiprocessor Systems"
The First Ann. Symp. on Comp. Arch., University of Florida, Gainesville, Florida (1973) 21-28.
20. Lawrie, D.H.
"Access and Alignment of Data in an Array Processor"
IEEE Trans. on Comp., Vol. C-24, No. 12 (Dec. 1975) 1145-1155.
21. Franklin, M.A.
"VLSI Performance Comparison of Banyan and Crossbar Communications Networks"
IEEE Trans. on Computers, C-30, 4 (April 1981) (Modified Version in Proc. 1980 Workshop on Interconnection Networks for Parallel and Distributed Processing, Purdue Univ., April 1980).
22. Padmanabhan, K.
"VLSI Interconnection Network Modeling and Performance Evaluation"
M.S. Thesis, Dept. of Elec. Engr., Washington University, St. Louis (Jan. 1982).
23. Thompson, C.D.
"Area-Time Complexity for VLSI"
Proc. 11th Ann. ACM Symp. on The Theory of Computing (April 1979).

Interconnection Networks

A Model and Survey

Krishnan Padmanabhan

1.0 Introduction

Due to LSI and VLSI technologies there now exists an abundance of low cost digital logic and microprocessor components. These components may have substantial logical complexity, and are rapidly becoming the basic building blocks in the design of more powerful computers and communications systems. In the computer area, high computational power appears achievable through the development of multiple processor systems (1,2,3,4). Central to such multiple processor systems is the connection network over which the processors exchange information. A poorly designed network can rapidly become a performance bottleneck as the number of processors and traffic in the system increases. Very high bandwidth networks, on the other hand, can be extremely costly, and may indeed have a cost which grows faster than the growth of processors in the systems. Under these conditions, the cost of the network may dominate the cost of the overall multiprocessor system.

This surging interest in multiprocessor systems has led to a renewed interest in the design of interconnection networks. Most of the earlier work in networks, particularly nonblocking networks has been carried out in the field of telephone switching. This is not surprising for telephone exchanges had been the most complex digital system in existence for quite some time. Fortunately however, most of the theoretical work done against this background can be transferred to a discussion of the issues involved in networks of computers. The reason for this, in a major part, is the fact that the complexity of a switching network (a term that itself needs

elaboration) is dependent mainly on the combinatorial and topological properties of the network, rather than on factors related to the application. However, the latter considerations are definitely important while choosing a specific network.

Two principal modes of network communications can be identified. The first is referred to as the circuit switched mode, and has been the dominant one used in telephone switching. In this mode connections are provided between a requesting set of inputs and the desired outputs by opening and closing switches or crosspoints. A path that is thus set up between an input and an output is "held" until the transaction is completed. Being held, no other transactions can use that path, and such transactions are thus blocked if they require that path, or any part of it. The second communications mode is called the packet switched mode. In the packet switched mode of communication the input or the sender puts the information to be sent to the receiver in a packet which contains the address of the destination. The interconnection network routes this packet to its destination in a manner that depends on the current network status. The packet itself moves through the network "holding" only a part of the path it traverses at any point in time. The efficiency of this scheme derives from two sources. First by holding only part of the path during transmission, a smaller part of the network is tied up by the ongoing transaction. Other transactions can use that part of the packet's path which has been released as the packet moves through the network. Second, there is generally no unique route between input and output. A requesting unit need not therefore be tied down to the availability of a single complete path to initiate communication. These are clearly advantages of the packet mode over the

circuit switched mode of communication. However there are a number of disadvantages, primarily with respect to higher logic and hardware costs. Furthermore there are a number of specific computer network applications where circuit switching is clearly faster than packet mode communications. It is not the objective of this paper to compare the performance of these different approaches and the discussion to follow will be restricted to networks operating in the circuit switched mode.

The next section presents a general approach to modeling such networks in terms of graphs and states. This is followed by a brief discussion of certain issues relating to network complexity. A network classification scheme is then given, and some results relating to the blocking characteristics of networks presented. A more through discussion of selected networks and their complexity is considered next and the entire paper is then summarized and concluded. The paper itself is a condensation of material found in reference 20.

2.0 Interconnection Network Models - Graphs and States

To be able to prove certain facts about networks, a formalized notion of a network is needed. A natural way to define a network is as a directed graph with nodes representing either an input or output module or a switching element and each directed edge standing for a physical link in the network. Choosing a digraph means we are restricting ourselves to unidirectional communication over a set of paths (which are collectively represented by an edge). This really does not restrict the model because a link between two elements that permits bidirectional transmission can be represented by two oppositely directed edges. It will be shown that this model is both sound and complete in the sense that there exists a one to one correspondence between the set of all finite digraphs and the set of interconnection networks.

A switching network N can be represented as a digraph $G_N(V_N, E_N)$ where V_N is the vertex set and E_N is the edge set.

The vertex set V_N is defined as:

$$V_N = I \cup SE \cup O$$

I is a nonempty set of vertices with each $i \in I$ standing for an input module in N . O is a nonempty set of vertices with each $o \in O$ representing an output module in N . SE is a switching element (module) set with each $se \in SE$ standing for an individual switching element in the network. The switching element is defined as having n inputs and m outputs ($n, m \geq 1$) with the capability of connecting any input to any output. Note that if SE is empty, it's not necessary for each of I and O to

be nonempty, but just that $I \cup O$ be $\neq \phi$. This is the classic case of the interconnection where the processors communicate with each other directly through links without any switching elements. In this case it may not be possible to identify input and output modules (e.g., star connection or ring).

Given a set of vertices V_N , with elements being designated as v_i , it is now necessary to specify how these elements are connected together. Let E_N be the set of directed edges $\{e_i\}$. The connection between E_N and V_N is defined in terms of an incidence function ψ . If an output of the module corresponding to v_i (i, se, or o) is connected to an input of the module corresponding to v_j then:

$$\psi(e) = v_i v_j \quad \left\{ \begin{array}{l} e \in E_N \\ v_i, v_j \in V_N \end{array} \right\}$$

Some examples of standard interconnection structures and their digraph equivalents are presented in Figures 1, 2 and 3.

Network graphs defined in this manner have a number of fairly evident properties. These are presented below without proof.

1. If each switching element is of size $n \times m$ (i.e., n inputs and m outputs), then the in-degree of each vertex of the type se is n and the out-degree of any vertex of the same type is m .
2. In general, the in-degree of a vertex of type i is zero and the out-degree of a vertex of type o is also zero.
3. $\forall i, \forall o$, out-degree of i = in-degree of o = 1.
4. Any network graph with $I \cap O = \phi$ is cycle free. This can be proved using properties 2 and 3. Also the number of cycles in the graph is equal to $|I \cap O|$.

5. This representation for a communication network is both sound and complete: that is, for every network there corresponds a unique digraph and a digraph with properties 2,3, and 4 always corresponds to a 'valid' communications network.

6. In N , input k can 'access' output l if and only if there exists a (directed) path between i_k and o_l in G_N .

The subgraph S of G_N with $V(S) = SE$ will be called the switch graph. Basically this is the network graph with the input and output nodes (and their associated edges) removed. More often than not, it's the switch graph that we are interested in because it is the structure that captures the intrinsic topological properties of the network. Note that G_{SE} may indeed be the null graph, if the interconnection does not make use of any switches. However, for our purposes, we will be interested in networks for which $SE \neq \phi$.

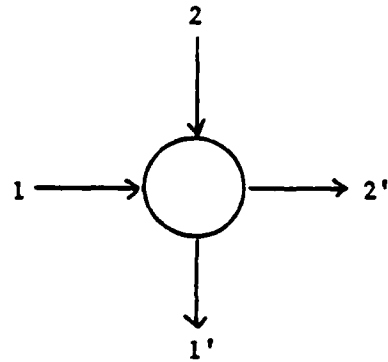
The model as presented above is concerned only with the 'static' properties of the network, i.e., with the topological aspects external to the switch. A connection is set up between an input unit and an output unit only when some (generally more than 0) switching elements are 'set' in particular positions. This is a dynamic property of the network since it depends on what inputs are currently active and to what outputs they are connected. This information can be specified, by associating with each switch node a table that gives the connections established within that node. The table need not contain entries corresponding to inputs and outputs that are not used (not connected).

Where each switching element has two inputs and two outputs, it is possible to condense the information in the table to just one bit. Call the two

inputs to the switch 1 and 2 and the two outputs 1' and 2'. If 1 is connected to 1' then the only connection for 2 is 2'. Similarly, if 1 - 2' is established, then 2 can be connected only to 1'. Thus the switching element can only be in one of two configurations and hence the state can be represented by a single bit. A 0 will represent a 'straight' position (1-1',2-2') while a 1 will represent a 'cross' position (1-2',2-1'). This fact can also be seen in another way; basically the entries in the table specify the permutation of the inputs to the switching element. If the switching element has n inputs (and n outputs) there are a total of $n!$ possible entries for the table. For a 2 input - 2 output switch then, there are just 2 permutations, one of which we represent by a 0 and the other by a 1.

We are now in a position to define the state of a network. The state is given by the ordered pair $\langle G_N, \{T_{se}\} \rangle$. G_N is the network graph defined earlier. $\{T_{se}\}$ is a set of tables that specify the permutation realized by each switching element. The first element of the state therefore gives the static topological characteristics of the network. The second element is what basically captures the dynamic properties of the network. In fact it is possible for us to define the state as just $\{T_{se}\}$ since G_N is fixed. G_N is however included in the specification of the state for sake of completeness. As an example consider the mesh network of Figure 1. If the input/output paths to be established are 1 - 1', 2 - 3', 3 - 2', the state of the network is given by G_N and $\{T_{se}\}$ where G_N is specified in Figure 1 and $\{T_{se}\}$ is given below along with the switch input/output labelling.

se	j	1	2	3
1		1	2	3
1		0	-	-
2		1	1	0
3		1	0	-



A dash indicates a don't care condition.

3.0 Issues of Complexity

The complexity of a system made up of interconnecting a large number of (atomic) elements has two aspects to it-- the number of the atomic components and the way they are interconnected together. Pippenger (5) gives a good account of complexity theory applied to telephone switching. Although the complexity of such a system is very much greater than the sum of its parts (essentially due to the type of interconnection), researchers have, up until very recently, been concerned just with this sum. The reason for this has been two fold: (1) It is much easier to analyze the complexity in terms of obtaining both asymptotic expressions and lower bounds on the number of components required than it is to quantify the complexity of interconnections in the general case; and (2) Till recently the component count did give a fairly realistic estimate of overall system complexity. As Pippenger points out, "...systems that have fewer components are obtained by interconnecting the components in a more intricate way. In complexity theory, this is considered an advantageous transformation." Note that when interconnection networks are entirely contained on a VLSI chip simple component count measures of complexity are inadequate. In general this is because the topology of the network can have a pronounced effect on the area required to lay out the network. This has led to the development of complexity measures which include both space (area) and time where space here includes space taken up both by the active components and the signal lines connecting the components (6,7). This aspect of the complexity of networks is not considered in this survey. Later, in section 6 many of the classical results pertaining to switch count complexity measures are presented.

4.0 Network Classification

There are several different characteristics and performance measures that can be used to group different networks in equivalence classes. Malek and Myre (7) discuss several of them. Some of the more common ones are switch count, delay (in terms of number of switches in a data path) and set of permutations (or at least the cardinality of that set) that can be achieved by a network.

Closely related to the permutations that can be realized by a network is the concept of blocking. Using this characteristic various network schemes are now classified.

Consider a (network) graph G_N . An input/output pair in N can be connected if a path exists in G_N from the input node to the output node. However, when the network is in a particular state (specified by $\{T_{se}\}$), it may not be possible to realize this path in the network if it is not permitted by the T_{se1} of some switch node. Consider the example in Figure 2. Assume $T_{se11} = 0$. (i.e., the straight through connection). This may be because we might currently have connected processor 0 to processor 2. If we now want to realize a path between processors 1 and 0 it will require T_{se11} to be 1 which is incompatible with the current state of the network. This leads to what is known as blocking-- the second request for connection has been blocked by the network. In general, blocking will occur if in G , two $i - o$ paths have at least one edge in common.

It is easy to see that blocking can never occur in the networks discussed in examples 1 and 3. Such networks are called nonblocking networks. In networks where there is more than one path from an input to an output, it might be possible to avoid blocking by choosing one or another

of the paths. This leads to different kinds of blocking exhibited by networks. These are discussed briefly now; more elaborate results will be presented a little later.

4.1 Network Classification Based on Blocking

- (a) Strict sense nonblocking networks: A network is nonblocking in the strict sense, if irrespective of what state it is in, an 'idle' input can be connected to an 'idle' output. This means, in G_N , given any set of paths between a subset of inputs and a subset of outputs, and an input i_1 and an output o_1 , (i_1 and o_1 are not members of the subsets), it is always possible to find a path between i_1 and o_1 that is edge disjoint with all the other paths. The networks in examples 1 and 3 are strict sense nonblocking networks. Example 1 is the extreme case where no path between an arbitrary input i_1 , and an output o_1 , has any edge in common with any path between another pair $i_2 - o_2$.
- (b) Wide sense nonblocking networks: In this type of networks states do exist where it will not be possible to establish a path between an unused input i and an unused output o , but such states can be avoided if a certain 'rule' for routing the calls is followed. As a necessity, such a property implies the existence of multiple paths between an $i - o$ pair. By following such a specified rule then, the networks would never be in a state where it is unable to service a request for a connection. Examples for this case and

also conditions for a network to exhibit this property will be derived in a later section.

- (a) Rearrangeably nonblocking networks: In both the cases above, we could route an input request without disturbing the paths for the transactions currently in progress. There are networks where in a particular state we may be able to service a request only if we 'rearrange' the existing configurations of the network, i.e., by modifying $\{T_{se}\}$. Such a network is nonblocking in that we would still be able to route any request, but it might incur a high cost.
- (d) Blocking networks: A network, in a sense, can be considered to permute the set of inputs and the permutation realized is given by the input/output connection. Nonblocking networks then can realize any arbitrary permutation of the inputs. A blocking network is one that is not nonblocking in any of the senses mentioned above. In other words, there do exist states of the network in which it will not be possible to satisfy a new request in any way. In terms of G_N , a necessary and sufficient condition for this property is the existence of two input/output pairs $i_1 - o_1$ and $i_2 - o_2$ such that there is no path between i_1 and o_1 that is edge disjoint from at least one path between i_2 and o_2 .

5.0 Nonblocking Networks: Some Theoretical Results

One of the earliest interconnection networks ever to be studied and used is the crosspoint switch. While its use now in large systems is limited to telephone switching, it provides a standard against which to compare other interconnection schemes. An $n \times m$ crosspoint switch is, as shown in Figure 4, an array of nm contacts, called crosspoints, with n inputs and m outputs. It has the ability to connect any input to any output in any state the switch is in. Each crosspoint is used to connect a unique input to a unique output. It can be considered as a single pole single throw switch between the input and output to which it is connected. Traditionally electromagnetic relays did the job in telephone switching. Pass transistors, less than a millionth of their size, can now handle the job.

The crosspoint switch quite obviously is a strict sense nonblocking network. An $N \times N$ square network that is strictly nonblocking can thus be constructed using N^2 crosspoints. (Note that we have implicitly assumed the number of crosspoints in the network to be our measure of complexity). Hence N^2 crosspoints can be taken as an upper bound on the complexity of a strictly nonblocking $N \times N$ network.

The next logical question concerns obtaining the lower bound on the complexity of a strictly nonblocking $N \times N$ network. A theoretical lower bound to this value was given by Shannon in 1950 (8). The derivation basically relates the minimum number of states the network should be able to assume to the number of inputs/outputs. As mentioned in section 2 with regard to T_{SE} , an $N \times N$ network has to be able to realize $N!$ different permutations. A single crosspoint can be in one of two states-- open or

closed (note that we are using the term 'state' here a little loosely!) and hence the overall network, if it has γ crosspoints, can exist in no more than 2^γ distinct states. Therefore:

$$2^\gamma \geq N! \quad \text{or} \quad \gamma \geq \log_2 N!$$

Applying Stirling's approximation to $N!$ we find that the lower bound is on the order of $N \log_2 N$. Hence any network which is strictly nonblocking with N inputs and N outputs must have at least on the order of $N \log_2 N$ crosspoints.

Given this perspective we now consider the actual complexities of various networks. Here we have different ways of proving the existence of a network with a certain complexity. The most important (in terms of practical application) is the constructive technique where we actually produce the particular network. We can thus physically build the network. The Clos construction that we discuss a little later comes under this category. It is interesting to see what other methods there can be of establishing facts like this. In 1973, L.A. Bassalygo and M.S. Pinsker of the Institute for Problems of Information Transmission in Moscow (9) proved that a strict sense nonblocking network with N inputs and N outputs exists that uses just $O(N \log N)$ crosspoints. The proof, a very brief outline of which follows, does not say how such a network can be built, however. A sparse crosspoint switch is obtained from a regular crosspoint switch by removing many crosspoints, and retaining the following property: every group of one third of the inputs can be connected to more than two thirds of the outputs. Bassalygo and Pinsker show the existence of $N \times N$ sparse crosspoint switches that make use of just $12N$ crosspoints and have the

above property. They use a combinatorial argument that involves considering the set of all possible arrangements of the $12N$ crosspoints. Most of these arrangements do satisfy the connection property above and therefore should result in valid structures. They then go on to show that by interconnecting sparse crosspoint switches, it is possible to construct strictly nonblocking networks that use $O(N \log N)$ crosspoints.

A sparse crosspoint switch with 6 inputs and 6 outputs is shown in Figure 5. Here we need less than $12N$ crosspoints because $N < 12$. For $N > 12$, every input has crosspoints between it and 12 of the N outputs (resulting in a total of $12N$ crosspoints).

Where does the catch in the above process lie? The problem is in testing a configuration. Out of the N inputs, there are $\binom{N}{N/3}$ ways of choosing a third of them and an equal number of ways of choosing two thirds of the outputs. That leaves us with $[\binom{N}{N/3}]^2$ different choices. This number is astronomically large even for reasonably large N . However, the theoretical proof of existence is quite important in itself. Pippenger (10) has refined the bound on the number of crosspoints to $90N \log_3 N$.

It was Charles Clos, in 1953 (11) who first came up with the design of a strict sense nonblocking network that uses fewer than N^2 crosspoints (N inputs/ N outputs). Basically his design involved interconnecting columns (stages) of rectangular crosspoint switches and he showed that given any $\epsilon > 0$, by using a sufficient number of stages, one can design strict sense nonblocking networks in which the number of crosspoints is $O(N^{1+\epsilon})$. We will now briefly consider Clos' analysis.

The basic approach in the Clos design is to apply the divide-and-conquer paradigm to the overall network required. A large network is built out of a number of subnetworks. Note that in the crosspoint switch, every path involved the switching of just one crosspoint. If we considered the number of crosspoints in the path to be a measure of delay, the crosspoint switch provides us with the fastest switching mechanism-- constant delay independent of network size. The saving in crosspoint count in the Clos design comes from increasing this delay by a fixed parameter. Inputs, rather than being switched by a single crosspoint are now switched by a series of subnetworks. Note that these subnetworks can be crosspoint switches, if they are small enough.

A general three-stage Clos network is shown in Figure 6. The three stages can be considered to be input, output, and intermediate switching. Any input-output path must be successively switched by a crosspoint in the input, intermediate, and output stages respectively leading to a delay of three crosspoints. (We're assuming that each subnetwork is implemented as a crosspoint switch-- this need not be the case. We can also recursively apply the divide-and-conquer paradigm to the intermediate stages).

The next question is how to get the parameters n, m and r so that the number of crosspoints is minimized. The only fact we do know is that the product of n and m equals N . The number of switching networks, r , required in the intermediate stage must be sufficient to avoid blocking under the worst set of conditions. What is the worst case condition? Assume that we have $(n-1)$ requests currently being serviced at an input switch and a new request comes in. The $(n-1)$ existing requests go off via a crosspoint each to $(n-1)$ switches in the intermediate stage. Consider the

destination of the new request and the output switch which contains that destination. The worst case here is when all the $(n-1)$ outputs at this switch other than the desired destination are busy and are connected to $(n-1)$ switches in the intermediate stage different from the $(n-1)$ switches previously considered. At this stage one more switch is needed in the middle column to service the new request. Hence given $(2n-1)$ switches in the intermediate switch, it is possible to service a request whatever state the network is in. Thus, for $m \geq 2n-1$, the Clos network is nonblocking in the strict sense.

The number of crosspoints required by the three stage Clos network (which, incidentally, is denoted by $C(3)$) is given by:

$$C(3) = 2rnm + nr^2 \quad [1]$$

Noting that $nr = N$ and arbitrarily choosing $n = \sqrt{N}$ one obtains:

$$C(3) = 2N(2N^{1/2} - 1) + N(2N^{1/2} - 1) = 6N^{3/2} - 3N$$

Note that this is a significant reduction in cost. For $N \geq 36$, this value is less than N^2 . The value of n which minimizes [1] can be found by rewriting [1] as:

$$C(3) = (2n-1)(2N + r^2) = (2n-1)\left(2N + \frac{N^2}{n}\right) \quad [2]$$

Differentiating [2] and setting zero gives

$$2n^3 - nN + N = 0 \quad [3]$$

For large N (and, large n), [3] can be approximated by

$$2n^2 - N = 0$$

or

$$n = (N/2)^{1/2}$$

Substituting this value of n in [2] give us

$$C(3) = 4N(2^{1/2}N^{1/2} - 1)$$

The next step is to apply recursion to the intermediate stages of networks too and synthesize them using smaller size networks. Clos shows that the casts of 5,7, and 9 stage networks are given by:

$$C(5) = 16N^{4/3} - 14N + 3N^{2/3}$$

$$C(7) = 36N^{5/4} - 46N + 20N^{3/4} - 3N^{1/2}$$

$$C(9) = 76N^{6/5} - 130N + 86N^{4/5} - 26N^{3/5} + 3N^{2/5}$$

(The number of stages considered is always odd because that's the way the networks grows: Each time the intermediate stage is decomposed into three stages.) Note that by going to a sufficiently large number of stages, we can make the order of magnitude growth as near to N as we wish. But however small ϵ is, $N^{1+\epsilon}$ still grows faster than $N \log N$ (even though it stays less than $N \log N$ until N becomes very large).

David Cantor (12) analyzed the Clos network as it's designed to the limits of recursion and found it takes on the order of $N e^{2(\log N \cdot \log 2)^{1/2}}$ crosspoints which simplifies to $O(N(\log N)^{2.269...})$. Cantor himself proposed an interconnection scheme involving subnetworks (crosspoint switches) arranged in more than three stages that grows on the order of $N(\log N)^2$. The design is much more complicated than the Clos network and will not be presented here.

Note that the advantage of the reduced exponent in the complexity figures cited above does not appear until N gets very large. The Clos network with three or five stages offers a fair compromise between the crosspoint count and the issues in complexity that are not considered in this figure (like the interconnection of the components). This concludes the discussion of strict sense nonblocking networks.

Practically useful wide-sense nonblocking networks have not yet been found. One example has been cited by Benes (13). Consider the three stage Clos network (Figure 6) with $r = 2$. If the rule that an empty middle switch is not to be used unless there is no partially filled middle switch that will route the request is used, then it can be shown that no call is blocked by the switch for $m > \lceil 3n/2 \rceil$. Thus the three stage Clos network with $r = 2$ is wide-sense nonblocking for $m > \lceil 3n/2 \rceil$.

That the Clos network is rearrangeable for $m \geq n$ was proved by A.M. Duguid in 1959 (14). The proof is based on a combinatorial argument that goes on to show that any permutation of the N inputs can be realized by a configuration of the network that makes use of just n switches in the middle row. It has been proved by M.C. Paull in 1962, that in the Clos network with $m = n = r$, at most $(n-1)$ existing input-output paths need be rerouted in order to connect an idle pair of terminals.

6.0 Blocking Networks

Blocking interconnection networks by definition have the property that there do exist $i - o$ pairs in their graphs which are not connectable when the network is in some particular state. This can arise from two cases.

(i) It is possible that there is no path at all between i_j and o_k in G_N . In this case it is never possible to connect input j and output k no matter what state the network is in. We call such networks as strong sense blocking networks. One such network configuration is the 'star' whose graph is shown in Figure 7. Though not a very practical scheme in large systems, it still is a perfectly valid example. Note that $SE = \phi$ in this case, so the nodes are labelled as p 's. In this case processors i and j are connectable if and only if there exists a link l of the form $p_i \xrightarrow{l} p_j$ or $p_j \xrightarrow{l} p_i$ in G_N . (In particular a path of the form $p_i \xrightarrow{l_1} p_j \xrightarrow{l_2} p_k$, even if present, does not connect i and k in the network.) Thus we see that none of the 'slave' processors (p_{s_i}) can communicate with each other (directly) while the 'master' (p_m) can interact with all the slaves. Some other examples of this class of networks are the mesh type interconnection (as in the Iliac IV) (2), the chordal ring network (15) and the tree configuration (the X-tree (16), for example).

Define the underlying graph of a digraph G to be the graph obtained from G by deleting the directions from the edges, i.e., by replacing directed edges by undirected edges. It then is easy to see that if $SE = \phi$ in G_N , then N will not be strong sense blocking if and only if the underlying graph of G_N is the complete graph K_n (where n is the number of nodes = number of processors). This is a reasonably strong condition to satisfy,

especially for large n ; other than the number of edges itself growing on the $O(n^2)$, each node will have to grow in size as $O(n)$ -- an almost complete antithesis of the design philosophy for large scale integration. However, the strong sense blocking characteristic of a network does not, quite often, pose great problems. This is especially true where a multiprocessor system is to be used (more often) in specific applications i.e., the alignments required of the network (by alignments, we mean the set of $p_i - p_j$ paths that have to exist simultaneously in the system) are known a priori and the links have been set up with this requirement in mind.

(ii) In this case, every $i - o$ pair is connected in G_N , but a specific path may not be allowed by the $\{T_{s_i}\}$ of the state of the network. This implies, by necessity, the existence of a nonempty SE. This more common form of blocking is exhibited by most proposed interconnection schemes and is what we will be concerned with in the rest of this section. We call this type of blocking as weak sense blocking. More formally, a network is weak sense blocking if and only if there exist states of the network in which it is not possible to connect an $i - o$ pair and for each such $i - o$ pair, there exists at least one other state in which it can be connected.

How does weak sense blocking differ from the rearrangeably nonblocking networks discussed in section 5? In the case of a rearrangeably nonblocking network, for every state of the network in which it is not possible to connect an $i - o$ pair, there exists at least one other state in which (a) all $(i - o)$ connections in the first state are maintained in the second (though not necessarily using the same paths) and (b) the new $i - o$ pair is also connected. Condition (a) is not necessary to be satisfied by a weak sense

blocking network. In fact, it is not satisfied by a weak sense blocking network, as otherwise the network would be rearrangeably nonblocking.

Note that in classifying network schemes based on their blocking characteristic, we include a network in the highest class of nonblocking networks possible. (We consider a weak sense blocking network to be more 'nonblocking' than a strong sense blocking network.) What we have established then is a hierarchy of equivalence classes based on the blocking characteristic. Any network in a particular class has the properties of all the classes below it in the hierarchy.

There is a plethora of networks of this type that have been proposed. All of these use multiple stages of switching elements and hence differ basically only in the type of connection between the stages. Three of these are shown in Figures 8,9, and 10-- the Omega network, the Indirect Binary 3-cube network, and the banyan network. Note that while the stages themselves are 'active', the connection between two stages is, in a sense, 'passive'. Most of these connections are variations of a basic structure known as a shuffle. Harold Stone, in (17) discusses the applications of the 'perfect' shuffle in parallel processing.

Define an $r \star m$ shuffle (19), denoted by $S_{r \star m}$, where r and m are some positive integers, to be the following permutation of rm indices $\langle 0, 1, \dots, (rm - 1) \rangle$:

$$S_{r \star m}(i) = (ri + \lfloor \frac{i}{m} \rfloor) \bmod (rm) \quad 0 \leq i \leq rm - 1$$

where $S_{r \star m}(i)$ is the position of i after the shuffle. An $r \star m$ shuffle can be viewed as a shuffle of rm cards in a deck in the following manner:

Divide the deck into blocks of m cards each. The order of the cards in the $r \times m$ shuffled version is given as follows: The first r cards are the first from each of the r blocks; the second r cards the second from each of the r blocks and so on.

Consider again the 3 stage Clos network shown in Figure 6. The connection between the input and intermediate stages is the $r \times m$ shuffle. That between the intermediate and output stages is the $m \times r$ shuffle (which, incidentally, is the inverse of the $r \times m$ shuffle).

The 'perfect' shuffle, as defined by Stone, is then the $2 \times \frac{N}{2}$ shuffle. Its pattern is shown in Figure 11. The permutation π is given by

$$\begin{aligned} \pi(i) &= 2i & 0 \leq i \leq N/2 - 1 \\ &= 2i + 1 - N & N/2 \leq i \leq N - 1 \end{aligned} \quad \dots 4$$

4 then tells us that if the index of an input i is represented in binary notation (of $\log_2 N$ bits), its position after the perfect shuffle is given by cyclically rotating the bits of i one position to the left. This indicates that $\log_2 N$ successive shuffles of a vector i will return to us i back.

The Omega network (Figure 8) then consists of $\log_2 N$ switching stages with $\log_2 N$ perfect shuffles connecting the stages. The Indirect binary n -cube is a Banyan network (with 2^n inputs) with an inverse shuffle appended at the end. This lets the IBNC realize the identity permutation which the Banyan network cannot.

The Indirect Binary n -cube network derives its name from the fact that it can simulate the transfers along the edges of a binary n -cube, Q_n , which is defined as follows: The vertices of Q_n are the n -tuples of 0 and 1 and there is an edge between two vertices iff their n -tuples differ in are

and only are component. A binary 4-cube is shown in Figure 12. Each stage in the IBNC network is identified with one axis of the binary n-cube. The transfers along a particular axis (if we assume a processor at each node of Q_n) are obtained by setting the switches in the corresponding stage in the 'cross' position and all other switches in the 'straight' through position. Of course, the IBNC can realize a lot more permutations than are permitted by the connections of the n-cube.

Each of the three networks considered above is 'self-routing'; that is, the input message along with the destination address routes itself through the network. This is typically done as follows. The paths from one particular input to all the outputs form a binary tree rooted at the input switch corresponding to the input. Each node is a switching element. This is basically the demultiplexer tree for the $\log_2 N$ bits which encode the address of the destination. Each switch along the path strips one bit off the address (the most significant bit), examines it and sets itself based on the bit. Typically, a 0 sets the switch in the straight position while a 1 sets it in the cross position. Note that the concept is identical to that of positional trees. (18)

We conclude this section with a brief discussion of the probability of blocking. The stochastic model assumed for the entire system (which includes the processor and memory modules) is important here in the final results that will be obtained. Patel (19) presents a model in which each processor generates random and independent requests which are uniformly distributed over all memory modules. The probability of accepting in this case (assuming a mean request generation rate per cycle of 1-- a saturated system) is shown in

Figure 13a. An alternate model, which considers the blocking characteristic introduced by the network alone (that is, leaves out the blocking inherent in input contention) has been considered by us where in we assume the input address unique output ports in each cycle. Simulation results for this model are presented in Figure 13b. Analytical results for the latter case are more difficult to obtain. The figures presented are for networks of the type Omega, IBNC or Banyan.

7.0 Summary and Conclusions

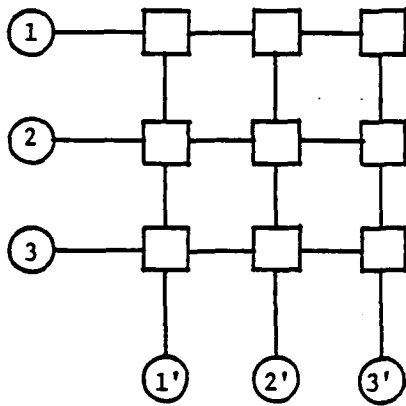
Increasing interest and design of multiprocessor systems has led to a renewed concern with the properties of interconnection networks. These networks though originally studied in the context of telephone systems are being re-examined with multiprocessor systems in mind.

This paper has presented a review and classification of some of the principal networks studied. The classification scheme is primarily based on the blocking characteristics of the network. In addition some discussion of network complexity was presented. Further research work is needed in this area. In particular the effects of changing technology (e.g., VLSI) need to be investigated in terms of how the new characteristics of these technologies relate to network performance. In addition it is not yet understood just how to relate network properties to applications requirements in any general manner. These and associated questions will be the examined for some time to come.

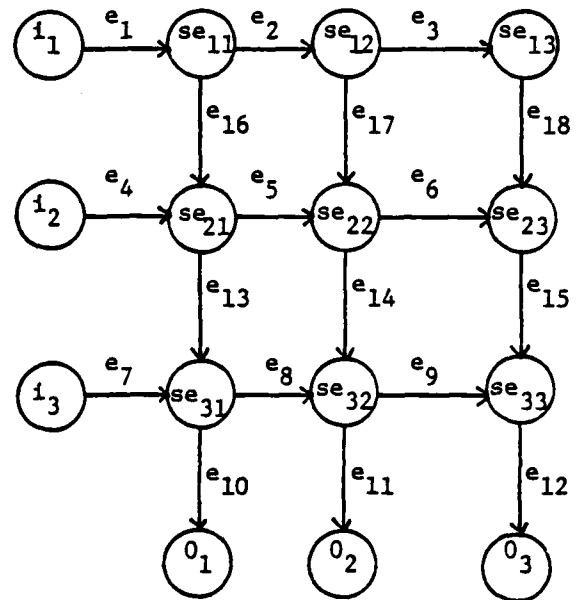
8.0 REFERENCES

- (1) Swan, R.J., et al. "Cm* - A modular, Multi-Microprocessor," Proc. National Comp. Conf., 1977, pp 637-644
- (2) Barnes, G., et al. "The Illiac IV Computer," IEEE Trans. Comput., Vol., C-17, No. 8, Aug 1968, pp 746-757
- (3) Batcher, K.E., "STARAN parallel processor system hardware," Proc. National Comp. Conf., 1974, pp 405-410
- (4) Widdoes, L.C., Jr., "The Minerva Multi-Microprocessor," 3rd Annual Symp. on Comp. Arch., Jan 1976, pp 34-39
- (5) Pippenger, N., "Complexity Theory," Scientific American, June 1978, pp 114-121
- (6) Franklin, M.A., "VLSI Performance comparison of Banyan and Crossbar Switching Networks"
Proc. of Workshop on Interconnection Networks, Purdue University, April 1980
- (7) Malek, M., and Myre, W.W., "Figures of Merit for Interconnection Networks"
Proc. of Workshop on Interconnection Networks, Purdue University, April 1980
- (8) Shannon, C., "Memory Requirements in a Telephone Exchange," Bell System Tech. J., Vol. 29, 1950, pp 343-349
- (9) Bassalygo, L.A., and Pinsker, M.S., "On the Complexity of an Optimum Non-blocking Switching Network without Reconnection"
Problemy Peredachi Informatsii Vol. 9, No. 1, pp 84-87, Jan 1973
English Translation in Problems of Information Transmission, Vol. 9 No. 1, Nov. 1976, pp 64-66
- (10) Pippenger, N., "On Rearrangeable and Nonblocking Switching Networks," Journal of Computer and System Sciences, Vol. 17, No. 2, Oct. 1978, pp 145-162
- (11) Clos, C., "A Study of Nonblocking Switching Networks," Bell System Tech. J., Vol. 32, pp 406-424 (1953)
- (12) Cantor, D., "On Nonblocking Switching Networks," Networks, Vol. 1, pp 367-377
- (13) Benes, V.E., Mathematical Theory of Connecting Networks and Telephone Traffic, Academic Press, New York, 1965
- (14) Duguid, A.M., "Structural Properties of Switching Networks," Brown Univ., Progress Rept., BTL-7, 1959

- (15) Arden, B.W., and Lee, H., "Analysis of Chordal Ring Network," Proc. of Workshop on Interconnection Networks, Purdue University, April 1980
- (16) Despian, A.M., Patterson, D.A., "X-Tree: A Tree Structured Multiprocessor Computer Arch.," 5th Ann. Symp., on Comp. Arch., 1978, pp 144-151
- (17) Stone, H.S. "Parallel Processing With the Perfect Shuffle," IEEE Trans. on Comput., Vol. C-20, No. 2, Feb. 1971, pp 153-161
- (18) Even, S., Graph Algorithms, Computer Science Press, 1969
- (19) Patel, J.H., "Processor-Memory Interconnections for Multiprocessors," 6th Ann. Symp., on Comp., Arch., Apr. 1979, pp 168-177
- (20) Padmanabhan, K. "VLSI Inconnection Network Modeling and Performing Evaluation", M.S. Thesis, Dept. of Elect. Engr., Washington University, St. Louis (Jan. 1982).



Mesh Network

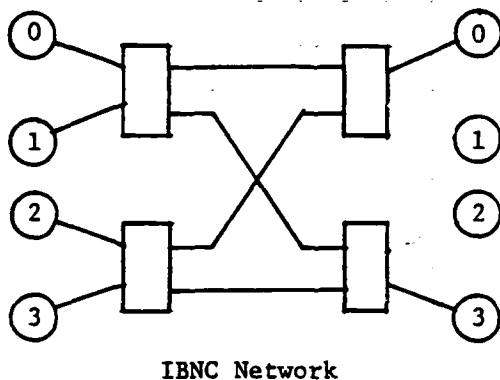


Mesh Graph G_N

Figure 1: The Mesh Connection $I = \{i_1, i_2, i_3\}$, $O = \{o_1, o_2, o_3\}$,

$SE = \{se_{11}, se_{12}, se_{13}, se_{21}, se_{22}, se_{23}, se_{31}, se_{32}, se_{33}\}$.

$E_N = \{e_1, e_2, \dots, e_{18}\}$, $\psi(e_1) = (i_1, se_{11})$, $\psi(e_2) = (se_{11}, se_{12})$ etc.



An Indirect Binary 2-cube Network

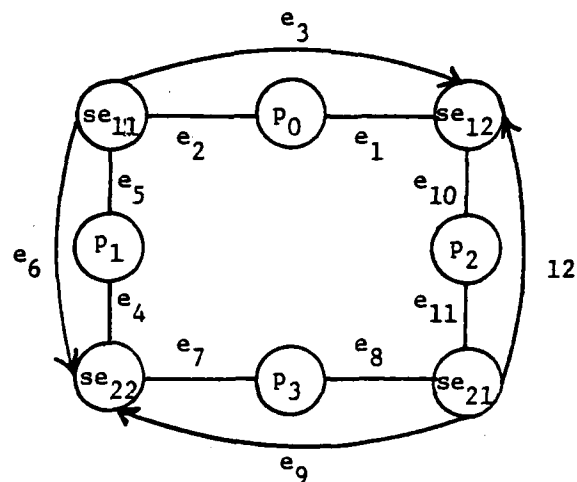


Figure 2: The Indirect Binary 2-cube network interconnecting four processing elements. In this case, $I = 0 =$
 $P = \{p_0, p_1, p_2, p_3\}$. Note that a path from any p -vertex to any other p -vertex has a length of 2. In general it is n long, where $n = \log|P|$. Also note that such a path is unique.
 $E_N = \{e_1, e_2, \dots, e_{12}\}$ $\psi(e_1) = se_{12}p_0$, $\psi(e_2) = p_0se_{11}$ etc.

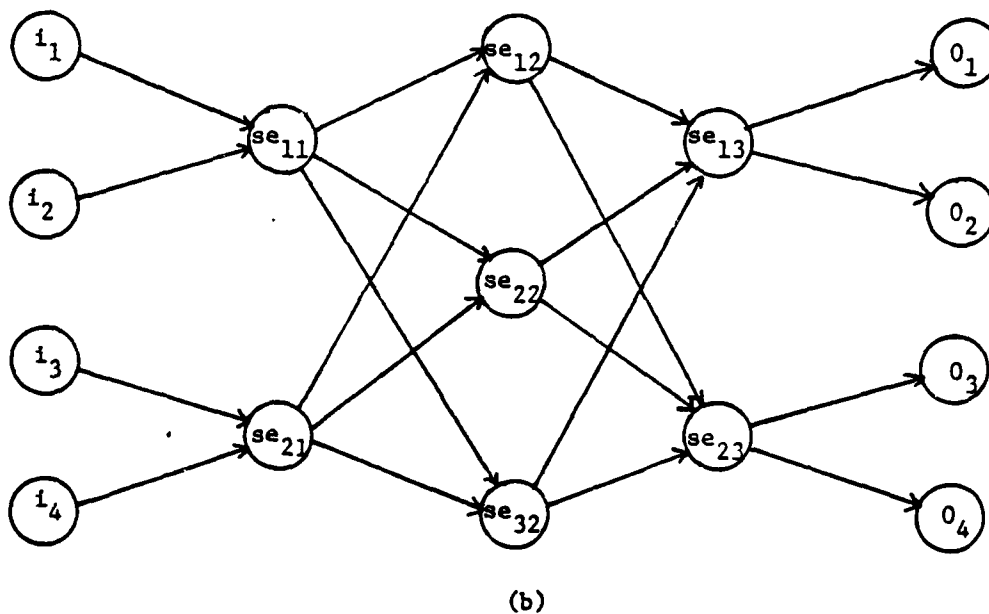
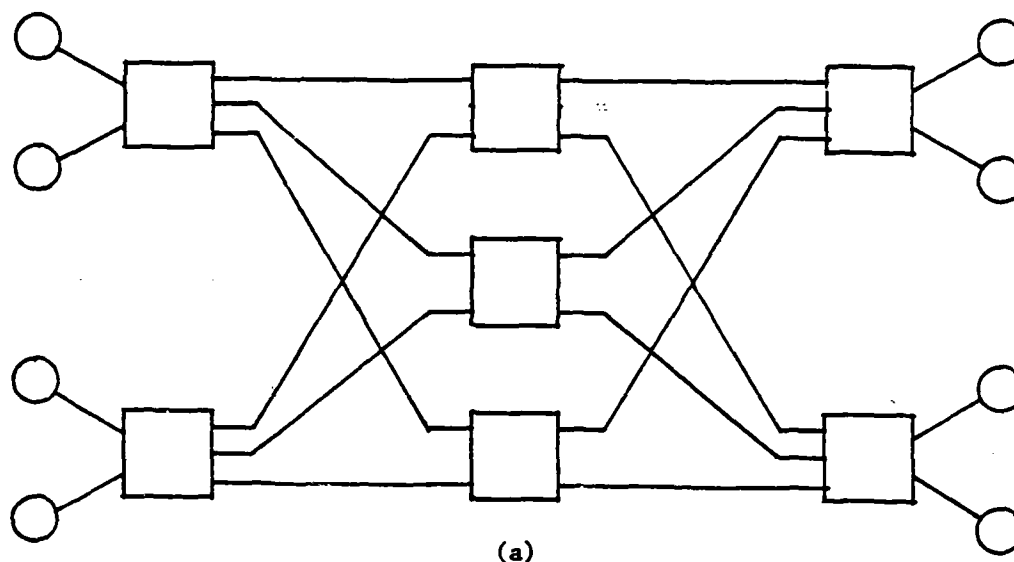


Figure 3: A 4x4 Clos Network (a) and its graph (b). It is a strict sense nonblocking network (a term that is explained in text.)

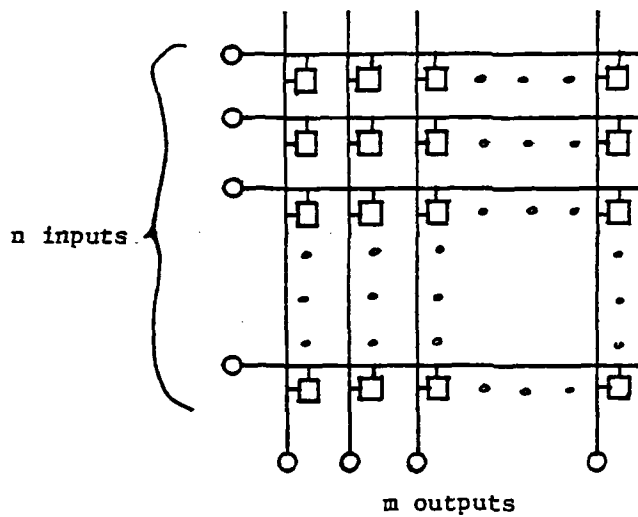


Figure 4: An $n \times m$ crosspoint switch

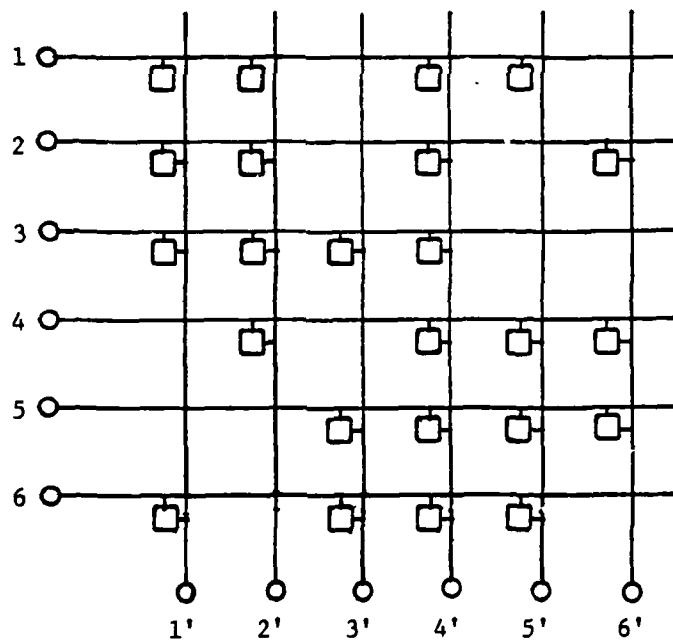


Figure 5: A 6×6 sparse crosspoint switch

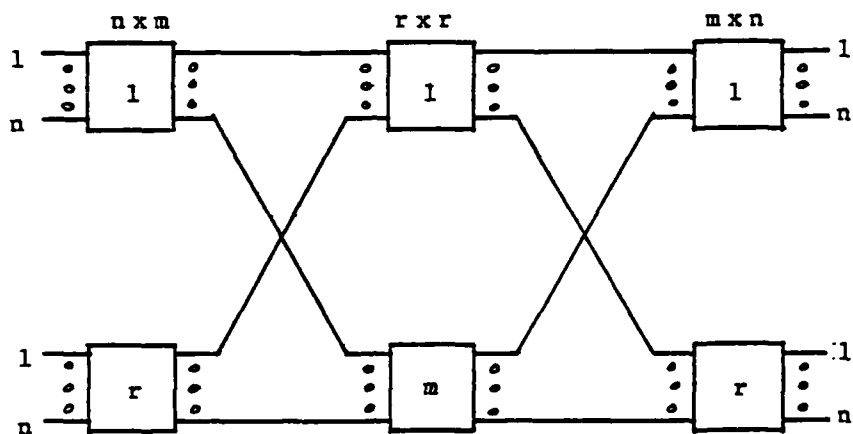


Figure 6: Three stage Clos network.
Each box is a subnetwork (crosspoint)
of the order shown.

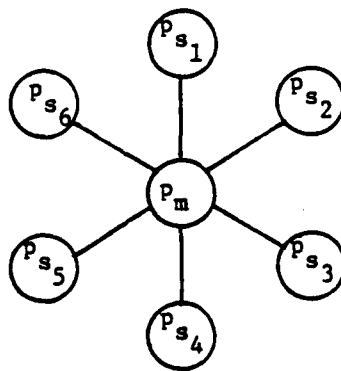


Figure 7: Network graph of a tree configuration.

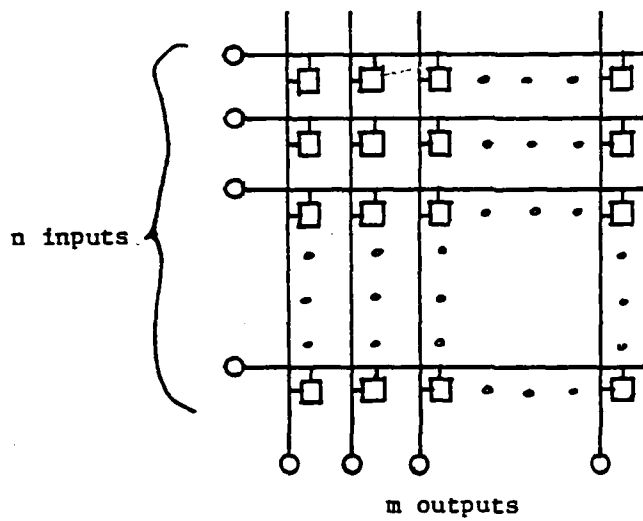


Figure 4: An $n \times m$ crosspoint switch

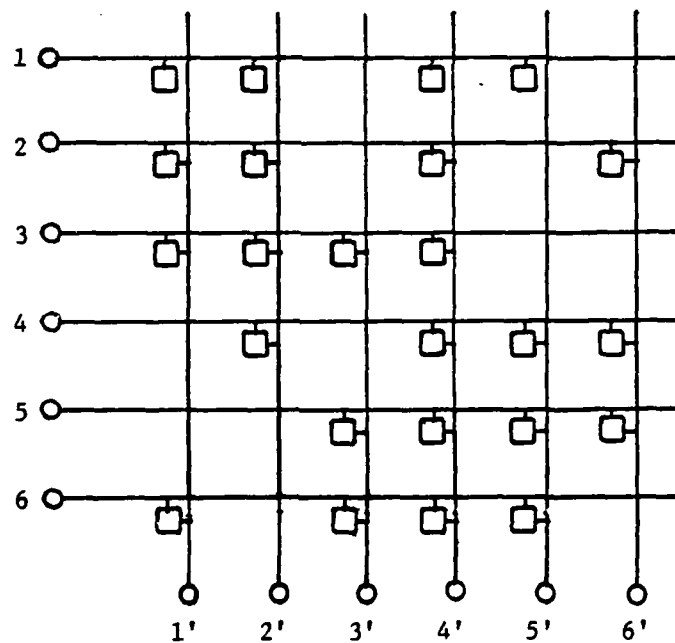


Figure 5: A 6 x 6 sparse crosspoint switch

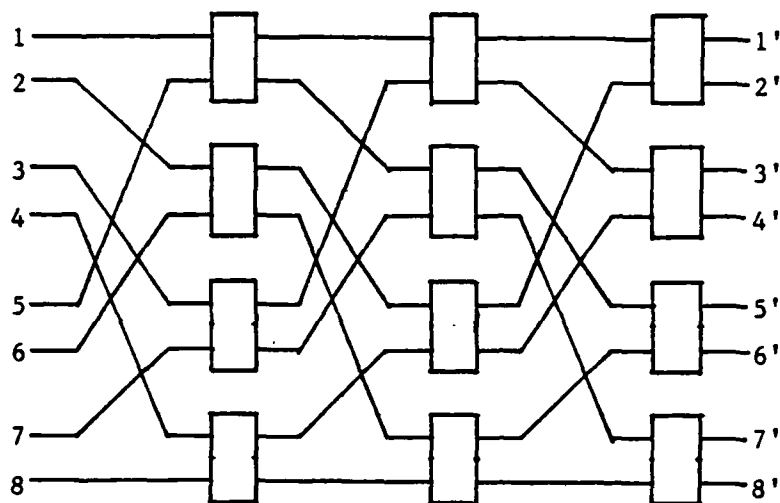


Figure 8: An 8 x 8 Omega Network

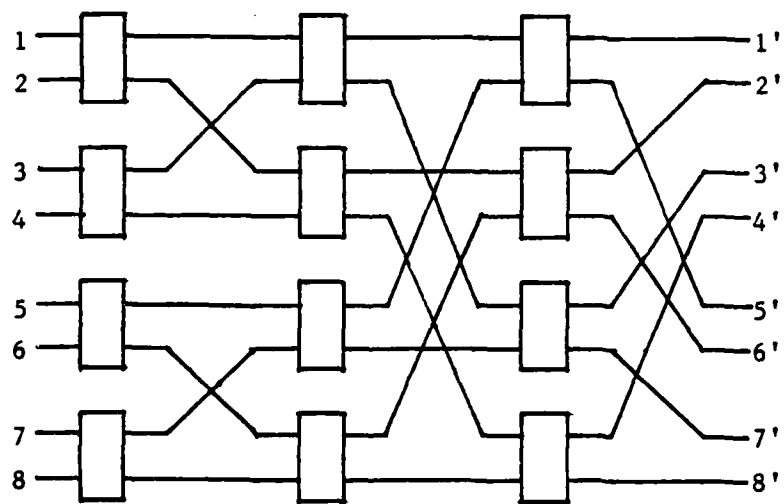


Figure 9: The indirect binary 3- cube array

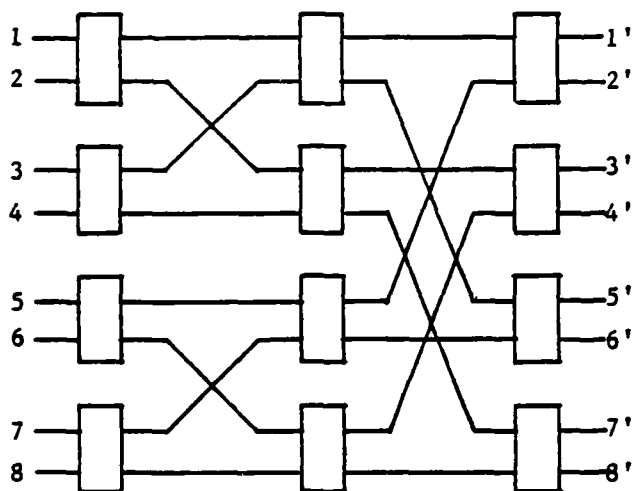


Figure 10: An 8 x 8 Banyan Network

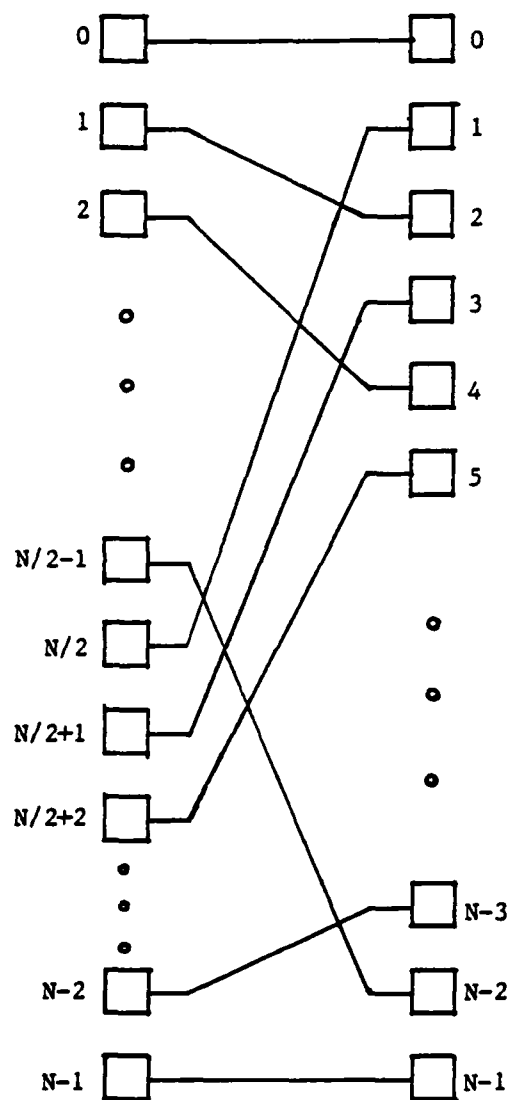


Figure 11: The perfect shuffle of an N element vector

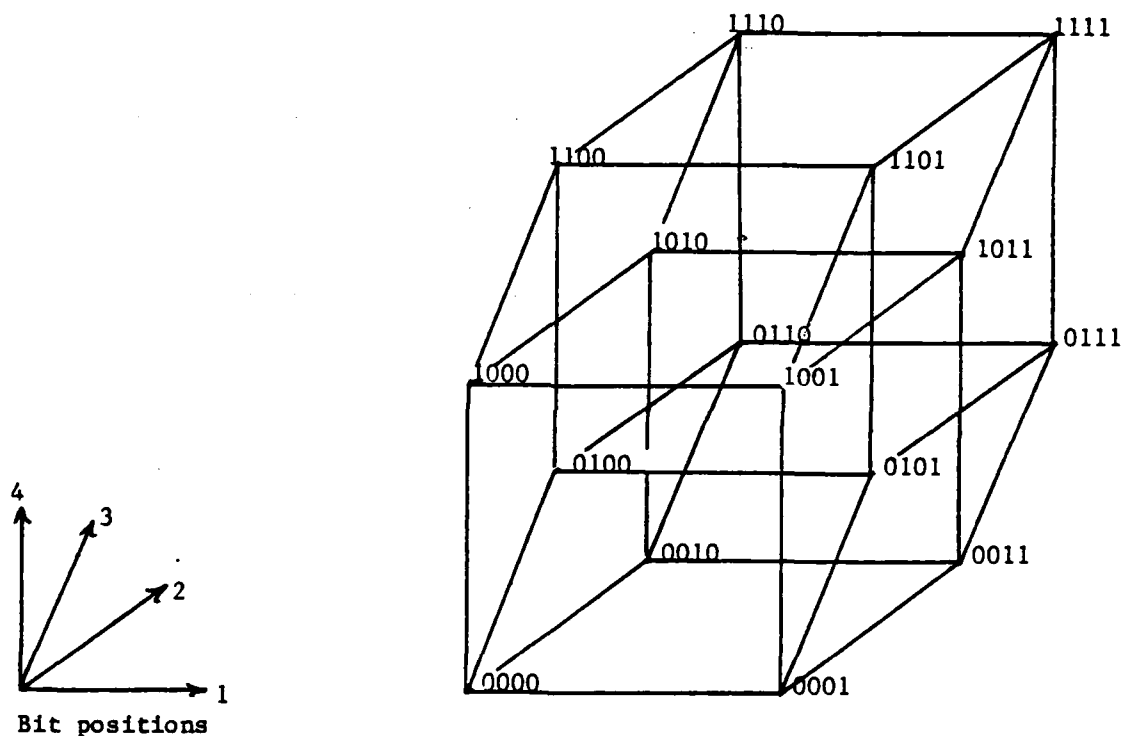


Figure 12: The edge assignment in a binary 4- cube network.

Note that every edge connects two nodes whose indices differ in exactly one bit. If we assume a processor in each node, the alignments along the edges parallel to any one axis are obtained by setting in the IB4C network, the switches in the corresponding column in the cross position and all the rest in the straight position.

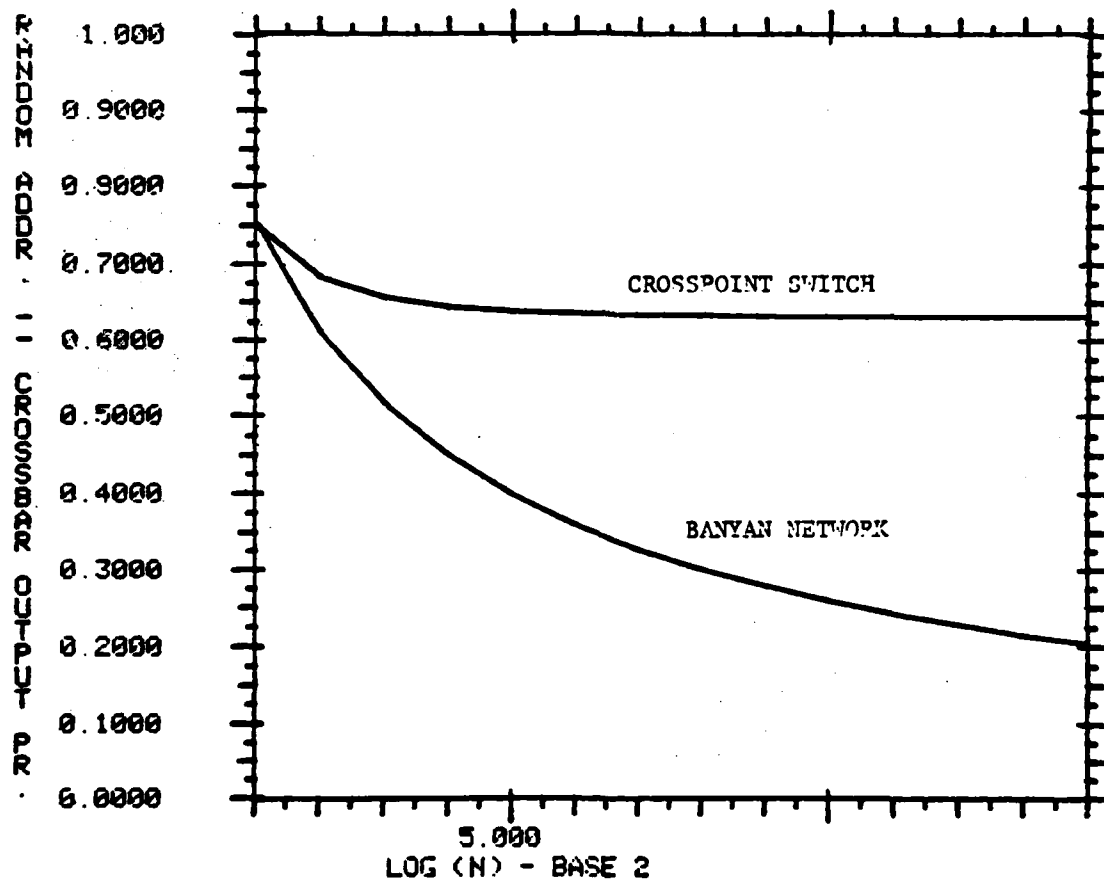


FIGURE 13: PROBABILITY OF ACCEPTANCE IN THE CASE OF CROSSPOINT SWITCH AND BANYAN TYPE NETWORKS. MODEL ASSUMES RANDOM ADDRESSING OF OUTPUTS.

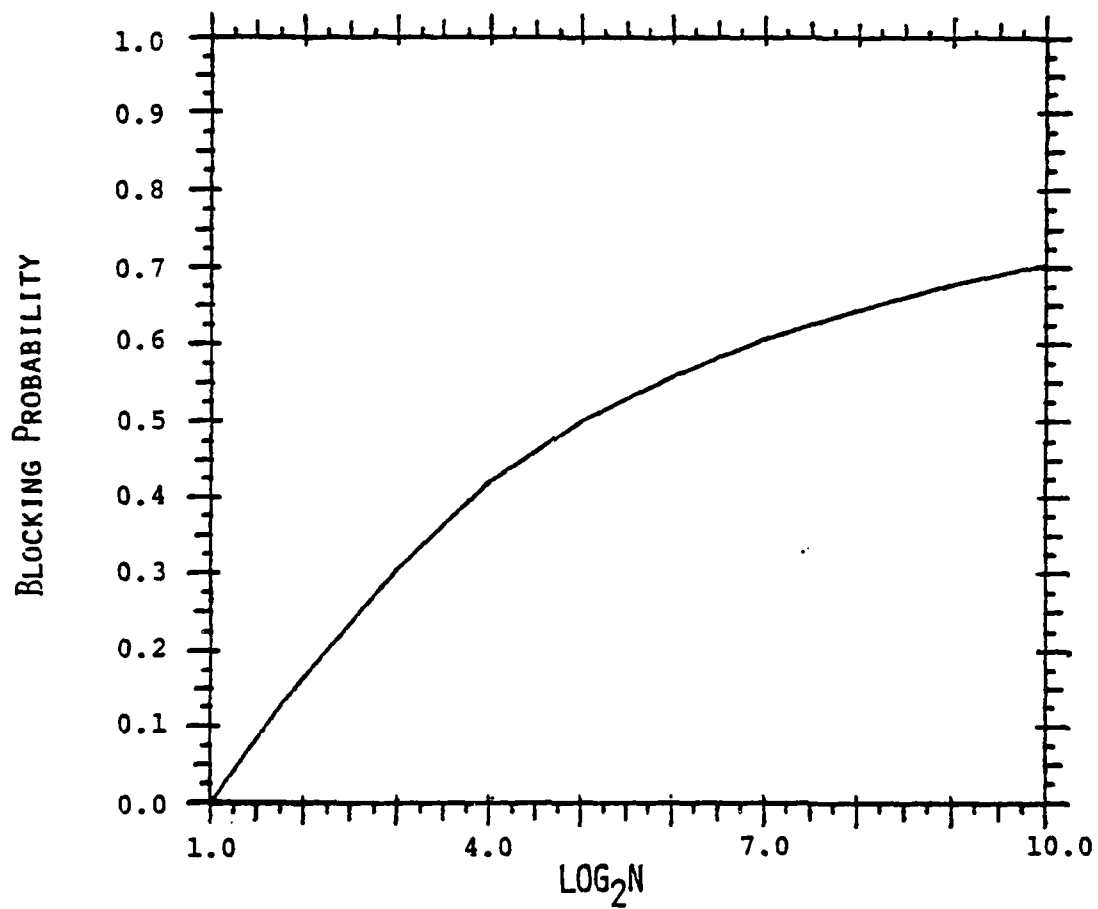


FIGURE 14: MESSAGE BLOCKING PROBABILITY (P_N) VERSUS NETWORK SIZE ($\text{LOG}_2 N$) (FOR BANYAN NETWORKS WITH UNIQUE AND UNIFORMLY DISTRIBUTED DESTINATION ADDRESSES)

VLSI BASED INTERCONNECTION NETWORKS*

Mark A. Franklin and Donald F. Wann

Washington University
St. Louis, Missouri,

ABSTRACT

Interest in tightly coupled multiprocessor computer systems has grown as the possibilities for high performance with such systems have been recognized. Central to their design is the structure of the network over which the processors communicate. Unless properly designed, such networks can become both a cost and performance bottleneck. This paper focuses on the design of VLSI communications networks, that is, on communications networks which can be placed on a single VLSI chip. Traditional SSI based cost and complexity measures for such networks have principally involved switch aggregate counts. In a VLSI domain, however, more appropriate measures involve chip area, and space-time product. The effects of network topology and VLSI layout on these measures are reviewed with regard to two network types. Another important question related to the VLSI communication network problem relates to chip pin constraints. This problem is discussed and some effects and options presented by bit slice network designs are described.

INTRODUCTION

In recent years there has been increasing interest in tightly coupled, physically local multiprocessor computer systems (1,2). This has been due both to the enhanced performance possibilities for such systems, (e.g., increased computational power resulting from parallel processing and higher reliability resulting from component redundancy) and the steady decrease in hardware costs associated with these systems. A central issue in the design of such systems concerns performance degradation due to costs associated with interprocessor communication. One aspect of this problem relates to the question of user problem decomposition and scheduling (3,4). Another relates to the structure and design of the network over which the multiple processors communicate. As the number of processors increases the characteristics of both the decomposition and scheduling algorithms, and the communications network, become critical in establishing acceptable overall system performance cost. This paper is concerned with certain communications network design questions which arise in context of multiprocessor systems designed in a VLSI environment.

Various studies aimed at characterizing and This work was supported in part by NSF Grant MCS-78-20731, ONR Contract N00014-80-C-0761 and NIH Grant RR00396.

quantifying the performance of SSI based networks have already been pursued (5,6,7). Typically the principal figures of merit used in these studies have been related to the number of switches required by the communications network and the bandwidth of the path between processors. For a given network architecture, determination of the number of switches is straightforward, while estimation of the bandwidth has, in most cases, been derived from an analysis of the average number of switches through which a signal must pass and the blocking characteristics of the network.

Use of these types of figure of merit make sense in an environment where the cost of switching elements is substantially greater than the cost of wires and connection paths. The situation changes however, with the introduction of VLSI technology. This fabrication methodology has the potential for economically placing large switching networks or subnetworks on a single chip. Cost here becomes related to chip area. Unfortunately, a new challenge appears: the implementation of the connection paths may use substantial amounts of the chip area, thus limiting the area available to the switch elements themselves. This has the effect of reducing the size of a switching network that can be fabricated on a chip of a given size. The time delay associated with the connection paths also contributes to the overall delay, thus directly affecting bandwidth. Area, topology and layout, mainly ignored in traditional communication network analysis, become important interrelated factors in VLSI network design (8).

The advent of VLSI has thus significantly changed the design space with which engineers must contend. More meaningful figures of merit based on parameters of time (on/off chip time) and chip area now seem more appropriate in many situations. In the next section two interconnection networks Crossbar (9) and Banyan (10) are compared in terms of their space-time products when implemented on a single VLSI chip.

While single chip design questions are important, when large networks requiring multiple chips are to be designed with single chip subnetworks as the building components, interchip communications delays can dominate overall delay times. Furthermore there is often a close relationship between intra and interchip communications network design. It may be advantageous, for instance, for the communications network on the chip to have a topology

different from the larger interchip network. Furthermore, this interacts directly with questions relating to chip pin limitations, network control, and communications path width. It may turn out that bit serial communications paths are best because they preserve pins and permit large networks to be placed on a single chip. Reducing the delays associated with interchip communications may more than offset the extra time necessary for serial data transmission. The question of pin limitations is explored later in this paper and some preliminary results reviewed.

SPACE-TIME NETWORK COMPARISONS

Banyan and Crossbar Networks

This section reviews certain research results comparing banyan and crossbar interconnection networks (11). The crossbar network considered is of the form shown in Figure 1. While there are many ways of designing a crossbar (e.g., demultiplexer/multiplexer designs, switched bus designs), the approach examined has a number of generally desirable capabilities. These include distributed (local) network path control, asynchronous and pipelined data transfer, and a high degree of modularity (9). Furthermore its naturally regular and planar layout appears to make it well suited to VLSI implementation. The banyan network considered is of the form shown in Figure 2. It too can be designed for distributed network path control, asynchronous data transfer and pipelining. On the other hand its modularity properties are not quite as straightforward as the crossbar and its topology, while regular in a certain sense, is inherently nonplanar. Note that both networks have a full interconnection capability in that any single input/output connection can be made by placing the appropriate switches in the proper positions.

While the properties mentioned above are important, most work to date has compared the networks principally on the basis of switch complexity (i.e. number of switches required for implementation), and network bandwidth. For square N input, N output networks switch complexity for the crossbar is $A_{CB} = O(N^2)$ while for the banyan it is $A_{BA} = O(N \log N)$.

Network bandwidth is associated with three items. First, pipeline characteristics of the network and message length distributions must be determined. For analysis simplicity, a nonpipelined design and circuit switched mode of operation are assumed. Message length considerations therefore do not enter into these bandwidth comparisons. The second item to consider relates to the average number of switches through which a message must pass assuming uniform addressing of input and output ports. For the crossbar this is $D_{CB} = O(N)$ while for the banyan it is $O(\log N)$. The final item to consider relates to the networks blocking characteristics and the protocol to be used when a message is blocked. The crossbar is a strict sense nonblocking network. That is, as long as each input port addresses a unique output port, no message is blocked in the network due to path contention (and no rearrangement of paths is necessary). The banyan network on the other hand is a blocking network, and under certain situations message blocking can occur. For N less than about 2000,

the probability of this blocking can be approximated by $P_N = 1 - b/N^a$ with $a = .19$ and $b = 1.05$. Assuming a saturated system, synchronized messages of equal length, and a message retry protocol where blocked messages reenter the system again with the next message batch, the average delay through a banyan network can be derived as $D_{BA} = O(\log N)/(1 - P_N)$.

Overall cost measures for the banyan and crossbar can now be obtained as:

$$C_{CB}' = A_{CB}' \cdot D_{CB}' = O(N^3) \quad [1]$$

$$C_{BA}' = A_{BA}' \cdot D_{BA}' = O(N^c \log^2 N) \quad (1 < c < 2) \quad [2]$$

From these equations it is clear that by traditional measures, the banyan is much less costly than the crossbar. This is also true for other blocking networks such as the Omega (12) and indirect binary N -cube (13) whose switch complexities are also $O(N \log N)$.

VLSI Network Implementation Model

Consider next that the layout of a crossbar network on a single chip directly follows the topology of Figure 1. Assuming the logic associated with each crosspoint fits into a square with sides of length L , and the spacing between squares follows the Mead and Conway (8) recommendations for spacing between metal interconnect lines (i.e., 3 feature sizes, 3λ), then the area in units of λ^2 is given by:

$$A_{CB} = [NL + 3(N-1)]^2 = O(N^2) \quad [3]$$

Unfortunately, for the banyan case the analysis is not as straightforward and one must refer to reference 11 for details. The thrust of the analysis, however, is as follows. First assume again that the switches in the banyan network fit into a square with sides of length L . Next assume that two layers of metal interconnect are available, one for horizontal and one for vertical lines. Various layouts may be proposed, but as long as the general form shown in Figure 2 is preserved (i.e., successive rows of switches) two things become evident. First, the horizontal distance required by the network will be $O(N)$ since this will vary directly with the number of input and output ports. Second, the vertical spacing between switch rows will increase as the network grows. This is because the number of horizontal lines which require routing between the right and left halves of the network increases as the network grows. These lines, being routed on the same plane, need more area as one moves from level (row) to level. This is illustrated in Figure 3. The result of this is that although there are $O(\log N)$ levels, the vertical distance grows as $O(N)$ and thus the area grows as $A_{BA} = O(N^2)$.

The interesting point to note here is that this is just the same as the crossbar, and is considerably different from what is predicted from switch aggregate counts. One other point to note is that these same results can be obtained by following a graph theoretic argument developed by Thompson (14).

Developing time delay models follows the same approach discussed above. As pointed out, for the

crossbar an average path contains N crosspoints. If each crosspoint is implemented in NMOS NOR gates with: a fanout f ; a transit time τ ; a pullup to pulldown transistor impedance ratio of 4; m levels of logic; a metalization capacitance/transistor gate capacitance ratio of α ; and intercrosspoint drivers of minimum area; then the crossbar delay can be derived as:

$$D_{CB} = 2.5Nmf\tau + (N+1)\tau(1+2.25\alpha) = O(N) \quad [4]$$

For the banyan case a more complex expression can be obtained. Here certain assumptions must be made concerning driving the metal lines between levels which increase in length from level to level. Assuming the metal lines present purely capacitive loads, and are driven by a matched sequence of driver stages (8) to minimize delay, it can be shown that the delay presented by the lines is $O(\log^2 N)$. Introducing the multiplicative factor related to the blocking probability yields an overall delay of $D_{BA} = O(N^d \log^2 N)$ where $0 < d < 1$. For large N this is less than D_{CB} , however it is greater than that predicted from traditional analysis.

The overall space-time product measure is given below:

$$C_{CB} = A_{CB} \cdot D_{CB} = O(N^3) \quad [5]$$

$$C_{BA} = A_{BA} \cdot D_{BA} = O(N^d \log^2 N) \quad (2 < d < 3) \quad [6]$$

These results indicate that while C_{BA} is still less than C_{CB} for large N , the costs are much closer than predicted from standard switch aggregate analysis. A more detailed analysis indicates that for reasonable values of N (i.e., values that could be currently implemented on a single chip), the two networks have roughly comparable space-time performance.

PIN LIMITATIONS

One of the key constraints on placing very large networks on a single chip is the limited number of pins supported by standard integrated circuit carriers. Consider for instance the interconnection network depicted in Figure 4 which establishes a B' -bit path between device x_i and device y_j where $1 \leq i, j \leq N'$. Our interest is in developing networks that are general in that we place minor, if any, conditions on the specific numerical values for N' and B' . If the network of Figure 4 were to be implemented on a single VLSI chip then the number of required pin connections (ignoring power, ground, and general control such as reset) is given by $2N'B'$. Suppose, for example, that we have a square interconnection network with $N' = 12$ and $B' = 16$. Then the number of required pins is 384, much larger than common commercially available integrated circuit carriers. The total number of pins is limited mainly by the increase in the physical length of the package; the pins are typically placed on 100 mil centers if the package is to be inserted in pads on printed circuit boards. (We ignore here certain more advanced schemes such as the array configuration used by IBM). For this pin placement, a 64 pin dual-in-line package is 3.2 inches in length. This becomes physically awkward to manipulate and also becomes more susceptible to breaking forces. The 384 pin example, for instance would require a 19.2 inch dual-in-line package!

There are a number of potential solutions to this pin limitation problem. We review here two of the more obvious ones with details being presented in reference 15. The first approach is to implement a large network requiring many pins as a collection of smaller networks where each of the smaller networks can be contained on a single chip in which the pin constraints of the chip are met. An $N' \times N'$ network would therefore be decomposed into a set of subnetworks (each subnetwork of size $N \times N$) which would themselves be interconnected in some fashion.

The second approach is to slice the network so that one creates a set of network planes, each plane handling one or more bits (e.g., B bits) of the B' -bit wide datapath. This is commonly done in memory designs. Note that a potential problem arises here due to the difficulty in synchronizing the multiple planes. Although details of this issue will not be discussed here, there are ways of dealing with this problem.

The question to be considered is what represents the "best" combination of datapath slice B and chip network size N given: an overall network size N' ; a data path width B' ; an intrachip network type T ; an interchip network type T' ; a maximum allowable number of pins N_p ; and a required number of pins for power, ground and control N_k .

A Chip Count Model

While the "best" B and N selection refers to both the chip count and bandwidth of the overall $N' \times N'$ network, due to space limitations only the chip count analysis is reviewed here. With regard to network types T and T' , the overall chip count is a function only of T' , the interchip network type. Although there are numerous ways of connecting subnetworks together to achieve an overall network, two basic network types are considered.

The first is the common crossbar network. An $8 \times 8 \times 1$ crossbar network for example can be configured using $4 \times 4 \times 1$ chip components. The number of $N \times N \times B$ chips required to implement an $N' \times N' \times B'$ network is given by:

$$N_{cb} = \left\lceil \frac{B'}{B} \right\rceil \left\lceil \frac{N'}{N} \right\rceil^2 \quad [7]$$

The second type of network considered is the banyan. The number of $N \times N \times B$ chips needed to implement an $N' \times N' \times B'$ network is given by:

$$N_{ba} = \left\lceil \frac{B'}{B} \right\rceil \left\lceil \frac{N'}{N} \right\rceil \lceil \log_2 N' \rceil \quad [8]$$

The first term in this expression is the number of bit slices or network planes; the second term is the number of chips at each level (row) and the third term is the number of levels necessary to achieve a full interconnection.

Pin constraints can be introduced by noting that:

$$N_p \geq K_1 3N + N_k \quad [9]$$

where $K_1 = 4$ for a fully modular crossbar network, and $K_1 = 2$ for a banyan network. Consider next two cases. For case I the number of power, ground and control pins are small compared to the data pins and thus, using all available pins, $N = N_p / K_1 B$. This is typical of clocked systems where a small number of clock lines are needed to synchronize all the data lines. For case II N_k is not negligible

and the number of control lines is proportional to the number of ports, N ; thus, $N = N_p / (K_1 B + Q)$. This would be an appropriate model if the network chips communicated with each other in an asynchronous manner and a request/acknowledge control line pair ($Q=2$) were associated with each port.

Each of the above expressions for N can now be substituted back into equations 7 and 8, and a value of B , and thus N , yielding the minimum number of chips obtained. To get some feeling for this one can assume that large values of B' and N' are present and that the number of chips can be approximated by expressions 7 and 8 with the ceiling functions removed. From these continuous functions it is clear that for case I the number of chips is minimized with $B=1$. This is reassuring since it corresponds to experience with memory chip design where the slice width is generally taken as one bit. A discrete optimization search procedure verifies that this is true with the ceiling functions in place for crossbar networks, and for banyan networks above a certain size ($N' > 256$). For case II, the $B=1$ result generally holds for the crossbar case. For the banyan case, however, the best value of B varies considerably depending on the particular N_p , N' and B' values being considered. For instance with $B'=16$, $N'=512$ and $N_p=60$, a $B=2$ ($N=10$) results in $N_{ba}=1248$ while a $B=1$ ($N=15$) results in $N_{ba}=1680$. Typically large differences in chip counts occur when a nonoptimum value of B is used in this situation. Two other points should be noted. First, the control pin overhead in case II ($Q=2$) is substantial. For instance with $B'=16$, $N'=256$ and $N_p=90$; $N_{ba}=192$ with $Q=0$ and $N_{ba}=348$ with $Q=2$. Second, from a chip count point of view, there is a heavy penalty associated with using a crossbar interchip network due to the $O(N^2)$ versus $O(N \log N)$ network growth in number of chips.

The above sort of chip minimization analysis suggests that in designing an interconnection network chip set, chip control procedures which are proportional to the number of I/O ports be avoided (i.e., no request/acknowledge pairs on a per port basis), and a banyan like interchip network be used. Under these conditions, a path slice of $B=1$ seems appropriate. Not considered here is the question of how this path width and interchip network selection effect overall bandwidth. Initial results (15) indicate that the selections above are also appropriate for bandwidth optimization.

Other questions remain to be answered. One relates to whether having separate network chips are appropriate given their pin requirements. Perhaps structures which include both networks and processors are more reasonable. A variety of questions relating to centralized versus decentralized network control, the tradeoffs associated with circuit switched, packet and pipelined network designs, and the various options associated with synchronous versus asynchronous/delay insensitive design remain to be explored.

References

1. Enslow, P.H., Jr., "Multiprocessor organization - a survey," ACM Comp. Sur. 9,1 (March 1977).
2. Kuck, D.J., "A Survey of Parallel Machine Organization and Programming," ACM Comp. Sur. 9,1 (March 1977).

3. Chu, W.W. et.al., "Task Allocation in Distributed Data Processing", Computer 13,1 (Nov.1980).
4. Cornett, D.H. and Franklin, M.A., "Scheduling Independent Tasks with Communications", Proc.17th Allerton Conf. on Comm., Cont. and Comp. (Oct.1979).
5. Siegel, H.J.; McMillen, R.J., and Mueller, P.T., Jr., "A Survey of interconnection methods for reconfigurable parallel processing systems," Proc. 1979 Nat. Comp. Conf. (June 1979).
6. Anderson, G.A., and Jensen, E.D., "Computer interconnection structures: taxonomy, characteristics and examples" ACM Comp. Sur. 7,4 (Dec. 1975).
7. Thurber, K.J., "Interconnection networks - a survey and assessment", Nat. Comp. Conf. (May 1974).
8. Mead, C. and Conway, L., Introduction to VLSI Systems, Addison-Wesley Pub. Co. Reading, MA (1980).
9. Franklin, M.A.; Kahn, S.A., and Stucki, M.J., "Design Issues in the Development of a Modular Multiprocessor Communications Network", Proc. Ann. Symp. on Comp. Arch. (April 1979).
10. Goke, L.R. and Lipoviski, G.J., "Banyan Networks for Partitioning Multiprocessor Systems", Proc. Ann. Symp. on Comp. Arch. (1973).
11. Franklin, M.A., "VLSI Performance Comparison of Banyan and Crossbar Communications Networks", IEEE Trans. on Comp. C-30,4 (April 1981).
12. Lawrie, D., "Access and alignment of data in an array processor", IEEE Trans on Comp. C-24,12 (Dec. 1975).
13. Pease, M.C., "The indirect binary n-cube microprocessor array," IEEE Trans. Comp. C-26,5 (May '75).
14. Thompson, C.D., "Area-Time Complexity for VLSI", Proc. 11th Ann. ACM Symp. on the Theory of Comp. (April 1979).
15. Franklin, M.A. and Wann, D.F., "Pin Limitations in VLSI Interconnection Networks", Int. Rept. #CCSD-81-01, Washington Univ., St. Louis, MO Ctr. for Comp. Sys. Design (Feb. 1981).

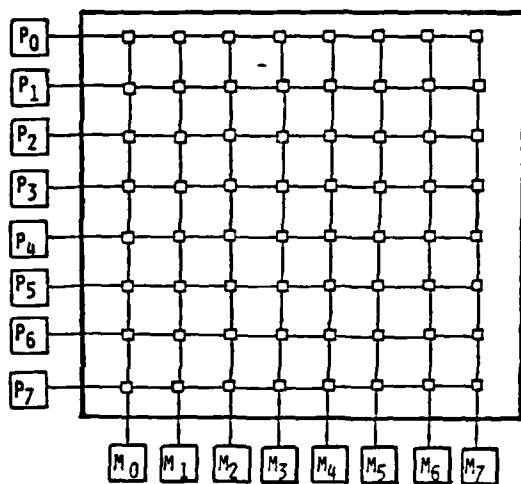


FIGURE 1: 8 * 8 CROSSBAR SYSTEM

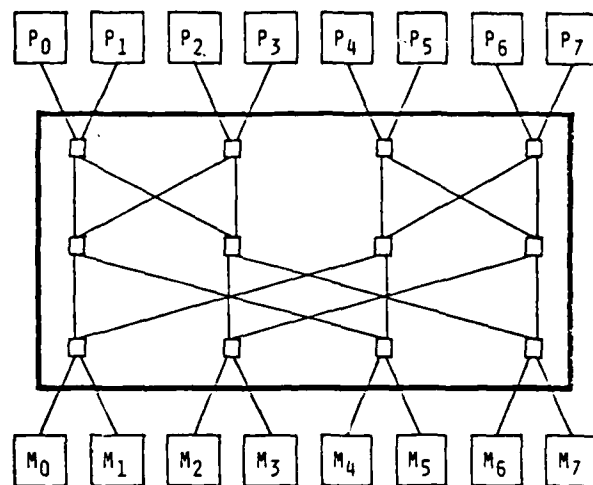


FIGURE 2: 8 * 8 BANYAN SYSTEM

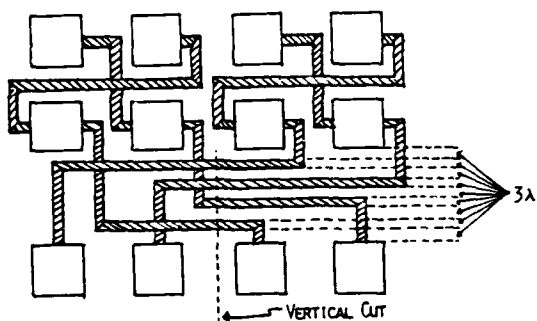


FIGURE 3: VERTICAL LAYOUT FOR BANYAN (B=1)

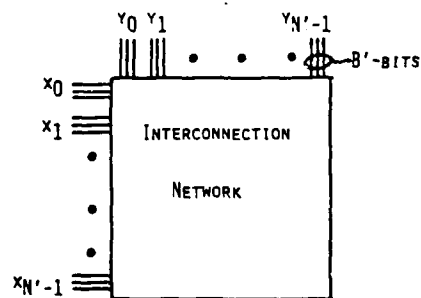


FIGURE 4: AN $N' \times N'$ NETWORK (B' WIDE DATAPATHS)

PIN LIMITATIONS AND VLSI INTERCONNECTION NETWORKS*

Mark A. Franklin and Donald F. Wann
Department of Electrical Engineering
Washington University
St. Louis, Missouri 63130

Abstract: Multiple processor interconnection networks can be characterized as having N' inputs and N' outputs, each B' bits wide. Construction of large networks requires partitioning of the $N' \times N' \times B'$ network into a collection of $N \times N$ switch modules of data size B ($B < B'$) each implemented on a single chip and interconnecting them with a specific interchip network type T' . The major constraint in the VLSI environment is the pin limitation, N_p , of the individual modules; these are allocated between data and control lines, Q . This paper presents a methodology for selecting the optimum values for N and B given values of the parameters, N' , B' , T' , Q , and N_p . Models for both the banyan and crossbar networks are developed and arrangements yielding minimum number of chips, average delay through the network, and product of number of chips and delay, are presented. A bit slice approach ($B = 1$) produces the optimum arrangement for the crossbar, while for the banyan the optimum is achieved with multiple bits per module.

Introduction

Over the past few years a variety of physically local, closely coupled multiple processor systems have been proposed (1,2,3,4). One key issue in the design of such systems concerns the communications network used by these multiprocessor systems. Various studies have focused on the functional properties of such networks (i.e., what permutations are possible, what control algorithms are needed, etc.), on their complexity, and to some extent on performance issues (5, 6, 7, 8, 9). In most cases network complexity has been measured by the number of elementary switching components needed by a network of a given size and type, while performance has been determined by the average number of elementary switching components through which a message must pass (i.e. average delay). Recently work has begun on examining complexity and performance questions in the context of VLSI implementation of such interconnection networks. Franklin (10) has compared two networks, crossbar and banyan, operating in a circuit switched mode in terms of their space (area) and time (delay) requirements. The networks were assumed to be implemented as complete modules on a single VLSI chip.

Closer examination of VLSI network implementation problems shows that pin limitations, rather than chip area or logical component limitations, are a major constraint in designing very large interconnection networks. Consider, for instance, a network with N' inputs, M' outputs and with each output being B' bits wide ($N' \times M' \times B'$). The number

of required pin connections (ignoring power, ground and general control) for a single chip implementation is given by $B'(N'+M')$. For a square network of size twelve with $B'=16$, the number of pins required would thus be 384; much larger than common commercially available integrated circuit carriers. Given that pins are typically placed on 100 mil centers along the periphery of the package, the total number of pins is limited mainly by the increase in the physical length of the package. For this pin placement and the 384 pin example, a 19.2 inch dual-in-line package would be required.

In this paper we focus on two of the more obvious solutions to this pin limitation problem. The first approach is to implement a large network ($N' \times N'$) requiring many pins as a interconnected set of smaller subnetworks ($N \times N$) where each of the smaller networks can be contained on a single chip in which the chip pin constraints are met.

The second approach is to slice the network so that one creates a set of network planes, each plane handling one or more bits (e.g., B bits) of the B' wide datapath. This is commonly done in memory designs. A potential problem arises in this approach due to the difficulty in synchronizing the multiple planes. This is discussed in reference 11.

The remainder of this paper deals with determining the "best" combination of datapath slice B and chip network size N given:

1. N' : An overall network size ($N \leq N'$),
2. B' : A data path width ($B \leq B'$),
3. T : An intrachip network type (e.g., the interconnection network implemented within the $N \times N$ chip might be a crossbar).
4. T' : An interchip network type (e.g., the interconnection network implemented between the $N \times N$ chips to achieve the overall $N' \times N'$ network might be an Omega network).
5. N_p : The maximum number of pins allowed on a chip.
6. N_k : The number of pins on a chip allocated to power, ground, and control.

"Best" in this context, refers to both chip count and bandwidth of the overall $N' \times N'$ network. Figures 1, 2 and 3 illustrate a general $N' \times N'$ network, and a possible decomposition of a sample 16×16 network. In the next section basic models

*This work was supported in part by NSF Grant MCS-78-20731 and ONR Contract N0014-80-C-0761.

for this problem are presented and used to determine the B and N combinations which minimize the total chip count, the overall network delay and an overall performance measure using the product of chip count and time delay.

The Basic Model

The basic model consists of two parts. The first relates to the chip count while the second concerns network time delay. For brevity, only square fully connected networks (i.e. there is a path from each input port to each output port) are considered. Note that certain input/output paths may have a common subpath and this may result in messages being temporarily blocked.

Let us refer to the $N \times N \times B$ chip as a switch module; a number of these modules will be interconnected to realize the N' network. This paper considers two types of interchip networks (T'): the incremental crossbar, CB, and the banyan BA (12,13,14). While there are many ways of designing a crossbar network (e.g., demultiplexer/multiplexer configuration, switched multiple busses, etc), the incremental crossbar design (Figure 4) can be expanded on a unit basis by adding basic switch modules in a row-column arrangement. This modularity property permits flexible expansion while retaining the nonblocking and full connection properties of the crossbar. A price is paid for these properties in terms of number of switches and pins required on a switch module. While the number of switches required per switch module may not be a serious constraint with VLSI technology, the problem of pin constraints is severe. For the incremental crossbar, the modularity property requires $4NB$ data pins to implement a $N \times N \times B$ switch module while the banyan, a blocking network, requires $2NB$ data pins.

To make global comparisons similar and to eliminate blocking at the switch module level, this paper examines cases in which the switch modules are constructed using an incremental crossbar architecture ($T = CB$). Two types of module interconnections are examined; the crossbar and the banyan ($T' = CB$ or $T' = BA$).

Chip Count Model

As illustrated in Figure 4, the number of $N \times N \times B$ chips required to implement an $N' \times N' \times B'$ incremental crossbar network is given by:

$$N_{cb} = \left\lceil \frac{B'}{B} \right\rceil \left\lceil \frac{N'}{N} \right\rceil^2 \quad [1]$$

The banyan network is one of the class of blocking networks whose logical component complexity grows as $O(N \log N)$ rather than $O(N^2)$. As illustrated in Figure 5, the number of $N \times N \times B$ chips needed to implement an $N' \times N' \times B'$ banyan network is given by:

$$N_{ba} = \left\lceil \frac{B'}{B} \right\rceil \left\lceil \frac{N'}{N} \right\rceil \left\lceil \log_N N' \right\rceil \quad [2]$$

The first term in this expression is the number of bit slices or network planes that are required. The second term represents the number of chips at each level (row), while the third term is the number of levels.

Time Delay Model

A model giving the average time for a signal to propagate through a network must include the time to traverse each of the chips, the time to propagate from chip to chip, and since the bit slice approach separates the bits in a data word, the additional time that is needed to make certain that all the data bits have completed their movement through the $N' \times N' \times B'$ network.

The average delay associated with a basic switch module will be designated as D_{cb} since these modules have a crossbar construction. Path setup delays (i.e., time to set switches in their desired positions) are not considered here. The delay of a pin driver and associated interconnection wires between modules (i.e. the intermodule delay) is denoted by D_{im} . The intermodule delays for the CB and BA networks are different and will be denoted as D_{imcb} and D_{imba} . Additional synchronization delay introduced by the designer to assure that all data bits have traversed the network will be represented by D_{syncb} and D_{synba} .

For the CB network the average delay can be determined by examining Figure 4. Note that this represents one of $\lceil B'/B \rceil$ planes. Assume that each switch module, implemented on a single chip, represents an $N \times N$ CB network. The pin drivers for each module are also located on the chip. For this arrangement the number of modules in an average path is $\lceil N'/N \rceil$ and each intermodule path has the same delay D_{imcb} . Therefore the average network delay D'_{cb} is given by:

$$D'_{cb} = \left\lceil \frac{N'}{N} \right\rceil D_{cb} + \left\lceil \frac{N'}{N} \right\rceil D_{imcb} + D_{syncb} \quad [3]$$

Note that a circuit switched design is assumed here with no pipelining between modules.

For the BA network the number of switch modules and the number of intermodule connections is $\log_N N'$. Here, because of the connection topology, the intermodule paths are not constant in length. The average delay, D'_{ba} , through such a network (assuming no delay penalty for blocking) is given by:

$$D'_{ba} = \left\lceil \log_N N' \right\rceil D_{cb} + \left\lceil \log_N N' \right\rceil D_{imba} + D_{synba} \quad [4]$$

Pin Constraints

For a square $N \times N \times B$ chip with N_k pins allocated to power, ground and control, the pin constraint is given by:

$$N_p \geq KBN + N_k \quad [5]$$

where $K = 4$ for the CB network and $K = 2$ for the BA network. The equality will be used since it is advantageous to utilize as many available pins as possible. Two cases may be considered. Case 1 is the situation where the number of data pins is much larger than N_k (i.e., $KBN \gg N_k$) and thus equation 5 becomes:

$$N = \left\lfloor \frac{N_p}{KB} \right\rfloor \quad [6]$$

This is typical of a clocked system where a small number of clock lines are needed to synchronize all the data lines.

Case 2 encompasses the situation where N_p is not negligible and there is a control line overhead associated with the data paths. Assuming that the number of control lines is proportional to the number of ports, N , on an individual chip (i.e. $N_k = QN$ where Q is a constant), N can be expressed as:

$$N = \left\lceil \frac{N_p}{(KB+Q)} \right\rceil \quad [7]$$

This would be the appropriate model if network chips communicated with each other in an asynchronous manner and the control line overhead consisted of request/acknowledge pairs ($Q = 2$).

Chip Count Minimization

For large networks with large datapath widths and chips with a large number of pins, the ceiling and floor functions can be removed from [1] and [2], and [6] and [7]. Then N_{cb} and N_{ba} can be approximated as continuous functions. Assume that all available pins are used and consider Case 1 where N is given by equation 6. Substituting equation 6 with $K = 4$ and $K = 2$ respectively into continuous versions of equations 1 and 2 yields:

$$N_{cb1} = 16BB'(N^{**2})/N_p^{**2} = K_{cb} B \quad [8]$$

$$N_{ba1} = \frac{2B'N' \log N'}{N_p (\log N_p - \log 2B)} = \frac{K_{ba}}{\log N_p - \log 2B} \quad [9]$$

For a given pin constraint N_p , and overall network requirements N' and B' , K_{cb} and K_{ba} are constants. Minimizing N_{cb1} and N_{ba1} for this case requires that B be minimized. The smallest datapath width possible is $B = 1$, hence with this model N should be selected to be N_p/K . This result corresponds to memory chip design where the slice width is generally taken as one bit. Note however, that this was obtained with a continuous approximation to equations 1 and 2; while $B = 1$ yields a minimum number of chips in most cases, there are situations where other values of B are better. For instance, with a BA network with $N_p = 60$, $N' = 128$ and $B' = 16$, a $B = 1$ solution yields $N_{ba1} = 160$, while a $B = 2$ solution yields $N_{ba1} = 142$.

For case 2 where N_p is not negligible equation 7 is used for N and substituted back into the continuous versions of [1] and [2] to give:

$$N_{cb2} = \frac{(4B+Q)^2 B' N'^2}{B N_p^2} = \frac{K_{cb} (4B+Q)^2}{16B} \quad [10]$$

$$N_{ba2} = \frac{K_{ba} (2B+Q)}{2B (\log N_p - \log (2B+Q))} \quad [11]$$

The derivatives of N_{cb2} and N_{ba2} with respect to B can now be taken, and the values of B and N which minimize the chip count obtained.

For the case of T' a CB, the number of chips N_{cb2} is minimized when $B = Q/4$. Thus for a request/acknowledge pair associated with each chip datapath ($Q = 2$), B would be selected as 1. While this is true for almost all cases considered, the continuous model approximation should be checked when N' is less than 64 or B' is less than 16 (e.g., for $N' = 32$, $N_p = 75$, $Q = 2$ and $B' = 16$; $B = 1$ yields N_{cb2}

= 144; $B = 2$ yields $N_{cb2} = 128$).

For the case of $cb2$ a BA network, an equation can be derived for obtaining the optimum B and N and indicates that the continuous model does not yield optimum values in many situations. For instance, for $N_p = 90$, $N' = 512$, $Q = 2$ and $B' = 16$, a search procedure working directly with equation 2 gives an optimum $B = 4$ and yields $N_{ba2} = 684$. Note that using $B = 1$ in this case results in $N_{ba2} = 1152$. This is not unusual, and in most cases $N_{ba2} (N < 140)$ where $Q \geq 2$, a choice of $B = 1$ will be nonoptimal.

Equations 1 and 2 were solved using optimal values of N and B , and the chip count was obtained as a function of the parameters N_p , N' and network type T' . Figure 6 illustrates how the total number of chips varies as a function of the network size. Plots for two different values of N_p and Q are also given. For a given N' , N_p and Q , the BA requires fewer chips than the CB implementation and the curves agree with the observation that the crossbar grows as $O(N^{**2})$ while the banyan grows as $O(N' \log N')$. As expected, increasing Q or N_p requires a larger number of chips for both the banyan and the crossbar. Although not shown explicitly in these graphs, the optimum value of B is 1 for the crossbar ($N_p \geq 64$), while for the banyan the optimum B ranged from 1 to 4 ($N_p \geq 64$).

Network Delay Minimization

Next we determine expressions for the delays, D_{cb} , D_{im} , and D_{syn} and incorporate these into equations 3 and 4 to compute the average delay through the two networks.

Crossbar Network

The value of D_{cb} has been developed by Franklin (10) using NMOS NOR gates for construction of the crossbar module and is given as:

$$D_{cb} = N[2.5m\tau + \tau(1+2.25\alpha_{cb})] = N\alpha_{cb} \quad [12]$$

The parameters are defined in Table 1 which also gives some typical values. The equation assumes a circuit switched CB, and uniformly distributed addressing of module output ports. The first term in the brackets represents the delay through an individual switch within a module, while the second term is the delay between switches in a module.

The delay encountered when a signal goes off the chip, propagates along an interconnecting path and enters another switch module is D_{imcb} . A buffer (e.g., a series of inverters) must be included within the switch module to allow the minimum size transistor to drive the module pin and associated load with minimum delay. The buffer delay is determined by the gate capacitance of the minimum size transistor, the number of stages in the buffer, the capacitance of the pin being driven, the capacitance along the interconnecting path, and the capacitance of the receiving module pin. This delay is minimized when exponentially sized cascaded inverters are used (14). The delay in this case is:

$$D_{imcb} = \tau \log_8 \beta_{cb} \quad [13]$$

where β_{cb} is the ratio of the buffer load capaci-

tance to the buffer input transistor gate capacitance. The transistor gate capacitance, C_g , is the capacitance per unit area times the gate area of the minimum transistor. To determine the load capacitance assume that the driving and receiving pin capacitances are equal and each has a value of C_{pin} . Further postulate that the modules for the CB will be placed on a circuit board and interconnected via printed circuit copper paths. Given the planar topology of the CB, the spacing between modules will generally be less than one inch. Pin capacitance will dominate in this case and $C_{cb} = (2C_{pin} + C_{path}) / C_g = 2C_{pin} / C_g$.

The synchronization delay depends upon the specific design technique used to determine that all bits have traversed the network. Assuming self-timed design strategy, a reasonable design practice is to include a tolerance or guard region that is proportional to the average delay time. The average delay can thus be expressed as:

$$D'_{cb} = K_1 \left[N'/N \right] (D_{cb} + D_{imcb}) \quad [14]$$

where $K_1 = 1 + K$, and D_{cb} and D_{imcb} are given in [12] and [13]. Numerical studies have shown that for the CB with $Q = 0$ the continuous form of [14] usually gives the same results as the discrete form. Therefore we shall replace $[N'/N]N$ by N' . Finally, using equation 7 with $K = 4$ gives:

$$D'_{cb} = K_1 N' [A_0 + D_{imcb} (4B+Q)/N_p] \quad [15]$$

To minimize D'_{cb} , $4B + Q$ should be minimized. With $Q = 0$, this means that D'_{cb} is minimized when $B = 1$. Notice that D'_{cb} is directly proportional to N' , and decreases to a minimum value as the number of pins N increases. Consider next the typical parameter values given in Table 1. For N large (i.e. ≥ 64), $Q = 0$ and $B = 1$, the average delay can be approximated as:

$$D'_{cb} = 6.2N' \text{ nsec} \quad [16]$$

Banyan Network

The average delay through the BA network is given by equation 4. The value of D_{cb} is known from [12] and we assign a value to D_{imcb} that is proportional to the average path delay through the network. The only remaining quantity to determine is the value for D_{imba} . The development follows that presented for the CB. In this case however the separation between switch modules in the BA is not constant, and C_{path} will vary according to the banyan level. Since the number of levels required for a specific configuration is not known a priori, the inclusion of a variable for C_{path} complicates the delay computation. The last level has the longest path (3 inches) and therefore the maximum capacitance. The capacitance of a typical printed circuit path is approximately 1 pf/inch thus the delay in driving this longest path is:

$$D_{imba} = t_e \log_e ((2C_{pin} + S)/C_g) \quad [17]$$

By decreasing the pin driver area the banyan level decreases this value applies to all levels.

The average delay through the banyan network can now be expressed as: [18]

$$D'_{BA} = K_1 \left[\log_{N'} \right] [NA_0 + t_e \log_e ((2C_{pin} + S)/C_g)]$$

The continuous version of this equation is a poor approximation to the discrete version, thus only the discrete will be used. Using the values from Table 1 the banyan delay can be expressed as:

$$D'_{BA} = 6.17 \left[\log_{N'} \right] (N + 1.78) \quad [19]$$

The discrete relations for the CB and the BA delays were solved using optimal values of N and B , and the delays obtained as a function of parameters N , N' and network type T' . The banyan delay is consistently smaller than the crossbar for networks of reasonable size.

Chip Count-Time Product Minimization

The chip count-time product P , can be obtained by multiplying the appropriate equations given previously. Earlier discussion indicated that for reasonable size networks, both chip count and delay were minimized in the CB case with $B = 1$. Consequently the product is also minimized with this choice (for $N' \geq 64$, $B' \geq 16$).

For the BA, the situation is more complex and a computer search for the optimum B and N values must be undertaken. Consider the case of $Q = 0$ and $N' = 512$. Table 2 shows the values of N, B which optimize the number of chips, the delay, and chip count-time product. The B and N values required for minimizing P fall between those needed for minimization of the chip count and delay measures by themselves. The count minimization is achieved by attempting to place as large a network as possible on a given chip. Delay minimization is achieved by balancing the delays associated with the module network and the delays associated with increasing the number of levels in the overall network. In this case placing as large a network on a chip as possible is not the best strategy from a delay point of view. Note that this analysis does not consider delays associated with network blocking which can have a significant effect in a saturated network.

Values for N and B which minimized P were obtained for both network types over a range of N', N and Q values (Figure 7). As expected P increases with increasing N' and increasing Q , and decreases with increasing N . Once again the banyan does better than the crossbar on this overall performance measure.

Summary and Conclusions

This paper concerned the design of multiple processor interconnection networks. Models for both the banyan and crossbar networks (T') were developed and arrangements yielding minimum: number of chips, average delay through the network, and product of number of chips and delay, were presented. The results show that for the crossbar a bit slice approach ($B = 1$) produces the optimum arrangement, while for the banyan the optimum is achieved with multiple bits per module. The impact of the number of control

lines on chip count, delay and product were also modelled.

The analysis presented made a number of assumptions whose effects are being further investigated. In particular the role of blocking in the banyan case, the potential gain which would accrue from a pipelined design, and the problem of synchronization between network planes is being studied.

Parameter	Symbol	Units	Typical Value
minimum feature size	$\lambda_{\min}$	μm	3
minimum gate area	$A_{\min} = 4\lambda^2$	$(\mu m)^2$	36
gate capacitance	C_g	pf	1.4×10^{-2}
switch module pin cap.	C_{pin}	pf	5
transit time	τ	nsec	0.5
NOR gate logic levels per crossgate	m	—	2
NOR gate fanout	f	—	2
Metal path cap. to transistor gate cap. ratio (switch module)	α_{CB}	—	0.1
guard region multiplier	K_s	—	0.1
printed circuit path cap.	C_{path}	pf	lpf/inch
length of longest BA path	S	inches	12

TABLE 1: TIME DELAY PARAMETERS

	N	B	CHIP DELAY COUNT (nsec)	CDP (*10 ³)
N = 60				
P COUNT MINIMIZATION	30	1	576	392
DELAY MINIMIZATION	5	6	1236	168
PRODUCT MINIMIZATION	10	3	936	218
N = 90				
P COUNT MINIMIZATION	45	1	384	578
DELAY MINIMIZATION	5	8	824	168
PRODUCT MINIMIZATION	11	4	564	237
N = 120				
P COUNT MINIMIZATION	60	1	288	763
DELAY MINIMIZATION	5	11	824	168
PRODUCT MINIMIZATION	10	6	468	218

TABLE 2: BANYAN NETWORK MINIMIZATION RESULTS
(N' = 512, Q = 0, B' = 16)
(CDP: Count Delay Product)

References	
1.	Swan, R.J., et. al., "Cm*-A Modular Multi-Micro-processor", Proc. NCC (1977).
2.	Dennis, J.B., and Misunas, D.P., "A preliminary architecture for a basic data-flow processor", Proc. 2nd Ann. Symp. on Comp. Arch. (Dec. 1974).
3.	Sejnowski, M.C., et. al., "An overview of the Texas Reconfigurable Computer", Proc. AFIPS Nat. Comp. Conf. (1980).
4.	Sullivan, H. and Baskow, T.R., "A Large Scale, Homogeneous, Fully Distributed Parallel Machine

I", Proc. 4th Ann. Symp. on Computer Architecture (March 1977).

5. Benes, Y.E., Mathematical Theory of Connecting Networks and Telephone Traffic, Academic Press, New York (1965).
6. Siegel, H.J.; McMillen, R.J., and Mueller, P.T., Jr., "A Survey of interconnection methods for reconfigurable parallel processing systems", Proc. 1979 Nat. Comp. Conf. (June 1979).
7. Anderson, G.A., and Jensen, E.D., "Computer interconnection structures: taxonomy, characteristics, and examples", ACM Comp. Sur. 7, 4 (Dec. 1975).
8. Thurber, K.J., "Interconnection networks - a survey and assessment", Nat. Comp. Conf. (May 1974).
9. Franklin, M.A.; Kahn, S.A. and Stucki, M.J., "Design Issues in the Development of a Modular Multiprocessor Communications Network", Proc. of the Annual Symposium on Comp. Architecture (April 1979).
10. Franklin, M.A., "VLSI Performance Comparison of Banyan and Crossbar Communications Networks", IEEE Trans. on Comp. C-30,4 (April 1981).
11. Franklin, M.A. and Wann, D.F., "Word Inconsistency in Partitioned VLSI Interconnection Networks", Center for Computer Systems Design, Washington University., St. Louis, Internal Report #81-101 (May 1981).
12. Goke, L.R. and Lipoviski, G.J., "Banyan Networks for Partitioning Multiprocessor Systems", The First Ann. Symp. on Comp. Arch., University of Florida, Gainesville, Florida (1973).
13. Lawrie, D.H., "Access and Alignment of Data in an Array Processor", IEEE Trans. on Comp., Vol. C-24, No. 12 (Dec. 1975).
14. Pease, M.C., "The indirect binary n-cube micro-processor array", IEEE Trans. Comp. C-26,5 (May 1977).
15. Mead, C. and Conway, L., Introduction to VLSI Systems, Addison-Wesley Pub. Co. Reading, MA (1980).

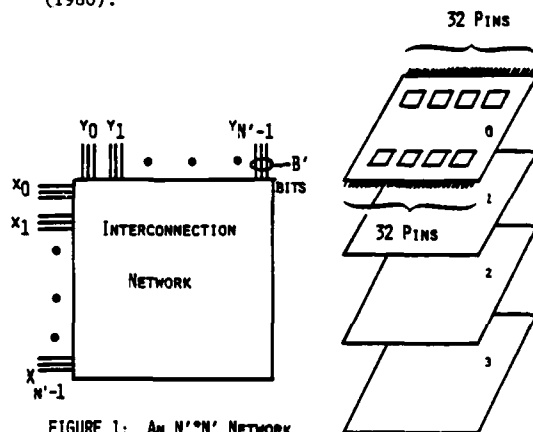


FIGURE 1: AN N' x N' NETWORK

FIGURE 3: A FOUR PLANE
16x16x8 NETWORK

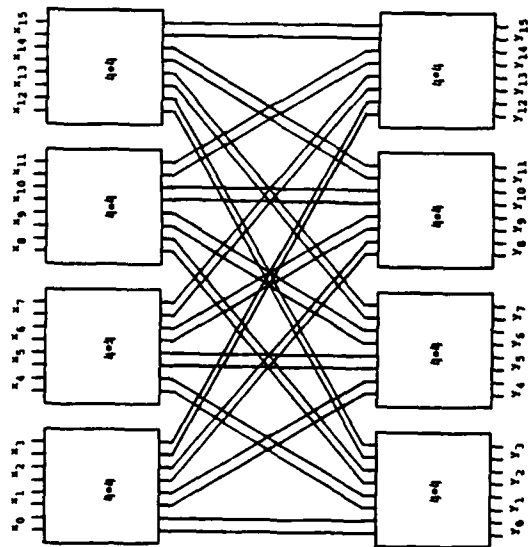


FIGURE 2: DECOMPOSITION OF A 16x16 NETWORK (USING 4x4x2 NETWORK CHIPS (ONE PLANE OF FOUR SHOWN. CONTROL AND POWER PINS NOT SHOWN)).

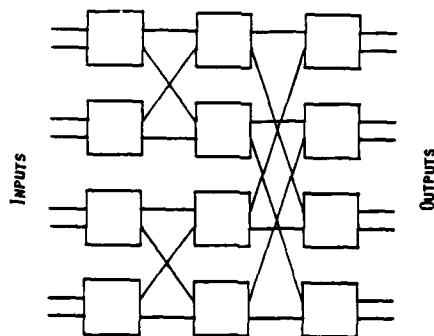


FIGURE 5: AN 8x8 NETWORK COMPOSED OF 2x2x1 CHIP COMPONENTS ARRANGED IN A BANYAN CONFIGURATION

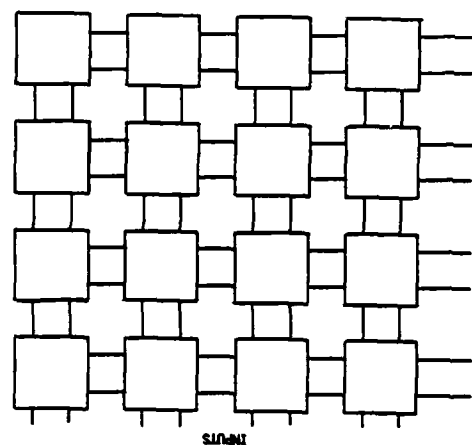


FIGURE 4: AN 8x8 NETWORK COMPOSED OF 2x2x1 CHIPS

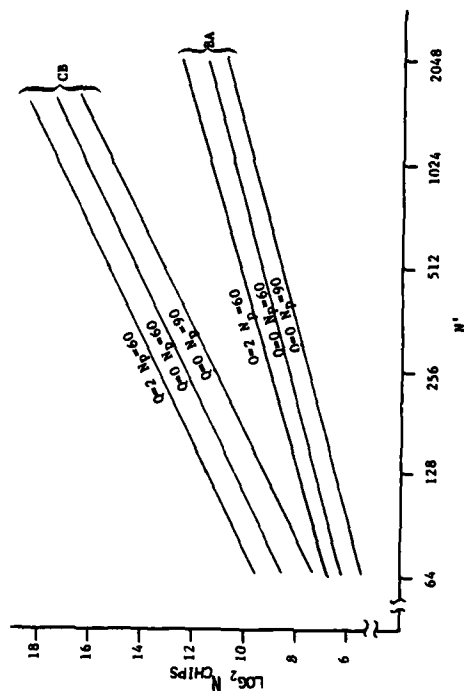


FIGURE 6: NUMBER OF CHIPS, N_{CHIPS} , VERSUS NETWORK SIZE N'

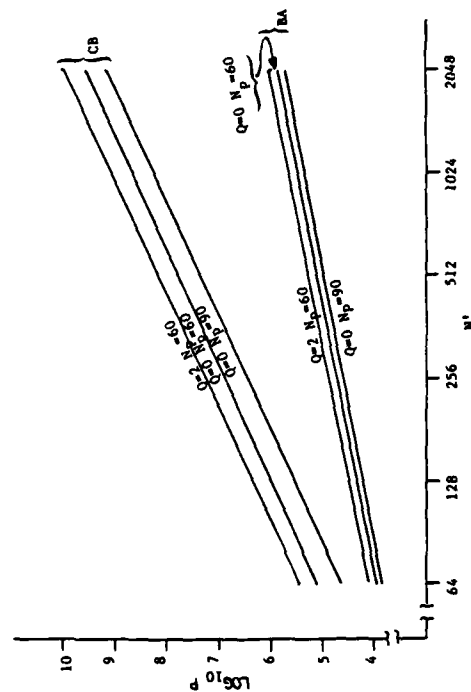


FIGURE 7: PERFORMANCE MEASURE, P , VERSUS NETWORK SIZE N'

WORD INCONSISTENCY IN PARTITIONED VLSI
INTERCONNECTION NETWORKS*

Mark A. Franklin, Donald F. Wann,
and William J. Thomas
Department of Electrical Engineering
Washington University
St. Louis, MO 63130

KEY WORDS

Interconnection Networks, Connecting Networks, VLSI, Multiprocessor
Architecture, Partitioning of Interconnection Networks, Partitioning
of VLSI Networks, Network Control

*This work was supported in part by NSF Grant MCS-78-20731, ONR Contract
N0014-80-C-0761 and NCHSR under Grant HS03792

WORD INCONSISTENCY IN PARTITIONED VLSI INTERCONNECTION NETWORKS

M.A. Franklin, D.F. Wann, and W.J. Thomas
Washington University
St. Louis, MO 63130

1.0 INTRODUCTION

Large, closely coupled multiprocessor systems typically require the presence of interconnection networks to provide for high bandwidth communications paths between processors. As the size of the systems under consideration has grown, the importance of the design of the interconnection network has become more apparent, and a number of studies have been undertaken to characterize such networks both from functional and VLSI implementation viewpoints (1,2,3,4,5). Due principally to pin constraints, such networks when implemented via VLSI methods require partitioning into multiple chips. In general a square $N' \times N' \times B'$ (N' inputs, N' outputs, B' bit wide data path) network (Figure 1) can be partitioned into multiple chips each of size $N \times N \times B$ ($N \leq N'$, $B \leq B'$) which can be interconnected to form the desired primed network. Figure 2 illustrates the $Z = \lceil B'/B \rceil$ planes needed to implement an $N' \times N' \times B'$ network. Given a performance criteria the optimum B and N can be shown to be a function of the network type (e.g., omega, banyan, crossbar), the form of the control structure, and the pin constraints. For example, if it is desired to minimize the number of chips and an incremental crossbar architecture of the type shown in Figure 1 is used both between and within the network chips, reference (6) demonstrates that for many cases the minimum is achieved by selecting $B = 1$. In this case the architecture becomes the common bit slice.

This paper focuses on an important problem which arises when a network is partitioned across the bits in a data word. Consider a particular source, S_i whose B' bits are partitioned into Z planes. Thus S_i can be represented as

$$S_i = \{S_i^1, S_i^2, \dots, S_i^p, \dots, S_i^Z\} \quad 1 \leq i \leq N' \quad [1]$$

where the superscript identifies the plane. A specific plane, p , then can interconnect only the bits from the p th partition of each source, (S_i^p) to the p th partition of each destination, (D_k^p) . This is illustrated in Figure 3. Since the interconnection network supports concurrent transactions, two sources, say S_i and S_j , may make simultaneous requests to be connected to, say D_k . The arbitration of these requests, and the appropriate path selection through the network, can be handled either on a central basis (i.e., individual crosspoints controlled from one central location) or on a distributed basis (i.e., each crosspoint making its own decision). Because of reliability and extendability considerations - of particular importance in VLSI - a modular

decentralized control structure has significant realization and perhaps performance advantages. Unfortunately, if a distributed control arbitration arrangement is used a non-homogeneous word can be received. To understand this, suppose that S_i and S_j are both requesting a path to D_k at about the same time. It is possible that a path from S_i to D_k will be established on chip p (S_i^p to D_k^p) while on chip q a path from S_j to D_k will be established (S_j^q to D_k^q). This is illustrated for the case where $B' = 16$, $B = 1$ in Figure 4 in which S_j captures the path of plane 14. The received D_k is connected to

$$D_k = \{S_i^1, S_i^2, \dots, S_i^{13}, S_j^{14}, S_i^{15}, S_i^{16}\} \quad [2]$$

and an inconsistency occurs at plane 14. The path from S_i^{14} to D_k^{14} is blocked because of the connection to S_j^{14} .

Notice that there is a switching module at which requests from S_i and S_j intersect, and at which an arbitration may occur. Neglecting the arbitration uncertainty, which is a small effect (7,8), the path to the destination is awarded to the request that arrives first. But a source request is divided into Z requests, one for each plane. Therefore, even if one of the sources makes a request prior to another source, the Z plane level requests may not arrive at their respective arbitration modules first on all planes. This can happen because the propagation delay along a path is not a constant from plane to plane. It is thus possible that the propagation delay from S_i to the arbitration module on plane p will be greater than the propagation delay from S_j , while on plane q the propagation delay from S_i will be less than from S_j . This problem is discussed in more detail later. Thus a received B' bit word can contain a nonhomogeneous set of bits if the interconnection network utilizes a distributed methodology for establishment of the paths through each of the partitions, rather than a single path controller assignment strategy. If this local distributed control structure is selected (because of its other desirable features) then some mechanism must be included to detect that each of the paths established to D_k through the partitions are from the same source. We define a word received at the destination as being inconsistent if all its B' bits are not from the same source.

In this paper we assume that pin limitations and minimization of the total number of chips dictate partitioning across the B' bits and that reliability and modularity considerations require decentralized control. We then investigate two problems:

- 1) What types of decentralized control allow modularity and also permit the detection of inconsistent words?
- 2) Given the statistical properties of the requests from the N' sources, what is the probability that an inconsistent word will be received by a destination?

A solution to the first problem permits us to implement locally controlled networks that can detect an inconsistent connection; if inconsistent a retry can be initiated. A solution to the second problem allows us to estimate how often an inconsistent interconnection is likely to occur, and thus permits us to judge if the benefits of decentralized control offset its limitations.

2.0 BIT PARTITIONING CONTROL STRATEGIES

One economical method for realizing distributed control is to carry it out via the same pathways that the data occupies. The message transmitted by a source therefore would contain, in sequence, the following: a) a header to request the path and identify the destination, b) data bits comprising the information to be sent to the destination, and c) an end of message word to indicate the end of transmission. In order to place the network in a state to receive the next path request by the next source, the end of message would also initiate a path clearing operation.

Decentralized control implies that there is no global sequencing, thus communication between chips is accomplished via some form of self-timed protocol. Within a chip, however, the transactions may be carried out with a local clock or with a self-timed methodology. Although, in general, a self-timed protocol requires more wires (and hence pins) than a clocked scheme, if properly designed it has the advantages of being able to be easily expanded (e.g., step and repeat) without a recomputation of timing constraints, and usually also has the highest bandwidth.

Here we describe a switching module and communications signalling protocol that contains provisions for:

- path establishment
- transfer of data from source to destination
- detection of a blocked path
- indication of end of transmission with path clearing
- initiation of retry for path establishment
- detection of word inconsistency

2.1 Switch Module Characteristics

Figure 5 illustrates the primitive switch module and associated signals that

are used to form the self-timed crossbar interconnection network. These modules are utilized as depicted in Figure 1. Each module has 4 connections per side for a total of 16 connections per module. The sides are identified via subscripts, L = Left, R = Right, T = Top, and B = Bottom. The left and top side connections correspond to communication links with a data source, while the right and bottom side connections correspond to communication links to a data destination. The left side of a switch module contains a path R_L^1 that is used if a binary one is to be sent to the module, and a path R_L^0 that is used if a binary zero is to be sent to the module. Likewise for the top side. To achieve the self-timed operation an acknowledgment of the receipt of a data bit must be returned to the data source and this is performed by the path A_L for the left side and the path A_T for the top side. When the data source receives a signal from the A line it indicates to it that the last transmitted data bit has been successfully received by the destination. All signalling events are encoded as changes, thus a logic change in a line represents the occurrence of a signal on that line. For example, if the line R_L^1 changes from a 0 to a 1 or vice versa it represents the transmission of a one to the switching module. Since no return to zero is required to achieve a quiescent condition when using such transition encoding, the number of transitions per bit can be minimized thus maximizing the data rate. In addition to the R_L^1 , R_L^0 and A signals, on each side of the module there is a negative acknowledge signal, NA, whose purpose will be described later.

2.1.1 Switch States

The switch module can be considered as having five data connection states. These are illustrated in Figure 6 and correspond to: the horizontal and vertical paths inactive (State I); the horizontal path active (State H); the vertical path active (State V); the horizontal and vertical paths both active (State HV); and the left side to bottom side corner path active (State C). Notice that a data connection and switch state correspond to data from the top side to the right side is not needed in the crossbar interconnection network.

2.2 Path Establishment

The goal is to establish a circuit switched path from source S_i to destination D_k . We now show how S_i requests that such a path be established.

Assume that all modules are in their inactive state and let the row, column identification of sources and destinations correspond to that shown in Figure 3. In order to establish a path from S_i to D_k it is necessary that: all switch modules in row i from column i to column k-i be set to their

horizontal active state, the module at the intersection of row i and column k be set to its corner state, and all modules in column k from row $k-i$ to row i be set to their vertical state. This will complete the path. To achieve these states, the source sends a header word (which is merely a special bit string) of length k into the network from its connection at port i . Thus, if the destination is D_6 , the bit string will be of length 6. This bit string is formatted such that its first $k-i$ bits are zeros and its k th bit is a one. For example, to request a path to D_6 the sequence 000001 would be sent by source S_i by making proper changes on lines R^0 and R^1 .

The module response to header bits arriving from the left is as follows: The module, which has an inactive horizontal path (it is in the I state) examines the first bit that it receives. If this bit is a zero the module changes to the horizontal state, absorbs (e.g., does not pass on) this first bit, and then generates an acknowledgment on A_L to indicate receipt of the first bit. Subsequent header bits arriving at this module find it in the horizontal active state and these bits (either zero or one) are merely passed on to R^0_R or R^1_R .

If the first header bit that an inactive module receives is a one, then the module enters the corner state and passes this bit out its bottom side via R^1_B . In the example under consideration, the first five modules in row i would enter the horizontal state and the sixth module would enter the corner state. Thus the proper column has been found and the data connections to this point have been established. The module immediately below the corner module will now receive a one via a change on its R^1_T line. A module which is in the inactive vertical state that receives a one from the top enters the vertically active state and passes this one onto the module beneath it via R^1_B . (Note that a module will never receive a zero as its first bit from the top.) This process is repeated along column k . When destination k receives its first one it knows that this is the last bit in the header word. The destination then produces a signal on line A_B of the first module in column k and this signal passes backward along the path through column k via A_B to A_T of each module, then at module i, k from A_B to A_L , along row i from A_R to A_L , eventually reaching the source S_i . Upon receipt of this acknowledgment to the k th bit of its header word, the source knows that the path to the destination has been completed and that the destination is ready to accept data. (If the destination were not ready, it would have delayed issuing the acknowledge). The source may now transmit data.

2.3 Transmission of Data

The source transmits data bits by sending events on R^1 to R^0 to the module

in row i column l . This module passes the bits on via the established pathway to the destination. Each data bit is acknowledged by the destination back to the source via the acknowledge lines.

2.4 Blocked Path

The above description assumes that there are no other paths in use at the time that the S_i to D_k connection is requested. If another source has previously established a connection to D_k then a provision must be introduced into the interconnection network to indicate to the source S_i that a blocked, or unavailable path has been encountered. We adopt this procedure rather than merely "waiting" at the module because, as will be discussed later, the waiting method can produce a deadlock condition. Thus, it is necessary to transmit a signal back from the blockage point to S_i . This means that in general S_i will either receive an acknowledge signal (path completed) or a negative acknowledge (path blocked), hence, this requires one bit of information. To transmit these two conditions in a delay insensitive manner requires the addition of a fourth line to the module configuration. This is the NA signal line illustrated in Figure 5. This signal is used during the path establishment phase of the protocol and, as will be described in a later section, is also used during the end of data transmission phase.

Observe that when we permit concurrent paths to be established in the cross-bar network the definition of inactivity must be interpreted to include just the inactivity of the path that is required. For example, suppose a header bit enters a module in which the vertical path is active and the horizontal path is inactive. If the header bit is a zero then the new state of the module becomes horizontal active and vertical active (State HV) and an A is returned to the source. If the header bit is a one (requesting the corner) the module state remains the same (State V) and a NA is returned to the source to indicate that the path is not available. In both cases the A or the NA move back along the partially established path to the source. However, as the NA signal moves through a module it must clear each module by placing it in the inactive horizontal, inactive vertical, or inactive horizontal and inactive vertical conditions, thereby making the path available for the next request.

2.5 End of Transmission

The protocol described so far allows for the establishment of a path, the transmission of data over a path, the detection of a blocked path, the automatic resetting of individual module states along a blocked path, all in a self-timed environment. Now we explain how a path that has been successfully

established between source S_i and destination D_k can be relinquished and placed in the inactive state at the conclusion of the data transfer.

The data transmission is arranged so that a special bit stream combination (e.g., a special character) is reserved to indicate to the destination an End of Transmission. For example, if ASCII characters were being sent over the interconnection path then the stream 00000001 (hexadecimal 04) might be used as an End of Transmission indicator. The destination continuously decodes the received data bit stream and provides an acknowledge signal after every bit. When the destination detects an EOT character, instead of supplying an A signal, the destination produces a NA signal. This signal then travels back along the path, placing each switch module in its appropriate inactive state (H, V or I) and arrives at the source indicating that the path is now clear. Note that this action of the NA on the individual modules is identical to its action when a blocked signal path is encountered, thus, no additional logic is required by the module to implement this function.

2.6 State Diagram of Switch Module

The specific state changes that an individual switch module must make as a function of the various input events and the output events that it must generate are summarized in the state diagram shown in Figure 7.

2.7 Conflict Resolution

The state transitions shown in Figure 7 are predicated on the assumption that requests to a specific module to establish a path from left to bottom and from top to bottom do not arrive at the module simultaneously. Notice that since the control is distributed such a request condition can occur and will result in a conflict as to which path will be completed. To accomodate these concurrent requests in a reliable manner, it is necessary to include an arbitration unit within each of the primitive switching modules. The placement of this unit is shown in Figure 8. There are three cases in which the behavior of the module depends upon the order in which the concurrent input signals arrive. These cases are illustrated in Figure 9. The first case is the situation described above in which the module is in the inactive state (I) and requests arrive from the left and top. The results of these concurrent requests are also illustrated in Figure 9. Suppose a logic one header bit arrives from source S_y at the top of a module and from source S_x at the left of a module simultaneously. Then the arbitration unit selects one of these requests, say S_y , and passes the request on. The other request, from S_x , is not granted and a NA signal is returned to S_x . Note that in contrast to a common class of arbitration units,

this arbitration unit does not hold the request from S_x for later action (e.g., for action when the switch path becomes free), but generates a NA immediately, thereby allowing the source S_x to make its own retry decision. The second and third cases occur when one path in the module is being cleared via an NA signal and part of this path is being requested. Figure 9 shows the two arbitration results. Construction and performance details of such arbitration units have been described in the literature.

2.8 Retry Protocol for Blocked Path

When a source requesting a connection to a destination receives a NA after sending the one in its header word, it knows that the path is blocked. It may then try to reestablish the path immediately by retransmitting the header word, or it may employ some backoff algorithm that generates a random delay before the next retry is issued.

2.9 Retry Protocol for Word Inconsistency

When a source attempts to establish a path to a destination it should receive an A signal from each of the Z planes. If the source receives one or more NA signals it recognizes that the desired pathway is blocked; thus, the other pathways, even though they have been established, are not useful. In order to retry the connection the source sends an EOT character over each of the pathways on which it received an A; this will cause the destination to generate an NA, thereby clearing them. When the source detects all NA signals from all Z planes, the path is clear. The return of a mixture of A and NA signals to a source indicates that another source is requesting the same destination and it must have also received a mixture. Also, note that this behavior would result in a deadlock situation in which no retry could be started if the waiting technique were adopted rather than this path clearing method. However, in the path clearing method if both sources issue a retry the probability is high that another blocking will occur. Therefore, backoff techniques (such as used in the Ether net) may also be adopted.

2.10 Pin Minimization

Notice that if the network is arranged in a bit slice the self-timed switching module requires four pins per port. In contrast, a central control, although presenting problems such as clock skew, only requires one pin per port. Thus, the pin efficiency of the self-timed module is very poor. What is the minimum number of pins that will still maintain the self-timed discipline? We have found that it is possible to convey the same information across one of the module sides with three bidirectional paths rather than the four unidirectional ones. The technique for this encoding is described next.

To guarantee a self-timed discipline we recognize that when a signal arrives on a line we cannot send the next signal on this same line since two changes on the same line (without a change on a second line) violate the self-timed rules. Let us postulate a module that has three bidirectional lines as shown in Figure 10, and assume that the last change occurred on line Z and represented an acknowledgment from the destination. Then the next signal will be from the source and it has two lines (X and Y) on which it may send R^1 or R^0 . Suppose we let the source send R^1 by changing line X. When the destination replies it now has two lines, W and Z, that it can use to indicate A or NA. Notice that both the source and destination always have two lines, other than the line on which the last change occurred, on which they can send their one bit of information. But the coding, in contrast to the four line unidirectional module, is not constant. During one transfer a change on X may represent an A while during the next transfer a change on Y may represent an A. It is the responsibility of the source and destination to keep track of this variation in coding in order to determine which line may be changed next. Although the network could be constructed entirely with these bidirectional line modules, this complicates the module design. Another approach that we are exploring is to merely use this technique at the interface between the crossbar network and the ports. This is illustrated in Figure 11. It does require the design of a second type of module, but the modules in column 1 and row 1 are usually special since they must receive and drive signals to the external pins.

This technique reduces the pin requirements for the self-timed module to three per port.

2.11 Nearly Self-Timed Module

Although we cannot reduce the pin count further and maintain the completely self-timed nature of the network, it is possible to create a network that has the minimum number of pins (1 per port) and have local control of the path establishment and data transfer. This is accomplished by means of a local clock in each chip. This local clock is distributed only through the chip (there is no clock signal or relative timing requirements between chips) and the communication between modules on a chip or between chips is via the standard UART type of serial data transmission. A module of this type is illustrated in Figure 12. Here the interconnection pathways are bidirectional and module communication is via serially encoded characters. After one character is sent the destination module acknowledges via the return of a serially encoded character thus providing handshaking. This is a very simple yet elegant technique and we are examining it further.

3.0 PROBABILITY OF RETRY

In Section 2.0 a scheme was presented which permitted use of a distributed routing control, bit-sliced architecture for interconnection networks. As indicated, propagation delay differences may occur across the bit slices resulting in different interconnections being established on the various network planes. While such inconsistencies can be detected and corrected through retry procedures, it is important to quantify how often such retries are encountered so that performance degradation can be evaluated. Therefore, the goal of the analysis presented here will be to evaluate the probability that an inconsistent word will develop for a source S_i upon requesting a destination D_k , given the statistical properties of the requests from the N sources and a simple propagation delay model for the interconnection network. For the purpose of this analysis, we will assume that the request process for each source has a Poisson distribution with rate λ , and that these processes are independent. We shall also assume that the requests are uniformly distributed across the destinations.

In the next section, a simple example is provided to illustrate the affect of propagation delay variations across the planes of a bit partitioned network. Following this, an expression is derived for the probability of an inconsistent word. Then a model is presented to account for propagation delay variability. In the last section, the probability of an inconsistent word is evaluated as a function of the number of network ports, bit plane partitions, and request arrival rate for a pipelined crossbar structure.

3.1 Illustration of the Affects of Path Delay Variability

There are two factors that influence the probability that an inconsistent word will develop: the variability in propagation delay of an interconnection path from plane to plane, and the probability that an arbitration module will not award the path to the first request. For large networks this latter factor can be shown to have negligible effect on the probability of retry and it is reasonable to assume that one has a perfect arbitration module. To illustrate the affect of the variability of the path propagation delay across the planes consider the following example:

Let S_i and S_j make simultaneous requests for a path to D_k . Let the delay, δ_i^p , along the path from S_i to the arbiter at row j column k in

plane p be a constant, K_i for all planes, that is:

$$\delta_i^p = K_i \quad \text{for } p = 1, 2, \dots, z.$$

Let the delay from S_j to the arbitration module be a constant for all but one of the planes, that is:

$$\delta_j^p = \begin{cases} K_j & \text{for } p = 1, 2, \dots, z-1 \\ K_j + \Delta & \text{for } p = z \end{cases}$$

where $\Delta \geq 0$.

The received message then can be expressed as

$$D_k = \begin{cases} \{S_j^1, \dots, S_j^z\} & \text{for } K_j + \Delta < K_i \\ \{S_i^1, \dots, S_i^z\} & \text{for } K_i < K_j \\ \{S_i^1, \dots, S_i^{z-1}, S_j^z\} & \text{otherwise} \end{cases}$$

Observe that for a range of propagation delays there will always be an inconsistent connection. If one wanted to tailor this set of paths from S_i to D_k and from S_j to D_k such an inconsistency could not occur, independent of the time separation between the source requests, it could only be accomplished by constructing each of the Z paths associated with S_i to have a constant delay, K_i , and each of the Z paths associated with S_j to have a constant delay K_j . Obviously, this is not possible. Therefore, it is important to quantify how often such inconsistencies will arise.

3.2 Probability of an Inconsistent Word

The goal of this section is to derive an expression for the probability that an inconsistent word will develop when a source makes a request for a destination. Due to the combinatorial complexity of this problem (especially when N' is large), we will not attempt to obtain an exact solution. Indeed, our overall goal is simply to show that the probability of a word inconsistency is small under typical operating conditions. To this end, we will only

determine an expression which represents an upper bound for this probability. Several models have been investigated and differ only in the extent to which approximations are used, and hence, in the tightness of their bounds. The highest level model, to be described here, provides the loosest bound, but is (to some extent) the asymptote that the other models approach as the number of pathwidth partitions become large (e.g. $Z \geq 16$).

3.2.1 Definition of PIW

Let us formally define what is meant by "the development of an inconsistent word". We distinguish three possible outcomes which can occur when a source S_i requests a destination D_k : S_i may capture 1) all, 2) some, or 3) none of the network path partitions required for communication with D_k . In the event that only some of the partitions are captured, we say an inconsistent word has developed. Thus we have:

$$\begin{aligned} \text{PIW} &\equiv \text{PROB}\{S_i \text{ must retry due to an inconsistent word}\} \\ &= \text{PROB}\{S_i \text{ captures some (but not all) of the network planes}\} \end{aligned}$$

Since these three outcomes are disjoint and describe the entire outcome space, then for computational purposes, we may evaluate PIW in terms of these other two events. Therefore, let:

$$\begin{aligned} \text{PIW} = 1 - &\left[\text{PROB}\{S_i \text{ captures all of the planes}\} \right. \\ &\left. + \text{PROB}\{S_i \text{ captures none of the planes}\} \right] \end{aligned}$$

For the sake of a shorter notation, this becomes:

$$\text{PIW} = 1 - [P\{\text{ALL}\} + P\{\text{NONE}\}]$$

In the next section, we investigate how $P\{\text{ALL}\}$ is determined.

3.2.2 Capture Process for Source S_i

To understand what must occur if S_i is to capture all of the partitions (planes), we refer to Figure 3. S_i must capture all of the switching modules

in the dotted path to D_k , and it must do this for all of the Z planes. Since the horizontal path through a module is captured without contention, the only switch modules that S_i might experience contention at are those in the destination column. At each of these column modules, it will compete for temporary use of the switch with the module's other request line. Let us identify a module by its row (R) and column (C), e.g. MOD $\langle R, C \rangle$. Then, for MOD $\langle i, k \rangle$, this other request can come from any one of the sources "above" S_i , i.e. any S_j for which $j > i$. All of these other sources will, so to speak, "fight it out" to see which source request gets to compete for this module. For MOD $\langle j, k \rangle$ where $j < i$, this request can only come from one source, S_j .

Before we can determine the probability that S_i will capture an arbitrary switch module, we must introduce the concept of an "uncertainty interval". This is done in the next section.

3.2.3 Definition of the "Uncertainty Interval"

If two sources S_i and S_j , $j < i$ request the same destination, there are two factors that determine which of the sources captures the required path on any given plane. These are:

- 1) The relative times that the requests enter the network
- 2) The path delays between the network inputs and the arbitrating switch module (e.g. MOD $\langle j, k \rangle$)

If both of these factors are known, we can determine which source obtains the path. If only the path delays are known, we can specify the request times of S_j (relative to the request time of S_i) which result in S_j capturing the path and those times which result in S_i capturing the path. This is shown in Figure 13, where t_i represents the request time of S_i . There is ideally only one request time for S_j which results in an unpredictable outcome, although even this time is eliminated under the assumption of perfect arbiters. The unpredictable time is labelled in the figure as t_* .

Let us now presume that there is some uncertainty in the path delays to MOD $\langle j, k \rangle$. For instance, let the path delay for S_i vary between $\delta_i(\min)$ and $\delta_i(\max)$, and the path delay for S_j vary between $\delta_j(\min)$ and $\delta_j(\max)$.

If t_j represents the request time of S_j , then this request will arrive at the arbiter sometime between $t_j + \delta_j(\min)$ and $t_j + \delta_j(\max)$. We shall refer to this as S_j 's "arrival interval". Similarly, S_i 's "arrival interval" will be between $t_i + \delta(\min)$ and $t_i + \delta(\max)$. These are shown in Figure 14. From the figure, we see that as long as t_j occurs before the time labelled t_A or after the time labelled t_B , the outcome as to which source captures the path is predictable: if t_j occurs before t_A , then S_j captures the path; if t_j occurs after t_B , then S_i captures the path. If, however, t_j occurs between t_A and t_B , the "arrival intervals" for the two sources will overlap, and the outcome will be uncertain. For this reason, the composition of these request times will be referred to as " S_j 's uncertainty interval with respect to S_i ".

Although we can directly determine the values of t_A and t_B in terms of the other parameters, our interest only lies in the length of this interval. From the figure, it is clear that the length, designated as ΔT_j , is equal to the sum of the lengths of the arrival intervals.

3.2.4 Evaluation of $P\{ALL\}$

Let us now determine the probability that S_i captures $MOD\langle j, k \rangle$ (for arbitrary j less than i) on all of the planes, given that the request reaches each module. If S_j does not make a request for D_k during its "uncertainty interval with respect to S_i ", then S_i captures the module outright (i.e. with probability = 1) on all of the planes*. If, however, S_j does request D_k during the uncertainty interval, then the probability that S_i captures the module for any individual plane is $p(j)$ where $p(j)$ will be determined by the type of request process assumed and the delay characteristics of the network. Thus, the probability in this case that S_i captures the module on all of the Z planes is $p(j)^Z$. Therefore, the probability that S_i captures $MOD\langle j, k \rangle$ on all of the network planes is equal to:

$$P\{S_j \text{ doesn't req. } D_k \text{ during its uncertainty interval}\} * 1 \\ + P\{S_j \text{ does req. } D_k \text{ during its uncertainty interval}\} * p(j)^Z$$

* Recall that we are working under the assumption that the destination D_k is idle. This eliminates the possibility that S_j makes a request before the uncertainty interval.

Let us now determine the probability that S_i captures $MOD\langle i, k \rangle$ on all of the planes, given that the request reaches each module. If none of the sources above S_i request D_k during their uncertainty intervals (each with respect to S_i), then S_i captures the module outright for all of the planes. Since these sources act independently, the probability that this occurs is just the product of the probabilities that each source has of not requesting D_k during its uncertainty interval. If one (or more) of the sources does request D_k (during its uncertainty interval), then the probability that S_i captures the module for any individual plane is $p(j)$, where j corresponds with the index of the source whose request reached the module. Thus the probability in this case that S_i captures the module on all of the Z planes is $p(j)^Z$. Therefore, the probability that S_i captures $MOD\langle i, k \rangle$ on all of the network planes is equal to:

$$\prod_{j=i+1}^{N'} P\{S_j \text{ doesn't req. } D_k \text{ during UI}\} + \left\{ 1 - \prod_{j=i+1} P\{S_j \text{ doesn't req. } D_k \text{ during UI}\} \right\} * p(j)^Z$$

We now evaluate the probability that an arbitrary source S_j does not make a request for destination D_k during its uncertainty interval with respect to source S_i , recalling that we have assumed independent Poisson request processes with rate λ for all of the sources. It is easily shown that the probability that S_j does not make a request for D_k during its uncertainty interval is related to the probability that S_j does not make a request at all during its uncertainty interval as follows:

$$P\{S_j \text{ doesn't req. } D_k \text{ during UI}\} = 1 - \frac{1}{N'} \left\{ 1 - P\{S_j \text{ doesn't make a req. during UI}\} \right\}$$

Then, using the fact that S_j 's request process is Poisson, and that the length of S_j 's uncertainty interval (from section 3.2.3) is ΔT_j ,

$$\begin{aligned} P\{S_j \text{ doesn't make a request during UI}\} &= P\{N_{\Delta T_j} = 0\} \\ &= e^{-\lambda * \Delta T_j} \end{aligned}$$

Finally, the probability that S_i captures the path to D_k on all of the network planes (i.e. $P\langle ALL \rangle$) equals the product (over $j = 1$ to i) of the probabilities that S_i captures switch module $MOD\langle j, k \rangle$ on all of the planes, given that the request arrived at each module, i.e.

$$P\langle ALL \rangle = \prod_{j=1}^i P\{S_i \text{ captures } MOD\langle j, k \rangle \text{ on all planes given } S_i \text{ arrives at each module}\}$$

3.2.5 Evaluation of $P\{NONE\}$

Let us now consider $P\{NONE\}$. Due to the combinatorial complexity of this probability, an analysis similar to that of $P\{ALL\}$ can not be done. Therefore we shall simply attempt to show that this term is no larger than $P\{ALL\}$, and in fact, is significantly smaller than $P\{ALL\}$ when the number of network partitions is large.

Consider first the probability that S_i does not capture an arbitrary plane P :

$$\begin{aligned} P\{S_i \text{ doesn't capture arbitrary plane } P\} \\ &= 1 - P\{S_i \text{ does capture arbitrary plane } P\} \\ &= 1 - P\{ALL\} \Big|_{z=1} \end{aligned}$$

Now, if all of the planes operated in lock step, i.e. were totally dependent, then:

$$\begin{aligned} P\{NONE\} &= P\{S_i \text{ doesn't capture arb. plane } P\} \\ &= 1 - P\{ALL\} \Big|_{z=1} \end{aligned}$$

If, on the other hand, all of the planes were totally independent, then:

$$\begin{aligned} P\{NONE\} &= \{P\{S_i \text{ doesn't capture arb. plane } P\}\}^z \\ &= \left\{ 1 - P\{ALL\} \Big|_{z=1} \right\}^z \end{aligned}$$

Thus:

$$\left\{1-P\{ALL\}\right\}_{z=1}^z \leq P\{NONE\} \leq \left\{1-P\{ALL\}\right\}_{z=1}$$

Hence, as long as $P\{ALL\}$ is greater than 0.5, $P\{NONE\}$ will be smaller than $P\{ALL\}$. Since in actuality there is only a slight dependence between planes, $P\{NONE\}$ tends toward the left hand side of this inequality, which implies that $P\{NONE\}$ will approach zero as the number of planes increases. For this reason, we will neglect this term in the evaluation of PIW when Z is large.

3.2.6 An Approximation for PIW Given Z is Large

A large reduction in the computation and complexity of PIW can be made when Z is large (e.g. ≥ 16). For this case, we can assume that $p(j)^Z \approx 0$. Substituting this approximation into the equations of Section 3.3, we get the following result:

$$\begin{aligned} PIW &\approx 1 - P\{ALL\} \\ &\quad N' \\ &\approx 1 - \prod_{j=1}^{j=1} P\{S_j \text{ doesn't req } D_k \text{ during UI}\} \\ &\quad (j \neq 1) \end{aligned}$$

That is, $P\{ALL\}$ is simply the probability that none of the other sources request D_k during their uncertainty intervals. Since the sources are independent, this is just the product of the probabilities that each source has of not requesting D_k during UI.

3.3 Propagation Delay Model

Here we present a simple model for propagation delay based on the physical characteristics of the network. As a signal propagates along one of the network paths, it will experience two types of delay: switch module delay and interchip path delay. (The intrachip path delay is very small due to the crossbar structure and can be included in the switch module delay.) Franklin and Wann, in (6), have derived equations for these two delays. If we use typical VLSI values in these expressions and assume that a switching module has three or four levels of "steering" logic, then both of these delays

are on the order of 10 nsec. However, due to stray capacitances, misaligned masks, etc., there will be some variability in these delays. Therefore, to model these, we will consider both the interchip and module delays to be normal random variables, with means of 10 nsec. If we assume that the actual delays will be within fifty percent of the mean 95% of the time, then the corresponding standard deviation is 2.5 nsec. We shall also assume that these delays are all independent of each other for the purpose of simplification. In general, this may not be true.

3.4 PIW for a Pipelined Crossbar Network: An Example

Let us now apply the equations that we have derived, together with the propagation delay model, to determine the probability that an inconsistent word will develop for source S_i upon requesting destination D_k (given that D_k is idle). Our initial research efforts in this area have been directed toward a pipelined network rather than the circuit switched network that was described in section 2.0. For this reason, the example to be presented here will assume a pipelined approach.

The main difference between the pipelined and circuit switched approach is that, in the pipelined model, each module has memory, and data is passed locally using a handshake between modules rather than globally using a handshake which always involves the source. Path establishment is done in a pipelined fashion as well. The source, handshaking with the first module, simply pushes the routing header into the pipe, followed by data (unless it receives a blocked signal). Therefore, in this approach, the source's request will reach the first arbiter ($MOD<i,k>$) after $k-1$ module delays and $\left\lceil \frac{k}{N} \right\rceil$ interchip path delays. Neglecting the interchip path delays and applying our propagation model for the switch module delays, the time it takes the request to reach the arbiter is then the sum of $k-1$ Normal random variables, each having a mean of 10 nsec and standard deviation of 2.5 nsec. Hence, the arrival time at the arbiter will be Normally distributed with mean $10(k-1)$ nsec and standard deviation $2.5\sqrt{k-1}$ nsec.

As a first order approximation, this path delay can be specified as a Uniform random variable having a minimum and maximum value which correspond with the 95% confidence interval cutoff points of the Normal delay distribution.

That is, let

$$\delta(\min) = 10(k-1)\text{nsec} - 2(2.5)\sqrt{k-1} \text{ nsec}$$

and

$$\delta(\max) = 10(k-1)\text{nsec} + 2(2.5)\sqrt{k-1} \text{ nsec}.$$

Hence, the "arrival interval" for this source at MOD<i,k> is $4(2.5)\sqrt{k-1}$ nsec long.

We can make a similar analysis to the one above for each of the other sources to acquire their arrival intervals to the various arbitors. We can then use these to determine each source's uncertainty interval. In general, this will be :

$$\Delta T_j = 4(2.5)\left(\sqrt{k-1} + \sqrt{k-1 + |j-i|}\right) \quad \text{for all } j.$$

We can then apply this to the equations derived in section 3.2 to obtain the probability of an inconsistent word (for source S_i requesting D_k) as a function of the number of network ports, bit plane partitions, and the request arrival rate, λ . Figure 15 shows the curves obtained when source N' requests destination N' , which yields the highest probability of inconsistency for the network. Note that, for the arrival rates shown, the probability of retry is relatively small.

References

1. Swan, R.J., et. al., "Cm*-A Modular Multi-Microprocessor", Proc. NCC (1977).
2. Dennis, J.B., and Misunas, D.P., "A preliminary architecture for a basic data-flow processor", Proc. 2nd Ann. Symp. on Comp. Arch. (Dec. 1974).
3. Sejnowski, M.C., et. al, "An overview of the Texas Reconfigurable Computer", Proc. AFIPS Nat. Comp. Conf. (1980).
4. Sullivan, H. and Baskow, T.R., "A Large Scale, Homogeneous, Fully Distributed Parallel Machine I", Proc. 4th Ann. Symp. on Computer Architecture (March 1977).
5. Franklin, M.A. "VLSI Performance Comparison of Banyan and Crossbar Communications Networks", IEEE Trans. Comp. C-30, April 1981 pp 283-291
6. Franklin, M.A. and Wann, D.F. "Pin Limitations and VLSI Interconnection Network" Proc 1981 Inter. Conf. on Parallel Processing", Proc. 1981 Inter. Conf. on Parallel Processing (Aug. (1981), pp.253-258.
7. Chaney, T.J. and Molnar, C.E. "Anomalous Behavior of Synchronizer and Arbiter Circuits" IEEE Trans. on Computers, April 1973 pp 421-422.
8. Couranz, G.R. and Wann, D.F. "Theoretical and Experimental Behavior of Synchronizers Operating in the Metastable Region" IEEE Trans. on Computers June, 1975 pp 604-616.

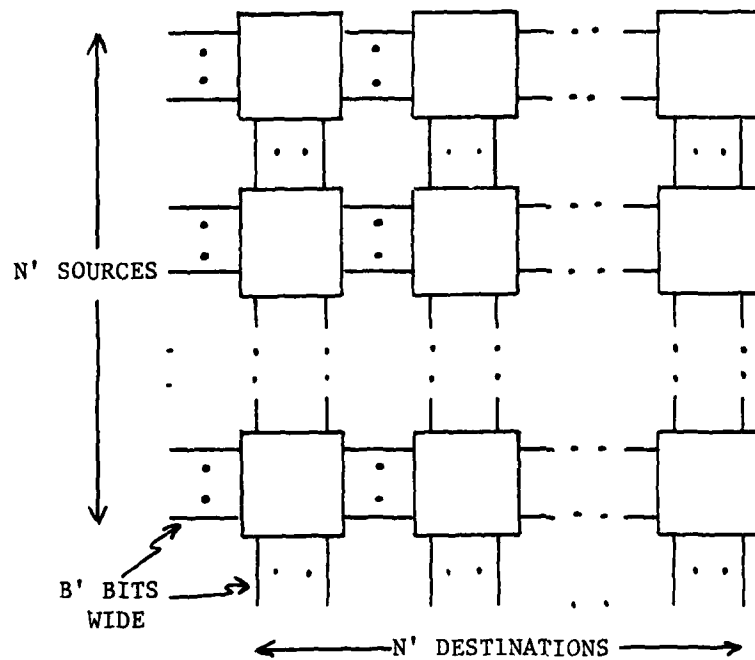


FIGURE 1. AN $N' \times N' \times B'$ INTERCONNECTION NETWORK

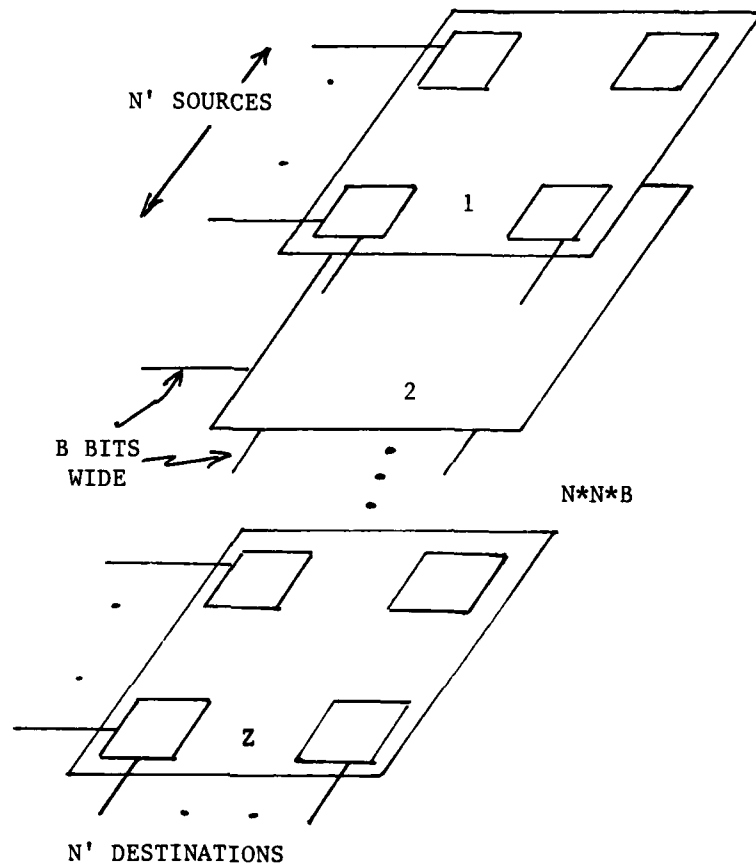


FIGURE 2. PARTITIONING OF $N' \times N' \times B$ NETWORK INTO Z PLANES USING $N \times N \times B$ CHIPS

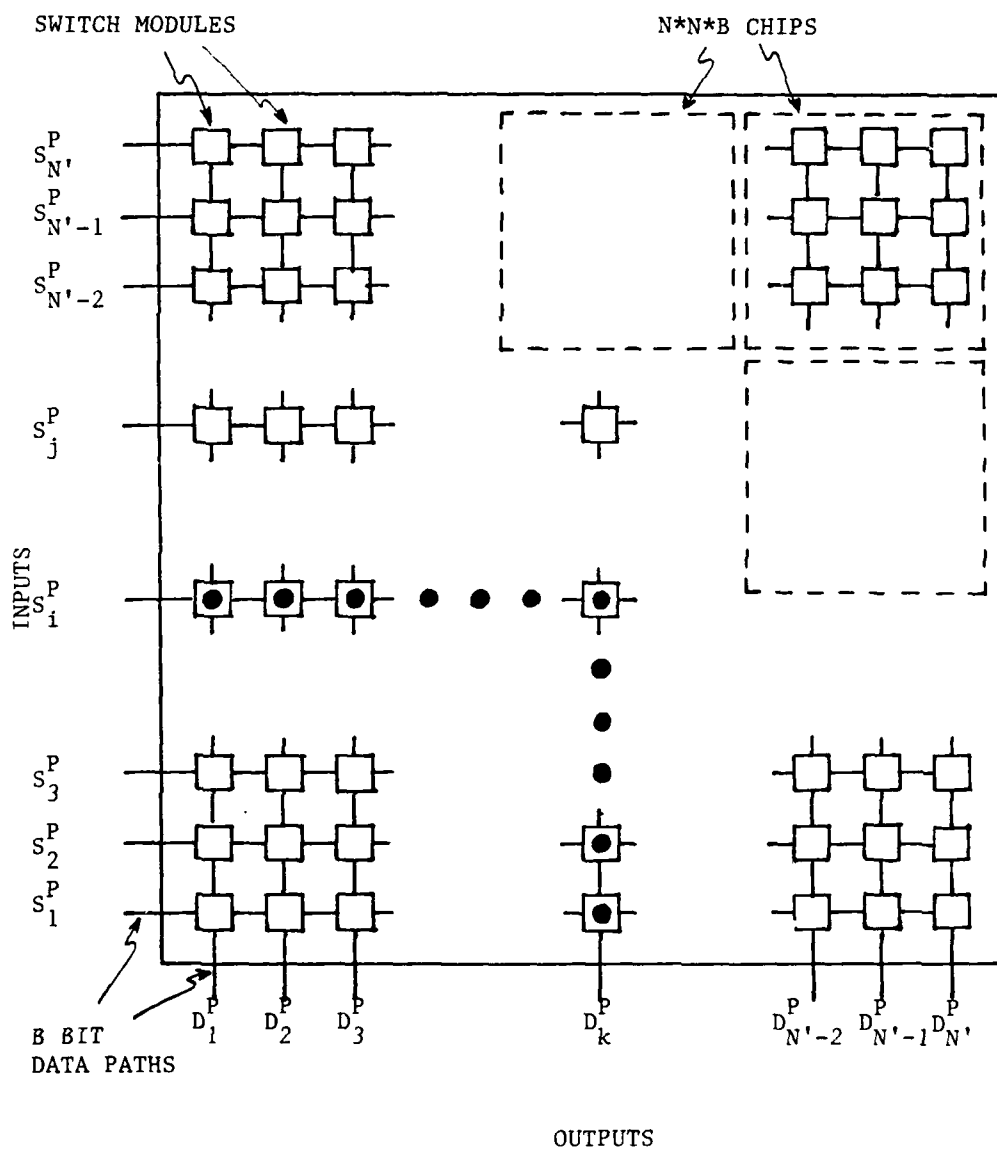


FIGURE 3. INTERCONNECTION NETWORK FOR P^{TH} PLANE
 USING $N*N*B$ CHIPS

AD-A104 747

WASHINGTON UNIV ST LOUIS MO CENTER FOR COMPUTER SYST--ETC F/6 9/2
VLSI BASED MULTIPROCESSOR COMMUNICATIONS NETWORKS.(U)
SEP 81 M A FRANKLIN, D F WANN

N00014-80-C-0761

NL

UNCLASSIFIED

2 * 2

AD-A104 747

■

END
DATE
FILMED
10-81
DTIC

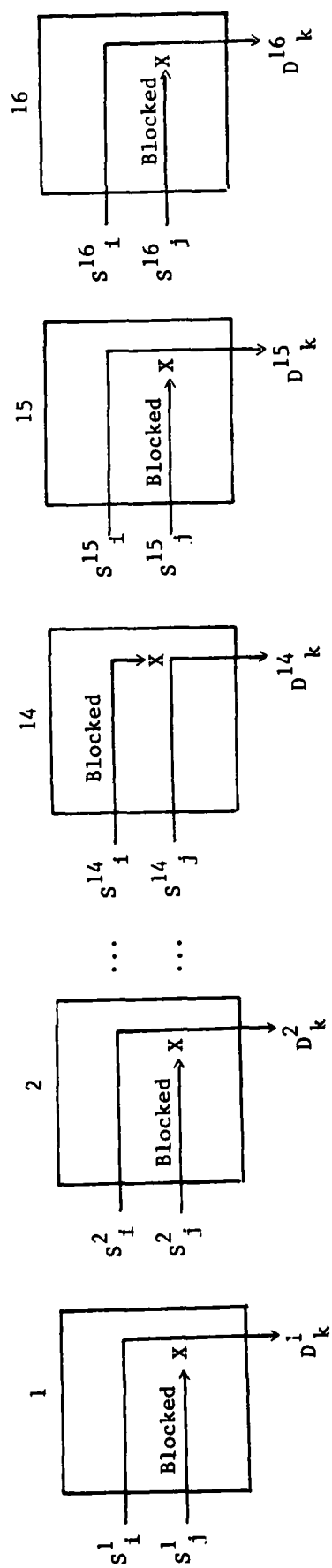


FIGURE 4. CONNECTIONS FOR S_i AND S_j REQUESTING D_k WITH
INCONSISTENCY IN 14th PLANE ($B' = 16, B = 1$)

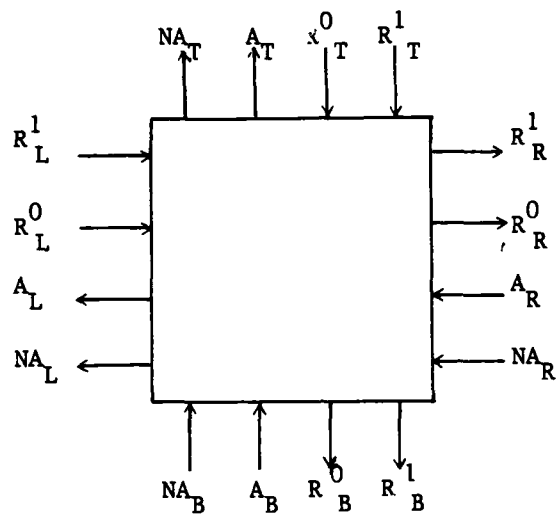


FIGURE 5. DELAY INSENSITIVE SWITCH MODULE
 (SUBSCRIPTS: L=LEFT, R=RIGHT, T=TOP,
 B=BOTTOM)

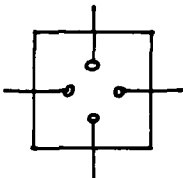
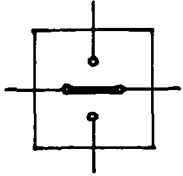
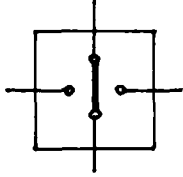
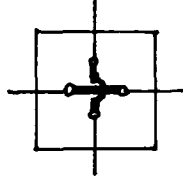
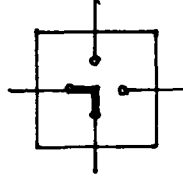
STATE	PATH CONNECTION
I	
H	
V	
HV	
C	

FIGURE 6. THE FIVE POSSIBLE DATA CONNECTION STATES OF A SWITCH MODULE

NEXT STATE: OUTPUT
FOR INPUT OF

PRESENT
STATE

	R_L^1	R_L^0	R_T^1	R_T^0	A_R	NA_R	A_B	NA_B
I	$C:R_B^1$	$H:A_L$	$V:R_B^1$	-	-	-	-	-
H	$H:R_R^1$	$H:R_R^0$	$HV:R_B^1$	-	$H:A_L$	$I:NA_L$	-	-
V	$V:NA_L$	$HV:A_L$	$V:R_B^1$	$V:R_B^0$	-	-	$V:A_T$	$I:NA_T$
HV	$HV:R_R^1$	$HV:R_R^0$	$HV:R_B^1$	$HV:R_B^0$	$HV:A_L$	$V:NA_L$	$HV:AT$	$H:NA_T$
C	$C:R_B^1$	$C:R_B^0$	$C:NA_T$	-	-	-	$C:A_L$	$I:NA_L$

FIGURE 7. STATE TABLE FOR SWITCH MODULE
(Dash represents "cannot occur" transition)

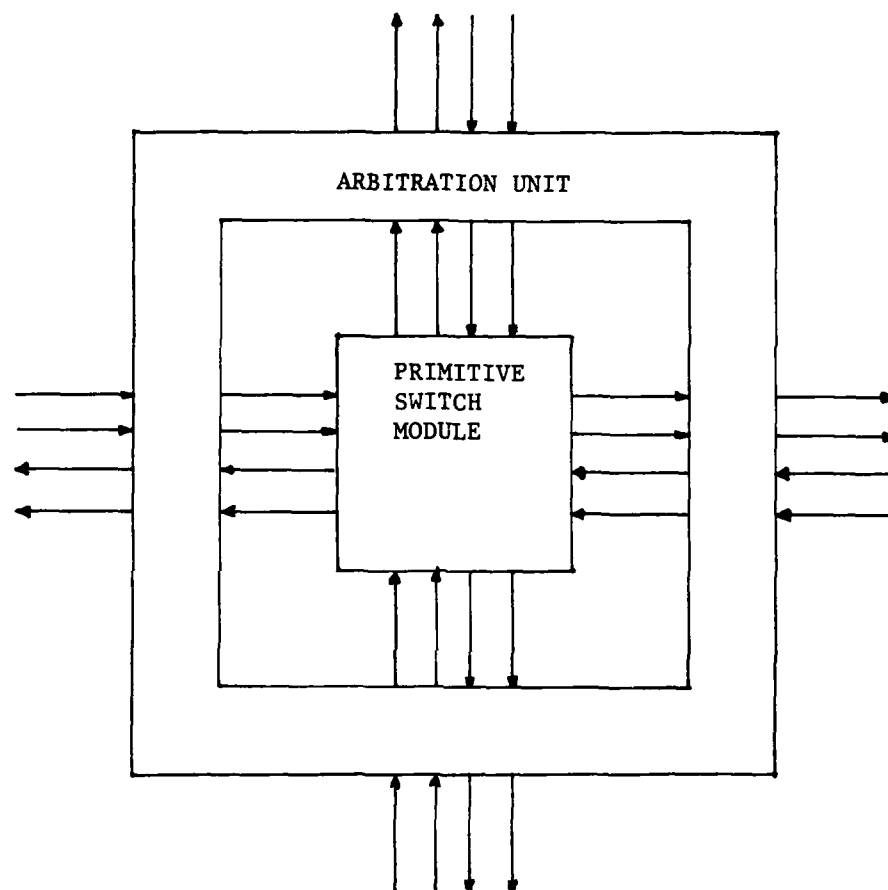


FIGURE 8. PLACEMENT OF ARBITRATION UNIT

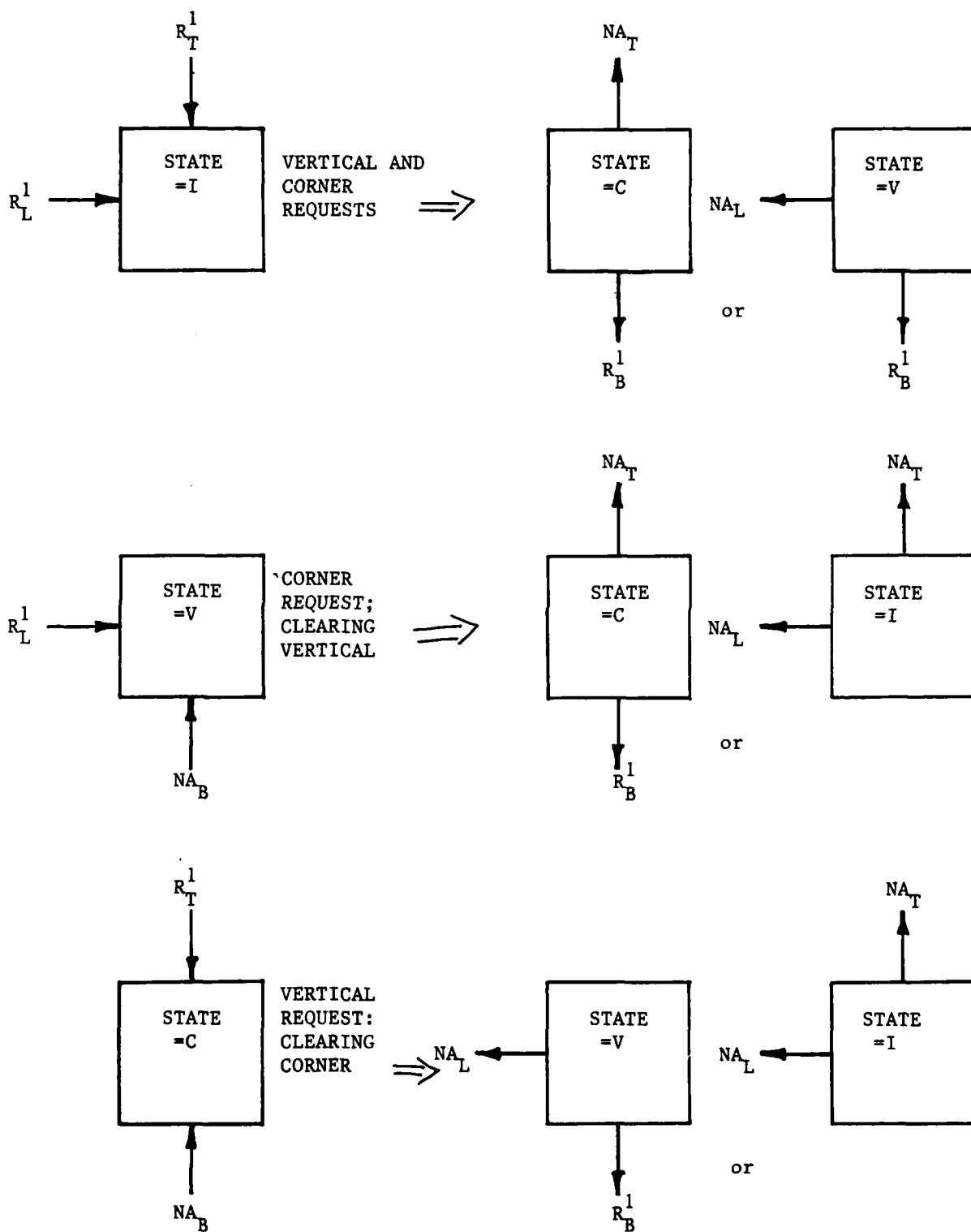


FIGURE 9. CONFLICT SITUATIONS AND POSSIBLE RESULTS

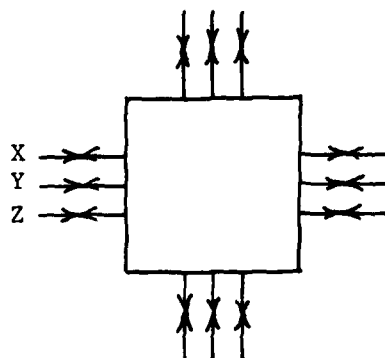


FIGURE 10. THREE LINE
BIDIRECTIONAL PATH SWITCH
MODULE

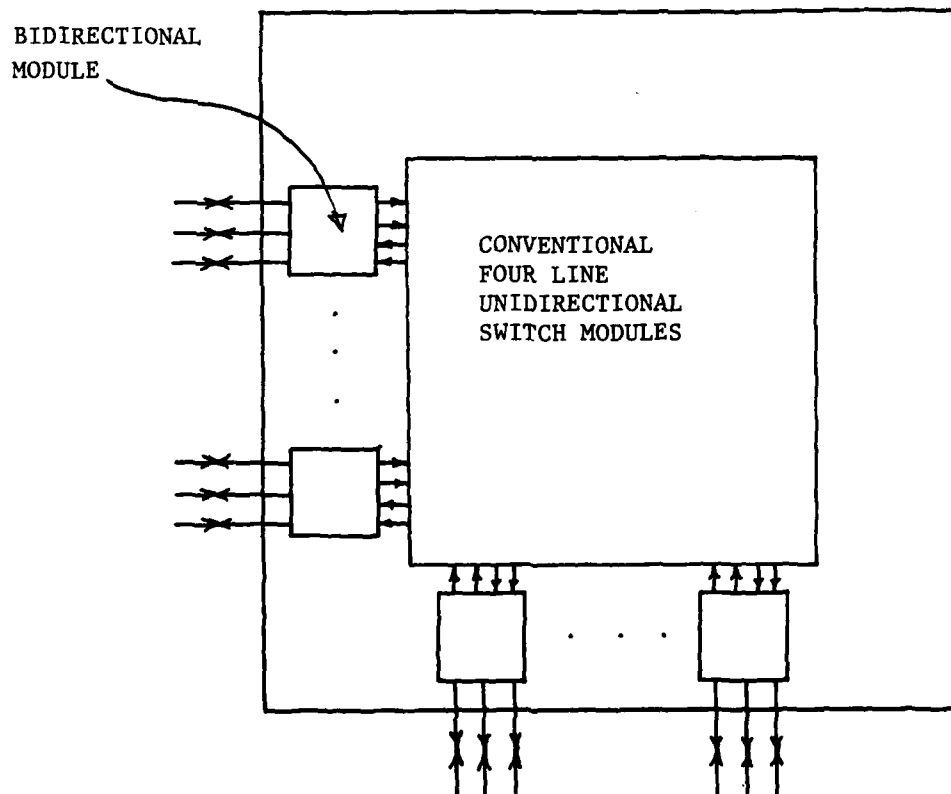


FIGURE 11. USE OF BIDIRECTIONAL AND UNIDIRECTIONAL
MODULES

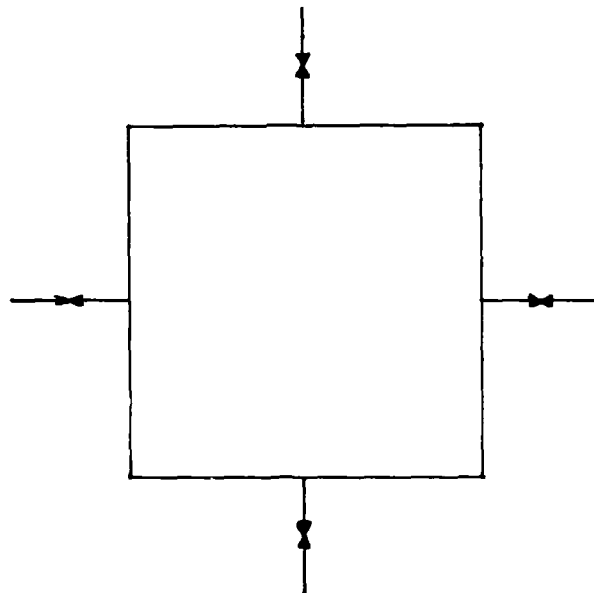


FIGURE 12. BIDIRECTIONAL SINGLE LINE
SWITCHING MODULE (NOT SELF-TIMED)

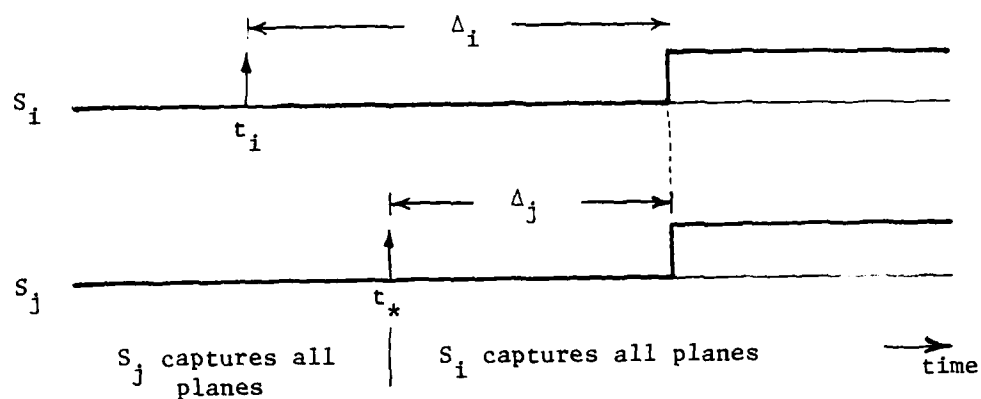


FIGURE 13. PATH DELAYS KNOWN (FOR PLANE P)

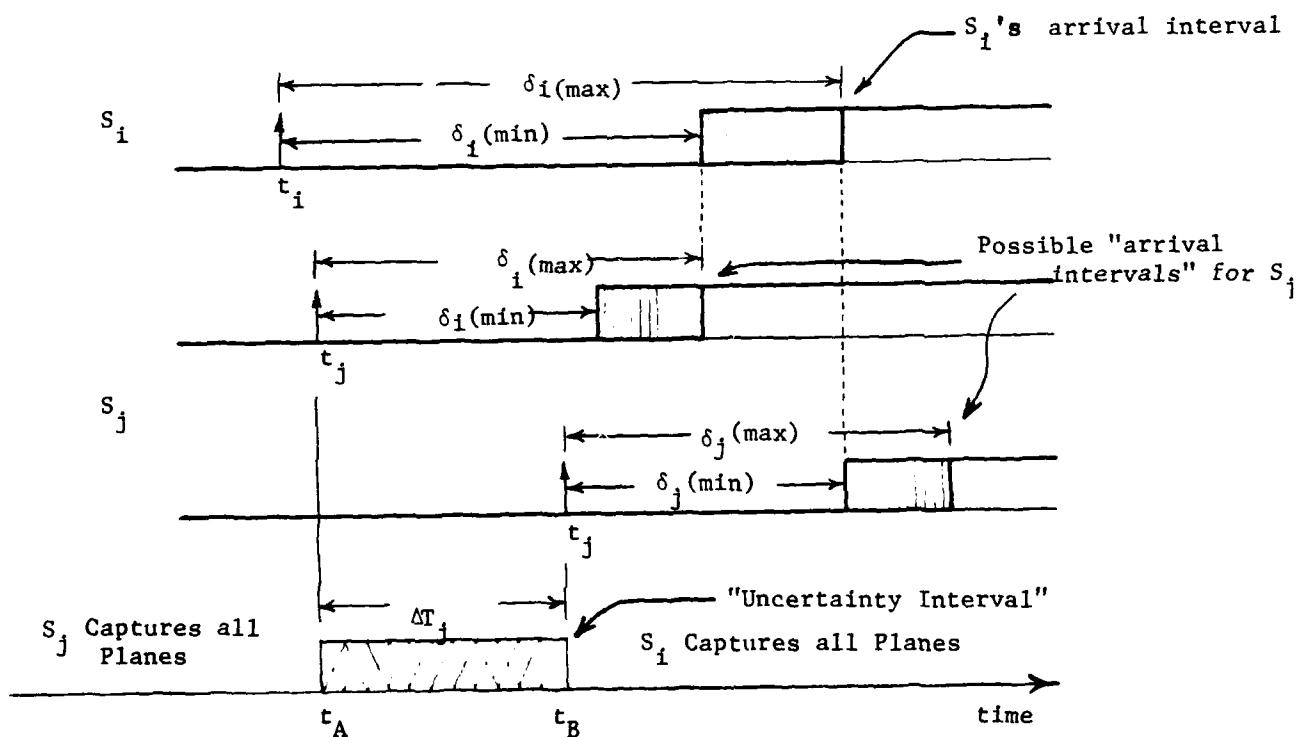


FIGURE 14. PATH DELAYS UNCERTAIN (FOR PLANE P)

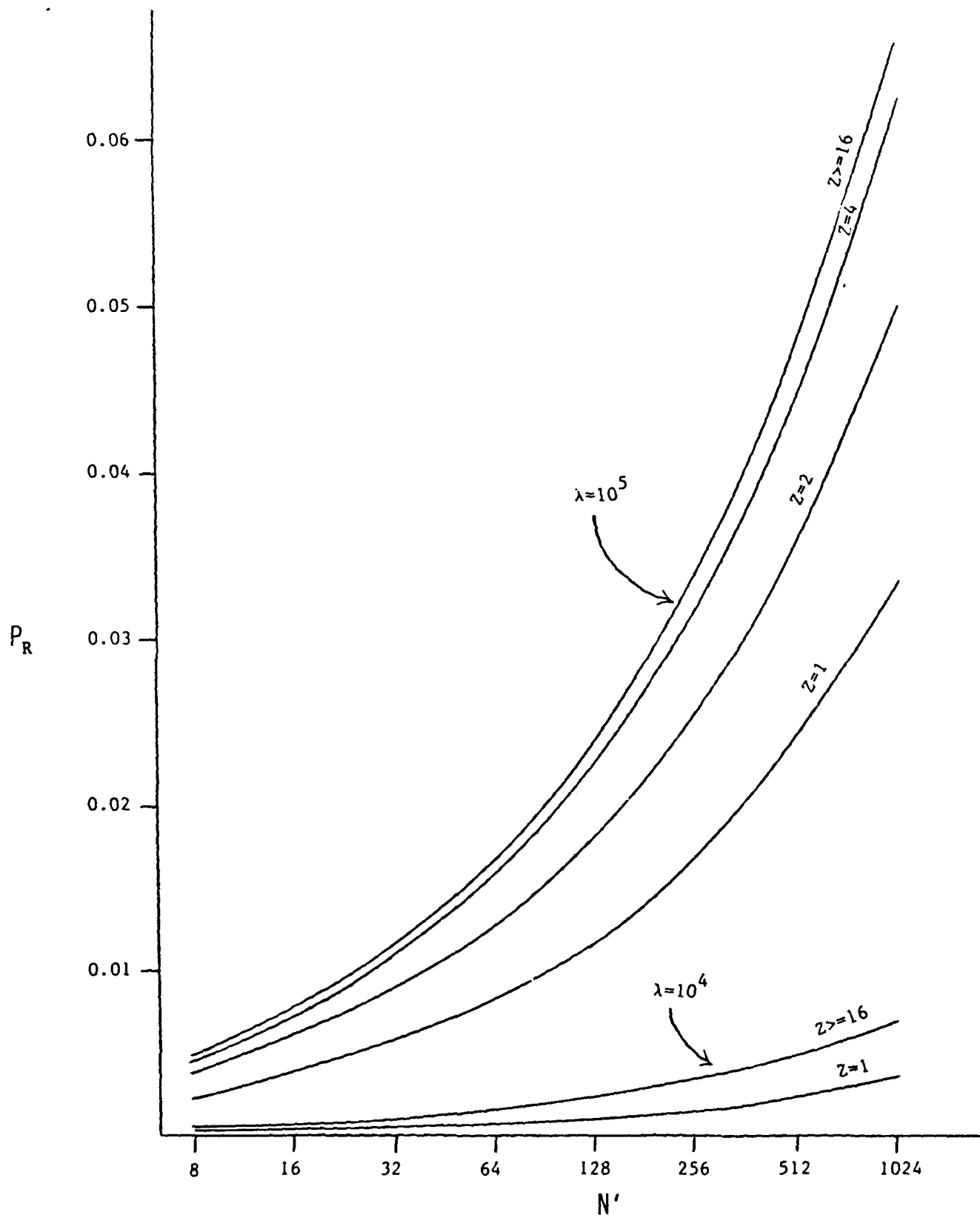


FIGURE 15. PROBABILITY P_R , THAT A WORD SENT BECOMES INCONSISTENT AND A RETRY IS NECESSARY AS A FUNCTION OF THE NETWORK SIZE N' , BIT PLANE PARTITIONS Z , AND SOURCE REQUEST RATE λ . SWITCH MODULE DELAY IS NORMALLY DISTRIBUTED WITH A MEAN OF 10 NSECS AND A 2.5 NSEC STANDARD DEVIATION. WORST CASE CONDITION IS USED WITH A SOURCE AT UPPER LEFT CORNER AND DESTINATION AT LOWER RIGHT CORNER OF INTERCONNECTION NETWORK.

Digital Systems
Research Review and Perspectives on the Future

M.A. Franklin and D.F. Wann
Washington University
St. Louis, Missouri 63130

Digital Systems Research Review and Perspectives on the Future

M.A. Franklin and D.F. Wann
Washington University - St. Louis

1.0 Research Review

Our research activities at Washington University (St. Louis) are centered on the analysis and synthesis of multiprocessor systems and associated communication networks. ONR support is principally in the communications network area.

National interest in tightly coupled MIMD (Multiple Instruction, Multiple Data stream) multiprocessor computer systems has increased greatly during the latter part of the 1970's. This has been due to 1) the enhanced performance possibilities for such systems (e.g., increased computational power and high reliability), 2) the steady decrease in hardware costs associated with these systems, and 3) the realization that the assembly of modular systems for the solution of computationally intensive problems is now feasible. This realization has resulted in the initiation of a number of multiprocessor projects around the country, several of which will be mentioned later in this discussion. Overviews of certain of the benefits (as well as limitations) of such systems have been presented by Enslow (1977), and Kuck (1977).

There are many technical issues involved in the design of these multiprocessor systems. One of the major constraints are those imposed by the network structure over which the multiple processors communicate. This is not to minimize the unique programming tasks required for such systems, but the programming is impacted less by technology and does not appreciably change when small alterations are made, for example, in the number of processors (e.g., say from 10 to 40). In contrast, this type of change could significantly influence the design and performance of the communication network - and thus the performance of the overall system. As a consequence, the

characteristics of this communication network becomes the critical factor in establishing both performance and cost. This has led to a variety of studies aimed at characterizing and quantifying the performance of such networks. The figure of merit of the system is determined, to a great extent, by the average bandwidth that is achieved between processors. Since this average bandwidth is primarily established by the architectural organization of the communication network, various structural configurations have been investigated and compared, for example, see Anderson, et. al., (1975), Thurber (1974), and Siegel, et. al., (1979A). The principal parameter used in these studies has been the effect that the number of switches (i.e., complexity) has on the bandwidth of the path between processors, or between processors and shared memories. Tradeoffs between bandwidth, switches, and path blocking have been presented in these and other papers. Major efforts in the area of network characterization and functional analysis are underway at the University of Illinois under Lawrie (1975), at Purdue University under Siegel (1979B), and at the University of Texas at Austin under Lipovski (1979).

Implementation efforts in the multiprocessor area are being under taken in a number of places. Carnegie-Mellon University has had an operational multiple processor system for several years now called C.MMP (Wulf and Bell, 1972). Processors are connected by use of a very complex crossbar switch which lacks modularity and expandability properties. Currently their efforts are centering on the implementation of a tree structured multiprocessor called CM.*. Another tree structured multiprocessor is being designed by Keller (1979) at the University of Utah with support from NSF. At Purdue, Siegel has been working, with Air Force sponsorship, on parallel processors for use

in image processing. Widdoes (1980) has been directing a large multiprocessor project at the Lawrence Livermore Laboratory under the sponsorship of the U.S. Navy. This multiprocessor uses a crossbar switch to connect up to sixteen computers each of which is itself of super computer power. The System Development Corporation at Huntsville, under contract to the Air Force, is investigating the design of a large multiprocessor for use in ABM applications. Dennis (1974) at MIT has been studying the design of multiprocessors of the "data flow" variety, and an experimental machine has been implemented by Texas Instruments. A similar approach is being studied by Sullivan and Bashkow (1977) at Columbia University. Processing elements in this case are connected in a Binary-K cube arrangement. Bolt, Beranek and Newman (BBN) is currently implementing a multiprocessor for ARPA. This system uses an indirect binary n-cube communications network and utilizes M68000 microprocessors (up to several hundred) as the computational element in a shared memory environment. At the University of Texas (Austin) implementation is proceeding on a "Reconfigurable Computer" (Sejnowski, 1980) which will eventually contain 16 bit slice processors of the 2900 variety, and 81 memories connected through a Banyan network. This work is being supported by NSF. An interesting commercial multiprocessor that recently became available is the HEP (Homogeneous Element Processor) which is being designed and implemented by the Denelcor Corporation of Denver under the auspices of the U.S. Army.

Finally at Washington University a multiprocessor is being designed and implemented. The processor building blocks are microprocessors (DEC-LSI-11s) while the connection network is a modular, pipelined crossbar (Franklin, 1979) which has been designed to have certain features which make it amenable to

VLSI implementation. The multiprocessor systems work has been supported by NSF, while aspects of the communications network are being supported by ONR.

All of these efforts reflect the general view that one important way of achieving computer power is through parallelism. In toto they represent at the national level a series of experiments, pursued by independent and competing groups, aimed at evaluating the potentials and problems of multiple processor systems. As indicated, one important aspect of these experiments is the interprocessor communications network used. While this is of critical importance, most efforts have concentrated on using relatively straightforward structures implemented in SSI and MSI technologies. This is acceptable as long as the number of processors in the system is limited, and little or no expansion is foreseen. This is the case with most of the multiprocessor experiments mentioned above. When one gets to systems where thousands of processors may be present, then the communications network, both in terms of its functionality, and implementation details becomes critical. In almost all designs for closely coupled multiprocessor systems, network complexity grows faster than processor complexity (i.e., number of processors). This fact plus our experience in multiprocessor design has reinforced our views on the importance of VLSI in achieving physically small, reliable, yet powerful, switching networks. The VLSI technology has the potential for economically placing large parts of a switching network for a multiprocessor system on a single chip. Cost here becomes related to chip area. Unfortunately, a new challenge appears: the implementation of the connection paths may use substantial amounts of the chip area, thus limiting the area available to the switch elements themselves. This has the effect of reducing the size of a switching network that

can be fabricated on a chip of a given size. The time delay associated with the connection paths also contributes to the overall delay, thus directly effecting bandwidth. Area, topology and layout, basically ignored in traditional communication network analysis, become important interrelated factors in VLSI network design. Some of these major issues have been discussed by Franklin (1980) and this is the research currently being supported by ONR.

We are not aware of any other effort in the United States in which the VLSI modular approach to communications network (including actual implementation issues) is being considered and that is principally what makes the effort at Washington University unique. Perhaps the most analogous work is that started by Kung, et. al., (1979) and his student Thompson (1979), in which they are exploring how certain conventional algorithms (such as matrix multiplication, tree searching and shuffle-exchange) can be efficiently mapped into VLSI. However, they are not explicitly investigating communication network problems.

Thus one of our major goals is to be able to provide both a theoretical and practical analysis of various communication network configurations in the VLSI domain, and hopefully to make recommendations regarding a modular approach to their design. If we are successful, this work could form the basis for the wide spread exploitation of the computational power of multiprocessor systems.

2.0 "Cutting Edges" in Multiprocessor Communications Network Research

The questions related to VLSI implementation of communications networks will remain a critically important research area for sometime to come. There are a variety of design options available including: a) network topology (e.g., crossbar, Banyan, binary K-cube, etc.), b) communications style (e.g., circuit switching, packet switching, pipelining), c) general organization (e.g., number of data/control lines, synchronous/asynchronous communications, pin usages and limitations, etc.).

It will likely take some-time before the most rewarding of these approaches is determined. Other research activities will be stimulated by this work and a broader view of these problems is presented in section 3.0.

A central issue for sometime will be the problem of pin limitations. To accomodate this constraint, while exploiting the large component densities available in VLSI, will require further investigations in multiple computer architectures. Architecture explorations which attempt to merge processing and network elements on the same chip will be pursued and just what the communications requirements are for different problem classes (Franklin, 1978) will be explored in depth.

This leads directly to the question of how one optimally can exploit parallelism in different applications areas. Communications networks which are tailored to application problem solution algorithms represents an important research area whose full potential is not known. In this area, "cutting edge" research will attempt to define how one effectively exploits VLSI technology in the context of multiprocessor architectures, particular application areas, and parallel solution algorithms.

3.0 Future Research Opportunities

While research into particular and specific problems in Electronic Systems Theory will continue to be fruitful (e.g., reliability theory, differential games, etc.) it appears that two interrelated general research themes will be of increasing importance in the future. The first concerns the design of high performance digital systems, while the second concerns the problem of designing and managing systems of rapidly growing complexity.

In the past the armed forces have been able to make effective use of digital computers and digital systems which have been developed and marketed primarily for nonmilitary purposes. General purpose computers, and more recently various popular microprocessors have had a sizeable role in many military systems. Increased system performance has stemmed in large part from developments in basic technology which have resulted in increased device performance. While this trend will continue, it is clear that many more complex architectural and design options which are potentially rewarding to the Navy will not be explored or developed for the civilian commercial market due to their cost, ongoing commercial compatibility requirements, and the specialized performance and environmental needs of the military. Of course this has always been true, however, with the advent of inexpensive microprocessors on the one hand, and custom VLSI chips on the other hand, the design options available appear to be growing rapidly. As a consequence, research opportunities should be directed towards exploring those options which are generally not being pursued actively by the civilian economy.

Super high performance digital systems can be achieved in a number of ways. At the systems and digital level two approaches stand out. The first

is to develop high performance systems by aggregating commercially available processors and microprocessors, into specialized multiprocessors suited to particular applications. Since the processors must communicate with each other in an effective manner, the design of the processor communications network is of particular importance. This is the basic motivation for the research work described in section 1.0 of this review. The second approach is to design specialized processors by developing customized VLSI chips or chip sets which satisfy the requirements of limited applications groups. In certain situations one or the other of these approaches might be advantageous. The key point however is that methodologies for selecting an approach, for predicting performance prior to implementation, and for performing design studies and selecting among design alternatives are by and large not available.

Thus, just when the possibilities are present for creative design resulting in large increases in performance, the methodologies, and tools for properly performing such design and for managing the complexity associated with this design environment are absent. This is an area where research opportunities exist, and where the long term payoff could be considerable.

Systems complexity is increasing at every level of the design cycle. At the chip level, component densities on the order of a million or more are likely to be achieved within the decade. At the processor level, multiprocessor systems containing a thousand or more processors will probably be attempted in the same general time frame. Ad hoc design methods are not suitable for such systems and more research into structured design techniques and methodologies is needed. One area of research requiring more work relates to the development of specification languages which permits the designer to specify and document systems at the hardware, software and interface boundaries.

Such specification languages should be highly modular and structured, and should be based on a hierarchical view of systems. This hierarchical view is essential so that designers at different levels (e.g., functional, timing/sequencing, and electrical) can develop and contribute to the system specification.

Such specification languages represent tools not only for documentation and management of the design process, but for conceptually developing models of the system to be designed. Such models, when represented in a specification language, should be capable of automatically generating simulation programs for the systems specified. This is critical if a better handle on system performance is to be achieved prior to implementation and if alternative designs are to be compared in a quantitative manner.

A number of critical research questions are present with regard to such specification/simulation systems. For instance it is not clear just how to achieve true hierarchical simulation capabilities. The high cost associated with running such simulations when even small systems are investigated is of concern. There may in fact be a need for certain types of special purpose computers tailored to the requirements of large system modeling and simulation. Finally the costs associated with generating simulation programs is very high. Research into automatic generation of such programs from system specification languages is needed.

Given system specification and simulation capabilities at a number of levels in the specification hierarchy, the problem of automatic design can be attacked. That is, having settled on a satisfactory system specification at one level, can the design, or system specification at successively lower levels be automatically generated? This is clearly a pressing problem in the VLSI domain. The current conventional wisdom is that while VLSI capabilities

will be exploited in the design of regularly structured digital systems (e.g., memories, programmable logic arrays), it may not be possible to fully exploit the available logic densities in special purpose, nonregular devices. The reason is that design time and costs go up drastically when one begins to design single chip systems whose complexity approaches a million devices. Note that except for certain high volume applications such as microprocessors, there is relatively little commercial incentive to design at this leading edge of VLSI capabilities. In many situations, packaging, power supply and similar costs already dominate the costs of the logic itself.

High performance, specialized military systems, however, require design at this leading edge. Thus design automation systems for VLSI are of importance. In particular more research should be directed at the problems of automatic chip design from systems specifications languages. For instance one should be able to specify a system in a register transfer language, specify a technology and its key parameters (e.g., for NMOS, feature size), and automatically generate the appropriate masks or chip layout instructions. Note that research is needed even at the level of defining just what represents an acceptable set of key parameters. At the level of automatic chip design and layout some research is under way, however much of it appears to be preliminary and many problems remain. For instance, assume that a register transfer language is used to specify that at different points in time and on various conditions, information is to be moved from each of several different registers and input pins to the input to an adder. Other data transfers involving these registers and inputs must also be performed at various times and on various conditions. Question: what is the best way (and layout) of setting up communications paths between these logical devices (e.g., use of a common

bus, use of multiplexers, demultiplexers, chip wide communications network, etc.)?

Such questions can often be answered in a satisfactory manner when the systems involved are of limited complexity. When the systems involve millions of components the specification, analysis and design questions become overwhelming. These, however, are the sort of systems we will want to design in the eighties, and the military in particular will have need to design such systems of a specialized nature to achieve its performance and reliability goals. To solve these problems of the middle and late 1980's it is imperative that appropriate research and tool development programs be initiated now.

References

1. Anderson, G.A., and Jensen, E.D., "Computer interconnection structures: taxonomy, characteristics, and examples," ACM Computing Surveys, 7,4 (Dec. 1975), 197-213.
2. Dennis, J.B., and Misunas, D.P., "A preliminary architecture for a basic data-flow processor," Proc. 2nd Ann. Symp. on Comp. Arch. (Dec. 1974), 126-132.
3. Enslow, P.H., Jr., "Multiprocessor organization - a survey," ACM Computing Surveys, 9,1 (March 1977), 103-129.
4. Franklin, M.A.; Kahn, S.A., and Stucki, M.J., "Design Issues in the Development of a Modular Multiprocessor Communications Network," Proc. of the Annual Symposium on Computer Architecture (April 1979), 182-187.
5. Franklin, M.A., "Parallel Solution of Ordinary Differential Equations," IEEE Trans. on Computers, C-27, 5 (May 1978).
6. Franklin, M.A., "VLSI Performance Comparison of Banyan and Crossbar Communications Networks," Proc. Workshop on Interconnection Networks (April 1980), 20-28.
7. Goke, L.R. and Lipoviski, G.J., "Banyan Networks for Partitioning Multiprocessor Systems," The First Ann. Symp. on Comp. Arch., University of Florida, Gainesville, Florida (1973), 21-28.
8. Keller, R.M., et. al., "A loosely-coupled applicative multi-processing system," Proc. AFIPS Nat. Comp. Conf. (1979), 861-869.
9. Kuck, D.J., "A Survey of Parallel Machine Organization and Programming," ACM Computing Surveys 9,1 (March 1977), 29-59.
10. Kung, H.T., "Let's Design Algorithms for VLSI Systems," Proc. of the Caltech Conf. on VLSI (Jan. 1979), 65-90.
11. Lawrie, D.H., "Access and Alignment of Data in an Array Processor," IEEE Trans. on Comp., Vol. C-24, No. 12 (Dec. 1975), 1145-1155.
12. Sejnowski, M.C., et. al., "An overview of the Texas Reconfigurable Computer," Proc. AFIPS Nat. Comp. Conf. (1980).
13. (A) Siegel, H.J.; McMillen, R.J., and Mueller, P.T., Jr., "A Survey of interconnection methods for reconfigurable parallel processing systems," Proc. 1979 Nat. Comp. Conf. (June 1979), 529-542.
14. (B) Siegel, H.J., "Interconnection Networks for SIMD Machines," Computer, Vol. 12, No. 6 (June 1979), 57-65.

15. Sullivan, H. and Baskow, T.R., "A Large Scale, Homogeneous, Fully Distributed Parallel Machine I," Proc. 4th Ann. Symp. on Computer Architecture (March 1977).
16. Thompson, C.D., "Area-Time Complexity for VLSI," Proc. of the Caltech Conf. on VLSI (Jan. 1979), 495-508.
17. Thurber, K.J., "Interconnection networks - a survey and assessment," Nat. Comp. Conf. (May 1974) 909-919.
18. Widdoes, L.C., Jr., "The S-1 Project: Developing High-Performance Digital Computers," Proc. of the Spring Computer Conf. 80 (Feb. 1980), 282-291.
19. Wulf, W.A. and Bell, C.G., "C.mmp-A Multi-Mini-Processor," Proc. AFIPS Fall Joint Comp. Conf., Vol. 41 (Fall 1972), 765-777.

ATE
LME