JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 2, 79–107 (1985)

Evaluation of Cube and Data Manipulator Networks*

ROBERT J. MCMILLEN

Hughes Aircraft Company, Long Beach, California 90810

AND

HOWARD JAY SIEGEL

PASM Parallel Processing Laboratory, School of Electrical Engineering, Purdue University, West Lafayette, Indiana 47907

The interconnection of a large number of processors and other devices to form a parallel/distributed computing system is a research area receiving a great deal of attention. One method is to use a multistage network. This paper compares two classes of multistage networks by examining two representative networks: the Generalized Cube and the Augmented Data Manipulator. The two topologies are compared using a graph model. By interpreting the graphical representations of the networks in different ways, different but functionally equivalent implementations result. The costs of the various implementations are compared taking VLSI considerations into account. Finally, the robustness (fault tolerance) of the different networks is measured and contrasted.

I. INTRODUCTION

The interconnection of a large number of processors and other devices to form a parallel/distributed computing system is a research area receiving a great deal of attention. Many different approaches to the interconnection method have been proposed and discussed including the use of buses [47], hierarchies of buses [44], direct links [13], single-stage networks [21], multistage networks [9, 22, 30, 38], and crossbars [49]. An important aspect of this research is the evaluation and comparison of the proposed approaches [6, 16, 40, 45]. The conclusion most often reached is that the best scheme to use in a particular design depends highly upon the intended application, performance requirements, and cost constraints. Once a connection method

*This work was supported by the United States Army Research Office, Department of the Army, under Grant DAAG29-82-K-0101; the National Science Foundation under Grant ECS 80-16580; and the Air Force Office of Scientific Research, Air Force Systems Commands, USAF, under Grant AFOSR-78-3581. The U.S. Government's right to retain a nonexclusive royalty-free license in and to this paper, for governmental purposes, is acknowledged.

0743-7315/85 $3.00

is chosen (e.g., single-stage network), a specific design must be decided upon and then implemented. During this phase of a system's specification, it is important for the designer to understand fully the differences and similarities between candidate designs.

This work is motivated by an ongoing study of methods to model distributed systems and an examination of networks suitable for use in the PASM [41] and PUMPS [10] systems. Two classes of multistage networks that have been considered for use in these and other systems, cube type and data manipulator type, are investigated in this paper. Specifically, graph models are used to quantify the difference between the Generalized Cube and Augmented Data Manipulator (ADM) networks in terms of cost and robustness (fault tolerance). Graph models are used because they are unencumbered by implementation details and are an excellent tool for representing an essential characteristic of a network: its topology. They also facilitate comparison of this work with other studies (e.g., [5, 19]).

The Generalized Cube and ADM networks are defined in Section II. Their relation to other multistage networks described in the literature is also discussed. Using a graphical representation, the networks' topologies are compared in Section III. In Section IV, two functionally equivalent implementations resulting from two different graph interpretations are examined to compare the cost of each network. Here, using VLSI chips is considered and costs are compared relative to the fraction of a stage that can be implemented on one chip. Finally, Section V contains an analysis of the robustness each network exhibits.

II. The Generalized Cube and ADM Networks

The Generalized Cube network is a multistage cube-type network topology that was introduced as a standard for comparing network topologies [39]. Assume the network has N inputs and N outputs: in Fig. 1, $N = 8$. The Generalized Cube topology has $n = \log_2 N$ stages, where each stage consists of a set of N lines connected to $N/2$ interchange boxes. Each interchange box is a two-input, two-output device. The labels of the input/output lines entering the upper and lower inputs of an interchange box serve as the labels for the upper and lower outputs, respectively. Each interchange box can be set to one of the four legitimate states shown [22].

The connections in this network are based on the *cube interconnection functions* [35]. Let $P = p_{n-1} \cdots p_1 p_0$ be the binary representation of an arbitrary I/O line label. Then the n cube interconnection functions can be defined as

$$cube_i(p_{n-1} \cdots p_1 p_0) = p_{n-1} \cdots p_{i+1} \overline{p}_i p_{i-1} \cdots p_1 p_0,$$

where $0 \leq i < n$, $0 \leq P < N$, and $\overline{p}_i$ denotes the complement of p_i. This

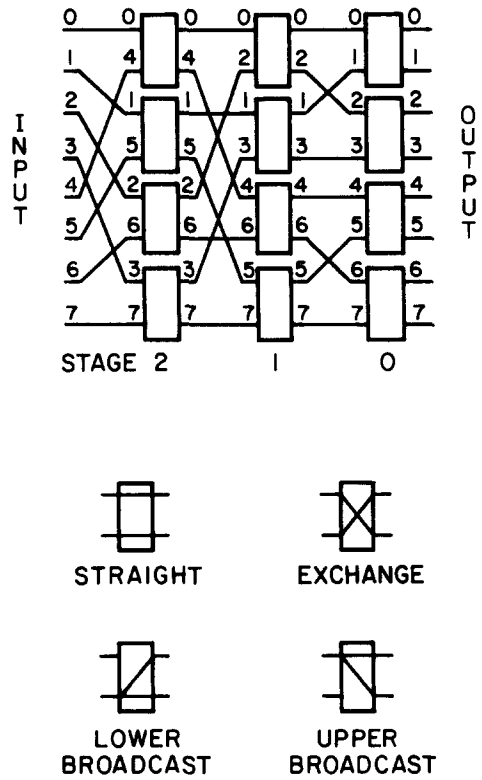

FIG. 1. Generalized Cube network for $N = 8$ [37]. The four legitimate states of an interchange box are shown.

means that the $cube_i$ interconnection function connects P to $cube_i(P)$, where $cube_i(P)$ is the I/O line whose label differs from P in just the ith bit position. Stage i of the Generalized Cube topology contains the cube interconnection function. That is, it pairs I/O lines that differ in the ith bit position.

The ADM network is shown in Fig. 2 for $N = 8$. It is based on Feng's data manipulator [15]. In this network, a *stage* consists of N *switching elements* or *nodes* and the $3N$ data paths that are connected to the inputs of a succeeding stage. Each node can connect one of its inputs to one or more of its outputs. At stage i of the ADM network, $0 \leq i < n$, the first output of node j is connected to the input of node $(j - 2^i) \bmod N$ of the next stage; the second output is connected to the input of node j; and the third output is connected to the input of node $(j + 2^i) \bmod N$. Because $(j - 2^{n-1})$ equals $(j + 2^{n-1}) \bmod N$, there are actually only two distinct data paths instead of three from each node in stage $n - 1$ (in the figure, stage 2). There is an additional set of N nodes at the output stage.

Both of these networks are based on the *PM2I interconnection functions* [35]. There are $2n$ of these functions defined by $PM2_{+i}(j) = j + 2^i \bmod N$

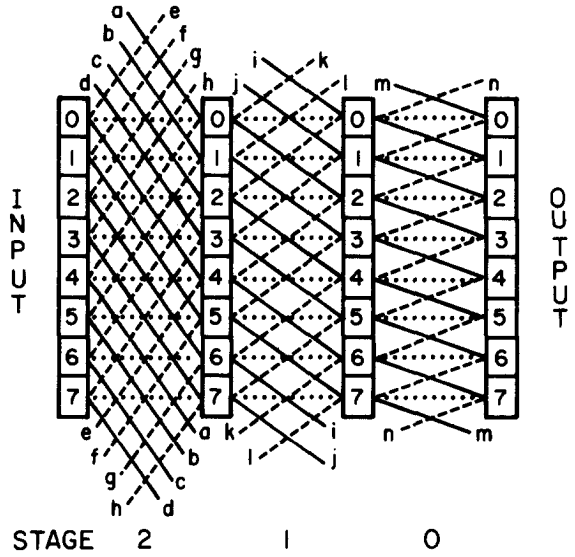

FIG. 2. Augmented Data Manipulator network for $N = 8$ [37]. (Lowercase letters represent end-around connections.)

and $PM2_{-i}(j) = j - 2^i \bmod N$ for $0 \leq j < N$, $0 \leq i < n$, where $-x \bmod N = N - x \bmod N$. (Note $PM2_{+(n-1)} = PM2_{-(n-1)}$.)

A number of systems have been proposed and/or built that use multistage networks (e.g., [7, 8, 24, 34, 41]). Among the networks that have been proposed are the ADM [38], baseline [48], binary n-cube [30], data manipulator [15], Gamma [29], Generalized Cube [39], inverse ADM (IADM) [27], omega [22], STARAN flip [9], and SW–banyan [19]. Studies have shown that the baseline, binary n-cube, Generalized Cube, omega, STARAN flip, and SW–banyan ($S = F = 2$) networks are all topologically equivalent [31, 36, 37, 42, 48]. Differences between these networks are due to proposed control schemes, whether or not a broadcast capability is included, and the method used to number input and output ports. All of these networks belong to the general class of cube-type networks. Because of the similarities among these networks, a designer is not faced with choosing between six different networks; rather the choice is whether or not to use a cube-type network.

The data manipulator, ADM, IADM, and Gamma networks are topologically identical. The differences between these networks are the control scheme, order in which stages are traversed, and switch complexity. The switches in each stage of the data manipulator are divided into two groups. Each group receives an independent set of control signals and all switches in a group respond identically. Each switching element of the ADM, IADM, and Gamma networks is controlled individually. The stages of the IADM and Gamma networks are traversed in an order opposite to that of the ADM and data manipulator. Also, the Gamma network's switching elements are 3×3

crossbars (as opposed to selecting one input at a time). One property that these networks have is that for all nontrivial source/destination pairs (i.e., source address $\neq$ destination address) there are multiple paths through the network. For that reason, none of the networks is a member of the general banyan class [19].

The capabilities of the Gamma network are a superset of the ADM and IADM networks. It has been shown in turn that their capabilities are a superset of all the cube-type networks as well as the data manipulator network [36, 37, 42]. Data manipulator-type networks, however, are more complex than cube-type networks.

A common feature of all cube-type networks is that there is exactly one path through the network for each source/destination pair. This property makes control schemes simple but any single failure of a link or switch will disallow the use of any path requiring the failed component.

Thus there exists the classic trade-off between cost and performance when choosing between the two network types. In this paper, the network types are compared, using one representative network from each type: the Generalized Cube and the ADM. Both networks have the same number of input and output ports and individual switching element control. Routing tag schemes are available for the networks [22, 28, 38, 39], so it is assumed that they are used to implement network control.

Some aspects of the Generalized Cube and the ADM networks have been compared elsewhere. The ability of the ADM network to perform all the functions a Generalized Cube can was demonstrated in [42]. In [1], the total number of unique permutation connections each network can perform was compared. In [5], graph models were used to study multistage interconnection networks which have the "buddy property" (cube-type networks have that property) and other networks including the ADM. In that paper emphasis was on comparing the networks' permutation capabilities. This paper is concerned with comparing cost and robustness or inherent fault tolerance. Cost is examined from two points of view. The first is the common method of counting links and switching nodes. In this case, the graph model with a consistent interpretation (two are possible) is used to ensure a "fair" comparison. The second point of view is oriented toward VLSI considerations. Modules for each network requiring roughly the same number of pins are compared. The change in relative cost is also examined when as much as one whole stage is placed on one chip. Robustness is measured by calculating the average number of network inputs and outputs affected by the removal of a single link or switching element. The calculations are performed for both of the graph interpretations to be defined.

III. Graph Modeling: A Common Basis for Comparing Networks

Graph models have been used by Goke and Lipovski [19] as the basis for defining a class of networks called banyans. The graphs used to represent

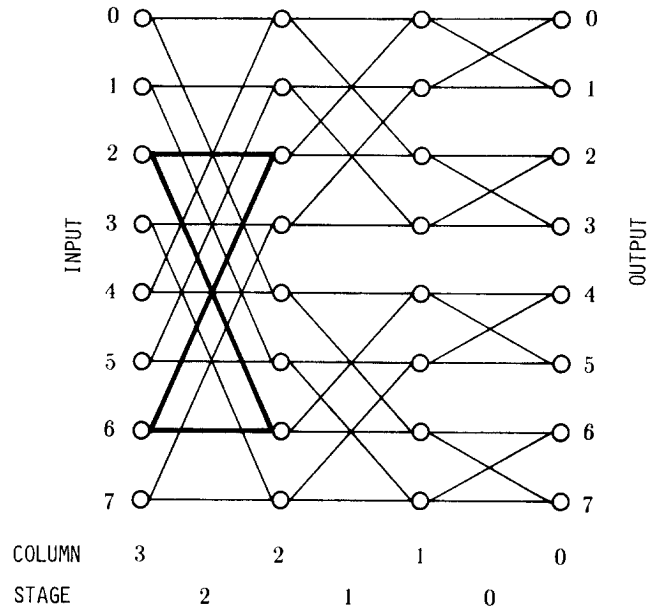

FIG. 3. Graphical representation of the Generalized Cube network for $N = 8$.

these networks consist of *nodes* connected by *directed arcs*. By definition, in a banyan there is one and only one path from input to output [19]. In this paper the arcs are undirected and there is no restriction on the number of paths from input to output.

It has been observed [20, 23] that the Generalized Cube network (Fig. 1) has the graphical representation shown in Fig. 3. This graph also represents an SW–banyan (with $S = F = 2$). The graph can be interpreted a number of different ways. One is to treat each node (vertex) (a circle in the figure) as a switch and each arc (edge) (a line in the figure) as a link. To model the network's behavior under this interpretation, the switch (node) shown in Fig. 4a should only connect one of the input links, a *or* b, to one of the output links, c *or* d. An implementation based on this interpretation, for an N input/output network, would consist of $n + 1$ stages of N switches, with $2N$ lines between stages. The TRAC reconfigurable, multimicroprocessor system contains an SW–banyan constructed from switches of this type (but that have two incoming and three outgoing links, i.e., $S = 2$ and $F = 3$) [32].

A second interpretation of the graph in Fig. 3 is to treat the nodes as links and the arcs as forming interchange boxes. For example, the thickened lines in Fig. 3 can be considered to represent the interchange box with inputs 2 and 6 (compare this to Fig. 1). In this case the SW–banyan implementation would have the same structure as specified here for the Generalized Cube (assuming a bidirectional network). This interpretation is illustrated in Figs. 4b and c.

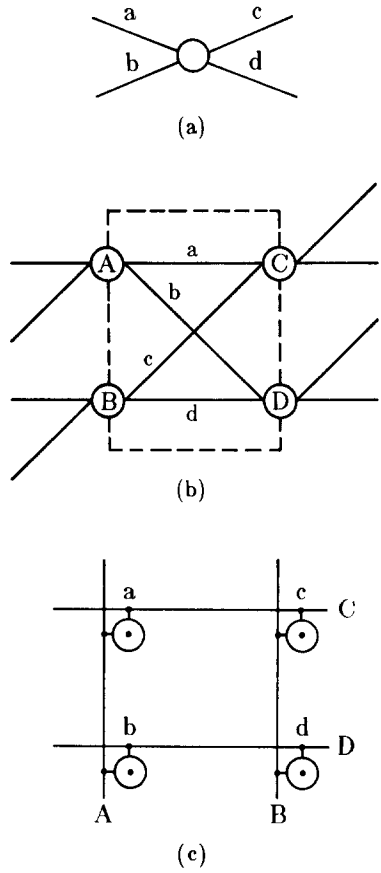

FIG. 4. (a) A node from the graph representing the Generalized Cube network. When equated with a switch, input *a or b* can be connected to output *c or d*. (b) Four nodes from the graph. When the arcs, *a*, *b*, *c*, and *d* are equated with switches, a 2 × 2 crossbar is obtained. (c) The components of a crossbar that correspond to the graph in (b).

Each of the arcs labeled a through d in Fig. 4b acts as a crosspoint switch in Fig. 4c. When viewed this way, the portion of the graph within the dashed lines of Fig. 4b behaves as a 2 × 2 crossbar or interchange box. If a and d are "on," the straight setting is obtained; b and c "on" corresponds to exchange; a and b "on" corresponds to upper broadcast; and c and d "on" corresponds to lower broadcast. Conflict occurs if a and c or b and d are on at the same time. It will be shown in the next section that implementations based on the first and second graph interpretations are functionally equivalent.

A third possible interpretation of the graph in Fig. 3 is to equate nodes with 2 × 2 interchange boxes and arcs with links. In that case, Fig. 3 would represent a size $N = 16$ Generalized Cube network. This interpretation will not be discussed further in this paper.

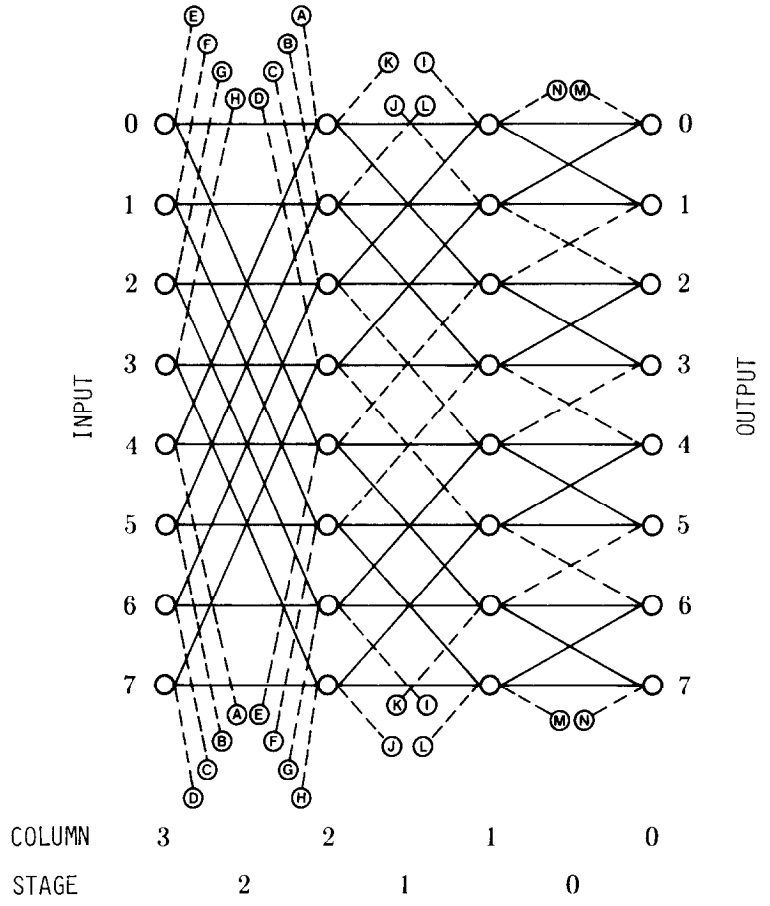

FIG. 5. Graphical representation of the Augmented Data Manipulator for $N = 8$.

The graphical representation of the ADM network (Fig. 2) is shown in Fig. 5. Since there are multiple paths from input to output, this is not a banyan graph. This graph can be obtained by adding the dashed lines shown in Fig. 5 to the graph in Fig. 3.[1] When switches are equated with nodes, the network depicted in Fig. 2 is obtained. When switches are equated with arcs, the network looks like that shown in Fig. 6. In the figure, two nodes directly connected by a solid line between stages are represented by a single node in Fig. 5. Note that the labels on end-around connections in both Fig. 5 and Fig. 6 are attached to the same arcs (links) in the network. This second type of

[1]We first published this observation in October 1982, in the Proceedings of the Third International Conference on Distributed Computing Systems, in a preliminary version of this material. It was also discovered independently and published in [5].

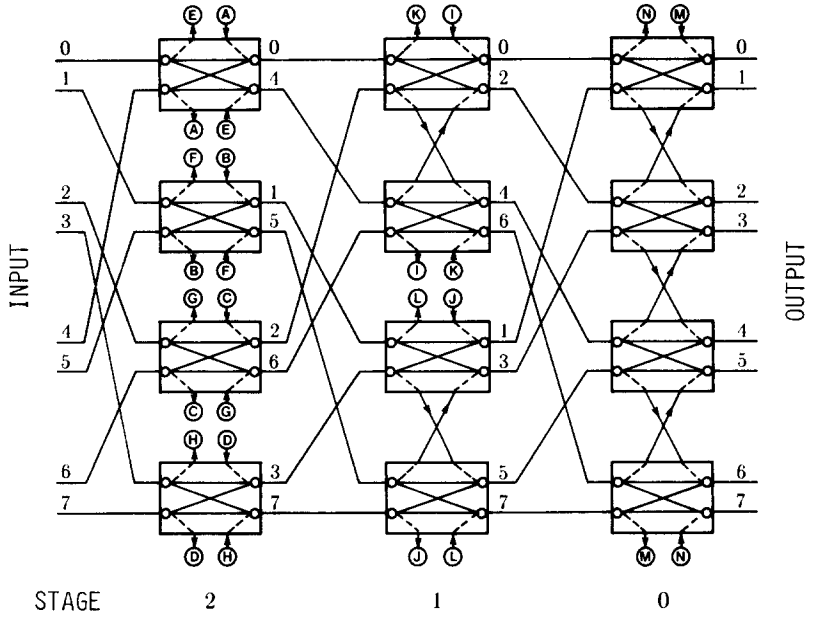

FIG. 6. Implementation of the Augmented Data Manipulator for $N = 8$ when the graph of Fig. 5 is interpreted with arcs equated to switches.

implementation is examined in [43], where LSI packaging of network building blocks is discussed.

Though the same ADM network is represented, Figs. 2 and 6 look rather different. Depending upon which representation is chosen, a comparison with the Generalized Cube in Fig. 1 could produce different conclusions. Comparing Figs. 1 and 2, one might conclude that, in addition to having an extra column of switches, the ADM has twice as many switching nodes and three times as many links as the Generalized Cube network. It would be easy to decide that the ADM network is considerably more expensive. On the other hand, comparing Figs. 1 and 6, it appears the only difference is N extra links that interconnect switches within each stage of the ADM network. The latter comparison is more accurate because the network depictions of Figs. 1 and 6 are based on the same interpretation of the networks' respective graphs. Thus when making comparisons, it is important to compare either graphical representations or consistent interpretations of those graphs. In the next section, the latter is done for both interpretations, so that the resulting implementations can be compared as well.

IV. Cost Comparison

A. *Introduction*

The purpose of this section is to compare the cost of the Generalized Cube network to that of the ADM network. To do this, implementations of each

network are examined. Two different criteria are used in the comparison. First, hardware requirements are examined. Since two basic implementations are possible for each network, to be fair, only implementations corresponding to the same graph interpretation are compared. Then, since VLSI implementation is being considered, the total number of data pins available on a chip is held constant and chip counts are compared for all the different implementations. It would be desirable to compare the gate densities required for each chip; however, that requires having a detailed design for each. In lieu of such details, the attempt was made to compare chips with comparable major architectural features (e.g., queues) which presumably require the same amount of logic and which can be compared at a gross level.

Although the discussion presented here is in terms of integrated circuit chips, it is not restricted to any particular technology. It is only presumed that a network is constructed from modular elements with I/O facilities (ports) proportional to that portion of the network graph (with an appropriate interpretation) intersected by the boundary of the module. For example, in the future, an I/O port may consist of a laser diode and a single optical fiber instead of many parallel wires.

B. *Hardware Realizations*

There are two basic ways to implement multistage networks. They can be circuit switched or packet switched. In circuit switching, a complete path is established from input to output and must be held for the duration of the communication. Circuit switching is often used when processors are connected to the network inputs and memories are connected to the outputs. Designs for circuit-switched interchange boxes have been discussed in [11, 26, 43]. In packet switching, messages are decomposed into packets which each make their way from stage to stage until the output is reached. This method is often used in configurations that connect processing element (processor/memory pair) j to input j and output j of a unidirectional network. Packet-switched network switching element designs have been discussed in [14, 26, 46].

In the remainder of this paper, implementations will be discussed primarily in terms of packet switching. Circuit-switched versions can be obtained by replacing any queues shown with buses. Other than this, remaining differences are in the control logic; however, the logic is shown only at the block diagram level. Only key elements of the implementations to be discussed are included since many variations of the basic designs are possible. For more detail see [14, 26, 46].

C. *Generalized Cube*

Figure 7 shows two designs for a Generalized Cube switching element. Figure 7a results when switches are equated with nodes in the graph (this corresponds to Figs. 4a and 3). One of the two inputs is selected depending

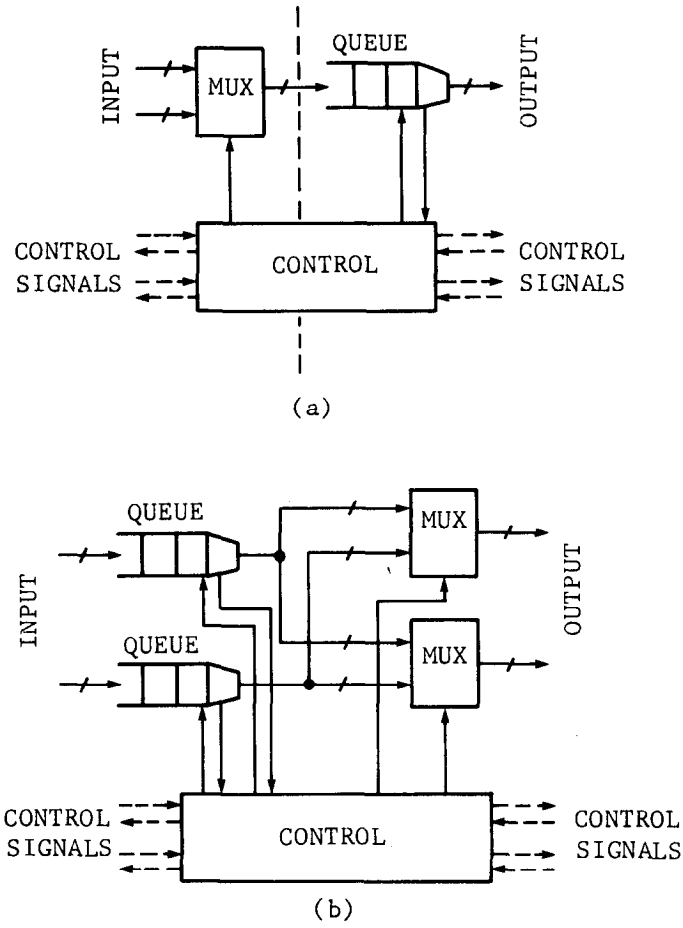

FIG. 7. Implementation of Generalized Cube switches. (a) Node = switch interpretation. (b) Arc = switch interpretation.

on the requests (if any) received by the (left half of the) control logic, which handles any needed arbitration. A single output link is shown, but it is to be connected to *two* other switches as shown in Fig. 8. A bit in the routing tag is examined by the control logic, which then determines to which switch a request for access should be made. The (right half of the) control logic maintains the queue, interprets the routing tag, generates access requests, and receives grants for access requests. Switches that implement nodes in column 3 of Fig. 3 only contain hardware to the right of the dashed line in Fig. 7a. Switches that implement column 0 nodes only contain hardware to the left of the dashed line. A detailed design of this type is discussed in [32].

If arcs in the graph are equated with switches, then four arcs form a 2×2 crossbar or interchange box (see Figs. 4b and c and 1). An implementation

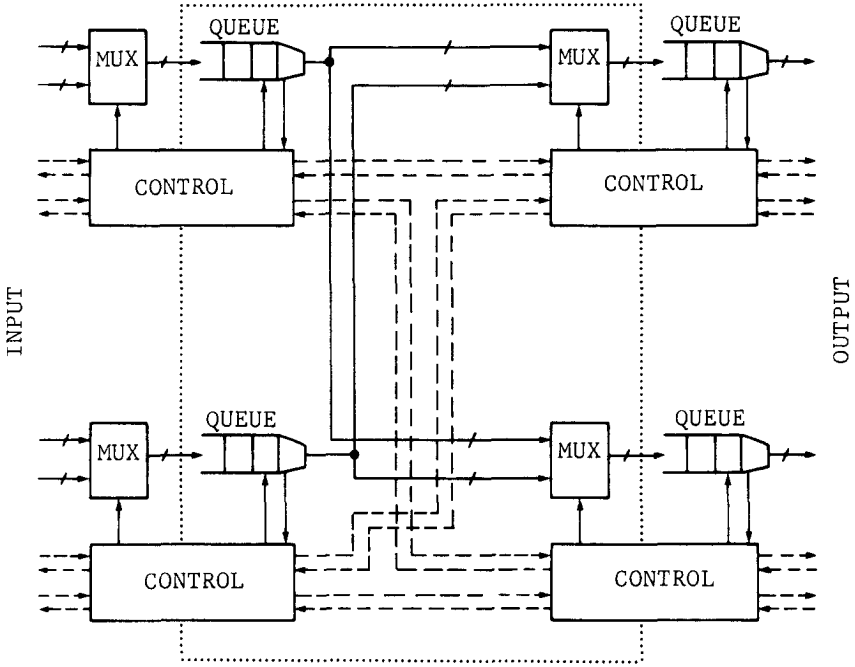

FIG. 8. Four switches from Fig. 7a combined to form one switch (within dashed lines) equivalent to that in Fig. 7b.

for this is shown in Fig. 7b. Here two input queues are required. As long as a given queue is not full, incoming packets for that queue will be accepted. Logic is required to handshake with other interchange boxes, maintain two queues, and interpret the routing tags at the head of each queue. This logic only interprets the tags in order to request the desired settings for the multiplexers. Logic asociated with the multiplexers performs any necessary arbitration. It also makes appropriate requests of other interchange boxes once the multiplexers are set. Different protocols and design variations for this type of switching element are discussed in [26]. The performance of networks implemented with these interchange boxes has been studied in [3, 14, 25].

The equivalence of two networks implemented with the two kinds of switching nodes is illustrated in Fig. 8. Four of the switching elements shown in Fig. 7a are connected as prescribed by the graph in Fig. 3. It can be seen that the hardware within the dashed lines is identical to that shown for the interchange box in Fig. 7b. The handshaking lines (directed dashed lines) shown connecting control units are equivalent to internal connections between the tag interpretation and queue control logic and the arbitration and output request logic in the control unit of Fig. 7b. It is thus apparent that the same total amount of hardware is required for either implementation, but that

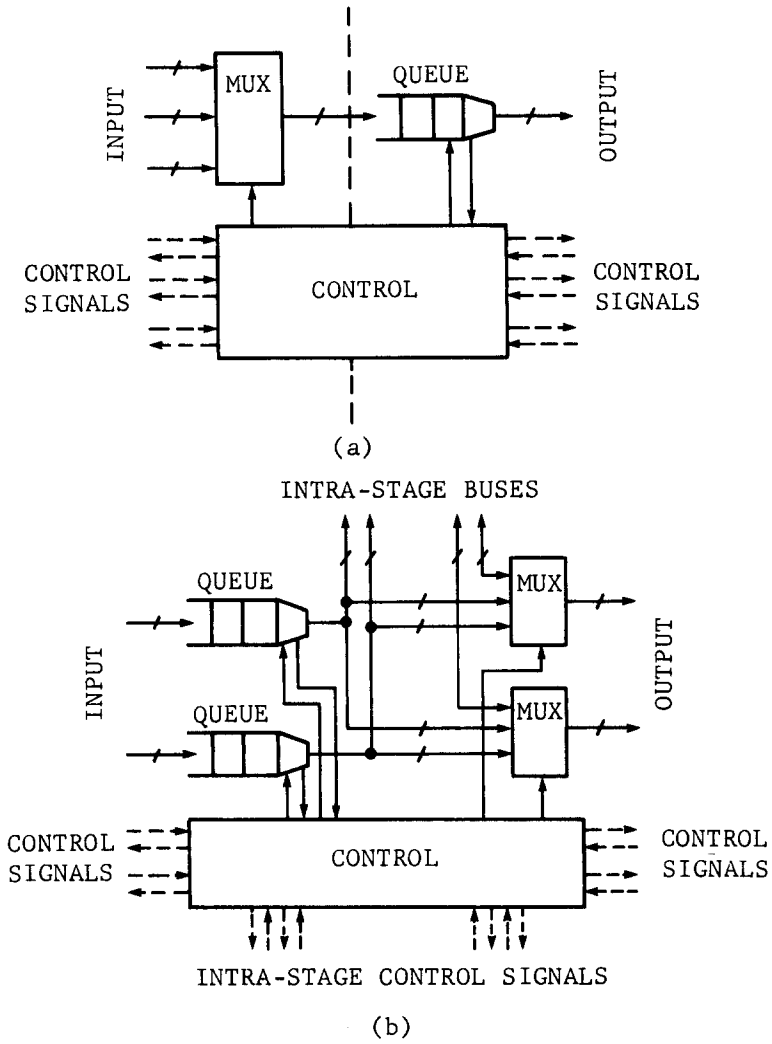

FIG. 9. Implementation of Augmented Data Manipulator switches. (a) Node = switch interpretation. (b) Arc = switch interpretation.

the two graph interpretations lead to different network building blocks or packagings for the components.

D. *Augmented Data Manipulator*

Two implementations for the ADM network are shown in Fig. 9. Figure 9a results from equating the nodes of Fig. 5 with switches. In this design, the multiplexer selects from among three input links and the output link is connected to three other switches. The control signals shown on the output side

in Fig. 9a are used to determine which of the switches is to read the data from the output link. A broadcast is performed by selecting more than one switch. The basic routing tag scheme for the ADM network [28] requires the routing tag logic to examine two bits, so it is slightly more complex than that required in the Generalized Cube. As with the Generalized Cube, the switches implementing nodes in columns 0 and 3 of Fig. 5 only require the logic to the left and right, respectively, of the dashed line in Fig. 9a. This was also observed in [15].

If arcs are equated with switches, an implementation similar to the interchange box is obtained as shown in Fig. 9b. Here, however, the outputs from the queues must be connected to multiplexers in two other switching elements (as shown in Fig. 6) via *intrastage buses*. Similarly, the two multiplexers shown here must accept connections from the queues of two other switching elements. Two control signals must also accompany each of the intrastage buses.

E. *Comparison*

An approximate cost comparison between the Generalized Cube and the ADM network can be made by comparing their respective switching elements. Since the choice is arbitrary, Figs. 7a and 9a will be compared. Both require a single queue. If the cost of the queue and its associated control logic dominates the cost of the switching element, then the ADM switch will cost only slightly more than a Generalized Cube switch in a discrete implementation. On the other hand, for a circuit-switched implementation, the multiplexer and control logic in an ADM switching element will cost about 50% more than that required in a Generalized Cube switching element.

The perspective changes somewhat when implementing these four designs in VLSI is considered. Input/Output (I/O) requirements and logic/pin ratio become important considerations. For constructing a Generalized Cube network, the interchange box in Fig. 7b is a better choice than the switch in Fig. 7a. The interchange box (Fig. 7b) has approximately 33% more pins but approximately 100% more logic than the switch (Fig. 7a). For the ADM network, the logic/pin ratio is nearly the same for both of the designs in Fig. 9. The design in Fig. 9b has approximately twice as many pins and twice as much logic as that shown in Fig. 9a. The extra links that give the ADM network its superior capabilities over the Generalized Cube require a larger number of pins on the VLSI chips being considered.

The design of Fig. 7b and that of 9a have approximately the same number of pins. If this number of pins (due to the data path width) is near technological limits (and thus the design of Fig. 9b will not fit on one chip), then the Generalized Cube interchange box is superior due to the logic/pin ratio. Assuming the cost of two chips with the same number of pins is about the same, an ADM network would be more than twice as expensive as a Generalized Cube network of the same size (when realized with these two re-

spective chips). The logic/pin ratio of the ADM chip (Fig. 9a) can be improved considerably by implementing extra capabilities the ADM network is known to support [27, 28]. These capabilities include dynamic rerouting of blocked messages and stage look-ahead with rerouting for blockage prediction. None of the additional features requires any extra pins. The additional capabilities are possible because of the extra paths between input and output and thus are *not* available for the Generalized Cube network.

The cost difference between the two implementations of each network due to pin limitations can be further quantified. Assume that one switching element is implemented on one chip and that the chips are bit sliced. For the sake of modularity, in the node-equals-switch implementation of both networks, this means the chip will be more complex than necessary for the switching elements in the input and output columns (3 and 0 in Figs. 3 and 5).

Let D be the number of pins available on the chip for data path connections and P be the number of I/O ports required by the switching element (see Figs. 7 and 9). The D/P is the number of pins available per port. It is assumed that data pins dominate the total pin count and that the chip has the capacity to accommodate the small number of control and power pins also needed. If the network data path width is W, the $W \cdot P/D$ is the number of chips required to construct one switching element. Multiplying this by the number of switching elements needed to implement the network gives the total chip count. The expressions for the chip count for the four implementations as a function of W, D, N, and n are given in Table I. A crossbar is included for comparison. The arc-equals-switch (interchange box) implementation of the Generalized Cube gives the lowest count regardless of the values of W, D, and N. As an example of the number of chips required in networks of size $N = 16$ and $N = 64$, assume the network path width is $W = 32$ bits and there are a total of $D = 64$ pins available on the chip for data connections. The resulting counts are shown in Table I.

TABLE I
COMPARISON OF CHIP COUNTS FOR TWO IMPLEMENTATIONS OF GENERALIZED CUBE AND ADM NETWORKS[a]

				$W = 32, D = 64$	
Network	Implementation	P	Chip count	$N = 16$	$N = 64$
Generalized Cube	Node = Switch	3	$3(W/D)N(n + 1)$	120	672
	Arc = Switch	4	$2(W/D)Nn$	64	384
ADM	Node = Switch	4	$4(W/D)N(n + 1)$	160	896
	Arc = Switch	8	$4(W/D)Nn$	128	768
Crossbar	Crosspoint linking two buses	2	$2(W/D)N^2$	256	4096

[a] P is the number of I/O ports per switching element, W is the network path width, D is the number of data pins per chip, and $n = \log_2 N$.

As advances in packaging technology continue, the cost difference between the Generalized Cube and the ADM will narrow considerably until the ADM is more cost effective. To see this, examine Fig. 6. The larger the number of switching elements (of the type in Fig. 9b), in the same stage, that can be placed on a single chip, the more intrastage buses can be internalized. This reduces the I/O overhead of the extra links. If a whole stage can be placed on one chip, then the ADM network requires the same number of chips and connections between chips as the Generalized Cube network. The assumption here is that the chip circuit density is not sufficient to support a crossbar but it will accommodate more logic than one stage of a Generalized Cube requires. The ADM network's structure thus fills a gap between the cube-type networks and crossbars. Until very large portions of an interconnection network can be placed on a single chip, it is clear that the ADM network will be more expensive to implement than the Generalized Cube, though the difference will continue to decline. Thus, it is important to determine the networks' cost effectiveness. It has already been pointed out that the ADM's capabilities are a superset of the Generalized Cube's. Another factor that is becoming more important as the construction of enormous systems is considered will be discussed in the next section: robustness of inherent fault tolerance.

V. Robustness: A Comparison of Degradation under Component Failure

A. *Introduction*

In this section, the robustness of each network is measured by removing a single component (link or switch) and counting the number of input and output ports that are affected. An input port is considered affected if it cannot send a message to all output ports. An output port is considered affected if there is at least one input port from which it cannot receive messages. Since the number of ports affected varies with the location of the removed component, averages are computed. Calculating the averages for the Generalized Cube network is relatively straightforward. Calculating the averages for the ADM network is complicated considerably by the varying numbers of multiple paths between ports of the network. Extensive use of and extension to the theoretical result in [28] were required to obtain the closed-form solutions presented here. To streamline this presentation, however, most of the mathematical derivations appear in the Appendix.

The average number of affected ports is calculated for both implementations of each network. These calculations are performed using two different rules for counting affected ports. The first rule requires all I/O ports to be considered. Under this rule, it has been shown that some permutation

connections can be routed around a faulty link in the ADM network, but this is not true in general [38].

The second rule allows "severely" affected ports to be disabled and thus not included in the count. It is implemented as follows. Referring to the graphs in Figs. 3 and 5, if a straight (or horizontal) arc at level j is removed, then input port j and output port j are disabled. If links are equated with arcs, one pair of I/O ports is disabled. If switches are equated with arcs, since two straight arcs are included in each switching element (Figs. 7b and 9b), two pairs of I/O ports are disabled. Thus, in Figs. 1 and 6, the I/O ports whose addresses correspond to the output labels on a given switching element are disabled if that switching element fails.

This second rule takes into account a practical system response to a network fault: the disabling of some components so that operation can continue, but in a degraded mode. This is feasible if the network is used for asynchronous communication by cooperating processors (MIMD mode [18]). If the network is used in a synchronous mode to establish permutation connections (SIMD mode [18]) disabling some components is not feasible. However, if the system is partitionable so that subsets of the processors, called submachines, operate synchronously but independent of other submachines, then certain submachines can be disabled when a fault occurs. PASM [41] and TRAC [34] are systems with this capability.

Since robustness is useless unless it can be exploited, it is implicit that faults can be detected and diagnosed and that the system can continue to function once a fault is detected. Detection and diagnosis have been investigated in [4, 17, 33, 39]. The latter requirement implies that measuring robustness is only meaningful for MIMD and partitionable SIMD environments.

The results using the first rule are shown in Table II and those using the second rule in Table III. Two examples of how to calculate the expressions

TABLE II
AVERAGE NUMBER OF AFFECTED I/O PORTS IN THE GENERALIZED CUBE AND ADM NETWORKS WHEN LINKS AND SWITCHES ARE REMOVED[a]

	Node = Switch		Arc = Switch	
Failure	Link	Switch	Link	Int. box
Generalized Cube	$\frac{2N-2}{n}$	$\frac{4N-2}{n+1}$	$\frac{4N-2}{n+1}$	$\frac{4N-4}{n}$
ADM	$\frac{N+n-1}{3n}$	$\frac{2N+n}{n+1}$	$\frac{2N+n}{n+1}$	$\frac{7N-8}{2n}+1$
Cube/ADM	$\simeq 6$	$\simeq 2$	$\simeq 2$	$\simeq 1.14$

[a] All ports are considered. Node = switch implementation corresponds to Figs. 3 and 2. Arc = switch implementation corresponds to Figs. 1 and 6.

TABLE III
AVERAGE NUMBER OF AFFECTED I/O PORTS IN THE GENERALIZED CUBE AND ADM NETWORKS WHEN LINKS AND SWITCHES ARE REMOVED[a]

	Node = Switch		Arc = Switch	
Failure	Link	Switch	Link	Int. box
Generalized Cube	$\frac{2N-2}{n}-1$	$\frac{2N}{n+1}-2$	$\frac{2N}{n+1}-2$	$\frac{2N}{n}-4$
ADM	0	0	0	$\frac{3N}{2n}-3$
Cube/ADM	∞	∞	∞	$\simeq 1.33$

[a] Severely affected ports are disabled and not counted.

in Table II are shown in the next subsection. The remaining derivations for both tables are presented in the Appendix. It should be noted that the entries in both tables under the "Node = Switch" column for a switch failure and under the "Arc = Switch" column for a link failure are identical. This is because both situations correspond to removing a single node from the graphical representation. Removing a link from the node-equals-switch implementation corresponds to removing a single arc from the graph, whereas removing an interchange box from the arc-equals-switch implementation corresponds to removing four or eight arcs from the graph, considerably different situations.

B. *Fault Effect Analysis Counting All Affected Ports*

Here the effects of a link fault are analyzed in detail for the Generalized Cube network and then for the ADM network. For the former, assume there is a link failure in stage i, the first rule applies, and the network is implemented by equating nodes with switches (Fig. 3). To see which inputs are affected, start with the failed link and move backward toward the input, tracing all links that are connected to the failed one. The number of affected inputs corresponds to the number of traced links in stage $n-1$. In general this number is 2^{n-i-1}. For example, if the link at level 4, stage 1 (Fig. 3), fails, inputs 0 and 4 are affected. The number of affected outputs is calculated by tracing links from the failed one to those to which it is connected in the last stage, i.e., stage 0. This number is expressed as 2^i. For example, failure of the link at level 6, stage 2 (Fig. 3), affects outputs 4, 5, 6, and 7. To calculate the average number of affected I/O ports, given a single link failure, a sum of these two terms taken over all stages is computed:

$$\frac{1}{n}\sum_{i=0}^{n-1}(2^{n-i-1}+2^i)=\frac{2}{n}\sum_{i=0}^{n-1}2^i=\frac{2(N-1)}{n}.$$

As a second example, consider the case of a link failure in the ADM network, using the first rule, and implemented by equating nodes with switches (see Fig. 2). A property of the ADM network is that there are at least two paths between every nontrivial (input address $\neq$ output address) input/output pair [28]. One of the existing paths consists of plus and straight links only and is called *positive dominant*. There is another path that consists of minus and straight links only and it is called *negative dominant*. The portions of the positive and negative dominant paths that are distinct depend on the relationship between the addresses. If they agree in the low-order $i + 1$ bits, then the paths converge at the input to stage i and follow the same set of straight links in stages i through 0. (The paths will be distinct in stages $n - 1$ through $i + 1$.) Thus if a nonstraight link fails, none of the I/O ports are affected because there will be a distinct path of the opposite dominance that avoids that link. (Routing schemes have been proposed that allow messages to *dynamically* switch between positive and negative dominant paths as they traverse the network [27, 28], allowing them to avoid busy or faulty links and switches.) If a straight link in stage i at level j fails, then all the input ports whose low-order $i + 1$ bits agree with output port j's low-order $i + 1$ bits will not be able to send a message to output j. There are 2^{n-i-1} such input ports. The other input ports can communicate with output port j since their paths to j do not converge until reaching a stage less than i. No output port other than j is affected by the failure. To see this, consider output port $k \neq j$. All input ports must be able to communicate with k. They can be divided into two classes: (1) those whose addresses agree with k's in less than $i + 1$ low-order bits; and (2) those whose addresses agree with k's in at least $i + 1$ low-order bits. In the first case, either a given path from the input to output k does not include the faulty straight link (in stage i, level j) or if it does, there is another path of opposite dominance that does not. In the second case, in stages i through 0 the required path uses straight links; however, they are all at level k. Thus all inputs can communicate with output k so k is unaffected. As an example, suppose the straight link in stage 1, level 4 (in Fig. 2), is bad ($i = 1, j = 4$). Consider three different situations: communication from inputs 0 and 4 to output 4, from input 0 to output 5, and from input 1 to output 5. Output 4 will be unable to receive messages from inputs 0 and 4, since 0 and 4 agree with j in the low-order $i + 1$ bits (2 bits). All the other output ports are unaffected. Consider connecting input 0 to output 5. Even though the positive dominant path, $+2^2$, straight, $+2^1$, from input 0 to output 5 includes the bad straight link, a message can simply take the negative dominant path, straight, -2^1, -2^0. Input 1 agrees with output 5 in the two low-order bits (bits 0 and 1) and therefore requires straight links in stages 0 and 1. However, the required links are at level 5, and thus the faulty straight link is not required.

The average number of I/O ports affected by a bad *straight* link, under the first rule, is calculated by adding the number of affected input and output ports

(as a function of the stage in which the fault is located) and summing over all stages:

$$\frac{1}{n}\sum_{i=0}^{n-1}(2^{n-i-1}+1)=\frac{1}{n}\sum_{i=0}^{n-1}(2^{i}+1)=\frac{N+n-1}{n}.$$

Since the failure of a $+2^i$ or a -2^i link does not affect any I/O ports, if link failures are equally likely, then the average over all links is one-third of the above value.

In Table II, the ratio of the average number of affected I/O ports in the Generalized Cube to those in the ADM is computed. Regardless of network size, in the node-equals-switch implementation, a link failure in the Generalized Cube network affects six times as many ports, on the average, as a link failure in the ADM. A given switch failure affects twice as many ports. In the arc-equals-switch implementation, a link failure in the Generalized Cube network affects twice as many ports as the same failure in the ADM network. An interchange box failure affects 1.14 times as many ports.

C. *Discussion of Fault Effects with Some Disabled Ports*

The measurement using the first rule is a very conservative indication of the robustness of the ADM network. Table III shows that under the second rule, the ADM network is very robust. When the pair of I/O ports connected to the network at the level of the failure is disabled in the node-equals-switch implementation, none of the remaining ports is affected by a link or a switch failure. A failure can only eliminate one of at least two paths that are always available between the enabled I/O ports (as illustrated in the example above connecting input 0 to output 5). In the arc-equals-switch implementation, link failures have no effect on enabled I/O ports, because (as pointed out earlier) the situation is equivalent to removing a switch in the node-equals-switch implementation. However, "interchange box" failures do affect some enabled ports. The reason this is the case is that there are situations in which both paths between input and output ports pass through the same box. It so happens that these situations only occur in networks larger than size $N = 8$. The full details are presented in the Appendix.

The entries in Table III for the Generalized Cube network are calculated in a fashion similar to those in Table II. The derivation for each entry is given in the Appendix.

The "interchange box" implementation fault analysis for the Generalized Cube and ADM networks considers only the worst case; i.e., the entire box is faulty. Whether both paths are acutally blocked due to a single fault in a real implementation depends on the nature of the fault. Assume that one interchange box is implemented on a single integrated circuit chip. If two data lines internal to the chip and coming from the same input become shorted, this will have no effect on the other internal data path and it is not necessary to

assume that the whole interchange box has failed. On the other hand, a mechanical failure could affect enough of the chip to render the entire device unusable. From a reliability point of view, this analysis shows that the implementation in Fig. 9a (which corresponds to the network in Fig. 2) is to be preferred over that in Fig. 9b (which corresponds to the network in Fig. 6). Since the logic/pin ratio is roughly the same for both implementations, nothing is lost. However, the total component count will be higher, leading to a less physically compact implementation.

The robustness measures for the ADM network are equally applicable to all the data manipulator-type networks with individual switching element control since all the properties used to derive them apply to each of the networks. Similarly, all the measures for the Generalized Cube network are applicable to all the cube-type networks that have individual switching element control.

The results presented here are for the basic cube-type and data manipulator-type topologies. It should be noted that variations on these topologies which are more fault tolerant have been proposed [2, 12, 27].

The above analysis assumed that the failure of one component was independent of the failure of any other component. If all or a large part of one stage is implemented on a single chip, this assumption may or may not be valid. If it is not, then the networks can be reanalyzed using the techniques presented here to account for the new failure pattern exhibited.

VI. Conclusions

This paper has examined two classes of multistage interconnection networks for use in parallel/distributed systems: the cube type and the data manipulator type. This was done by comparing a representative network from each class: the Generalized Cube and the Augmented Data Manipulator (ADM). This paper has attempted to quantify the differences in implementation costs by considering comparable implementation models for both networks. It was found that a discrete, circuit-switched implementation of the ADM network costs approximately 50% more than the same type of implementation of the Generalized Cube network. For discrete, packet-switched implementations, assuming the packet buffer cost dominates, the two networks cost about the *same* amount (ADM would be slightly higher). If the networks are to be constructed from VLSI chips, assuming the network's building block chips are to have nearly equal numbers of pins, the ADM network requires more than twice as many chips as the Generalized Cube.

Both networks can benefit from VLSI implementation. Each can be partitioned into complex building blocks that have higher logic/pin ratios than partitions of simple building blocks. Though the ADM building block requires more I/O ports on a chip than a Generalized Cube building block,

present and future predicted pin capacities are sufficient for the ADM network's needs. Using bit slicing, arbitrarily wide networks of either type can be constructed.

Using a graph model as a basis, two quantitative measures of comparative robustness were applied to the networks assuming they are used in MIMD or partitioned SIMD environments. Applying the measures to two different (functionally equivalent) implementations of each network under different faults it was found that the ADM network is always more robust. Using the first measure, the Generalized Cube network varied from having 1.14 to 6 times as many affected I/O ports due to a single failure as the ADM network. Using the second measure, in which some I/O ports are disabled, one implementation of the ADM network was shown to be able to fully support communication among the remaining enabled I/O ports.

In summary, a graph model has been used as a basis for quantifying the differences between cube- and data manipulator-type networks. Both implementation costs and robustness have been compared.

APPENDIX: DERIVATIONS OF ROBUSTNESS RESULTS

The following are derivations of each of the results in Tables II and III (excluding those already presented in Section V, B: the average number of affected I/O ports in the Generalized Cube and ADM networks when links or switches fail. Two different implementations and two different rules for disabling I/O ports are considered. Recall that a port is affected by a failure if it cannot send a message to all of the other ports or if it cannot receive a message from all of the other ports.

I. Rule 1: When a link or a switch fails, no I/O ports are disabled.
   A. Implementation: Node = Switch (Arc = Link).
      1. A single link fails. This case was considered in Section V of the text.
      2. A single switch fails. Under this implementation there are $n + 1$ columns of switches (see Figs. 3 and 5) so summations in the average are taken from 0 to n.
         a. Generalized Cube
            Starting at the failed switch, trace links and switches back to the input to determine the number of affected inputs. This number is 2^{n-i} if the failed switch is in column i. Using this same method, but tracing to the output, the number of affected output ports is 2^i. The average number affected is thus

$$\frac{1}{n+1}\sum_{i=0}^{n}(2^{n-i}+2^i)=\frac{2}{n+1}\sum_{i=0}^{n}2^i=\frac{2(2N-1)}{n+1}.$$

b. Augmented Data Manipulator
The same reasoning discussed in the text (Section V) applies here. There are 2^{n-i} affected input ports and only 1 affected output port. The average is thus

$$\frac{1}{n+1}\sum_{i=0}^{n}(2^{n-i}+1) = \frac{1}{n+1}\sum_{i=0}^{n}2^i + 1$$
$$= \frac{2N-1}{n+1} + 1 = \frac{2N+n}{n+1}.$$

B. Implementation: Arc = Switch (Node = Link).
1. A single link fails. This case is completely analogous to I.A.2 above.
2. A single interchange box fails. The effects of this are determined by examining Figs. 1 and 6.
   a. Generalized Cube
   The effects of a failed interchange box in stage i are determined by tracing both input links to the box back to the input and the two output links to the output. The number of affected input ports is 2^{n-i} and output ports is 2^{i+1}. The average number affected is

$$\frac{1}{n}\sum_{i=0}^{n-1}(2^{n-i}+2^{i+1}) = \frac{2}{n}\sum_{i=0}^{n-1}2^{i+1} = \frac{4}{n}\sum_{i=0}^{n-1}2^i = \frac{4(N-1)}{n}.$$

   b. Augmented Data Manipulator
   In this implementation, the straight arcs (from Fig. 5) that are paired at stage i (for the network in Fig. 6) are $p_{n-1}\cdots p_{i+1}p_ip_{i-1}\cdots p_0$ and $p_{n-1}\cdots p_{i+1}\bar{p}_ip_{i-1}\cdots p_0$. The logical "distance" between these links is 2^i. Thus, if $j = p_{n-1}\cdots p_{i+1}0p_{i-1}\cdots p_0$ is the address of the upper input to an "interchange" box in stage i, then $j + 2^i = p_{n-1}\cdots p_{i+1}1p_{i-1}\cdots p_0$ is the address of the lower input. For example, in Fig. 6, the second box from the top in stage 1 has inputs with addresses 4 and 6. In binary the addresses are $p_20p_0 = 100$ and $p_21p_0 = 110$, respectively. Notice that each box has two other inputs from nonstraight links. To consider all of the inputs that possibly could be affected by the failure of a box with inputs j and $j + 2^i$ in stage i, trace links backward from each box input to the input of the network. The easiest way to do this is to use Fig. 5. Start with the nodes at levels j and $j + 2^i$ in column i. For the example above, these are

nodes 4 and 6 in column 1. Trace the three links backward to column $i + 1$ and mark the appropriate nodes. Repeat the procedure for each marked node. There will be 2^{n-i} inputs marked in column n, so this is the upper bound on the number of affected inputs. For the example, nodes 0, 2, 4, and 6 are marked in column 3 of Figure 5. This translates to inputs with these numbers in Fig. 6. None of the other inputs can be affected by the failure of this box because they have no physical connection to it. All of the marked inputs are affected. This is because any input whose low-order $i + 1$ bits match either j (bits $0p_{i-1} \cdots p_0$) or $j + 2^i$ (bits $1p_{i-1} \cdots p_0$) will require straight connections in stages i through 0 at level j or $j + 2^i$ when these inputs communicate with outputs j or $j + 2^i$. They will be forced to use the faulty interchange box in stage i. Calculation shows that there are 2^{n-i} addresses that meet this criterion so the number of affected inputs equals the upper bound.

To determine which outputs are affected requires two observations: (1) inputs that can reach output j or $j + 2^i$ of the faulty interchange box can only get to levels in stage i of the form $j \pm b2^i \bmod N$, b any integer; and (2) regardless of the path taken in stages $n - 1$ though i, when the path reaches the output of stage i, it must be less than a distance of 2^i (i.e., 0 to $2^i - 1$) of the destination D. Observation (1) is a result of the fact that the inputs agree with j in the i low-order bits. In stages $n - 1$ through i the smallest increment by which a path can change levels is 2^i. Thus all the levels it can get to in stage i agree with j in the i low-order bits. Observation (2) is a result of the fact that the maximum distance stages $i - 1$ through 0 can change a path is $\sum_{k=0}^{i-1} 2^k = 2^i - 1$.

Now consider five cases regarding the relationship between D, j, and $j + 2^i$. First, note that any interchange box in stage $n - 1$ that fails will affect all the outputs since none of them can receive messages from inputs j and $j + 2^{n-1}$. So, assume $i < n - 1$.

Case 1. $D = j$. Output j is the only output from the faulty box less than a distance of 2^i from D; therefore (as shown in the affected inputs analysis) inputs that agree with j in the low-order i bits cannot communicate with D.

Case 2. $D = j + 2^i$. The argument is the same as for Case 1; thus D is affected.

Case 3. $j < D < j + 2^i$. The only outputs in stage i less than 2^i from D are j and $j + 2^i$. Thus both potential paths from an input that agrees with j in the low-order i bits must route through the faulty interchange box, so D is affected.

Case 4. $0 \leq D < j$ (if $j = 0 = D$ see Case 1). If D is a distance of 2^{i+1} or more from j, it is completely unaffected because there is no physical path from the faulty box to D. If D is a distance of less than 2^{i+1} from j, the only outputs from the faulty box less than 2^i from D are $j - 2^i$ and j. One of the paths to output $j - 2^i$ comes from the faulty box. However, it is known that there are at least two ways to get from an affected source to output $j - 2^i$ in stage i (which is input $j - 2^i$ of the next stage, stage $i - 1$). This follows from the facts that (1) there are at least two paths between every nonequal network input and output [28], and (2) the only way to reach network output $j - 2^i$ from an affected source is to go through input $j - 2^i$ from stage $i - 1$ and then "straight" through the rest of the network. Therefore, there are at least two physical paths from an affected source to input $j - 2^i$ at stage $i - 1$. Thus, every affected input must be able to communicate with D through the other path to input $j - 2^i$ in stage $i - 1$. Therefore, D is unaffected.

Case 5. $j + 2^i < D \leq N - 1$ (if $j + 2^i = N - 1 = D$ see Case 2). This case is completely analogous to Case 4. If D is a distance of 2^{i+1} or more from $j + 2^i$ then it is completely unaffected. Otherwise, the only outputs in stage i less than 2^i from D are $j + 2^i$ and $j + 2^{i+1}$. Thus D is unaffected.

To summarize, if $i = n - 1$, two inputs and all N outputs are affected. For $0 \leq i < n - 1$, the outputs affected have an address D such that $j \leq D \leq j + 2^i$. There are $2^i + 1$ such outputs. The inputs affected agree with j in the low-order i bits. There are 2^{n-i} such inputs. The average number of affected I/O ports is

$$\frac{1}{n}\left[\sum_{i=0}^{n-2} (2^{n-i} + 2^i + 1) + N + 2\right]$$

$$= \frac{1}{n}\left[\sum_{i=0}^{n-2} (4(2^i) + 2^i + 1) + N + 2\right] = \frac{7N - 8}{2n} + 1.$$

II. Rule 2: If a straight link or switching element at level j fails, disable input j and output j. If an interchange box with inputs j and k fails, disable inputs j and k and outputs j and k.

A. Implementation: Node = Switch (Arc = Link).

1. A single link fails.

a. Generalized Cube

When a straight link in stage i fails there are $2^{n-i-1} - 1$ affected inputs and when a nonstraight link fails there are 2^{n-i-1} affected inputs (since no ports are disabled). Similarly there are $2^i - 1$ and 2^i affected outputs, respectively. The average is thus

$$\frac{1}{2n}\sum_{i=0}^{n-1}(2^{n-i-1} - 1 + 2^{n-i-1} + 2^i - 1 + 2^i)$$

$$= \frac{1}{n}\sum_{i=0}^{n-1}(2^{n-i-1} + 2^{i-1}) = \frac{2N-2}{n} - 1.$$

b. Augmented Data Manipulator

If any straight link at level j fails, the only output some of the inputs cannot communicate with is output j. Since it is disabled and it was the only affected output (see the discussion in Section V), none of the remaining enabled input or output ports is affected.

2. A single switch fails.

a. Generalized Cube

This case is similar to case I.A.2.a except that there is one less affected input and one less affected output. Also, the failure of a switch in column n or 0 has no effect on any inputs or outputs. This is because when a column n switch fails, any input other than the one entering that switch can reach all outputs. Similarly, when a column 0 switch fails, the only output that cannot be reached is the one attached to the failed switch. The average is thus

$$\frac{1}{n+1}\sum_{i=1}^{n-1}(2^{n-i} + 2^i - 2) = \frac{2}{n+1}\sum_{i=1}^{n-1}(2^i - 1)$$

$$= \frac{2N}{n+1} - 2.$$

b. Augmented Data Manipulator

No input or output links are affected. The reasoning is similar to case II.A.1.b. If a switch at level j fails, the

only affected output port is that connected to the faulty switch by straight links, namely, output j.

B. Implementation: Arc = Switch (Node = Link).

1. A single link fails.
   This case is completely analogous to case II.A.2 above.
2. An interchange box fails.
   a. Generalized Cube
      This is similar to case I.B.2.a except that two less inputs and two less outputs are affected. Also, the failure of an interchange box in stage $n-1$ or 0 has no effect on any inputs or outputs. This is because when a stage $n-1$ box fails, any input other than those entering that box can reach all outputs. Similarly, when a stage 0 box fails, the only outputs that cannot be reached are those attached to the failed box. The average is thus

$$\frac{1}{n}\sum_{i=1}^{n-2}(2^{n-i}+2^{i+1}-4)=\frac{2}{n}\sum_{i=1}^{n-2}(2^{i+1}-2)=\frac{2N}{n}-4.$$

   b. Augmented Data Manipulator
      This is similar to case I.B.2.b except that two less inputs and two less outputs are affected. As in case II.B.2.a, the failure of an interchange box in stage $n-1$ or 0 has no effect on any inputs or outputs. Therefore, the average is

$$\frac{1}{n}\sum_{i=1}^{n-2}(2^{n-i}+2^{i}-3)=\frac{1}{n}\sum_{i=1}^{n-2}(3\cdot 2^{i}-3)=\frac{3N}{2n}-3.$$

ACKNOWLEDGMENT

A preliminary version of this material was presented at the Third International Conference on Distributed Computing Systems, October 1982.

REFERENCES

1. Adams, G. B., III, and Siegel, H. J. On the number of permutations performable by the augmented data manipulator network. *IEEE Trans. Comput.* **C-31** (Apr. 1982), 270–277.
2. Adams, G. B., III, and Siegel, H. J. The extra stage cube: A fault-tolerant interconnection network for supersystems. *IEEE Trans. Comput.* **C-31** (May 1982), 443–454.
3. Adams, G. B., III, and Siegel, H. J. The use of 4 × 4 switching elements in the multistage cube networks. Proc. 1st Int. Conf. Computers and Applications, June 1984, pp. 585–592.
4. Agrawal, D. P. Testing and fault-tolerance of multistage interconnection networks. *Computer* **15** (Apr. 1982), 41–53.
5. Agrawal, D. P. Graph theoretical analysis and design of multistage interconnection networks. *IEEE Trans. Comput.* **C-32** (July 1983), 637–648.

6. Anderson, G. A., and Jensen, E. D. Computer interconnection structures: Taxonomy, characteristics, and examples. *ACM Comput. Surveys* **7** (Dec. 1975), 197–213.
7. Barnes, G. H., and Lundstrom, S. F. Design and validation of a connection network for many-processor multiprocessor systems. *Computer* **14** (Dec. 1981), 31–41.
8. Batcher, K. E., STARAN parallel processor system hardware. Proc. AFIPS 1974 Nat. Computer Conf., May 1974, pp. 405–410.
9. Batcher, K. E. The flip network in STARAN. Proc. 1976 Int. Conf. Parallel Processing, Aug. 1976, pp. 65–71.
10. Briggs, F., Fu, K. S., Hwang, K., and Wah, B. W. PUMPS architecture for pattern analysis and image data-base management. *IEEE Trans. Comput.* **C-31** (Oct. 1982), 969–983.
11. Ciminiera, L., and Serra, A. Modular interconnection networks with asynchronous control. 14th Annual Hawaii Int. Conf. System Science, Jan. 1981, pp. 210–218.
12. Ciminiera, L., and Serra, A. A fault-tolerant connecting network for multiprocessor systems. 1982 Int. Conf. Parallel Processing, Aug. 1982, pp. 113–122.
13. Despain, A. M., and Patterson, D. A. X-tree: A tree structured multi-processor computer architecture. Proc. 5th Annual Int. Symp. Computer Architecture, Apr. 1978, pp. 144–151.
14. Dias, D. M., and Jump, J. R. Analysis and simulation of buffered delta networks. *IEEE Trans. Comput.* **C-30** (Apr. 1981), 273–282.
15. Feng, T. Data manipulating functions in parallel processors and their implementations. *IEEE Trans. Comput.* **C-23** (Mar. 1974), 309–318.
16. Feng, T. A survey of interconnection networks. *Computer* **14** (Dec. 1981), 12–27.
17. Feng, T., and Wu, C. Fault-diagnosis for a class of multistage interconnection networks. *IEEE Trans. Comput.* **C-30** (Oct. 1981), 743–758.
18. Flynn, M. J., Very high-speed computing systems. *Proc. IEEE* **54** (Dec. 1966), 1901–1909.
19. Goke, L. R., and Lipovski, G. J. Banyan networks for partitioning multiprocessor systems. Proc. 1st Annual Int. Symp. Computer Architecture, Dec. 1973, pp. 21–28.
20. Jenevein, R. M., and Browne, J. C. A control processor for a reconfigurable array computer. Proc. 9th Annual Int. Symp. Computer Architecture, Apr. 1982, pp. 81–89.
21. Lang, T., and Stone, H. S. A shuffle-exchange network with simplified control. *IEEE Trans. Comput.* **C-25** (Jan. 1976), 55–65.
22. Lawrie, D. H. Access and alignment of data in an array processor. *IEEE Trans. Comput.* **C-24** (Dec. 1975), 1145–1155.
23. Malek, M., and Myre, W. W. A description method of interconnection networks. *IEEE Tech. Committee Distrib. Process. Quart.* **1** (Feb. 1981), 1–6.
24. McDonald, W. C., and Williams, J. M. The advanced data processing test bed. Proc. Compsac, Mar. 1978, pp. 346–351.
25. McMillen, R. J., Adams, G. B., III, and Siegel, H. J. Performance and implementation of 4 × 4 switching nodes in an interconnection network for PASM. Proc. 1981 Int. Conf. Parallel Processing, Aug. 1981, pp. 229–233.
26. McMillen, R. J., and Siegel, H. J. The hybrid cube network. Proc. Distributed Data Acquisition, Computing, and Control Symp., Dec. 1980, pp. 11–22.
27. McMillen, R. J., and Siegel, H. J. Performance and fault tolerance improvements in the inverse augmented data manipulator network. Proc. 9th Annual Int. Symp. Computer Architecture, Apr. 1982, pp. 63–72.
28. McMillen, R. J., and Siegel, H. J. Routing schemes for the augmented data manipulator network in an MIMD system. *IEEE Trans Comput.* **C-31** (Dec. 1982), 1202–1214.

29. Parker, D. S., and Raghavendra, C. S. The Gamma network: A multiprocessor network with redundant paths. Proc. 9th Annual Int. Symp. Computer Architecture, Apr. 1982, pp. 73–80.

30. Pease, M. C. The indirect binary n-cube microprocessor array. *IEEE Trans. Comput.* **C-26** (May 1977), 458–473.

31. Pradhan, D. K. and Kodandapani, K. L. A uniform representation of single- and multistage interconnection networks used in SIMD machines. *IEEE Trans. Comput.* **C-29** (Sept. 1980), 777–791.

32. Premkumar, U. V., Kapur, R., Malek, M., Lipovski, G. J., and Horne, P. Design and implementation of the banyan interconnection network in TRAC. Proc. AFIPS 1980 Nat. Computer Conf., June 1980, pp. 643–653.

33. Rathi, B. D., and Malek, M. Fault diagnosis of networks. Proc. Distributed Data Acquisition, Computing, and Control Symp., Dec. 1980, pp. 110–119.

34. Sejnowski, M. C., Upchurch, E. T., Kapur, R. N., Charlu, D. P. S., and Lipovski, G. J. An overview of the Texas Reconfigurable Array Computer. Proc. AFIPS 1980 Nat. Computer Conf., June 1980, pp. 631–641.

35. Siegel, H. J. Analysis techniques for SIMD machine interconnection networks and the effects of processor address masks. *IEEE Trans. Comput.* **C-26** (Feb. 1977), 153–161.

36. Siegel, H. J. Interconnection networks for SIMD machines. *Computer* **12** (June 1979), 57–65.

37. Siegel, H. J. *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies*. Lexington Books, Lexington, Mass., 1984.

38. Siegel, H. J., and McMillen R. J. Using the augmented data manipulator network in PASM. *Computer* **14** (Feb. 1981), 25–33.

39. Siegel, H. J., and McMillen R. J. The multistage cube: A versatile interconnection network. *Computer* **14** (Dec. 1981), 65–76.

40. Siegel, H. J., McMillen, R. J., and Mueller, P. T., Jr. A survey of interconnection methods for reconfigurable parallel processing systems. Proc. AFIPS 1979 Nat. Computer Conf., June 1979, pp. 529–542.

41. Siegel, H. J., Siegel, L. J., Kemmerer, F. C., Mueller, P. T., Jr., Smalley, H. E., Jr., and Smith. S. D. PASM: A partitionable SIMD/MIMD system for image processing and pattern recognition. *IEEE Trans. Comput.* **C-30** (Dec. 1981), 934–947.

42. Siegel, H. J., and Smith, S. D. Study of multistage SIMD interconnection networks. Proc. 5th Annual Int. Symp. Computer Architecture, Apr. 1978, pp. 223–229.

43. Smith, S. D. LSI design considerations for multistage interconnection networks for parallel processing systems. Proc. 14th Annual Hawaii Int. Conf. System Science, Jan. 1981, pp. 219–227.

44. Swan, R. J., Fuller, S. H., and Siewiorek, D. P. Cm*: A modular, multimicroprocessor. Proc. AFIPS 1977 Nat. Computer Conf., June 1977, pp. 637–644.

45. Thurber, K. J., Interconnection networks—A survey and assessment. Proc. AFIPS 1974 Nat. Computer Conf., May 1974, pp. 909–919.

46. Tripathi, A. R., and Lipovski, G. J. Packet switching in banyan networks. Proc. 6th. Annual Int. Symp. Computer Architecture, Apr. 1979, pp. 160–167.

47. Widdoes, L. C., Jr. The Minerva multi-microprocessor. Proc. 3rd Annual Int. Symp. Computer Architecture, Jan. 1976, pp. 34–39.

48. Wu, C., and Feng, T. On a class of multistage interconnection networks. *IEEE Trans. Comput.* **C-29** (Aug. 1980), 694–702.

49. Wulf, W. A., and Bell, C. G. C.mmp—A multi-miniprocessor. Proc. AFIPS 1972 Fall Joint Computer Conf., Dec. 1972, pp. 765–777.