FAULT LOCATION IN DISTRIBUTED CONTROL INTERCONNECTION NETWORKS

Nathaniel J. Davis IV
William Tsun-Yuk Hsu
Howard Jay Siegel

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

Abstract -- One class of networks suitable for use in parallel processing systems is the multistage cube network. Unfortunately, the cube network is not fault tolerant and any single failure within the network can prevent some source-destination communications. Cube networks with "extra" stages can be constructed that permit faults to be bypassed -- providing the exact location of the fault is known. This paper focuses on fault location procedures suitable for use in networks that employ distributed routing control through the use of routing tags and message transmission protocols. Faults occurring in the data lines can corrupt message routing tags transmitted over them and thereby cause misrouting of messages. Protocol lines (used in handshaking between network sources and destinations), if faulty, can prevent a message path from being established or can cause the path to "lock-up" once transmission of data has begun. These faults have more pronounced effects on the network performance than faults previously considered for centralized routing control systems. The fault location procedures presented form a logical superset to those of the centralized control systems (where message routing is dictated by the actions of a global control unit) and can be adapted for use in both circuit and packet switching networks.

1. Introduction

With the advent of very large scale integrated circuit technology, relatively inexpensive hardware systems and subsystems are now readily available. The result has been the greater use of multiple-processor system designs that employ processing elements, operating in parallel, to achieve high levels of computational power. The ability of these parallel systems to continue operations, despite the occurrence of faults, is of critical importance.

One class of interconnection networks suitable for use in parallel processing systems is the multistage cube network. This class includes topologically equivalent networks such as the baseline [1], the indirect binary n-cube [2], the generalized cube [3], the omega [4], the STARAN flip [5], and the SW-banyan (S=F=2) [6]. Differences in the networks are due to the control schemes, the inclusion of the broadcast capability, and the numbering of the input and the output ports [7]. The generalized cube will be used as a model of this network class. The cube network is not fault tolerant. Any single point failure in the network will prevent some source-destination pair of functional subsystems from communicating.

Fault tolerance can be introduced into the network through the use of one or more "extra" stages of switches [8, 9, 10]. The extra stages create redundant paths between sources and destinations that can be used to route communications around faults within the network. Effective use of the redundant paths in the network requires that each source know the exact location of any network faults. In most cases, the location of the faults can be determined explicitly through the use of "off-line" test procedures.

Previous work in fault location has concentrated on networks operating under a centralized control scheme [11, 12, 13]. Systems such as PASM [14, 15] and Ultracomputer [16] implement the network using a distributed control methodology. Network faults in a distributed system, especially faults occurring in the interconnecting links within the network, can cause much more severe errors in network operations than could a similar fault in a centralized control system (a result of message misrouting due to the corruption of data tags). In addition, faulty protocol lines (used in handshaking between network sources and destinations) can prevent a path from being established or can cause the path to "lock-up" once the transmission of data has begun. In this paper, the fault location procedures necessary for distributed control networks are considered. These procedures form a logical superset to those of the centralized systems.

In Section 2, the system and network models are defined and the fault model is presented. Section 3 overviews the centralized control fault location procedures of [11]. An outline of the testing procedure for use in a distributed control network is presented in Section 4. Potential network faults, their ensuing effects on the network, and the testing procedure outputs they would produce are presented in Section 5. Section 6 discusses fault location techniques based on the output responses of the testing procedures. Section 7 summarizes these results.

2. The interconnection network model

Consider a parallel processing system consisting of N functional subsystems, where $N=2^n$. The subsystems will be assumed to be *processing elements (PEs)*, processors paired with their own local memories. The interconnection network will have N inputs (sources) and N outputs (destinations). PE i will be connected to network input i and output i. The *generalized cube* network [3, 7] consists of n stages with each stage being composed of N/2 2-by-2 interchange boxes. Interchange boxes in stage *i* pair I/O lines with link labels that differ only in the *i-th* bit position. The same labeling is used for both the input and output lines connected to an interchange box. A generalized cube network is shown in Figure 1, with N=8. Each data path through the network will be m-bits wide, where m is a function of the system hardware used. Circuit switched data transmission is assumed, where a complete path linking the source and destination PEs must be established before the actual data transfer can begin.

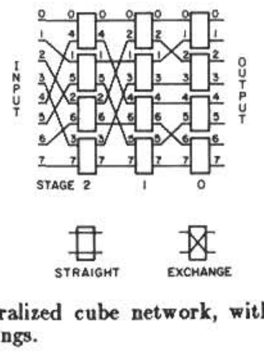


Figure 1. A generalized cube network, with N=8, and the allowable box settings.

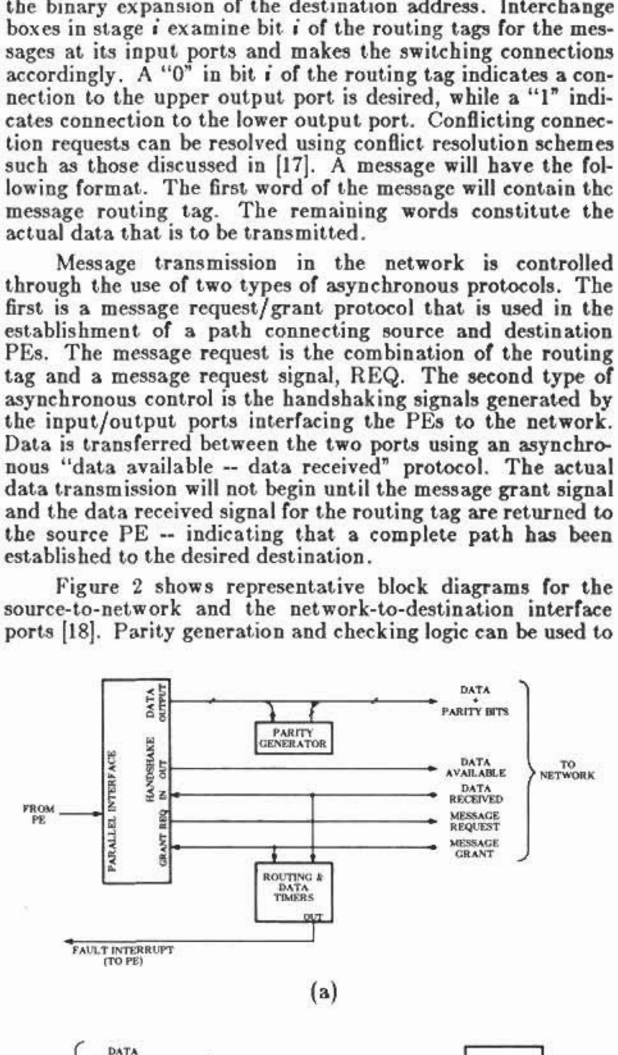
This research was supported by the Rome Air Development Center, under contract number F30602-83-K-0119.

Two different approaches can be used to govern the manner in which connections are made in the network -- centralized routing control or distributed routing control. In centralized control, a global network controller is used to mediate between message requests and establish the desired network connections. In contrast, distributed control removes this serial bottleneck by allowing the individual interchange boxes to establish their own connections based on the use of routing tags associated with each message.

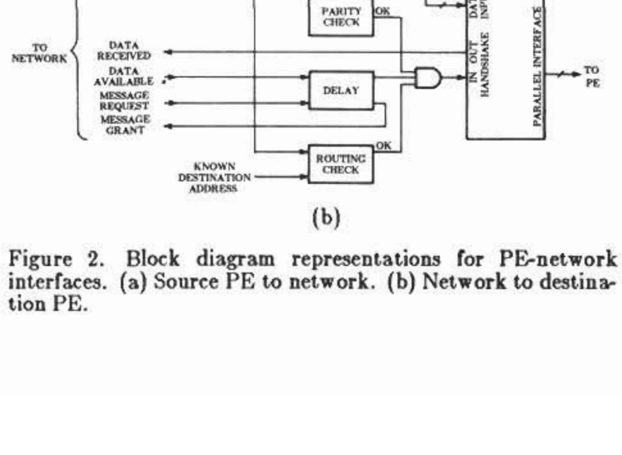
Two functionally equivalent forms of routing tags are the *destination address tag* [4] and the *exclusive-or tag* [3]. Distributed routing control using destination address routing tags is assumed for the rest of this paper. The routing tag is equal to the binary expansion of the destination address. Interchange boxes in stage i examine bit i of the routing tags for the messages at its input ports and makes the switching connections accordingly. A "0" in bit i of the routing tag indicates a connection to the upper output port is desired, while a "1" indicates connection to the lower output port. Conflicting connection requests can be resolved using conflict resolution schemes such as those discussed in [17]. A message will have the following format. The first word of the message will contain the message routing tag. The remaining words constitute the actual data that is to be transmitted.

Message transmission in the network is controlled through the use of two types of asynchronous protocols. The first is a message request/grant protocol that is used in the establishment of a path connecting source and destination PEs. The message request is the combination of the routing tag and a message request signal, REQ. The second type of asynchronous control is the handshaking signals generated by the input/output ports interfacing the PEs to the network. Data is transferred between the two ports using an asynchronous "data available -- data received" protocol. The actual data transmission will not begin until the message grant signal and the data received signal for the routing tag are returned to the source PE -- indicating that a complete path has been established to the desired destination.

Figure 2 shows representative block diagrams for the source-to-network and the network-to-destination interface ports [18]. Parity generation and checking logic can be used to



(a)



(b)

Figure 2. Block diagram representations for PE-network interfaces. (a) Source PE to network. (b) Network to destination PE.

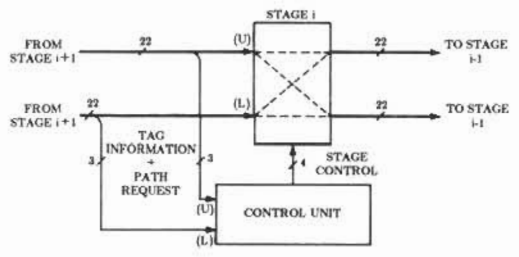


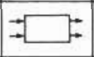
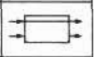
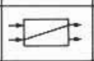
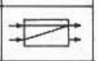
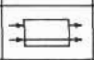
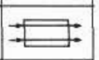
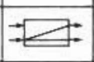
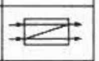
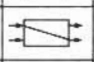
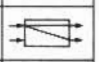
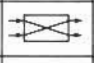
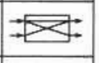
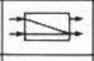
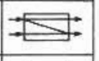
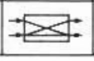
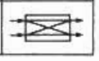
Figure 3. Block diagram of an interchange box.

validate the data received at the output port. For the first word of a transmission (the routing tag) a comparison is also made to the known destination address of the port to validate the routing of the message path. Correct parity and routing results in both the message grant and the data received signals being returned to the source, disabling the routing and data timers. Thus, in normal operations, blockages and potential faults in the network are detected when a routing and/or data timer does not detect the anticipated return protocol signals in a prescribed time period (non-acknowledge signals could also be used in place of the timers). Detection of a possible fault will initiate the execution of the fault location diagnostic routine.

A block diagram of a representative interchange box is shown in Figure 3. The box consists of a 2-by-2 switching element plus the necessary control hardware. The actual width of the information path through the box will be m bits plus one bit for each of the protocol signals and parity bits. The actions of the control unit are determined by the routing information present at its input ports, as stated earlier. This information is "pulled off" of the data path, as depicted in the figure.

A fault within the interconnection network can occur in either an interchange box or in one of the interconnection links within the information paths. All faults will be assumed to be non-transient. As delineated in [11], there are 16 possible ways of connecting the inputs of an interchange box to its outputs, as shown in Table 1. Only the straight and the exchange connection patterns, labeled S_{10} and S_5, are considered to be valid connections (broadcast states S_3 and S_{12} are not considered valid states for this model) [11]. A faulty box can become stuck in one particular invalid state, regardless of the routing information at its input ports, or it can respond incorrectly (but consistently) to the routing information. For

Table 1. Sixteen possible settings for a 2-by-2 interchange box.

State	Interchange Box Setting	State	Interchange Box Setting
S_0		S_8	
S_1		S_9	
S_2		S_{10}	
S_3		S_{11}	
S_4		S_{12}	
S_5		S_{13}	
S_6		S_{14}	
S_7		S_{15}	

an example of the later situation, it is possible for a box to respond correctly when the straight state (S_{10}) is requested, but incorrectly when the exchange state (S_5) is requested. Furthermore, a faulty box may enter one incorrect state (say, S_{11}) in response to a straight state request, and enter a different incorrect state (say, S_{12}) in response to an exchange state request.

A link fault occurs in an information line when it becomes stuck at either logical "0" or "1," regardless of the actual input signal that is applied to it. In centralized control, link faults only occur on lines used to carry data and, as a result, cannot affect message routing. This is a subset of the possible faults when distributed control is used, where faults can occur in either the data lines themselves or the lines carrying the protocol and parity bit signals. As will be seen in Section 4, link faults in distributed control systems can create a large number of network errors due to the corruption of routing tags as they are transmitted over links or the failure of the message handling protocol procedures.

3. Fault location in centralized control networks

In [11], Feng and Wu present a comprehensive method for detecting and locating both link and box faults in a centralized control interconnection network. A two-phase testing strategy is developed to detect faults within the network. The testing of the network is performed in an SIMD (synchronous) operating mode under the direction of a global system control unit. This procedure is illustrated in Figure 4 and described below.

In phase 1, all interchange boxes are set to the straight connection. A logical "0" and a logical "1" are transmitted over each data link through the network. The transmitted data is arranged at the input of the network in such a manner as to insure that, without faults, data at the two inputs of any interchange box is complementary. If the two data items are the same, an error can be assumed to have occurred in a previous stage that caused one of the items to change value. By comparing the received output words (at the destinations) with the known correct output specified by the system control unit, the presence of a fault can be readily determined. In phase 2, the process is repeated with the interchange boxes set to exchange. This two-phase process requires the transmission

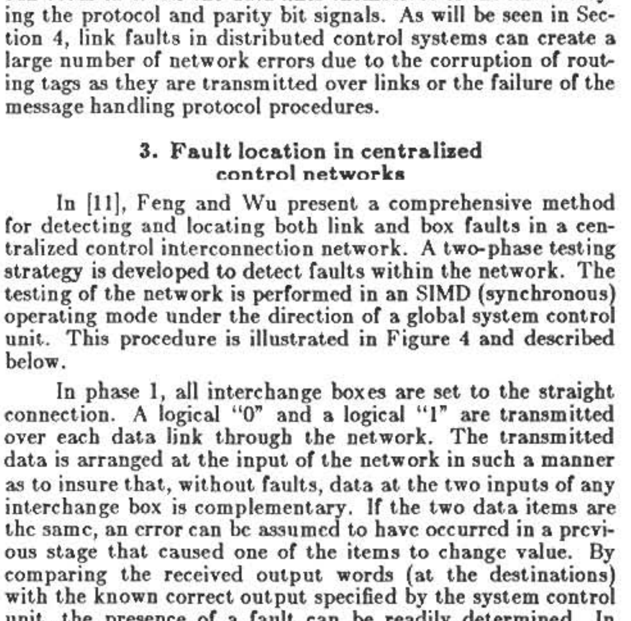


Figure 4. Example network response to test messages in a faulty environment. (a) Link fault detected by intersection of faulty paths. (b) Box fault, erroneous state S_3 in phase 2.

of exactly four data words and has been shown to detect all single faults within the network.

The two-phase test also provides the necessary information to specify the location of any link fault. The location can be obtained by comparing the paths on which detected errors occurred. The paths intersect in exactly one link, identifying the fault location. Interchange box faults can cause one or more errors to be detected at the destination ports. This is a result of the potential misrouting of messages by a faulty box and the ensuing blockages that could then occur (a properly functioning network will not have blockages in either of the two test phases). A binary tree search algorithm can be used to isolate the box faults in no more than $\max(12, 6+2\lceil\log(\log N)\rceil)$ tests. Complete details of the testing procedures can be found in [11].

4. Network fault detection in distributed control systems

The set of possible fault patterns in a distributed control network is more complex than that of a centralized control network. This is because the routing tags which direct the path establishment through the network and the data transmission protocol lines are carried over the same information paths as the data. For example, a link fault may produce an erroneous routing tag which, in turn, may cause the message to be misrouted -- an error that is not possible in a centralized control system.

The procedures and methodology described here for detecting and locating faults in a distributed control network are based on the network model of Section 2 but can be adapted for other cube-type interconnection networks and for different formats of the data path and protocol lines. Specific examples will refer to a generalized cube network, with 16 PEs and a 16-bit data path. The routing tag word will contain the four-bit destination address in bits 3 to 0 of the word. The other bits in the word (bits 15 to 4) are not used. There are two parity bits, a high order bit for bits 15 to 8, and a low order bit for bits 7 to 0 of the 16-bit word. This format is similar to the routing tag format used in the PASM prototype network [18].

The fundamental testing procedure remains similar to that described in [11]. To check for link stuck-at faults, each set of links carrying data or parity bits must have two bit-wise complementary words transmitted over it. Any link that is stuck at logical "1" or "0" will cause a fault message to be generated. The links carrying the protocol signals for distributed routing control are not considered to be fault-free. Procedures for isolating these types of faults will be discussed in Section 5. To test for faults in the interchange boxes, we shall attempt to set each box to the valid states S_{10} and S_5 (straight and exchange). Faults will be detected by examining the test patterns which propagate through the network and are received by the destination PEs and combining this information with the blockage/timeout information available from the source PEs.

The basic testing procedure is divided into two phases. In each phase we shall attempt to detect if there are one or more faulty paths and, through their intersection, isolate the faulty component. In *phase 1,* all boxes are preset to the "straight" setting through the actions of routing tags submitted to the network by the source PEs. Call the process of sending routing tags through the network to preset all paths the ***setup.***

The setup in phase 1 involves several steps. Each source PE sends its own address as the destination address tag. In a generalized cube network, this is equivalent to having the PEs request that all the interchange boxes be set to straight (setting S_{10}). All unused bits in the routing tag data word are set to "0" for convenience.

If some block or routing error is detected (via a timeout or other means), all paths are immediately dropped (by negating REQ) and phase 2 of the test is begun. If, however, no block or routing error occurred, each PE will begin the ***data transfer*** testing subphase of phase 1. The first data word to be transmitted will be the bit-wise complement of the routing word. If again no error is detected, a second data word, with different parity, is sent through the network to ensure the links carrying the parity bits have been tested properly. This is necessary because, for a data path with an even number of bits (as assumed here), the parity bit values will remain the same for the destination tag in setup and the first data word. As an example, consider a binary word with an even number of bits. There are either an even number of both "1"s and "0"s or an odd number of both "1"s and "0"s. Complementing such a word does not change its parity bit. Hence the routing tag word has the same parity as the first data word. The second data word is formed by complementing bits 0 and 8 of the first data word, i.e., the low order bit from each byte.

As an example of this phase of the testing procedure, consider a 16 PE system. PE 6 would have a phase 1 routing tag word of 0000000000000110 with parity bits 00. The first data word would be 1111111111111001 with parity bits 00. The second data word would be 1111111011111000 with parity bits 11. Every data and parity link in the information path will have had both logical "0" and logical "1" signal levels transmitted over it.

In *phase 2*, the setup procedure involves having each source PE send the complement of its address as the destination tag. This is equivalent to requesting that all interchange boxes be set to exchange (setting S_5). As in phase 1, all unused bits are set to "0" for convenience. If no block or routing error occurs in the setup subphase, the processing elements will send as the first data word the bitwise complement of the routing word.

If no error is detected with the transmission of the first data word, an extra data word with different parity is sent through the network to ensure that the links carrying the parity bits have been properly tested. This second data word is formed in the same way as that in phase 1, i.e., by complementing bits 0 and 8 of the first data word. Continuing with the example from phase 1, PE 6 would have a phase 2 routing tag word of 0000000000001001 with parity bits 00. The first data word would be 1111111111110110 with parity bits 00. The second data word would be 1111111011110111 with parity bits 11.

Thus, in each test phase, the word received by a destination PE during setup should be all "0"s except for the low order bits, which will contain the destination's address. The first data word should be all "1"s except for the low order bits, which will contain the bitwise complement of the destination's address. The second data word received should be the same as the first data word except bits 0 and 8 and the parity bits are complemented. The destination's network interface verifies the message routing during setup (by examing the destination address portion of the received setup word) and the parity for every received word. It, in turn, generates the return protocol signals for the source PE. The effects of a fault occurring in the network will depend on its exact location. These effects are discussed in detail in the next section.

5. The effects of network faults

Faults occurring in a network that uses distributed control can cause much more serious operational errors than in a comparable centralized control network. Faults and their concomitant error patterns are discussed below for both link and box faults.

5.1 Error patterns for link stuck faults

Two types of link faults can occur: faults in the data path or parity bit links, or faults in the links which carry the message protocol signals. Table 2 is a complete listing of the errors caused by each type of link fault. In the table, columns "one phase" and "other phase" record the received error signals (if any) and do not necessarily correspond to the phase 1 - phase 2 testing sequence.

Table 2. Fault patterns for link stuck faults.

Fault patterns for Link Stuck Faults					
Case	Faulty Link	one phase		other phase	
		setup	data transfer	setup	data transfer
TYPE 1					
1a	unused bits	E		E	
1b		OK	E	OK	E
2a	destination bit,	E		E	
2b	does not affect routing	OK	E	OK	E
3a	destination bit,	B		OK	E
3b	affects routing	EB		OK	E
4a		OK	E	OK	E
4b	parity bit	E		E	
4c		E		OK	E
TYPE 2					
1	message request stuck negated	B		B	
2	message grant stuck negated	B		B	
3	data available stuck negated	E		E	
4	data received stuck negated	E		E	
5a	message request stuck asserted	OK	OK	EB	
5b		EB		EB	
6	message grant stuck asserted	OK	OK	OK	OK
7	data available stuck asserted, before stage 0	OK	E	OK	E
8	data available stuck asserted, after stage 0	E		E	
9	data received stuck asserted, after stage n−1	OK	E	OK	E
10	data received stuck asserted, before stage n−1	E		E	

Fault pattern notation:
"OK" -- No errors or blocks in that testing subphase.
"E" -- Only 1 PE detects a routing or parity error.
"B" -- Only 1 PE detects a block.
"EB" -- One PE detects a routing error, 1 or more PEs detects blocks.

5.1.1 Type 1: Faults on the data path and parity bits.

Link faults on the data path and link faults on the parity bits generate very similar types of fault patterns, and can be grouped together. Functionally, Type 1 faults can be divided into four cases:

- Case 1: Fault in an unused bit. This is a bit which is not used in the routing word as a bit of the destination tag. No matter how the unused bits are scrambled, messages will still be routed to their correct destinations. Since all unused bits are set to "0" in setup, if a link carrying one of these bits is stuck at "1," there will only be one parity error in each test phase (Type 1 Case 1a of Table 2). If a link carrying the bit is stuck at "0," no error will occur during setup. When the first data word is sent through the network, all unused bits are set to "1." Parity checks will then result in one error in each phase (Type 1 Case 1b of Table 2). Since unused bits are set to specific values, this case is easily recognized by examining the test words received at each destination PE.
- Case 2: Destination bit which does not affect routing. This refers to a bit in a word which has already been examined for routing purposes before it became scrambled by a stuck link. For example, if a link carrying bit 3 of the destination tag has a stuck-at fault between stage 1 and stage 0, the message routing word will not be affected (for routing purposes, bit 3 would have already been examined at stage 3). This type of fault produces the same fault patterns as case 1; the stuck bit produces only one parity error during setup in each phase (Type 1 Case 2a of Table 2), or only one error in each phase when transferring the first data word (Type 1 Case 2b of Table 2).
- Case 3: Destination bit which affects routing. In this case, a bit is scrambled before it has been examined in the setup process. Since a link of this nature carries a "1" in the setup of one phase and a "0" in the setup of the other, in one of the phases there will be no setup error but an error will occur when transferring the first data word. In the other phase, there will be an error in setup. Two routing possibilities may happen. The erroneous bit may successfully request an erroneous path and, in turn, block an otherwise good path -- causing a block and a routing error (Type 1 Case 3b of Table

2), or, if the good path has already been established, the erroneous path will be blocked and the only error in that phase would be a block (Type 1 Case 3a of Table 2).

• Case 4: Faults on links carrying parity bits. Here, a link carrying a parity bit is stuck at "1" or "0." In both test phases, this will either be detected during setup or when transmitting one of the two data words. There will either be errors in both phases when doing data transfers (Type 1 Case 4a of Table 2), routing errors in setup in both phases (Type 1 Case 4b of Table 2), or an error in the setup in one phase and an error in data transfer in the other phase (Type 1 Case 4c of Table 2).

5.1.2 Type 2: Link faults in control links. There are four control lines which can be stuck at asserted or negated: message request, message grant, data available, and data received.

• Case 1: Request stuck at negated. A link carrying a message request signal is stuck in the negated state. To set up an interchange box, a message's request signal must be asserted. If not, the message will not propagate through the box, causing a blocking error to be detected. This will occur in both test phases (Type 2 Case 1 of Table 2).

• Case 2: Grant stuck at negated. A link carrying a message grant signal is stuck in the negated state. A block error will be detected in each phase (Type 2 Case 2 of Table 2). This is immediately identifiable because the routing tag arrives and is gated into the destination, but the source does not receive the grant signal.

• Case 3: Data Available stuck at negated. A link carrying a data available signal is stuck in the negated state. A routing error will occur in each phase because the assertion of the data available signal is never detected by the destination port and, as a result, the data received signal is never returned to the source PE (Type 2 Case 3 of Table 2). This is immediately detected and identified since the source port will receive the message grant signal, indicating that the tag does get to the destination port and should have been gated in.

• Case 4: Data Received stuck at negated. A link carrying a data received signal is stuck in the negated state. This will result in one routing error in each phase since no data received signal returns to indicate that the correct message has been received (Type 2 Case 4 of Table 2).

• Case 5: Message Request stuck at asserted. A link carrying a message request signal is stuck in the asserted state. Because message request is always asserted, the last path established before the link failed will not be dropped and will remain held through the fault isolation procedure. Depending on what this last path was, there are several different fault patterns. If the held path consists of all straight or all exchange box settings, there will be no routing errors or blocks in one phase but a routing error and a block in the other phase (Type 2 Case 5a of Table 2). If the held path consists of a combination of straights and exchanges, there will be a routing error and one or more blocks in each phase (Type 2 Case 5b of Table 2). For example, if the request signal for the upper input of a box is stuck at asserted and is holding the box in the straight setting, this box will appear to be operating correctly in the phase 1 testing. In phase 2, however, the straight setting is still held, causing the request on the lower input (which wants to establish an exchange setting) to become blocked.

• Case 6: Message Grant stuck at asserted. A link carrying the message grant signal is stuck in the asserted state. This will produce no errors or blocks in the normal fault location procedure. The only way to detect this is to deliberately attempt to set up paths which will be blocked in the network and check for the signal being stuck (Type 2 Case 6 of Table 2).

• Cases 7 and 8: Data available stuck at asserted. A link carrying the data available signal is stuck at the asserted state. Assume that the data available signal is edge-sensitive and that active low logic is being used in the network implementation (both reasonable assumptions for typical port handshaking signals). The errors that can be generated will depend on whether the stuck link is before or after stage zero.

In Case 7, a Data available link is stuck at asserted before stage zero (recall from Figure 1 that stage 0 is the network output stage). Since the stuck link is before stage zero, an edge is still produced on subsequent links when the path is first set up. Hence in both phases, no error or block occurs in setup, but errors will occur when transmitting the first data word, since the destination port will not receive the required negated-to-asserted edge on the data available line to gate the data word in (Type 2 Case 7 of Table 2). This is immediately recognized because when transmitting data words, the path has already been set up properly and the data word should always get gated in at the destination.

For Case 8, a Data available link is stuck at asserted after stage zero (on the line connecting stage 0 to its respective network-destination interface port). As a result of the fault, the port never detects an edge transition being produced on this control link. In each phase, there will be one routing error only. The fault patterns generated here are identical to those in Case 3 (Type 2 Case 8 of Table 2).

• Cases 9 and 10: Data received stuck at asserted. Similar to Cases 7 and 8, the errors generated depend on whether the bad link is before stage n−1.

In Case 9, assume that a Data received link is stuck at asserted after stage n−1 (the input stage of the network). As with the data available signal, assume that the data received signal is edge sensitive. An edge will be produced when the path is first established. Hence in both phases, no error or block occurs in setup, but errors will occur when transmitting the first data word, since the source port will not receive the required negated-to-asserted edge on the data received line (Type 2 Case 9 of Table 2).

For Case 10, let a Data received link be stuck asserted before stage n−1. Since the stuck link is between stage n−1 and the source port, no edge is ever produced on this bad link. In each phase, there will be one routing error only (Type 2 Case 10 of Table 2).

5.2 Error patterns for interchange box faults

Interchange box faults in interconnection networks with distributed routing schemes are handled in much the same way as are Feng and Wu's switching element faults in [11]. Differences lie primarily in the additional effects of the m-bit data path and the routing and protocol schemes not addressed in [11]. This class of faults will be described briefly with emphasis being placed on these differences.

Two groups will be considered: a faulty state when S_{10} is the desired state and a faulty state when S_5 is desired. The possible fault patterns for interchange box faults are summarized in Table 3. In the analysis, it is assumed that a box fault will effect all lines of an input port in the same way.

5.2.1 Faulty state in S_{10}. This refers to the condition where the combination of the routing tag and REQ signals request setting a box to S_{10} but, because of a fault in the internal logic of the box, it is set to some other state instead.

From Table 1, there are 15 possible erroneous states. Erroneous states S_0, S_1, S_2, S_4, S_5 and S_8 are straightforward -- messages are either misrouted or blocked, and error-checking hardware at the PEs detect routing errors or blocks. For erroneous states S_3 and S_{12}, one of the messages requesting passage through the faulty box is blocked, while the other message is sent to both output lines. The effect of this will be a blocked message and a routing error signal from the second incorrect output. The box input port hardware is assumed to perform a logical AND operation on the returning protocol signals from each of the box output ports to detect the presence of an error signal from either output link. The error signal is, in turn, propagated towards the respective source PE. The error patterns resulting from these erroneous states are similar to those of [11].

Table 3. Fault patterns for interchange box faults.

Interchange Box Faults					
Desired Setting is S_{10}			Desired Setting is S_5		
Erroneous	Observed Errors		Erroneous	Observed Errors	
	Setup	Data Transfer		Setup	Data Transfer
S_0	2B		S_0	2B	
S_1	EB		S_1	B	
S_2	B		S_2	EB	
S_3	EB		S_3	EB	
S_4	EB		S_4	B	
S_5	2E		S_6a	2E	
			S_6b	OK	2E
S_6a	2E		S_7a	2E	
S_6b	OK	2E	S_7b	OK	2E
S_7a	2E		S_8	EB	
S_7b	E				
S_8	B		S_9a	2E	
			S_9b	OK	2E
S_9a	2E		S_{10}	2E	
S_9b	OK	2E			
S_{11}a	2E		S_{11}a	2E	
S_{11}b	OK	2E	S_{11}b	E	
S_{12}	EB		S_{12}	EB	
S_{13}a	2E		S_{13}a	2E	
S_{13}b	E		S_{13}b	OK	2E
S_{14}a	2E		S_{14}a	2E	
S_{14}b	OK	2E	S_{14}b	E	
S_{15}a	2E		S_{15}a	2E	
S_{15}b	OK	2E	S_{15}b	OK	2E

Fault pattern notation:
"OK" -- No errors or blocks in that testing subphase.
"E" -- Only 1 PE detects a routing or parity error.
"2E" -- Two PEs detect routing or parity errors.
"B" -- Only 1 PE detects a block.
"2B" -- Two PEs detect a block.
"EB" -- One PE detects a routing error, 1 or more PEs detects blocks.

The remaining seven erroneous states, S_6, S_7, S_9, S_{11}, S_{13}, S_{14}, and S_{15}, may involve changes to the routing tags and, therefore, new considerations come to bear. The routing tags can become corrupted when one of the messages at the output links of the faulty box is the result of the two input messages overwriting each other. The effect of having two input bits write to the same output bit is defined similarly to Feng and Wu in [11]. If the two input bits are identical, the output bit would be equal to either one of the inputs. However, if the input bits are different, the output bit would always be a "1" or always a "0," i.e., a overwritten bit position always sticks at the same value. When two m-bit routing tags are transferred to the same output link, some bits in the resulting tag may be scrambled by the overwrite. Depending on the message at the output link of the faulty box, there are two resultant subcases.

In the first case, the message is changed by the overwrite so that an error in the bits of the setup word which specify the destination address is generated. A routing error is detected. For example, in erroneous state S_6, bit i of the setup word using the lower input should be a "1." Assume the two setup words entering the box are merged so that bit i is overwritten and is stuck at "0." Thus, a destination with a "1" in its ith address bit position will receive a setup word with a "0" in the ith bit position. The source PEs requesting a path through the faulty interchange box will detect this error since the error signal is propagated back to both PEs.

In the second case, the message can be changed by the overwrite so that an error in the destination address part of the setup word is not generated as a result of the overwritten bits (e.g., bit i of the message at the lower output of S_6 in Table 1 should be "1" and is stuck at the same value because of the overwrite). One of the setup words (e.g., the lower input in erroneous state S_6) arrives successfully at its expected destination, and the proper return protocol signals are propagated back to both of the source PEs. In this case, no routing error or block is detected in the setup for S_6, S_9, S_{11}, S_{14} and S_{15}. However, the first data word is a bitwise complement of the routing tag, while the overwritten bits always stay at the value they took in the setup. When the first data word propagates through the network, some of its bits are different from what they are expected to be. An error results from the parity check and both source PEs receive the error signal. The result is two routing errors. However, in the setup for S_7 and S_{13}, one of the messages is misrouted regardless of whether the overwritten message was scrambled For example, the message at the upper output of S_7 in Table 1 is misrouted, and the lower input performs a logical AND operation on both returning protocol signals and informs the PE connected to the lower input of the error. In these two states, a single routing error in the setup will result.

Overwrites may also occur on the message request and data available control lines. If the returning message grant and data received control lines are run through a multiplexer, overwrites will not occur. For the message request and data available lines, if the signal resulting from the overwrite is in the asserted state, no extra anomalies will occur. If the resulting signal is in the negated state, two blocks or two routing errors will occur, depending on which of the message request and data available lines are changed by the overwrite.

5.2.2 Faulty state in S_5. This refers to the condition where the combination of the routing tag and REQ signals request setting the interchange box to S_5 but the box is set to some other state instead. As for the case where the desired state was S_{10}, there are 15 possible erroneous states. The way in which fault patterns are generated are very similar to those of S_{10}. Refer to Table 3 for a complete description of the fault patterns generated by each type of fault.

6. Isolation of single faults

Tables 2 and 3 summarize the errors resulting from the different types of faults after the two phases of the testing procedures. The principle behind the testing procedure is to determine the fault type, i.e., whether a link or an interchange box is faulty, and the location of the fault. This is done by recording the path on which faults are detected in each phase and intersecting the faulty paths.

Notice that in the condition denoted by "EB" in the tables, i.e., one PE detects a routing error with one or more PEs detecting blocks, only the path which resulted in the routing error is considered to be the faulty path. This is because the blocked path may be fault free, but it was blocked by a misrouted message. If no routing errors are detected in a test phase but blocks occurred (i.e., in the conditions denoted by "B" and "2B"), paths with blocks are considered faulty paths. This is because there is no detected error in the network which could have caused a fault-free path to be blocked. The exception to this rule is for faults belonging to Group 3, as explained below. In the conditions denoted by "2E," two paths are considered to be faulty paths.

Looking over the tables of results for link faults and box faults, the faulty responses can be divided into several groups. These are considered below.

6.1 Group 1: Two faulty paths in either the setup or data transfer of a single phase

This refers to situations where the error conditions denoted by "2E" and "2B" are detected, either in the setup or in the data transfer. Referring to Tables 2 and 3, these conditions will only be registered if an interchange box were faulty, i.e.,

(a) if the desired state is S_{10}, and the erroneous state is S_0, S_5, S_6, S_7 subcase (a), S_9, S_{11}, S_{13} subcase (a),S_{14} and S_{15}, or

(b) if the desired state is S_5, and the erroneous state is S_0, S_6 , S_7, S_9 , S_{10}, S_{11} subcase (a), S_{13} ,S_{14} subcase (a), and S_{15}.

Since two faulty paths were registered in a single phase, intersection of these two paths will pinpoint the faulty box.

6.2 Group 2: No setup errors in one or both phases, data transfer error in phase(s) where no setup error occurred

In this group, no setup errors occur in one or both setup subphases. Where the setup is valid, a single error will be detected in the ensuing data transfer. This group includes link faults of Type 1 Cases 1b, 2b, 3, 4a and c, and Type 2 Cases 7 and 9. Interchange box faults are not in Group 2 since they never generate a single error in data transfer in one or both phases (see Table 3).

Referring to Table 2, a Group 2 link fault always generates one faulty path in each phase. Since fault patterns of this type can be definitely identified as being caused by link faults, intersection of the faulty path obtained in phase 1 and that obtained in phase 2 will isolate the faulty link.

6.3 Group 3: No anomaly in one phase, only one faulty path in the other phase

This group includes all those conditions where no anomaly (i.e., no routing error, parity error or block) was detected in one phase, but an error and/or block in setup or data transfer was detected in the other phase. Hence if faulty test patterns of this group were detected, only one faulty path will be registered and further tests are necessary to isolate the fault.

From Table 2, only one type of link fault might produce fault patterns of this group: Type 2 Case 5a. The fault pattern produced here in the faulty phase is the condition denoted by "EB". If the fault were in an interchange box, several types of faults would generate patterns belonging to this group.

(a) The faulty box works normally if it were set to S_{10} in the phase 1 test, but in the phase 2 test, instead of being set to S_5, it is set to S_1, S_2, S_3, S_4, S_8, S_{11}b, S_{12} or S_{14}b. Of these eight possibilities, S_1, S_4 ,S_{11}b, and S_{14}b do not produce the condition denoted by "EB" and are thus easily recognized as box faults instead of a link fault of Type 2 Case 5a.

(b) The faulty box works normally if it were set to S_5 in the phase 2 test, but in the phase 1 test, instead of being set to S_{10}, it is set to S_1, S_2, S_3, S_4, S_7b, S_8, S_{12} or S_{13}b. Of these eight possibilities, S_2, S_8, S_7b and S_{13}b do not produce the condition denoted by "EB" and are thus easily recognized as box faults instead of a link fault of Type 2 Case 5a.

For faults belonging to this group, the path that obtained the block error for a condition "EB" can be used to locate the fault because the types of faults that cause fault patterns of Group 3 are all "localized". If an interchange box fault caused an "EB" condition, the location of the faulty component is exactly pinpointed by the intersection of the two paths on which the routing/parity error and the block lie. If the "EB" condition were caused by link fault Type 2 Case 5a, the intersection of the faulty path and the blocked path will give the interchange box immediately after the stuck link. In *this type of fault, if more than one block is detected,* each blocked path is intersected with the path with the routing/parity error, and the component(s) obtained by the intersection that is closest to the input stage of the network is considered to be the faulty component. Hence for faults in this group, if the phase with the faulty path produced an "EB" condition, the faulty component is either the interchange box indicated by the intersection of the two bad paths, or an input link on that box.

For the remaining faults in this group (the non-"EB" cases), a very detailed discussion is given in [11] under the section on "Switching Element Faults," case 2.

6.4 Group 4: Anomalies in the setup of both phases

In this group, one faulty path is generated in the setup of both phase 1 and phase 2. This group covers all the remaining link faults, and also the interchange box faults in which a particular box is set to an erroneous state(s) in both phase 1 and phase 2. The three error conditions possible in the setup are "E," "EB," and "B." All possible combinations of these conditions are summarized in Table 4. From Table 2, it can be observed that link faults which generate fault patterns in the setup of both phases always produce identical fault patterns in both phases. Hence, combinations of fault patterns which are not identical in both phases (Table 4, subgroups 2,3,4,6,7 and 8) are immediately identifiable as interchange box faults. Since two faulty paths are obtained after the two-phase test, the intersection of the faulty paths will locate either the faulty box or a pair of interchange boxes connected by a link. The latter condition requires special procedures to further isolate the faulty component. These procedures, which involve trying to pinpoint the fault by setting boxes in different stages to different valid states, are described in detail in [11].

Table 4. Group 4 fault patterns.

Fault Patterns for Group 4		
Subgroup	phase 1 setup	phase 2 setup
1	E	E
2	E	EB
3	E	B
4	EB	E
5	EB	EB
6	EB	B
7	B	E
8	B	EB
9	B	B

Three subgroups from Table 4 are left. These are handled as follows.

(a) For fault patterns of subgroup 1, the faulty component is either a link fault (Type 1 Cases 1a or 2a, Type 2 Cases 3, 4, 8, or 10), or a faulty interchange box (which in phase 1 takes on the erroneous states S_{13}b or S_7b instead of the desired state S_{10}, and in phase 2 takes on the erroneous states S_{11}b or S_{14}b instead of the desired state S_5). The intersection of the faulty paths would narrow the fault location to a pair of interchange boxes and their connecting links. In this one particular instance, the testing procedure should be repeated, sending the data words despite the setup errors. If the faulty component were an interchange box, the fault type can be determined immediately by sending the first data word through the network after the faulty setup. Referring to Section 5.2 on box faults, one of the data words will be overwritten so that a parity error will result. This error will be detected by both source PEs which are accessing the faulty interchange box. The error condition of having two routing/parity errors in the same phase (i.e., the "2E" condition) is similar to the Group 1 faults, and the intersection of the two bad paths will pinpoint the faulty interchange box. If a link fault were responsible for the fault patterns, only one error would be detected in each phase. Intersection of the two faulty paths obtained from the two test phases will isolate the bad link.

(b) For fault patterns of subgroup 5, the faulty component is either a link fault of Type 2 Case 5b or an interchange box which is not set correctly in both phases (settings S_1, S_3, S_4, or S_{12} instead of S_{10} in phase 1, and setting S_2, S_3, S_8, or S_{12} instead of S_5 in phase 2). In some cases it is impossible to correctly determine the fault location in this subgroup. The faulty component can only be isolated down to a pair of interchange boxes and a link joining the pair, as described in [11] for their faulty box subset of network faults.

(c) For fault patterns of subgroup 9, the faulty component is either a link fault of Type 2 Cases 1 or 2 or a faulty box which takes on settings S_2 or S_8 in phase 1 and settings S_1 or S_4 in phase 2. The faulty component for subgroup 9 (one block in each phase) can only be isolated down to a pair of interchange boxes and a link joining the pair, a condition described in [11]. Only if the error were caused by a link fault of Type 2 Case 2 can the faulty component be further isolated. In this case an illegal combination of control signals is received by the source PE. The lack of message grant implies that a path was not successfully set up through the network, yet the data received signal is returned to the source PE, indicating that the routing tag did arrive successfully at the destination PE. This implies that the link carrying the message grant signal is faulty. Intersection of the faulty paths obtained would locate the bad link. Otherwise, it is difficult to tell if the fault were a link fault of Type 2 Case 1 or some combination of interchange box faults in both phase 1 and phase 2.

After the two-phase test, it is possible to detect all single faults and correctly locate the faulty component (by examining the combination of error signals) or, at worst case, narrow the location down to a pair of interchange boxes and their connecting links. Group 4 presents the most difficult combination of error patterns. In those cases where the fault has not been completely isolated, the special procedures described in [11] are necessary to further isolate the fault.

6.5 Group 5: No anomalies detected

In this group, no errors are detected in any phase of the testing procedure. This indicates the presence of a Type 2 Case 6 link fault (a message grant protocol line is stuck asserted). Knowing the source PE that requested the diagnostic testing and the path that it was trying to set up when the initial network error was detected (the fault was in that particular path), the location of the fault can be determined. As indicated in Section 5.1.2, blocking paths can be systematically established in the network that are known to conflict with the path containing the fault. If the faulty path and a blocking path intersect before the fault location, the grant signal should become negated (as a result of the blockage). If the paths intersect after the fault location, the grant signal will remain asserted, despite the path being blocked. The fault is identified as being in a particular link when a block in the preceding interchange box causes the grant signal to be negated while a block in the succeeding box does not negate the signal. The search for the fault location can be performed in a binary tree search fashion and will require O(log n) steps to complete.

7. Summary

As integrated circuit technology has matured, the design and construction of complex parallel processing systems has become feasible. In these systems, it is crucial to insure that operations can continue in the presence of faults that might occur. This is particularly important in the interconnection networks supporting such systems. A number of techniques exist to introduce fault tolerance into the multistage cube networks class. To be effective, however, the location of a network fault must be explicitly known.

In this paper, an existing fault detection and location procedure for centralized routing control networks has been overviewed. Networks that employ distributed routing control transmit both message routing and protocol information and data through the network. The errors that could occur in these networks were analyzed and shown to be a superset of the errors occurring in a centrally controlled system. Faults occurring in the data lines could corrupt the message routing tags transmitted over them and thereby cause the misrouting of messages. Additionally, protocol lines used in the handshaking between source-destination PE pairs, if faulty, could prevent a message path from being established or could cause the path to become "locked-up" once the transmission of data had begun. A fault detection and location procedure, patterned after the one presented in [11], was developed. The procedure is executed in two phases, where each phase involved the transmission of routing control information to set up the desired network connections and the transmission of up to two data words to test the integrity of the paths. Response patterns to the test messages were derived for link faults in the data path as well as in the protocol links. Interchange box faults and their associated fault patterns were also investigated.

While the procedures described in this paper were specifically targeted to a circuit switched network implementation, the approach could be modified for packet switched networks using similar control protocol structures. In packet switching, protocol lines connect interchange boxes in adjacent stages rather than the sources and destinations in circuit switching. As a result, the effects of faults in these lines will tend to be more localized than in the circuit switched networks discussed in this paper.

References

[1] C-L. Wu and T. Y. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comp.*, Vol. C-29, Aug. 1980, pp. 694-702.

[2] M. C. Pease III, "The indirect binary n-cube microprocessor array," *IEEE Trans. Comp.*, Vol. C-26, May 1977, pp. 458-473.

[3] H. J. Siegel and R. J. McMillen, "The multistage cube: a versatile interconnection network," *Computer*, Vol. 14, Dec. 1981, pp. 65-76.

[4] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comp.*, Vol. C-24, Dec. 1975, pp. 1145-1155.

[5] K. E. Batcher, "The flip network in STARAN," *1976 Int'l. Conf. Parallel Processing*, Aug. 1976, pp. 65-71.

[6] G. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," *1st Symp. Comp. Arch.*, Dec. 1973, pp. 21-28.

[7] H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies*, Lexington Books, Lexington, MA, 1985.

[8] G. B. Adams III and H. J. Siegel, "The extra stage cube: a fault-tolerant interconnection network for supersystems," *IEEE Trans. Comp.*, Vol. C-31, May 1982, pp. 443-454.

[9] K. Padmanabhan and D. H. Lawrie, "A class of redundant path multistage interconnection networks," *IEEE Trans. Comp.*, Vol. C-32, Dec. 1983, pp. 1099-1108.

[10] C-Y. Chin and K. Hwang, "Packet switching networks for multiprocessors and data flow computers," *IEEE Trans. Comp.*, Vol. C-33, Nov. 1984, pp. 991-1003.

[11] T. Y. Feng and C-L. Wu, "Fault-diagnosis for a class of multistage interconnection networks," *IEEE Trans. Comp.*, Vol. C-30, Oct. 1981, pp. 743-758.

[12] D. P. Agrawal, "Testing and fault tolerance of multistage interconnection networks," *Computer*, Vol. 15, Apr. 1982, pp. 41-53.

[13] W. Y-P. Lim, "A test strategy for packet switching networks," *1982 Int'l. Conf. Parallel Processing*, Aug. 1982, pp. 96-98.

[14] H. J. Siegel, L. J. Siegel, F. C. Kemmerer, P. T. Mueller, Jr., H. E. Smalley, Jr., and S. D. Smith, "PASM: a partitionable SIMD/MIMD system for image processing and pattern recognition," *IEEE Trans. Comp.*, Vol. C-30, Dec. 1981, pp. 934-947.

[15] D. G. Meyer, H. J. Siegel, T. Schwederski, N. J. Davis IV, and J. T. Kuehn, "The PASM parallel system prototype," *IEEE Comp. Soc. Spring Compcon 85*, Feb. 1985, pp. 429-434.

[16] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir, "The NYU Ultracomputer -- designing an MIMD shared-memory parallel computer," *IEEE Trans. Comp.*, Vol. C-32, Feb. 1983, pp. 175-189.

[17] M. Lee and C-L. Wu, "Performance analysis of circuit switching baseline interconnection networks," *11th Symp. Comp. Arch.*, June 1984, pp. 82-90.

[18] N. J. Davis IV and H. J. Siegel, "The PASM prototype interconnection network," *1985 Nat'l. Comp. Conf.*, to appear, July 1985.