Fault-Diagnosis in a Multiple-Path Interconnection Network

Harsh Sadawarti
Department of Computer Science & Engeneering, RIMT-Institute of Engineering & Technology,
Mandi Gobindgarh, Punjab, India, harshsada@yahoo.com., Tel :+91-1765-241405

Annotation:
Two pass routing scheme is described for communication in a multiprocessor system employing a unique-path multistage interconnection network in the presence of faults in the network. It is capable of tolerating all single faults and many multiple faults in all except the first and last stages of the network. The routing scheme is useful for tolerating both permanent as well as intermittent faults in the network. The hardware over head for implementing the scheme is very small and no time-penalty is paid in the fault-free case.
While a multiple-path interconnection network is capable of tolerating any single fault, the knowledge of the fault's location is required before it can adapt itself. An approach of on line single-fault detection is given. Based on a new fault-model, a system wide diagnostic procedure is developed to effectively detect and locate a single fault throughout a fault tolerant network as that proposed in (6). The model is realistic and has potential usefulness as a tool for modeling faulty states of larger switching elements (e.g., n x n switching elements, n > 2). Networks under diagnosis behave in a distributed control manner, i.e., a tag needed for establishing a path is conveyed by the same resources (switching elements and links) as those for transmitting data. Test vectors for appropriately setting switching elements when the procedure is conducted are presented. Faults are classified into two different groups each of which is dealt with separately to ease our diagnostic procedure.

Keywords: *parallel processing, multistage interconnection networks, fault-tolerant routing.*

1. INTRODUCTION

Suitable networks for interconnecting lots of processors and/or memory modules have long been recognized as one of the key issues in designing a multiprocessing system (1-3). As system size grows, multistage interconnection networks (MINs) become increasingly attractive because they are more cost-effective than full crossbars and can be controlled in a distributed fashion. Inherently, most of MINs possess the property that a unique path, exists between a primary input (input of the first stage) and a primary output (output of the last stage) (1-5). To such an MIN, any single fault at a switching element (SE) or a link may render some outputs inaccessible from certain inputs. Therefore, featuring an interconnection network with fault-tolerant capabilities is desirable, especially for a high-performance multiprocessor. We have proposed a scheme which is applicable to a wide range of MINs to enhance their fault-tolerant capabilities in (6). It will provide redundant paths between each input/output pair of a network. A network reconstructed by the scheme can tolerate any single fault and many multiple faults as well.
To diagnose a single fault in a network composed of 2 x 2 SEs, a method based on a general fault model is proposed in (9). However, the control signals for setting all SEs are assumed to be provided from a fault-free contralized control unit. Recently, a fault locating technique for distributed control networks in which routing tags are delivered through SEs and links is presented in (13). However, a network under distributed control can be fully diagnosed only when all of the possible situations in an SE are covered. To completely diagnose a 2 x 2 SE, 2^3 possible situations in an SE, i.e., each input port of an SE may have (i) no request; (ii) a request heading for the upper

output; or (iii) a request for the lower output, should all be tested.

In this paper, we propose a new diagnostic procedure for a fault tolerant multiple-path network. In section 2, the fault tolerant network is briefly reviewed. We also present a fault model for such a network. The proposed diagnostic procedure is presented in section 3. Finally, concluding remarks are given in section 4.

2. FAULT-TOLERANT INTERCONNECTION NETWORKS

In section 2.1, we shall give a brief review of our proposed fault-tolerant networks of size M x N for the case of $M = N = 2^n$. A fault model for the augmented 2 x 2 SEs will be described in section 2.2 This fault model can also be applied to SEs of any size (i.e., n x n SEs, n > 2).

A 2 x 2 SE has two input ports and two output ports. As in (14), each port in an SE is attached with a link of B bits wide, B > 1. A request consists of several smaller packets each with B-bit width. However, all the packets of a request are transmitted as a whole in a network.

2.1. Network configuration and routing procedure

An 8 x 8 Baseline network (5) composed of 2 x 2 SEs has three stages. In each stage, an SE is named by $t = (\log_2 N)-1$ bits, $b_0b_1\ldots b_{t-1}$. Each input/output link is named by t + 1 bits, $b_0b_1b_{t-1}b_t$, in which the left most t bits are the same as the binary representation of the SE; the last bit b_1 is 0 if

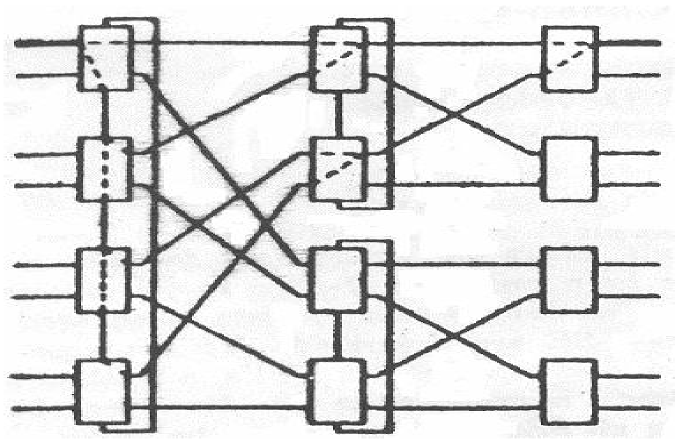

Fig.1. One way to chain SEs in each stage To allow multiple paths.

The link is the upper link and b_t is 1 if it is the lower link. All of the SEs in stage i, $0<i<\log_2 N$, belong to the same partition if $b_0b_1 \ldots b_{i-1}$ is the binary representation of their name, $b_0b_1 \ldots b_{t-1}$, has the same value. All of the SEs in stage 0 constitute a partition.

Our proposed fault-tolerant scheme is simply to chain the SEs in a partition together so that multiple paths between every input/output pair are provided. A network implemented with the scheme is shown in Fig. 1. The dotted lines in the figure show four of the possible redundant paths between a particular input/output pair in the network.
To allow SEs to be chained together, we introduce a chain in link and a chain-out link in each SE. The “augmented” switching element functions like a 3 x 3 crossbar with a modified destination-tag control algorithm. The network cycle is defined to be the switching time of an SE plus time for a packet to be delivered from an input (or chain-in) port through the link to the connected input (or chain-in) port. It is the basic time unit in a network. Any number of SEs in the same partition can form a chain by connecting the chain-in link of an SE to the chain-out link of itself or another SE within the same partition. A complete chain is a chain formed by connecting together all of the SEs in the same partition. For example, all of the chains in Fig.1 are complete chains. A network with chains inside is called a chained network.

Assume a network input, labeled $i_0i_1…i_t$ is to be connected to a network output, labeled $d_0d_1…d_t$. A tag with value d_0d_t is transmitted through the network with the data packets. To simplify our presentation, we will assume that the data path is wide enough to convey an entire tag, i.e., $B \geq t+1$. At stage I, bit d_i of the tag is used to route a request in an "augmented" SE. If $d_i = 0$, it is routed to the upper output link of the SE; if $d_i = 1$, it is switched to the lower output link. If the request can not be routed to the destined output link due to a conflict, a link failure, or a failure of the SE in the next stage to which the output link is connected, it is routed through the chain out link to another SE within the chain. Same d_i bit is utilized with an identical routing algorithm in the new SE. If the request fails again to gain connection to the destined output link in the new SE, it is routed to yet another new SE in the same chain. Eventually a "good" output link can be found and the request proceeds to the next stage. Because an identical routing algorithm is employed, the request will try to use the same (i.e., upper or lower) output link in each SE. Clearly, the routing algorithm proposed above can route a request from an input $i_0i_1...i_t$ to its destination $d_0d_1…d_t$ through multiple paths. More detailed description and analyses of the network can be found in (15).

2.2. Fault model

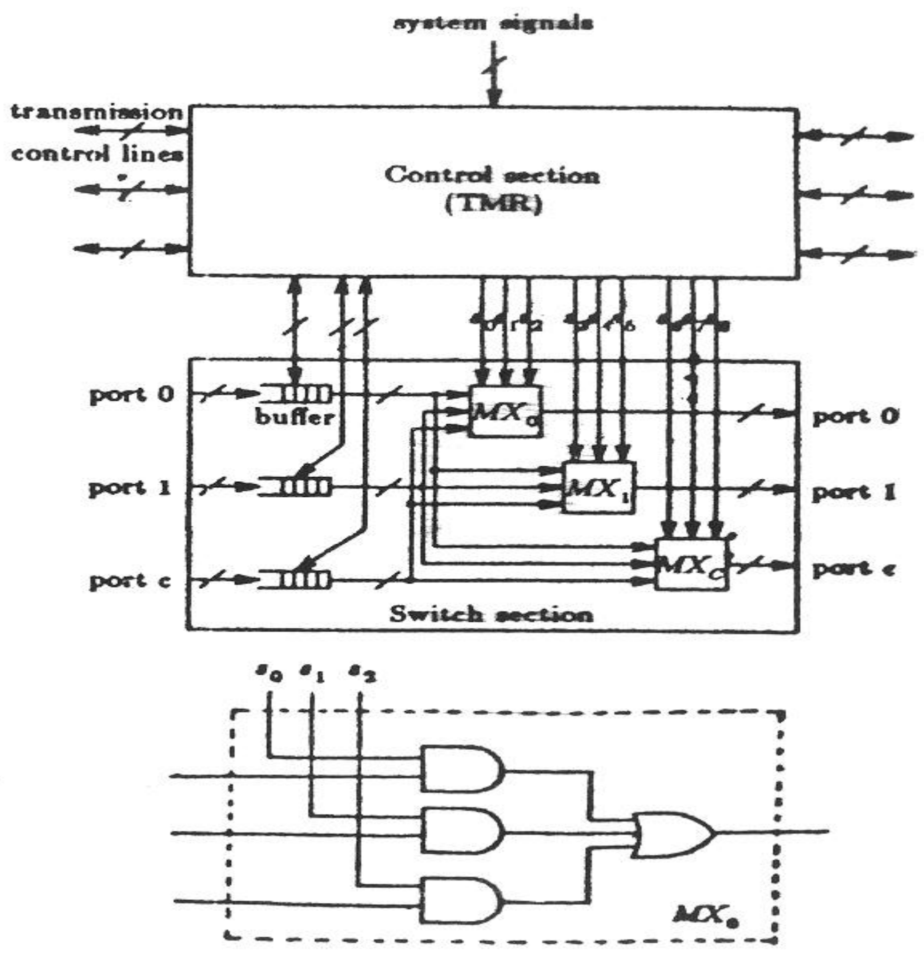

Fig. 2. The configuration of an "segmented" SE.

An "augmented" SE has three input ports and three output ports. Totally, there are2^{3x3} = 512 possible states. To treat the whole SE as an entity and discuss each valid state as in (9) is far too complex. Moreover, it is not easy to generate simple test vectors such that the chain-in port as well as two input ports of an SE have a request each simultaneously to diagnose all potential faults inside because a request is routed to a chain-out port only when there is a conflict in the SE.

Hence, the following fault model (see Fig.2) is assumed to facilitate our diagnostic procedure:
An SE is composed of two principal parts – a control section and a switch section (as in (14)). Communication protocol and other system signals, which handle the handshaking between stages and provide power and clock signals, are connected to the control section. The data path is included in the switch section. Incoming requests are stored at input buffers in the switch section. An input buffer can hold an entire request. Each request supplies its routing tag-bits to the control section where those tag-bits are decoded and the routing signals are sent back to the switch section.

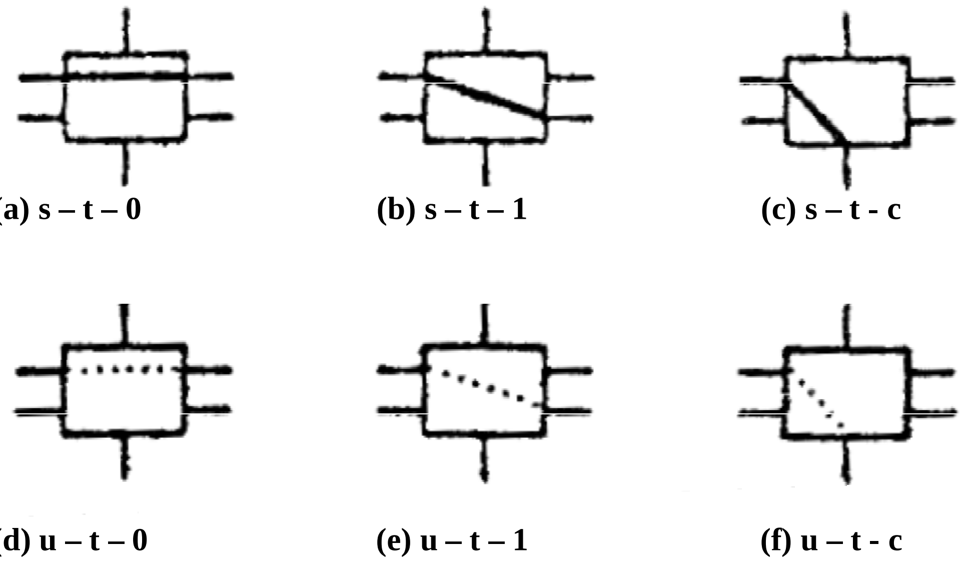

Fig. 3. Cases of faulty input port 0.

The behaviour of faulty SE depends very much on the implementation of the SE. To have a realistic fault model and, in the mean time, to avoid as many implementation dependent aspects as possible, we assume that any fault arising in the control section results either as a stuck-at-0 or a stuck-at-1 fault on one of those routing signals ($s_0 \ldots s_2$ in Fig. 2) This is not unreasonable because one way to achieve this is via a triplicate modular redundant (TMR) design of the control section (since the major portion of the hardware of an SE is in its switch section which usually has a wide data path). In such a design, the solely vulnerable part of the control section lies in the final routing signals generated after the TMR voting mechanism.

Those routing signals enable one of the paths in each “multiplexer” (MX for short, see Fig. 2). Notice that only one of the routing signals to an MX can be active at any time, i.e., only one of the AND gates can be enabled to allow only one input port to be connected to the output port. Faults occur when those routing signals are faulty. An input port could be “stuck” to one of the output ports (or chain-out port) if the corresponding routing signal is stuck-at-1. For example, input port 0 could be stuck to output port 0 (s-t-0), output port 1 (s-t-1), or chain-out port (s-t-c) if s_0, s_1 or s_2 is stuck-at-1, respectively. On the other hand, an input port may not be able to reach one of the output ports (or chain-out port) if the corresponding routing signal is stuck-at-0. For example, input port 0 is unable to reach output port 0 (u-t-0). Output port 1 (u-t-1) or chain-out port (u-t-c) if s_0, s_1, or s_2 is stuck-at-0.

Depending on the situation of an SE an input port or the chain-in port may be with one the following faults : s-t-0, s-t-1, s-t-c, u-t-0, u-t-1 and u-t-c as shown in Fig. 3.

Other parts of an SE are also possible to fail

(1) Control lines which handle the handshaking process between stages may fail. A set of control lines is employed to control data transmission over a link. The employed control lines are treated as a whole. A set of protocol lines is said to be faulty if any one in it fails. We assume such a failure will cause ports under its control to be inaccessible. Hence, a request is lost when it is sent to a port with faulty protocol control lines.
(2) System lines such as clock signals or power lines may fail. We assume the whole SE becomes inoperative and any request sent to the SE is lost.
(3) Input buffers may fail. We assume it can be detected by some kind of error detecting scheme like cyclic redundancy code (CRC).
(4) A data link which connects an output port to an input port of the next stage may have one of its data lines.(1 data link is of B-bit width) stuck-at-0 (s-a-0) or at-1 (s-a-1). Since routing tags are transmitted through the network along with data packets, a faulty data link not only damages the integrity of delivered data but also corrupts the routing tags. It may result in the whole request being directed to a wrong destination.
(5) A gate in a multiplexor of the switch section may fail. We assume it can only lead to the stuck-at-0 or stuck-at-1 type of faults and behaves like the link connected to the multiplexer being faulty.

3. DIAGNOSIS IN CHAINED NETWORKS

A fault-diagnostic procedure is invoked when (i) a fault is detected by an online fault detecting scheme or (ii) a fixed span of time has elapsed since the last invocation. An on-line fault detection can be achieved by the following :
(1) Each request carries both a destination tag and the one’s complement of the tag. Whenever a request teaches a primary output, its tag is checked against the address (label) of the primary output. Misrouted requests can be detected immediately. Carrying one’s complement of the destination tag is for detecting stuck-at type of faults on the data path.
(2) Each request also carries some kind of error detecting code like CRC (cyclic redundancy code) or SECDED (single error correction double error detection) code to check the integrity of the data.

The on-line fault detecting scheme can detect most of the faults occurring in a network. For a fault-tolerant network, however, if a request is directed simultaneously to two or more paths which lead to the same primary output, the situation is impossible to be detected. For instance, if an input port of a certain SE is s-t-c, a request on the faulty port may be routed to an output port and the chain-out port at the same time. Hence, a periodical invocation of a fault-diagnostic procedure is necessary.
Since the time between two regular invocations can be appropriately chosen so that the probability of multiple faults arising between two invocations is very small, we consider only the diagnosis of single-faults in this paper. The possible faults are :

Group 1 : Faults on input/output parts or their associated links faults in input buffers and in operative SE due to faulty system signals or faulty control signals.

First of all, let us rename the inputs of a Baseline network to facilitate our diagnostic procedure. The primary inputs of an 8 x 8 chained network are relabled through a bit-reversal pattern (the new network is an Omega network [4]) During the diagnostic course, all primary inputs of the network will generate test vectors synchronously. We also assume the network is the size N ($=2^{t+1}$).

3.1. Diagnosis for faults in Group 1

A test request contains multiple packets : the first two packets carry destination tag and its one's complement; the last packet carries an error detecting code like CRC and the rest of the packets are data. Following diagnostic procedure is for the faults.

Procedure :

Step 1 : Every network primary input *I* generates a request of maximum allowable length to primary output *I*. Data in each request is a string of "0".

Step 2 : Every network primary input *I* generates a request of maximum allowable length to primary output $J = N\text{-}1\text{-}I$ (i.e., the one's complement of *I*). Data in each request is a string of "1".

These two diagnostic steps are basically the same as in [9]. They are employed to set each SE to a "straight" state (i.e., input port 0 connected to output port 0 and input port 1 connected to output port 1) and a "cross" state (i.e., input port 0 connected to output port 1 and input port 1 connected to output port 0) during Step 1 and Step 2, respectively. In each test step, every primary output checks the arriving request. It may receive no request or more than one request due to the presence of a fault. An observed request at a primary output is called a **response**. A fault **syndrome** of a faulty network is the observed faulty situation at its primary outputs. Let TD denote the tag received at a primary output D and C_D the one's complement of the tag received. A response is referred to as a wrong response if (i) $T_D \neq D$ and $C_D \neq \overline{D}$; (ii) Type B: $T_D = D$ and $C_D \neq \overline{D}$ (iii) Type C: $T_D \neq D$ and $CD = \overline{D}$.

Faults are further divided into 6 classes. Their fault syndromes and locating processes are illustrated below.

Faults in Class 1 are those faults arising at input ports which may be s-t-0/1 or u-t-0/1. During Step 1 test, a s-t-1 upper input port (or a s-t-0 lower input port) results in its request and the request from the lower input ports (or the upper input port) being routed to the same output port. They are merged (by the OR gate in a multiplexer, see Fig.2) into a request which proceeds to an undetermined primary output because the outcome of two mixed requests is logically unpredictable. During Step 2 test, however, same two input ports can route their requests to the correct output ports as if there were no fault.

An u-t-0/1 input port behaves similarly. During Step1 test, an u-t-0 upper input port (or an u-t-1 lower input port) prevents its request from going to the upper (or lower) output port. The result of the output port, thus, is unpredictable. During Step 2 test, however, both requests can be directed to the proper output ports as if there were no fault.

Hence, for this type of faults, only one of the two steps can create observable fault syndromes. During the test step which a wrong response is observed, (i) if every primary output received a request, then, the output which got the fault response is of interest; or (ii) if a primary output received no request, then, the one is of interest. From the address of the primary output of interest, we can derive the primary input which issued the request that (i) was mingled for a s-t-0/1 input port or (ii) was missing for an u-t-0/1 input port. Even though a unique path starting from the primary input can be identified, the exact fault location on the path is still impossible to be pinpointed. Therefore, a binary search by applying more test vectors to find its actual location is required.

In order to locate the fault among the possible t+1 input ports along the path, we divide the input ports along the path into two equal (or near equal) halves and try to see which half the fault resides. Then, divide that half into two equal (or near equal) parts again and repeat this process until the fault is located.

At search step r, when the candidate port may stay in stages between stage 0 and stage m, or in the rest stages, each primary input $I = b_0b_1...b_t$ generates a test request with tag J = $b_0b_1...,b_m, \overline{b}_{m+1}...\overline{b}_t$, where $\overline{b}$ is the one's complement of b. During the course of the search, every issued request contains only its tag and the one's complement of the tag. It is clear that the number of search steps required is at most $[\log_2(t+2)]$, where $[z]$ stands for the smallest integer larger then or equal to z. Knowing the stage which the fault is in, we can obtain a unique SE on the path. The faulty input port is one of the two input ports of the SE. The input port on the path is faulty for a s-t-0/1 case; whereas the input port on the path is faulty for an u-t-0/1 case.

Faults in Class 2 are input ports with the s-t-c (stuck to chain-out port) fault. During either step of Procedure, the request reaching a faulty input port with spawn another request to the chain-out port because a request on the faulty port is always directed to the chain-out port as well. This newly generated request will also arrive without damage at the destined primary output, i.e., the output will receive two identical requests. In this case, we are able to recognize two primary inputs, one at each test step, which generated the requests that passed through the faulty port. By tracing the paths of the two requests generated by the two primary inputs, we can locate the faulty input port because both the two requests passed the fault.

Class 3 faults are link faults. A link is considered faulty if one of the B data lines in the link is s-a-0/1. Since only a single fault in a network are considered, we assume only on data line in a link may fail at a time. Each link in a size N ($=2^{t+1}$) network is composed of no less than t+1 data lines (i.e., a link can carry a tag entirely). For simplicity, we assume that a link comprises t+1 data lines: $d_0,d_1,..., d_t$. A link is said to be in stage s if it is an input link of an SE in stage s. The next theorem summarizes the syndromes of link fault.

Theorem 1: For a network with a link fault, exactly one primary output observes a wrong response during each diagnostic step. Furthermore, those two wrong responses observed belong to the same type (i.e., either Type B or C) and there is at most a conflict in the network when procedure is applied.

Because one primary output in each test step can observe a wrong response, we can identify two primary inputs from which the requests passing the faulty link are generated. These two requests traverse solely a common link. i.e., the faulty link. By comparing T_D and C_D in a wrong response collected, we may acquire the failed data line number in the faulty link because two corresponding bits, one in T_D and the other C_D, transferred by the failed data line are equalized.

Class 4 faults are caused by faulty protocol control lines. There is a missing request during each test step because exactly one request passes through a port (and a link). The request which encounters the fault is trapped there because a sender (i.e., a primary input, an input, or a chain-in port) may fail to get a grant to start transmitting a request all the time or fail to finish its transmission once it begins due to a fault in protocol signals. The cost request is definitely unable to proceed forwards and is lost. Notice that an output port in a stage can be treated as an input port in the subsequent stage. An inaccessible output port and an inaccessible connected input port are not distinguishable because they result in the same faulty syndrome. If a handshaking process fails, each share which participates in the process has no way to be exactly isolated. By following the paths that are traversed by the two missing requests as before, we can localize a region that consists of an input port, an output port, and the set of protocol lines monitoring data transmission between them. It is a component of the region that fails.

Class 5 faults are caused by faulty system lines like clock signals or power lines. They make the whole SE inoperative. All of the requests passing through the SE are lost. Hence, in each test step of Procedure, two missing requests are observed. The inoperative SE should be at the point where those paths of missing requests cross each other.

Input buffers in an SE are utilized to hold requests waiting for proceeding forwards. Recall that a string of “0” (or “1”) is included in each test request during Step 1 (or 2). The string of data is only for detecting any fault at an input buffer. Faulty input buffers (Class 6 faults) are assumed to be detectable by an error detecting scheme (e.g., the CRC checking). Depending on the implementation of buffers and the position of the fault, one or two wrong responses can be observed during the entire course of Procedure. If merely one wrong response is observed, a binary search is required to locate the faulty buffer; otherwise, a direct trace is applicable.

4. CONCLUDING REMARKS

We have proposed an on-line fault detection scheme, a fault model, and a diagnostic procedure for a multiple-path network [6]. Even though we assume our networks to be packet switching multiple-path networks, the diagnostic procedure proposed can also be applied to a circuit switching network or a network without multiple paths.

REFERENCES:

(1) IEEE Trans. Computers, Special Issue on Interconnection Networks for Parallel and Distributed Processing, Vol. C-30, Apr. 1981.

(2) K. Hwang and F.A. Briggs, Computer Architecture and Parallel Processing, McGraw-Hill, 1984.

(3) H.J. Siegel, Interconnection Networks for Large-seals Parallel Processing: Theory and Case Studies, Lexington Books, 1985.

(4) D.H. Lawris, “Access and Alignment of Data is as Array Processor,” IEEE Trans. Computer Vol C-24, pp-1145-1155, Dec. 1975.

(5) C.L. Wu and T.Y.Feng, “On a Class of Multistage Interconnection Networks,” IEEE Trans. Computers, Vol. C-29, pp. 694-702, aug. 1980.

(6) N.F. Tseng, P.c. Yew, and C.q. Zhu, “A Fault-Teleras Scheme for Multistage Interconnection Networks,” Psec. 12th Int’l Symp. Computer Architecture, June 1985, pp-368-375.

(7) D.P. Agrawal, “Testing and Fault Tolerance of multistage Interconnection Networks,” IEEE Computer, Vol. 15, pp. 41-53, Apr. 1982.

(8) K..M. Falaverjani and D.K. Pradhan, “Fault-Diagnosis of Parallel Processor Interconnection Networks,” Prec. 11th Int’l Symp. Fault-Tolerant Computing, June 1961, pp. 209-212.

(9) T.Y. Feng and C.L. Wu, “ault-Diagnosis for a Class of Multistage Interconnection Networks,” IEEE Trans. Computers, Vol. C-30, pp. 743-758, Oct. 1981.

(10) W.Y.P. Lim, “A Test Strategy for Packet Switching Networks,” Proc. 1982, Int’l Conf. Parallel Processine, Aug. 1982, pp. 96-98.

(11) T.Y. Feng and I.P. Kao, “On Fault-Diagnosis of Some Multistage Networks,” Proc. 1982 Int’l Conf. Parallel Processing, Aug. 1982, pp. 99-101.

(12) T.Y. Feng and Q. Zhang, “Fault-Diagnosis of Multistage Interconnection Networks with Four Valid States,” Proc. 5th Int’l Conf. Distributed comput. Systems, May 1985, pp. 218-226.

(13) N.J. Davis TV, W.T.Y. Hsu, and H.J. Siegel, “Fault Location Techniques for Distributed Control Interconnection Networks,” IEEE Trans. Computers, Vol. C-34, pp. 902-910, Oct. 1985.

(14) W.Lin and C.L. Wu, “ Design of a 2 x 2 Fault-Tolerant Switching Element,” Proc. 9th Ann Symp. Computer Architecture, June 1982, pp. 181-189.

(15) N.F. Tzeng, P.C. Yew, and C.Q. Zhu, “Fault-Tolerant Interconnection Networks for High- Speed Multiprocessors,”. Submitted to IEEE Trans. Computers for publications.

Article received: 2005-12-20