Fault identification and routing in Benes networks

Nabanita Das *, Jayasree Dattagupta

Electronics Unit, Indian Statistical Institute, 203, B.T. Road, Calcutta 700035, India

Received 13 March 1995; revised 18 May 1995; accepted 22 January 1996

Abstract

An $N \times N$ Benes network $B(n)$ $(n = \log_2 N)$, being a rearrangeable network, can realize any $N \times N$ permutation in a single pass. But even in the presence of a single switch fault in $B(n)$, two passes are necessary to realize any $N \times N$ permutation. In this paper, we attempt to characterize the switch fault sets in $B(n)$, in the presence of which the network is always capable of realizing any arbitrary $N \times N$ permutation P in two passes, such that every source-destination connection is set up in a single pass, i.e., no recirculation is needed, but the whole set of N source-destination connections of P is partitioned in two subsets and are realized in two successive passes. An algorithm has been developed to test if the faulty $B(n)$ is capable of realizing any arbitrary permutation in two passes by our technique; if it is yes, another algorithm also has been presented that computes the fault-tolerant routing to realize any arbitrary permutation P in two passes, through the faulty $B(n)$. Finally, for this routing technique, the exact locations of the faults are not important, only the information of some optimal regions around the fault is sufficient. This feature actually enables us to develop very fast and simple procedures for identification of faulty regions of $B(n)$, in the presence of multiple switch faults. Therefore, this fault-tolerant routing scheme enables us to make $B(n)$ fault-tolerant in the presence of an easily-testable class of multiple switch faults, without any recirculation through intermediate nodes, or any reconfiguration of the system.

Keywords: Multistage interconnection networks (MINs); Rearrangeable networks; Unique-path full-access networks; The Benes network; Fault-tolerant routing; Looping algorithm; Conflicts

1. Introduction

Multistage Interconnection Networks (MINs), are used in a multiprocessor or a multicomputer system for communicating among different modules of the system. Several $N \times N$ MINs with $O(n = \log_2 N)$ stages and $N/2$ switches in each stage have been proposed in the literature for use in SIMD or MIMD supercomputers. For example, the Memphis switch of the GF11 supercomputer is a three-stage Benes network, whereas the NYU ultracomputer uses an

Omega network, to interconnect N processors to N memory modules. Now, the failure of a switch or a link in these interconnection networks can bring down the entire system or cause a severe degradation in the performance, unless sufficient measures are provided to make the network tolerant to such failures.

In the n-stage unique-path full-access $N \times N$ MINs, like baseline, omega, reverse-baseline, etc., even a single fault in a switching element or a link, destroys the full-access property. On the other hand, in the rearrangeable networks, such as Benes, $\Omega\Omega^{-1}$, etc., with $(2n-1)$ stages of switches, multiple paths originally exist between any source-destination pair. Therefore, pure software approach of fault-tolerant routing is possible for these networks, without any recirculation, or reconfiguration of the system in a degraded mode. Most of the existing fault-tolerant routing schemes for rearrangeable networks, investigated the Benes network, with very limited fault-tolerant capability [1,4,5]; the serious limitation of all these fault-tolerant routing schemes is that, only the *control-line-stuck-at* fault in Benes network has been considered. This fault model does capture only a very small subset of all the faulty situations that may arise in real practice. However, another more realistic fault model, the *switch fault model*, the strongest among all the fault models of MINs, also has been analyzed in some literatures. In [3], the looping algorithm [6] has been extended to achieve fault tolerance and graceful degradation in the system performance, in the presence of both *switch fault* and *control-line faults*. In [7], a fault-tolerant routing scheme is proposed to tolerate multiple control-line faults in each stage, and in the presence of switch faults, it develops a graceful degradation routing scheme to make the loss of resources minimal. But these strategies of reconfiguring the system in a degraded mode, may result in an enormous wastage of resources and loss of parallelism; moreover, it might require redesign of algorithms to be executed by the system.

In this paper, we analyze the behaviour of Benes network in the presence of switch faults. Earlier studies on fault-tolerance in the presence of switch faults, reported some routing schemes that reconfigure the system in a degraded mode. But here, we will show that it is possible to exploit the large amount of inherent redundancy present in Benes network to tolerate multiple switch faults without reconfiguring the system in a degraded mode. By our technique, each source–destination path is realized in a single pass, i.e., no recirculation through intermediate modules is required. Here, we characterize the switch fault sets, in the presence of which, the Benes network is capable of realizing any permutation in two passes; an algorithm is also presented that checks if a given fault set belongs to this class. Given any arbitrary permutation P, we assume that the routing of P through the fault-free $B(n)$ is already known. Now, in the presence of switch faults of this class, some of the paths will fail in the first pass, when P is routed according to the original routing tags. An algorithm is developed to find alternative paths for the unsuccessful connections such that all the remaining paths can be set up simultaneously in the second pass.

It is true that this technique is applicable for a restricted class of noncritical multiple switch faults only. A fault set is noncritical, if in the presence of the fault set, at least one path exists between every possible source–destination pair. In the presence of critical faults, only recirculation may resolve the problem in some cases, but there are situations, e.g., faults in either first and/or last stages of the network, where some inputs/outputs become totally disconnected from the rest of the network and even recirculation fails to connect them. If we assume that for any $N \times N$ Benes network, the first and last stages are always fault-free, it can be shown that the class of multiple switch faults, considered here, covers all single faults, and more than 79% of all possible double faults, and more than 75% of all possible triple faults for $N = 8$ and 16; and also it

shows that for a given number of faults, the coverage increases with increase in N. Therefore, it is evident that the number of multiple faults considered here, is quite large.

Another interesting feature of this fault-tolerant routing technique is that, for its implementation, it is not necessary to know the exact location of the switch fault, but only the identification of a subnet, which is a localized region of the network around the fault, is sufficient. It simplifies the fault detection and location procedures for this technique extensively. Earlier results on testing and fault-diagnosis of Benes network, considered either control-line stuck-at faults [1,7], or a single switch fault only [2]. Here, it has been shown that just two one-bit test vectors are sufficient to detect and locate all single faults, all double faults and also some other cases with 3, 4, 5 and 6 switch faults.

It is evident that this fault-tolerant routing technique is more powerful than the earlier ones. Here, we need not reconfigure the existing system in a degraded mode, in the presence of faults. Moreover, no recirculation is required, i.e., each source–destination path is set up in single pass. The only degradation in the performance of the system comes into in the form of an additional pass. It also covers all single faults and more than 70% of double and triple faults. Moreover, the test strategies required for this fault-tolerant routing, are also very fast and simple.

Section 2 presents some preliminaries. The fault-tolerant routing and the fault-identification procedures have been described in Section 3. Section 4 summarizes the results.

2. Preliminaries

The recursive structure of an $N \times N$ Benes network $B(n)$ is shown in Fig. 1: the switches are labelled as $S_{i,j}$, where i denotes the stage, and j represents the position in a stage, $0 \leqslant i < 2n-1$ and $0 \leqslant j < N/2$. In $B(n)$, the two $B(n-1)$s are designated as $B_0(n-1)$ (the upper one), and $B_1(n-1)$ (the lower one) respectively, as shown in Fig. 1.

The structure of $B(n)$ shows that at any stage i, $0 \leqslant i < n-1$, the switches $S_{i,j}$, $0 \leqslant j < N/2$, can be divided into 2^i disjoint subsets $SS_{i,k}$, $0 \leqslant k < 2^i$, such that each subset $SS_{i,k}$ forms the first stage of a $B(n-i)$, designated as $B_k(n-i)$. Now $SS_{i,k} = \{S_{i,j}$, such that $k.2^{n-i-1} \leqslant j < (k+1).2^{n-i-1}\}$. The upper and lower outputs of all the switches of $SS_{i,k}$, are the inputs to the $B_{2k}(n-i-1)$ (the upper one), and $B_{2k+1}(n-i-1)$ (the lower one), respectively. The

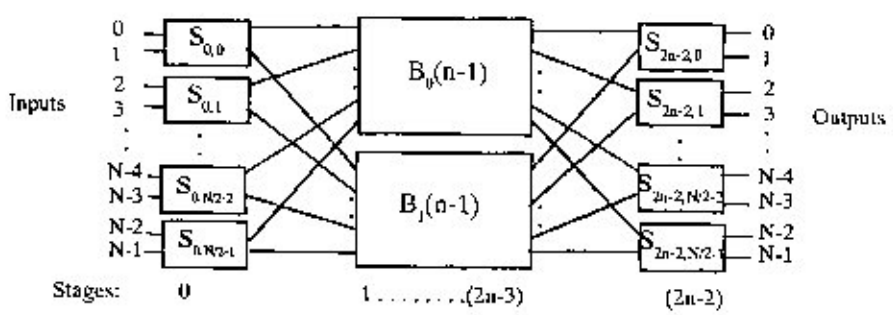


Fig. 1. An $N \times N$ Benes network $B(n)$, $n = \log_2 N$.

same is also true from reverse side for switches in stages i, $n-1 < i < 2n-1$.

Definition 2.1. In a $B(n)$, a $B_p(n-i)$, $0 \leqslant p < 2^i$, is said to be the *subnet* of a $B_q(n-j)$, $0 \leqslant q < 2^j$ and $j < i$, if and only if $B_p(n-i)$ is completely included in $B_q(n-j)$.

For any i, a pair of subnets of $B(n)$, namely $B_{2k}(n-i-1)$ and $B_{2k+1}(n-i-1)$, are said to be the *conjugate* of each other, if the inputs of both are the outputs of the same switch subset $SS_{i,k}$, and the outputs of both are the inputs of the switch subset $SS_{2n-2-i,k}$.

Fig. 2 shows a Benes network $B(4)$ and, the subset of switches $SS_{1,0}$, as well as the corresponding conjugate subnets $B_0(2)$ and $B_1(2)$.

Given an arbitrary $N \times N$ permutation P to be realized in a $B(n)$, we apply the looping algorithm [6] to generate the conflict-free set-up for the switches. We assume that for a given P, each source–destination path is represented by a $(2n-1)$-bit tag, referred here as the routing tag, and is formally defined below:

Definition 2.2. For an $N \times N$ permutation P, realized in a $B(n)$, each source–destination path is represented by a $(2n-1)$-bit tag $x_{2n-2}x_{2n-3}\ldots x_1x_0$, such that at any stage i, $0 \leqslant i < 2n-1$, if the path traverses via the upper output link of the switch, then $x_{2n-2-i}=0$, otherwise $x_{2n-2-i}=1$. The bit string $x_{2n-2}x_{2n-3}\ldots x_1x_0$ is referred to as the *routing tag* (*R-tag*) of the source–destination path.

Lemma 2.3. *For any source–destination path in $B(n)$, if a bit x_{2n-2-i} of the R-tag, $0 \leqslant i < n-1$, is complemented, the modified R-tag will set up the same source–destination connection.*

Proof. In a $B(n)$, let the R-tag $x_{2n-2}x_{2n-3}\ldots x_1x_0$ represent a given source–destination path. At stage

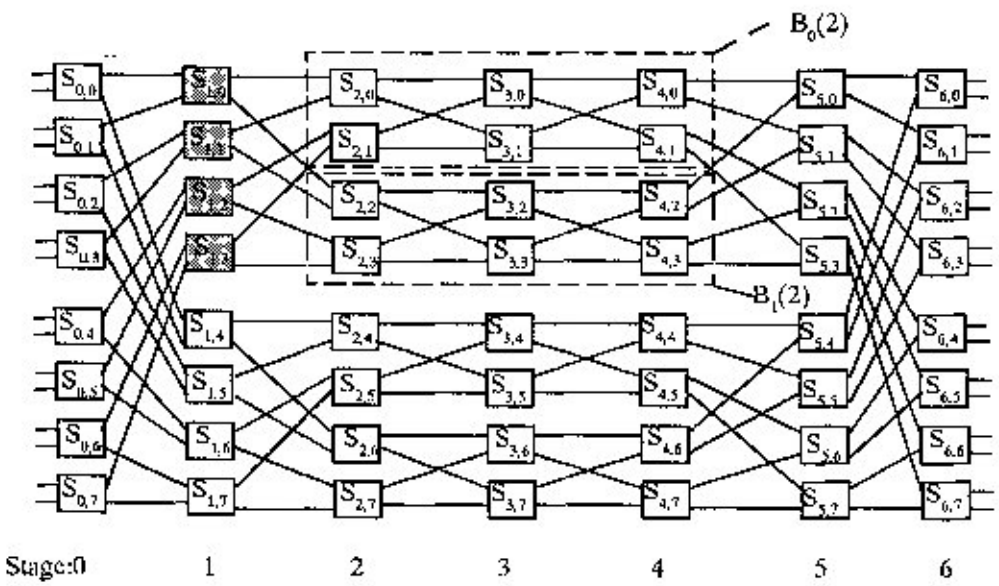


Fig. 2. A Benes network $B(4)$, the dashed blocks show two conjugate $B(2)$'s corresponding to a set of switches $SS_{1,0}$ (shaded ones).

i, $0 \leq i < n-1$, the path passes through a switch belonging to some $B_k(n-i)$. It is clear that if $x_{2n-2-i}=0$, the path enters the subnet $B_{2k}(n-i-1)$, otherwise it goes through the $B_{2k+1}(n-i-1)$. If the bit x_i is complemented, the path changes from $B_{2k}(n-i-1)$ to $B_{2k+1}(n-i-1)$, or vice versa. Since the part $x_{n-1}x_{n-2}\ldots x_1x_0$ remains unaltered, by the last n stages of $B(n)$, which is a unique-path full-access MIN, the path will ultimately lead to the same destination. This proves the lemma. □

Lemma 2.4. *Given any set of conflict-free paths in $B(n)$, if some bit x_{2n-2-i} of each R-tag, $0 \leq i < n-1$, is complemented, the modified paths will remain conflict-free.*

Proof. Let a set of m conflict-free paths have been routed in a $B(n)$, in pass 1, $m \leq N$.

Next, say some bit x_{2n-2-i}, $0 \leq i < n-1$, of each R-tag has been complemented, and again these m connections are routed in pass 2.

According to Lemma 2.3, in pass 2, due to the change in bit x_{2n-2-i} of each R-tag, the sets of inputs of the subnets $B_{2k}(n-i-1)$, and $B_{2k+1}(n-i-1)$, $0 \leq k < 2^i$, will get interchanged. Therefore, the outputs of $B_{2k}(n-i-1)$ in pass 2 are same as those of $B_{2k+1}(n-i-1)$ in pass 1, and vice versa. Hence, in pass 2, the upper (lower) input of each switch at stage $(2n-2-i)$ will be the same as the lower (upper) input of the same switch in pass 1. Therefore, the setting of each switch of this stage will be reversed. Each switch at the next stage will have the same inputs in pass 1 and pass 2 respectively. Since now onwards, it is just destination tag routing, the same source–destination connections will be set up in pass 2, as those in pass 1. This proves our lemma. □

These ideas about the redundant paths existing in a $B(n)$, lead us to develop our fault-tolerant routing scheme for $B(n)$.

3. Fault-tolerant routing in $B(n)$

Before going into the details of fault-tolerant routing, first let us present the fault model assumed by us. Three fault models are commonly used to represent all faulty situations of a MIN. These are the *stuck-at fault model*, the *link fault model*, and the *switch fault model*. The *switch fault model*, is the strongest of the three. In fact, the effect of control-line stuck-at fault or the link fault can be subsumed by the effects of switch fault. Here, we will consider the switch fault model, assuming that the switch is so designed that any physical defection of it causes both the outputs of the switch to be faulty. In this fault model, a failure makes the switch totally unusable, i.e., both the outputs of the switch are disconnected from its inputs. Therefore, in the presence of any number of switch faults, at least two input–output paths will fail.

The fault-tolerant routing technique developed here, is designed to tolerate the switch faults. In the presence of switch faults, it is evident that no $N \times N$ permutation can be implemented in a single pass. We are interested in finding out alternative paths in the faulty network, such that all the faulty paths are realized in one additional pass only. Here follow some definitions.

Definition 3.1. The set of faulty switching elements in a Benes network, is defined as the *fault set F*.

Definition 3.2. A *critical fault set F* is one, under which there exists at least one source–destination pair, such that each possible path for it, passes through some switch in F, i.e., the full-access property of the network is lost in the presence of F.

Definition 3.3. Any fault set F, in the presence of which there exists at least one fault-free path for any source–destination pair in a $B(n)$, is defined as a *noncritical fault set*.

Here, we will consider the routing in the presence of *noncritical* fault set in Benes network. In the presence of critical fault sets, either recirculation, and/or reconfiguration of the system in a degraded mode are the only possible solutions.

Here, we assume that given any $N \times N$ permutation P, the R-tag for each source–destination path is already computed, by the looping algorithm [6], that realizes P in a single pass in $B(n)$, under a fault-free-condition. In the presence of faults, if P is routed according to the computed R-tags, some paths will be unsuccessful. Now let us propose the routing technique that will modify the original R-tags for the faulty paths to find out alternative paths under different noncritical fault sets. Before that, here are some definitions and a lemma that will help us in describing our fault-tolerant routing techniques.

Definition 3.4. For a single fault $F = \{S_{i,j}\}$, with $0 \leqslant i < n$, a subnet $B_k(n-i)$ is the *cover* of F, such that $S_{i,j}$ is a switch at the first stage of $B_k(n-i)$; for $i \geqslant n$, $B_k(i-n+2)$ is the *cover* of F, such that $S_{i,j}$ is a switch at the last stage of $B_k(i \quad n+2)$.

Definition 3.5. In $B(n)$, given a noncritical fault set F, let MC be the minimal set of subnets, such that the cover of each member of F is either a member of MC or a subnet of a member of MC. Then MC is said to be the *minimal cover*.

By definition, for a given fault set F in $B(n)$, MC is unique.

Example 3.6. The fault set $F = \{S_{2,1}, S_{3,0}, S_{3,1}, S_{3,7}, S_{4,0}\}$ in $B(4)$, is shown in Fig. 3.

We find that the covers for $S_{2,1}$ is $B_0(2)$, for $S_{3,0}$ is $B_0(1)$ (a subnet of $B_0(2)$), for $S_{3,1}$ is $B_1(1)$ (a subnet of $B_0(2)$), for $S_{3,7}$ is $B_3(1)$, for $S_{4,0}$ is $B_0(2)$.

Hence the minimal cover of F is given by MC = $\{B_0(2), B_3(1)\}$.

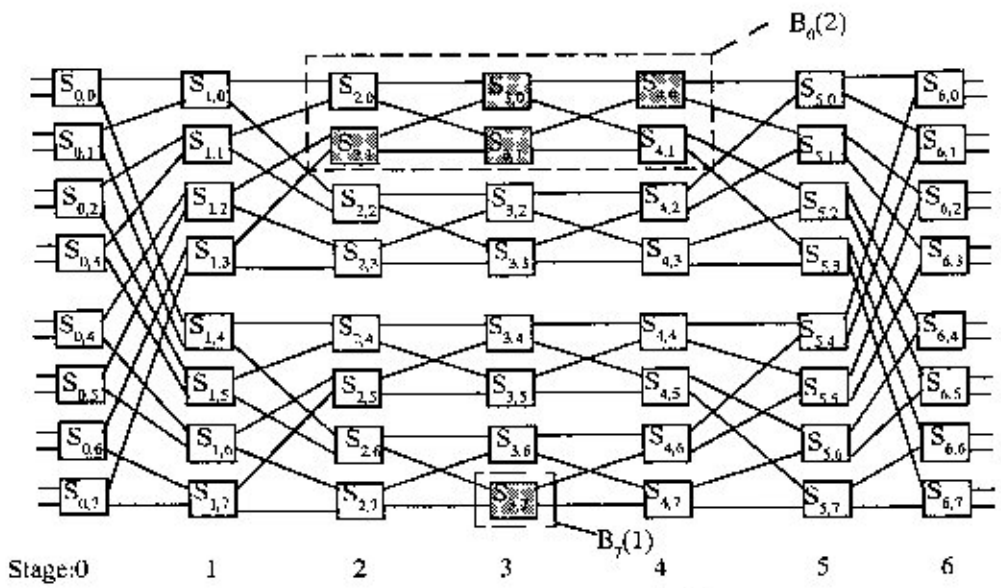


Fig. 3. A fault set F and its minimal cover in $B(4)$.

Definition 3.7. In a $B(n)$, if the minimal cover MC of a given noncritical fault set F, contains any pair of conjugate subnets $B_{2k}(n-i)$ and $B_{2k+1}(n-i)$, we replace the pair by $B_k(n-i+1)$: replacing every such pair of MC, we get the *optimal cover OC* of F; OC is same as the MC, if there exists no pair of conjugate subnets in MC.

Example 3.8. The minimal cover of the fault set $F=\{S_{3,0}, S_{3,1}, S_{3,3}\}$ in $B(4)$ is $\mathrm{MC}=\{B_0(1), B_1(1), B_3(1)\}$.

Since it contains a pair of conjugates $\{B_0(1), B_1(1)\}$, the optimal cover of F is $\mathrm{OC}=\{B_0(2), B_3(1)\}$.

For another fault set $F=\{S_{2,0}, S_{3,7}, S_{3,1}\}$, $\mathrm{MC}'=\{B_0(2), B_7(1),\}=\mathrm{OC}'$, since MC' does not contain any pair of conjugate subnets.

Theorem 3.9. *Two passes are sufficient to realize any permutation P in $B(n)$, in the presence of a fault set F, if, for every $B_k(n-i)\in \mathrm{OC}$, the optimal cover of F, the conjugate $B_{k'}(n-i)$ is fault-free.*

Proof. Let us consider a permutation P. Let $\mathrm{OR}(P)$ denote the set of R-tags for routing P in fault-free $B(n)$. In the first pass, route P through $B(n)$, according to $\mathrm{OR}(P)$. Let a noncritical fault set F be present in $B(n)$; OC be the optimal cover of F, such that for every $B_k(n-i)\in \mathrm{OC}$, the conjugate $B_{k'}(n-i)$ is fault-free. Now, in the presence of F, $\mathrm{OR}(P)$ will fail to set up some paths of P. Let $\mathrm{FOR}(P, F)$ denote the set of R-tags for the faulty paths.

Now, for every $B_k(n-i)\in \mathrm{OC}$, let $\mathrm{SOR}_j(P, F)$ represent the subset of $\mathrm{FOR}(P, F)$, that includes the R-tags of all the paths passing through the faulty $B_k(n-i)$. By definition, $\mathrm{SOR}_j(P, F)\cap \mathrm{SOR}_m(P, F)=\emptyset$, for $j\neq m$.

For each R-tag of $\mathrm{SOR}_j(P, F)$, the bit x_{2n-i-1} is complemented. These new R-tags will divert all the paths in $\mathrm{SOR}_j(P, F)$ from $B_k(n-i)$ to $B_{k'}(n-i)$. Since $B_{k'}(n-i)$ is fault-free, by Lemmas 2.3 and 2.4, all the paths in $\mathrm{SOR}_j(P, F)$ can be realized in the same pass.

The same technique is applied for each subset of $\mathrm{FOR}(P, F)$, corresponding to each member of OC, to find out alternative paths for the faulty ones. By definition, no two members of OC are subnets of each other, i.e., all members of OC are disjoint. It implies that all the alternative paths, given by the modified R-tags of $\mathrm{FOR}(P, F)$ will be conflict-free and hence can be realized in the same pass. Therefore, it proves the theorem. □

The above theorem leads us to identify the fault sets, in presence of which we can always realize any $N\times N$ permutation through a $B(n)$, in two passes. The condition stated in Theorem 3.9, is referred to as the *condition for two-passability*.

Remark. In the presence of any noncritical single fault in $B(n)$, it is evident that two passes are necessary and sufficient to realize any arbitrary permutation.

3.1. Algorithm for finding the optimal cover

For a $B(n)$, given a fault set $F=\{S_{i,j}, 0\leq i\leq 2n-1, 0\leq j<N/2\}$, the following algorithm finds the optimal cover OC for F. The variable C denotes the cardinality of OC.

Algorithm *Optimal Cover*

Input: the fault set F
Output: the optimal cover OC

1. $\mathrm{OC}:=\emptyset$; $C:=0$;
2. For each $S_{i,j}\in F$, if $i>n-1$ then $S_{i,j}\leftarrow S_{2n-2-i,j}$
3. Sort all $S_{i,j}$ s of F in ascending order of i; switches with same i are sorted in ascending order of j, and multiple occurrences of same element are replaced by a single occurrence. Let $F=\{f_0, f_1, \ldots, f_m\}$ be the sorted list.
4. For $q=0$ to m repeat:
   4.1. Take $f_q=S_{i,j}$; $k:=\lfloor j/2^{n-i-1}\rfloor$;

If $k \bmod 2 = 0$ then $k' = k + 1$ else $k' = k - 1$;

4.2. If $B_k(n-i) \in$ OC then goto step 4.5 else if $B_{k'}(n-i) \in$ OC then replace $B_{k'}(n-i)$ of OC by $B_{\lfloor k/2 \rfloor}(n-i+1)$ and goto step 4.5

4.3. For $r = 1$ to C repeat:

4.3.1. Take $OC(r) = B_v(v)$; if $v > (n-i)$ then if $\lfloor k/2^{v-(n-i)} \rfloor = u$ then goto step 4.5

4.3.2. Next r

4.4. $C = C + 1$; $OC(C) = B_k(n-i)$;

4.5. Next q

5. *Terminate*

It is evident from the above algorithm, that if $|F| = p$, the worst case time complexity of the algorithm *Optimal Cover* will be $O(p^2)$.

3.2. Algorithm to check two-passability condition

For a $B(n)$, given a fault set F, the algorithm *Optimal Cover* presented in the preceding subsection, will construct the set $OC = \{B_u(v),\ 1 \leqslant v \leqslant n$ and $0 \leqslant u \leqslant 2^{n-v}\}$. Next we present an algorithm which will test whether a given OC satisfies the condition for two passability, as is stated in Theorem 3.9. The boolean variable "*success*" is true, if the condition is satisfied, otherwise, it is false. C denotes the cardinality of OC.

Algorithm *Two-Passability*

Input: optimal cover OC
Output: the boolean variable "*success*"

1. *success* := 0;
2. If $B_0(n) \in$ OC then terminate
3. Sort all $B_u(v)$ s of OC in descending order of v.
4. For $x = 1$ to C repeat:

4.1. Take $OC(x) = B_k(j)$; If $k \bmod 2 = 0$ then $k' = k + 1$ else $k' = k - 1$;

4.2. For $y = x + 1$ to C repeat:

4.2.1. Take $OC(y) = B_u(v)$; If $j \geqslant v$ then if $\lfloor u/2^{j-v} \rfloor = k'$ then terminate

4.2.2. Next y

4.3. Next x

5. *success* := 1 and terminate

For a fault set F with cardinality p, the maximum cardinality possible for its optimal cover OC is also p. It is easy to see that the algorithm is of time complexity $O(p^2)$, for an OC with cardinality p.

3.3. Algorithm for two-pass-routing

Here, we will assume that the given fault set F, present in $B(n)$, has been processed by algorithm *Optimal Cover*, that outputs OC, the optimal cover of F. Next, we assume that OC satisfies the two-passability condition, i.e., any arbitrary $N \times N$ permutation P is routable in two passes through the faulty $B(n)$.

Now let us consider an $N \times N$ permutation P. Let $OR(P)$, the set of R-tags to route P in fault-free $B(n)$, be known. P is routed by OR(P) in the faulty $B(n)$. Some paths will fail. Let $FOR(P, F)$ be the set of R-tags of the faulty paths. Our algorithm will utilize OC to modify the R-tags of $FOR(P, F)$, so that the new R-tags can route all the faulty paths in second pass. In order to achieve that, we are to identify the set of faulty paths, that pass through each subnet belonging to OC. Now a path with R-tag $R = x_{2n-2}x_{2n-3}\ldots x_0$, passes through a subnet $B_k(n-i)$, $1 \leqslant i < n$, if and only if $\lfloor R/2^{2n-2-(i-1)} \rfloor = k$. The R-tags are then modified accordingly, as has been described in the proof of Theorem 3.9.

Algorithm *Two-Pass Routing*

Input: (i) array $OC(C)$, each element $OC(q)$ is a subnet $B_k(n-i)$, $1 \leqslant q \leqslant C$, and

(ii) array FOR(P, F)(m), each element is of the form $R_i = x_{2n-2}x_{2n-3}\ldots x_0$, $1 \leqslant i \leqslant m$

Output: array FOR(P, F)

1. For $p = 1$ to m repeat:
   1.1. For $q = 1$ to C repeat:
      1.1.1. Take OC(q) = $B_i(n-i)$;
         If $\lfloor R_p/2^{2n-2-(i-1)} \rfloor = k$ then complement the bit x_{2n-i-1} of R_p
      1.1.2. Next q
   1.2. *Next p*
2. *Terminate*

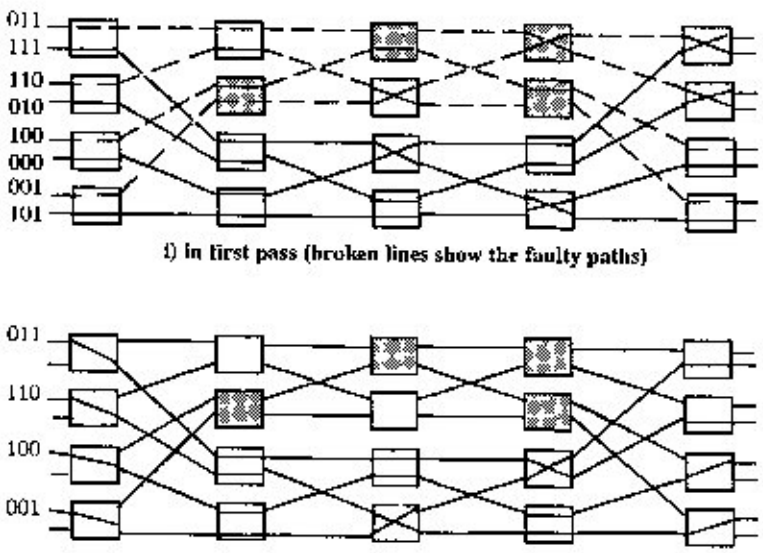
The time complexity of the above algorithm is $O(C \cdot m)$, where C and m are the cardinalities of OC and FOR(p, F) respectively.

Example 3.10. Let $F = \{S_{1,1}, S_{2,0}, S_{3,0}, S_{3,1}\}$ in a $B(3)$. The optimal cover of F is given by OC = $\{B_0(2)\}$. OC satisfies the condition for two-passability.

The routing of P: (3 7 6 2 4 0 1 5) in two passes through the faulty $B(3)$, are shown in Fig. 4.

3.4. Fault coverage

Now we are capable of routing any arbitrary permutation through a Benes network $B(n)$ in the presence of any noncritical switch fault set satisfying the condition of two-passability. It is obviously true that the fault set satisfying the condition for two-passability is a restricted class of faults. Now to find out the exact values of fault-coverage, we assume that all the switches in the first and last stages of a $B(n)$ are robust enough, so that they always remain fault-free. In fact, a failure in a switch of the first (last) stage, essentially disconnects two inputs (outputs) from the rest of the network, and there is no way out except



i) in first pass (broken lines show the faulty paths)

ii) in second pass

Fig. 4. Routing of P: (3 7 6 2 4 0 1 5) in presence of $F = \{S_{1,1}, S_{2,0}, S_{3,0}, S_{3,1}\}$.

Table 1

	n	No. of all possible faults	No. of faults observed	Fault-coverge (%)
Double faults	3	66	52	79
	4	780	160	93
	5	6,216	6,032	97
Triple faults	3	220	160	73
	4	9,880	8,384	88
	5	2,27,920	1,95,392	86

reconfiguration of the $N \times N$ Benes network into a $(N-2) \times (N-2)$ network. It is a critical fault, where even recirculation fails to connect those detached inputs (outputs). As we are concerned with fault-tolerant routing without any reconfiguration and/or recirculation, the assumption about the fault-free first and last stages is essential here.

With this assumption, Table 1 shows the fault-coverages for different number of switch faults with different values of N. The table shows that the coverage is quite good for cases with double faults and triple faults as well, and more interestingly, as N increases, the coverage increases. Therefore, this technique will be very much beneficial for use in supercomputers with large values of n, that employs a Benes network, for interconnecting different modules.

3.5. Fault detection and location in $B(n)$

From the algorithms described above, it is evident that the specific locations of the faulty switches are

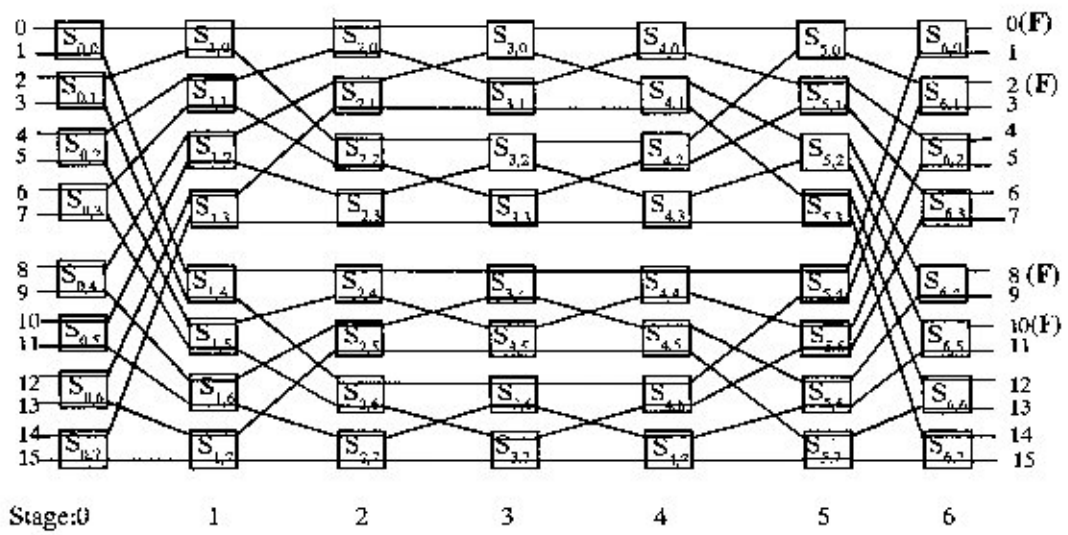


Fig. 5. A fault set $F = \{S_{5,6}, S_{5,2}\}$ and faulty paths {0, 2, 8, 10} in test phase 1.

not important here, instead the *optimal cover* of the given fault set actually determines whether the faulty $B(n)$ satisfies the *two-passability* condition; and if it is yes, what will be the alternative routes for the faulty paths. Here follows a definition.

Definition 3.11. In a $B(n)$, two fault sets are called *equivalent*, if and only if, their optimal covers are same.

Therefore, for implementing our fault tolerant routing scheme, it will be sufficient, if we can locate a fault set equivalent to that actually present in the $B(n)$.

With our switch fault model, if any switch is faulty, both of its outputs will be disconnected from its inputs. Also, we are assuming that the links are fault-free. Therefore, just one test bit (either a 0 or a 1) at each input of $B(n)$, setting all switches in straight mode, will be sufficient to detect the idle outputs, which correspond to the paths containing at least one faulty switch. Therefore fault detection will be complete just in one step. We call it *test phase* 1. It is evident that, in test phase 1, a $B(n)$ tries to realize the identity permutation.

Example 3.12. Fig. 5 shows that in the presence of a fault set $F = \{S_{3,0}, S_{1,2}\}$ in $B(4)$, *test phase* 1 detects the set of faulty outputs $S = \{0, 2, 8, 10\}$, i.e., these outputs are disconnected from the respective inputs.

After the detection of the faulty outputs in *test phase* 1, we are to detect the switch fault set F, or any of its *equivalent* fault sets F', which will be sufficient to follow the fault-tolerant routing technique, described above to determine the alternative routes.

Let in the presence of a fault set F in $B(n)$, *test phase* 1 detects that k outputs $\{O_1, O_2, \ldots, O_k\}$, $k \leqslant N$, are disconnected from the respective inputs. It signifies that each of the paths $I_j \rightarrow O_j$, $1 \leqslant j \leqslant k$, passes through at least one faulty switch in $B(n)$.

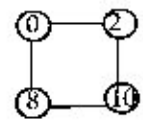


Fig. 6. Intersection graph for faulty paths {0, 2, 8, 10}.

where $I_j \rightarrow O_j$ represents a path from input j to output j. From now onwards, we will represent the path $I_j \rightarrow O_j$ by O_j only.

Definition 3.13. In a $B(n)$, in the presence of a fault set F, let k outputs $\{O_1, O_2, \ldots, O_k\}$, $k \leqslant N$, be faulty. Now let us construct a graph $G(V, E)$, where $V = \{O_1, O_2, \ldots, O_k\}$, and two vertices O_i and O_j are adjacent, if and only if, the two paths $I_i \rightarrow O_i$ and $I_j \rightarrow O_j$ in *test phase* 1, meet at least in one switch in $B(n)$. $G(V, E)$ is called the *intersection graph of the faulty paths* (IG) of $B(n)$, in the presence of the fault set F.

Example 3.14. In the presence of $F = \{S_{3,0}, S_{3,2}\}$ in $B(4)$, as has been shown in Fig. 5, *test phase* 1 detects the set of faulty outputs $S = \{0, 2, 8, 10\}$. The *intersection graph of the faulty paths* IG, is shown in Fig. 6.

Remark. In *test phase* 1,

(i) at any switch $S_{i,j}$, if the upper input (output) be x, the lower input (output) is $(x + 2^m)$, where, $m = i$, for $0 \leqslant i \leqslant n - 1$, and $m = 2n - 2 - i$, for $n \leqslant i < 2n - 1$.

(ii) exactly two switches of $B(n)$, $S_{i,j}$ and $S_{2n-2-i,j}$ for $0 \leqslant i < n - 1$, and $0 \leqslant j < N/2$, have identical pair of inputs (outputs).

Now for any $B(n)$, with a fault set F, we can summarize the properties of IG as follows:

(i) there exists no isolated vertex.

(ii) the degree (d) of any vertex lies in the range, $1 \leqslant d \leqslant n$.

(iii) two vertices are adjacent if and only if they are at unit hamming distance.

(iv) IG is an incomplete hypercube of order n with number of vertices p, $2 \leqslant p \leqslant 2^n$.

Note that an edge of IG, between vertices, say x and $x + 2^i$, corresponds to a pair of switches $S_{i,j}$ and $S_{2n-2-i,j}$ for $0 \leqslant i \leqslant n-1$, i.e., the presence of the vertices x and $x + 2^i$, in IG, signifies that either $S_{i,j}$ or $S_{2n-2-i,j}$ or both may be faulty in the network. Fortunately, all the three cases result the same OC (*optimal cover*), i.e., they all are equivalent. Therefore, the presence of the edge between x and $x + 2^i$, will be interpreted by us as a single faulty switch $S_{i,j}$ only.

Remark. Each line cover of IG, corresponds to a probable switch fault set existing in $B(n)$ that results in the same set of faulty outputs in *test phase* 1, all of them are not necessarily *equivalent*.

Example 3.15. Let us consider the *intersection graph of the faulty paths IG* shown in Fig. 6.

The line cover {0–8, 2–10} corresponds to the actual fault set $F = \{S_{3,0}, S_{3,2}\}$ existing in $B(4)$ as has been mentioned in Example 3.12.

Note that {0- 2, 8–10} is another line cover of IG, that corresponds to the switch fault set $F' = \{S_{1,0}, S_{1,2}\}$.

But F and F' are not *equivalent*, since their optimal *covers* are different.

Therefore, given the set S of failed outputs in *test phase* 1, we may construct IG uniquely, but if there exist a number of possible line covers of IG, which are not equivalent, to select the proper one, we need an additional test phase, namely *test phase* 2.

In *test phase* 2, all switches are set as cross/interchange mode, and a single bit test vector (either a *0*, or a *1*) is presented at the input. Again due to the fault set present (F), some outputs will remain disconnected. This set of failed outputs S' may be different from S, as was obtained in *test phase* 1. In fact, a switch $S_{i,j}$ having the upper input x in *test phase* 1, will get an input x' in *test phase* 2, where,

$$x' = \begin{cases} x \oplus (2^i + 2^{i-1} + \cdots + 2^0), \\ \quad \text{for } i \leqslant n-1 \\ x \oplus (2^{n-1} + 2^{2n-2-i} + \cdots + 2^0), \\ \quad \text{for } n-1 < i \leqslant 2n \quad 2, \end{cases}$$

($\oplus$ denotes bit-wise Ex-OR opertion).

For detailed analysis, we will categorize the results of *test phase* 1, according to k, the number of faulty outputs. It is evident that different fault sets

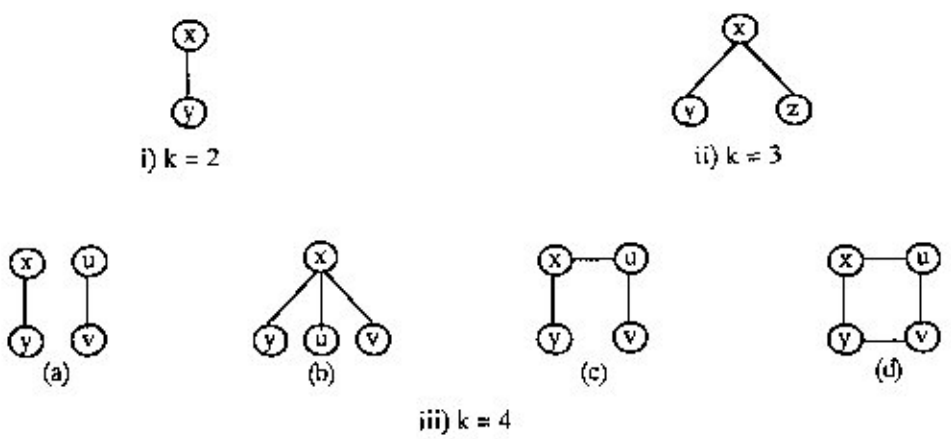


Fig. 7. Intersection graphs for different values of k.

including different number of faulty switches may result the same number of faulty outputs in *test phase* 1. Here, we will analyze only the cases of 2, 3, and 4 faulty outputs, that include all single switch faults, all double faults, and some of the cases with 3, 4, 5, and 6 switch faults as well.

For an edge $(x-y)$ of the line cover selected, the faulty switch will be $\{S_{i,j}\}$, where $2^i = x \oplus y$, and

$$j = \lfloor \{x_0 x_1 \ldots x_{i-1} x_{n-1} x_{n-2} \ldots x_i\}/2 \rfloor,$$

$\{x_{n-1} x_{n-2} \ldots x_0\}$ being the binary representation of x.

Case 1: $k = 2$.

IG for this case is shown in Fig. 7(i). Just one line cover exists, $(x-y)$: that corresponds to the fault set $\{S_{i,j}\}$. *Test phase* 1 is sufficient for detection and location of the fault set. It covers all single switch faults, and also double faults of the type $\{S_{i,j}, S_{2n-2-i,j}\}$.

Example 3.16. Let $F = \{S_{4,2}\}$ in $B(4)$. *IG* is as shown in Fig. 7(i), where the set of faulty outputs $S = \{2, 6\}$. Hence, $F' = \{S_{2,2}\}$. The same F' follows for $F = \{S_{2,2}, S_{4,2}\}$ as well.

Case 2: $k = 3$. IG for this case is shown in Fig. 7(ii). Just one line cover exists, $\{(x-y), (x-z)\}$; that corresponds to a fault set $\{S_{i,j}, S_{k,j}\}$. *Test phase* 1 is sufficient for detection and location of the fault set. It covers some cases with two, three and four switch faults.

Example 3.17. Let $F = \{S_{2,1}, S_{5,2}\}$ in $B(4)$. *IG* is as shown in Fig. 7(ii), where $S = \{8, 10, 12\}$. Hence, $F' = \{S_{1,2}, S_{2,1}\}$. The same F' follows for $F = \{S_{1,2}, S_{2,1}, S_{4,1}\}$ and $F = \{S_{1,2}, S_{2,2}, S_{4,2}, S_{5,2}\}$ as well.

Case 3: $k = 4$. *IG* for this case may have four possible configurations, as are shown in Fig. 7(iii).

(a) Just one line cover exists, $\{(x-y), (u-v)\}$; that corresponds to a fault set $\{S_{i,j}, S_{k,j}\}$, *Test phase* 1 is sufficient for detection and location of the fault set. It covers some cases with two, three and four switch faults.

Example 3.18. Let $F = \{S_{1,1}, S_{1,2}\}$ in $B(4)$. *IG* is as shown in Fig. 7(iii)(a), where $S = \{4, 6, 8, 10\}$. Hence, $F' = \{S_{1,1}, S_{1,2}\}$. The same F' follows for $F = \{S_{1,1}, S_{1,2}, S_{5,1}, S_{5,2}\}$ as well.

(b) Just one line cover exists, $\{(x-y), (x-u), (x-v)\}$; it corresponds to a fault set $\{S_{i,j}, S_{k,j}, S_{p,q}\}$. *Test phase* 1 is sufficient for detection and location of the fault set. It covers some cases with three, four, five and six switch faults.

Example 3.19. Let $F = \{S_{1,0}, S_{2,0}, S_{3,0}\}$ in $B(4)$. *IG* is as shown in Fig. 7(iii)(b), where $S = \{0, 2, 4, 8\}$. Hence, $F' = \{S_{1,0}, S_{2,0}, S_{3,0}\}$. The same F' follows for $F = \{S_{1,0}, S_{2,0}, S_{3,0}, S_{4,0}, S_{5,0}\}$ as well.

(c) Two line covers exist: (i) $\{(x\ y), (u-v)\}$, and (ii) $\{(x-y), (x-u), (u-v)\}$. Say, $x \oplus y = 2^i$, $x \oplus u = 2^j$, and $u \oplus v = 2^k$.

If either $i < j$ or $k < j$, (i) will be the required line cover, same as case (a).

Else, we are to apply *test phase* 2, if the set of faulty outputs contains the outputs corresponding to the outputs x and u of *test phase* 1, then only line cover (ii) would be selected, (it covers some cases of three, four, five and six switch faults), otherwise (i) will be the solution.

Example 3.20. Let $F = \{S_{2,0}, S_{3,2}\}$ in B(4). *IG* is as shown in Fig. 7(iii)(c), where $S = \{0, 2, 4, 10\}$. *Test phase* 2 results $S' = \{3, 7, 5, 13\}$. Since it contains neither the pair $\{1, 3\}$, nor $\{9, 11\}$, the vertex pair corresponding to $\{0, 2\}$ of S, $F = \{S_{2,0}, S_{3,2}\}$.

(d) Here also several line covers exist. But all the solutions can be partitioned in two equivalent cases (i) $\{(x-y), (u-v)\}$ and (ii) $\{(x-u), (y-v)\}$. Here, $x \oplus y = u \oplus v = 2^i$, and $x \oplus u = y \oplus v = 2^j$, $i \neq j$. If $i < j$, we need *test phase* 2, to check if, S' contains

any vertex-pair corresponding to either (x, y) or (u, v) of S; if yes (i) gives the solution, otherwise (ii) will be the solution.

Example 3.21. Let $F = \{S_{2,0}, S_{2,1}\}$ in $B(4)$. IG is as shown in Fig. 7(iii)(d), where $S = \{0, 4, 8, 12\}$. *Test phase 2* results $S' = \{3, 7, 11, 15\}$. Since it contains the pairs $\{3, 7\}$ and $\{11, 15\}$, corresponding to $\{0, 4\}$ and $\{8, 12\}$ of S respectively, $F' = \{S_{2,0}, S_{2,1}\}$. Same F' results also for $F = \{S_{2,0}, S_{2,1}, S_{3,0}, S_{3,1}, S_{4,0}, S_{4,1}\}$.

Therefore, test phase 1, is sufficient to locate all single faults, that results just two faulty outputs in test phase 1. But in case of double faults, two, three or four outputs may be faulty in test phase 1, depending on the location of the faulty switches in the network. It has been shown, that in all the cases resulting two, or three faulty outputs, and also in most of the cases with four faulty outputs, the outcome of test phase 1 will be sufficient to locate the faults. Only in some cases with four faulty outputs, we will need one additional test phase, to locate the switch faults. Moreover, these two phases are actually sufficient to locate some cases with 3, 4, 5 and 6 switch faults as well.

4. Conclusion

In this paper, fault-tolerant routing in Benes network is considered in the presence of noncritical multiple switch faults. In the presence of a single noncritical fault, two passes are always sufficient to route any permutation through the faulty network. In the presence of multiple switch faults, a condition for *two-passability* is presented here, and also algorithms have been developed for two-pass fault-tolerant routing. The novelty of the routing technique lies in the fact that it requires neither the reconfiguration of the system in a degraded mode, nor any recirculation through intermediate nodes. This class covers all possible single faults, assuming the first and last stages of $B(n)$ to be fault-free. The fault-coverages for double and triple faults are well above 70%, and with increase in the value of n, it increases in all the cases.

Another attractive feature of this routing scheme is that it does not reqire the exact locations of the faulty switches in $B(n)$. In fact, the knowledge about a minimal region of $B(n)$ covering the faulty swsitches is sufficient for finding out the alternative routes for the faulty paths. This feature enables us to develop very simple testing procedures for detecting and locating all single and double faults, as well as some cases with 3, 4, 5 and 6 switch faults.

Here, the definition for noncritical fault set, is independent on permutations. If we replace it by introducing the concept of noncritical fault sets relative to a given permutation P, i.e., if, for a given P, we define the set of noncritical faults as one, in the presence of which, every source-destination pair of P has a fault-free path, then we may cover a larger number of multiple faults, in the presence of which, P is routable in two passes.

References

[1] D.P. Agrawal, Testing and fault tolerance of Multistage Interconnection Networks, *Computer* (Apr. 1982) 41–53.

[2] T. Feng and W. Young, Fault diagnosis for a class of rearrangeable networks, *J. Parallel Distributed Comput.* 3 (1986) 23–47.

[3] T. Feng, Fault tolerance in rearrangeable networks, in: *Proc. IEEE Region 10 Conf. on Computer and Communication Systems*, (1990) 399–403.

[4] S.T. Huang and C.H. Tung, On fault tolerant routing of Benes network, *J. Inform. Sci. and Engineering*, 4 (1988) 1–13.

[5] D. Nassimi, A fault-tolerant routing algorithm for BPC permutations on multistage interconnection networks, in: *Proc. 1989 Internat. Conf. on Parallel Processing* (1989) 278–287.

[6] D.C. Opferman and N.T. Tsao-Wu, On a class of rearrangeable switching networks, *The Bell System Tech. J.* 50 (1971) 1579–1618.

[7] Y. Yeh and T.Y. Feng, Fault-tolerant routing on a class of rearrangeable networks, in: *Proc. 1991 Internat. Conf. on Parallel Processing* (1991) 305–312.

Nabanita Das received the B.Sc. (Hons.) degree in Physics, the B. Tech. degree in Radio Physics and Electronics from the University of Calcutta, the M.E. degree in Electronics and Telecomm. Engineering and the Ph.D. degree in Computer Science from the Jadavpur University. Since 1986, she has been on the faculty of the Electronics unit, Indian Statistical Institute, Calcutta. Her research interests include parallel processing, interconnection networks, testing and distributed architectures.

Jayasree Dattagupta received the B.E. degree in Electronics and Telecomm. Engineering and a Post-Graduate Diploma in Computer Science, both from Jadavpur University, Calcutta in 1968 and 1969 respectively. She received the M. Phil. degree from the Brunel University, UK in 1976 and the Ph.D. degree in Computer Science from Calcutta University in 1986. In 1970 she joined the faculty of Indian Statistical Institute, Calcutta, India where she is currently an Associate Professor. She had been with the Informatik Kolleg, GMD, Bonn during 1982–83 as a visiting scientist. She served as the chairman of the Indian sub-committee of the International Test Conferences in 1989. She was also the vice-chairman of the Indian sub-committee of the International test conferences during 1987–89 and a member of the programme committees of the International test conferences during 1987–90. Her research interests are in the areas of computer networks, fault-tolerant computing and parallel processing.