Diagnosis of Single Faults in Bitonic Sorters

Tsern-Huei Lee, *Member, IEEE,* and Jin-Jye Chou

***Abstract*—Bitonic sorters have recently been proposed to construct along with banyan networks the switching fabric of future broadband networks [2]. Unfortunately, a single fault in a bitonic sorter may have disastrous consequences for the switching system. Therefore, a bitonic sorter must be proved to be free of faults before it can be used. In this paper we study the topological properties of bitonic sorters and present an efficient fault diagnosis procedure to detect, locate, and identify the fault type of single faults. Our diagnosis procedure can detect most single faults in two tests. Faults which cannot be detected in two tests can always be detected in four tests. Several binary search techniques are developed to locate a faulty sorting element (i.e., a 2×2 sorter).**

I. Introduction

BITONIC SORTERS [1] have recently been proposed to construct along with banyan networks the switching fabric of future broadband networks [2]. The purpose of using sorters is to resolve the internal blocking of banyan networks and consequently enhance the performance of a switching system. Unfortunately, a fault occurring in a bitonic sorter may have disastrous consequences for the switching system. Therefore, a bitonic sorter must be proved to be free of faults before it can be used.

An efficient and effective fault diagnosis procedure has been proposed to detect, locate, and identify the fault type for banyan networks suffering from a single solid logical fault [3]. It is natural to wonder if this procedure can be modified to diagnose bitonic sorters. It turns out that it can [4]. However, this procedure can be applied only to banyan networks having state control lines. The number of state control lines in a banyan network is equal to $n = \log_2 N$, where N denotes the number of inputs/outputs, because each stage must have a separate line. Since the complexity of adding state control lines is not negligible, the procedure proposed in [3] may be unsuitable for large networks. In [5], the authors presented an efficient diagnosis procedure for banyan networks without state control lines. For a bitonic sorter with N inputs/outputs, the number of stages is equal to $\log_2 N(\log_2 N + 1)/2$, which is much larger than that in a banyan network with the same number of inputs/outputs. Thus, a diagnosis procedure which relies on state control lines is unrealistic for large bitonic sorters.

Manuscript received July 2, 1993; revised June 8, 1994; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Turner. This work was supported in part by the National Science Council, Republic of China, under contract number NSC 82-0404-E-009-334.

T. H. Lee is with the Department of Communication, National Chiao Tung University, Hsinchu, Taiwan, China (e-mail: thlee @ banyan.cm.nctu.edu.tw).

J. J. Chou is with the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, China (e-mail: choujj @ banyan.cm.nctu.edu.tw).

IEEE Log Number 9404856.

This paper presents an efficient fault diagnosis procedure for bitonic sorters without state control lines. The proposed procedure can be used to detect, locate, and identify the fault type of any single fault. Owing to space limitations, we present in this paper the procedures for detecting and locating single faults. A procedure for identifying the fault type can be found in [6]. Since a large bitonic sorter is often built from several IC chips, the faulty chip can be replaced once the fault is located.

Section II describes the fault model and operation of a sorting element (SE), i.e., a 2×2 sorter. Section III studies some properties of bitonic sorters which lead to the design of test vectors. Diagnosis of link faults and SE faults are presented in Sections IV and V, respectively. In Section VI, we study some examples. Section VII concludes the paper.

II. Fault Model and Operation of a Sorting Element

A. Fault Model

The fault model considered in this paper is exactly the same as that studied in [3]. That is, faults occurring in a bitonic sorter are classified into link faults and SE faults. A link fault is either a stuck-at-zero (s-a-0) fault or a stuck-at-one (s-a-1) fault. A functional approach is used to study SE faults. An SE is considered to be a 2×2 crosspoint switching matrix which has as many as 16 possible states, as illustrated in Table I. For each SE only the direct state (S_{10}) and the cross state (S_5) are valid states. A faulty SE can be in any one of the 16 possible states. Thus, for an SE with two valid states, there are 256 possible state combinations. Let S denote the set of all 16 states. As in [3], we use the set $\{(s_1, s_2) \mid s_i \in S\}$ to describe the state combinations and refer to each (s_1, s_2)-pair as a functional state. Assume the first valid state is S_{10} and the second valid state is S_5. As a consequence, only the state combination (S_{10}, S_5) is the normal functional state and the other 255 state combinations are faulty functional states.

In Table I, x_1 and x_2 are binary bits applied respectively to the upper and the lower inputs. Similarly, $\hat{x_1}$ and $\hat{x_2}$ are binary bits appearing at the upper and the lower outputs, respectively. The symbol "?" denotes logically unidentified output and "ϕ" represents logically erroneous output, where 0 and 1 are simultaneous inputs. The output values of "?" and "ϕ" depend on the particular circuit implementation. However, both of them are assumed to be constant values in a given bitonic sorter. Note that an erroneous fault cannot be observed at the outputs if x_1 and x_2 are identical, because "ϕ" = 1 if $x_1 = x_2 = 1$ and "ϕ" = 0 if $x_1 = x_2 = 0$. In addition to erroneous and unidentified faults, there is another type of faults called binary fault which results from misconnecting an input

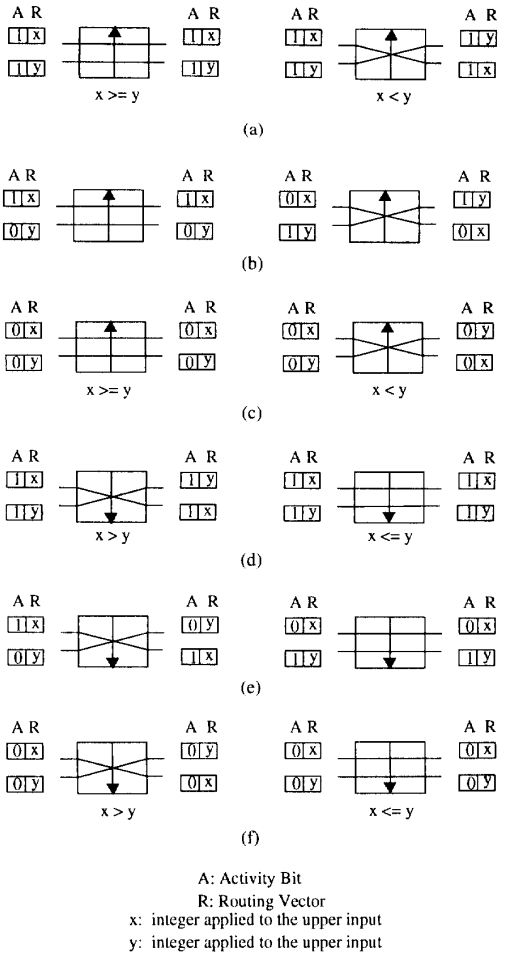


Fig. 1. Operations of sorting elements.

to an undesired output. For example, a binary fault occurs if an SE is in state S_3 while its valid state is S_{10}.

B. Operation

There are two types of SE's in a bitonic sorter: the up SE (indicated by an upward arrow) and down SE (indicated by a downward arrow). The state of a fault-free SE is controlled by the activity bits and the numbers applied to both inputs. Fig. 1 illustrates the operation of the two types of fault-free SE's. In Fig. 1, x and y represent the integers applied to the upper and lower inputs, respectively. An input is said to be active if the activity bit of the vector applied to the input is one. When both inputs are active or inactive, a down (up) SE will be in state S_{10} if $x \leq y$ $(x \geq y)$ or state S_5 if $x > y$ $(x < y)$. If only one input is active, then the active input is connected to the lower output of a down SE or the upper output of an up SE and the inactive input is connected to the other output. In other words, we treat the activity bit as the most significant bit. In addition, an SE is always in state S_{10} when the two applied vectors are identical.

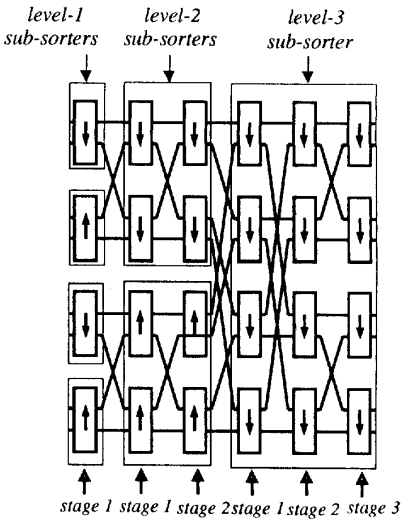


Fig. 2. A three-level bitonic sorter.

III. Some Properties of Bitonic Sorters

Consider a bitonic sorter with N inputs/outputs. Such a sorter consists of $n = \log_2 N$ levels of sub-sorters and thus is called an *n-level* bitonic sorter. Fig. 2 illustrates a three-level bitonic sorter. There are 2^{n-i} *level-i* sub-sorters and each *level-i* sub-sorter is similar to a banyan network with 2^i inputs/outputs. Therefore, for convenience, a sub-sorter is also referred to as a banyan sorter (BS). A sorter is called an ascending (descending) sorter if it sorts the inputs in ascending (descending) order. In this paper, we consider ascending sorters.

To diagnose a fault, one has to check both valid states for each SE. According to the operation of SEs described in the last section, one can set all the SE's to be in state S_{10} by applying identical numbers to all the inputs. However, a fault may not be detected if all the numbers applied are identical. Moreover, it is impossible to set all the SE's to state S_5 for $n \geq 2$. This can easily be proved by showing that it is impossible for $n = 2$. Therefore, one has to find two sets of numbers that can be used for fault diagnosis. It turns out that monotonic sequences can serve this purpose. Throughout this paper, we use $\{a_i\}_{i=1}^N$ to denote a strictly increasing sequence with $a_i = i - 1$ and $\{b_i\}_{i=1}^N$ to denote a strictly decreasing sequence with $b_i = a_{N-i+1}$.

The following propositions concerning the topological properties of bitonic sorters are very useful in designing test vectors. In these propositions, we assume the activity bit of each applied vector is one. In this paper, the leftmost bit of a vector is the most significant bit and we always count from the left or the top. For example, we may refer to the ith (from the left) bit of a vector or the jth (from the top) *level-i* BS. In addition, when a sequence, say $\{a_i\}_{i=1}^N$, is partitioned into k (k is a divisor of N) groups $G_1, G_2, \cdots,$ and G_k, we mean all the groups are of equal size and G_1 contains the first $\frac{N}{k}$ elements of $\{a_i\}_{i=1}^N$, G_2 contains the second $\frac{N}{k}$

elements of $\{a_i\}_{i=1}^N$, and so on. Further, the elements of G_i are arranged in ascending order. We use G_i' to denote a group whose elements are identical to those in G_i but may not be arranged in ascending order. For example, if $\{a_i\}_{i=1}^8$ is partitioned into two groups G_1 and G_2, then $G_1 = [a_1a_2a_3a_4]$ and $G_2 = [a_5a_6a_7a_8]$. A G_1' can be $[a_1a_3a_4a_2]$. When a sequence of numbers (say $\{a_i\}_{i=1}^N$) is applied to the inputs (of a bitonic sorter), we mean that the jth number (a_j) is applied to the jth input. In the following propositions, the sorter is assumed to be fault-free unless otherwise stated. The proofs of these propositions can be found in [6].

Proposition 1: If the state of an SE is S_{10} (or S_5) when $\{a_i\}_{i=1}^N$is applied to the inputs, then its state is S_5 (S_{10}) when $\{b_i\}_{i=1}^N$ is applied.

Proposition 1 suggests that one might be able to apply $\{a_i\}_{i=1}^N$ and $\{b_i\}_{i=1}^N$separately to the inputs to test the two valid states of each SE. Unfortunately, as will be seen later, not all of the SE faults can be detected by this approach.

For the following two propositions, we partition $\{a_i\}_{i=1}^N$ into 2^kgroups $G_1, G_2, \cdots$, and G_{2^k}. Define $G_l \ominus [x]$ to be the group obtained by removing x from G_l. Moreover, for any two groups X and Y, define $X \oplus Y$ to be the group obtained by adding the elements of group Y to the end of group X. The output of a fault-free bitonic sorter can be represented by $G_1G_2 \cdots G_{2^k}$. An output is said to be faulty if it is different from $G_1G_2 \cdots G_{2^k}$.

Proposition 2: Let $\{a_i\}_{i=1}^N$ or $\{b_i\}_{i=1}^N$ be applied to the inputs. Suppose an SE in stage $k-j$ ($0 \le j \le k-1$) of a *level-*$(n-j)$ BS is faulty such that it is in state S_{10} (S_5) while its valid state is S_5 (S_{10}). Then the output is correct if $k = n$ and $j \ge 1$ or becomes $G_1G_2 \cdots G_{2m-1}^* G_{2m}^* \cdots G_{2^k}$ for some m, where $G_{2m-1}^* = (G_{2m-1}' \ominus [x]) \oplus [y]$ and $G_{2m}^* = [x] \oplus (G_{2m}' \ominus [y])$ for some $x \in G_{2m-1}$ and $y \in G_{2m}$. Besides, there are 2^{n-k} SEs in each of the stages considered that can result in the same value of m. An SE in each of the stages considered can result in the value m if and only if (iff) its two input numbers are both in $G_{2m-1} \oplus G_{2m}$. That is, the i^{th} SE in each of the stages considered can result in the value m iff $(m-1)2^{n-k}+1 \le i \le m2^{n-k}$ when $\{a_i\}_{i=1}^N$ is applied or $(2^{n-k}-m)2^{n-k}+1 \le i \le (2^{n-k}-m+1)2^{n-k}$ when $\{b_i\}_{i=1}^N$ is applied.

Fig. 3 illustrates an example of the stages considered in Proposition 2 for n =4 and k =3. Several remarks should be made concerning Proposition 2. First, although there are 2^{n-k} SE's in each of the stages considered in the proposition that can result in the same value of m, the contents of G_{2m-1}^* and G_{2m}^* could be different for different SEs. Figs. 4 and 5 show respectively examples of SE faults which result in the same or different G_{2m-1}^* and G_{2m}^*. The sequence $\{a_i\}_{i=1}^N$ is partitioned into 8 groups in Fig. 4 or 4 groups in Fig. 5. Therefore, one may be able to tell which SE is faulty by observing the output. However, there are too many cases to be distinguished and thus, in our diagnosis procedure, we ignore this possibility. In other words, two faulty outputs are considered to be distinguishable only if they have different m values. Second, an SE fault cannot be detected (by applying $\{a_i\}_{i=1}^N$ or $\{b_i\}_{i=1}^N$) if the faulty SE is in the last stage of a *level-i* ($i \le n-1$) BS and its faulty state is S_{10} or S_5

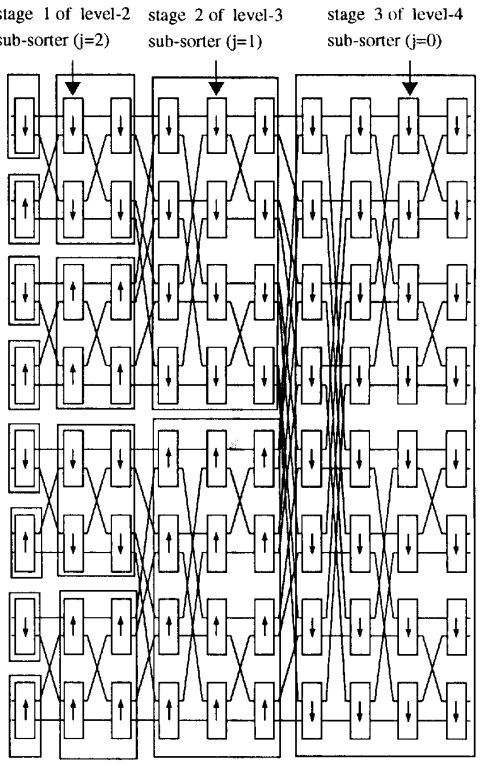


Fig. 3. Stages considered in Proposition 2 for n =4 and k =3.

(because the fault will be corrected by a succeeding *level-*$(i+1)$ BS). Fig. 6 shows an example of an SE fault which results in a correct output. Third, assuming the faulty BS (i.e., the BS which contains the faulty SE) is in *level-i* and has been identified, one can know only that the faulty SE is in an ambiguity set which contains 2^{n-k} elements in stage $i-n+k$ of the faulty BS.

We now describe how to detect an SE fault when the faulty SE is in the last stage of a *level-i* ($i \le n-1$) BS and its faulty state is S_{10} or S_5. Define a permutation, called the Bit Reverse Permutation (BRP) by BRP($\{a_i\}_{i=1}^N$) = (BRP($\{a_{2i-1}\}_{i=1}^{i=N/2}$), BRP($\{a_{2i}\}_{i=1}^{i=N/2}$)) and BRP($a_1, a_2$) = (a_1, a_2). The permutation is called BRP because if $\{c_i\}_{i=1}^N$ = BRP($\{a_i\}_{i=1}^N$) and $a_i = i-1$, then the binary representation of c_i is the bit reverse of that of a_i. For the rest of the paper, we define $\{c_i\}_{i=1}^N$ = BRP($\{a_i\}_{i=1}^N$) and $\{d_i\}_{i=1}^N$ = BRP($\{b_i\}_{i=1}^N$).

Proposition 3: Let $\{c_i\}_{i=1}^N$ or $\{d_i\}_{i=1}^N$ be applied to the inputs. Suppose an SE in the last stage of a *level-k* BS is faulty and its faulty state is S_{10} or S_5. Then the output becomes $G_1G_2 \cdots G_{2m-1}^* G_{2m}^* \cdots G_{2^k}$, where $G_{2m-1}^* = (G_{2m-1}' \ominus [x]) \oplus [y]$ and $G_{2m}^* = [x] \oplus (G_{2m}' \ominus [y])$ for some $x \in G_{2m-1}$ and $y \in G_{2m}$. Moreover, there are 2^{n-k} SE's which can result in identical G_{2m-1}^* and G_{2m}^*. These 2^{n-k} SE's belong to 2^{n-k} distinct *level-k* BSs and the two numbers entering into any of these SE's are both in $G_{2m-1} \oplus G_{2m}$.

Proposition 4: There is at most one common link shared by a path when $\{a_i\}_{i=1}^N$ is applied and another path when $\{b_i\}_{i=1}^N$ is applied.

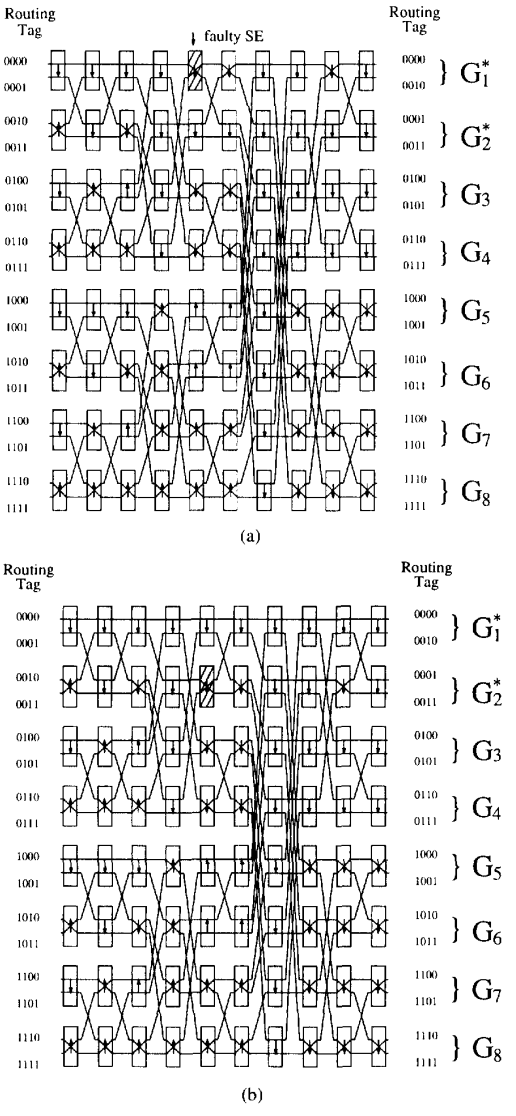


Fig. 4. SE faults which result in m =1 with the same G^*_{2m-1} and G^*_{2m}.

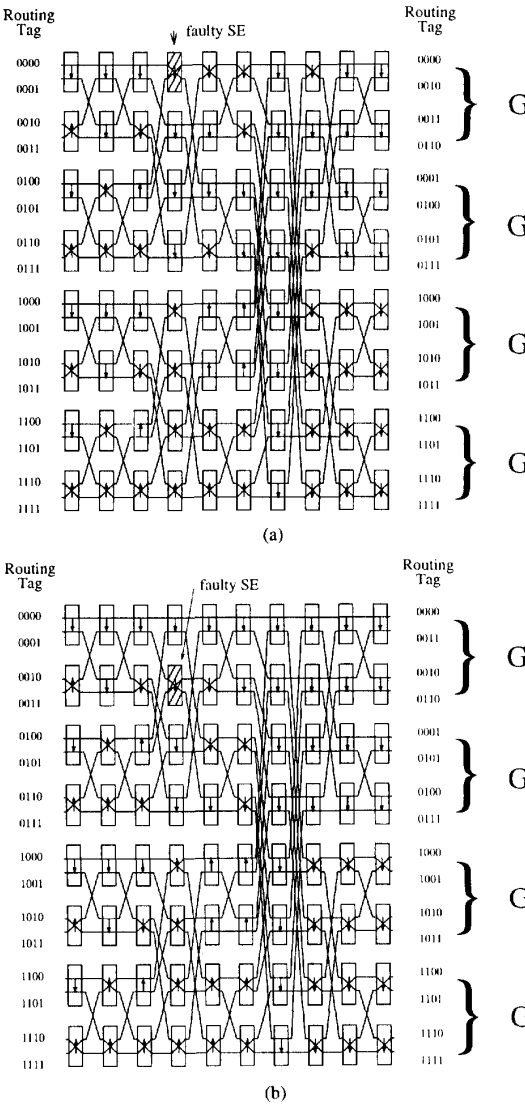


Fig. 5. Se faults which result in m =1 with different G^*_{2m-1} and G^*_{2m}.

Proposition 5: Let $\{a_i\}_{i=1}^N$ or $\{b_i\}_{i=1}^N$ be applied to the inputs. Any two paths can meet at most once at some SE within a BS. If two paths meet at an SE in stage k $(1 < k \leq i)$ of a *level-i* BS, then they meet again at an SE in stage k+1 of a *level-*(i+1) BS.

Although Proposition 5 is stated for $k > 1$, it is valid for $k = 1$ and $i = 1$.

Proposition 6: Let $\{a_i\}_{i=1}^N$ or $\{b_i\}_{i=1}^N$ be applied to the inputs. If two paths meet at an SE in stage 1 of a *level-i* ($i >$ 1) BS, then they meet only once.

Proposition 7: Let $\{a_i\}_{i=1}^N$ or $\{b_i\}_{i=1}^N$ be applied to the inputs. The binary representations of the two numbers entering into an SE in the first stage of a *level-i* BS differ in exactly i bits, the rightmost i bits.

Proposition 8: Let $\{a_i\}_{i=1}^N$ or $\{b_i\}_{i=1}^N$ be applied to the inputs. The binary representations of the two numbers entering into an SE in stage k ($k > 1$) of a *level-i* BS differ in exactly one bit, the $(n - i + k)$th bit.

To conclude this section, we describe our diagnosis procedure. The procedure consists of two phases. A test vector in each phase consists of four fields, namely, the activity bit, the routing tag, the checking data which is the 1's complement of the routing tag, and a two-bit constant data field whose content is 01. Fig. 7 shows the format of a test vector. In phase I and phase II, the sequences $\{a_i\}_{i=1}^N$ and $\{b_i\}_{i=1}^N$, respectively , are selected as the routing tags. Additional tests may have to be performed in some cases. For example, we need to perform at least two more tests by applying $\{c_i\}_{i=1}^N$ and $\{d_i\}_{i=1}^N$ to diagnose the SEs in the last stage of BSs if both phase I and phase II result in correct outputs. As mentioned before, the binary representation of c_i (d_i) is the bit reverse of that of a_i (b_i). For example, for a four-level bitonic sorter, the binary

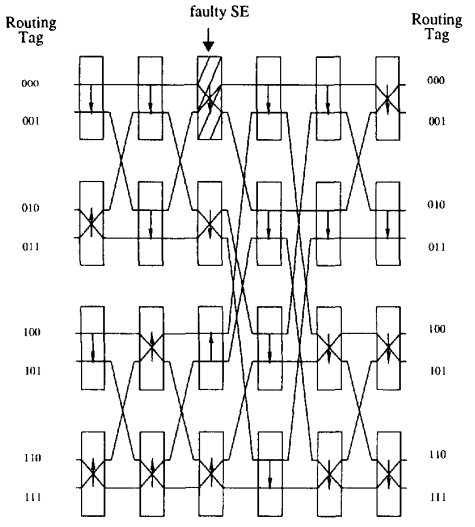


Fig. 6. An SE fault which results in a correct output.

Activity Bit	Routing Tag	Checking Data	Constant Data

Fig. 7. Format of a test vector.

representation of a_9 is 1000 and that of c_9 is 0001. Note that from the routing tag and the checking data, one can determine (according to Propositions 7 and 8) the stages the faulty SE can possibly be in if there is an erroneous fault. Furthermore, the constant data field is added so that an erroneous fault can always be distinguished from an unidentified fault. For example, without the constant data field, states S_1 and S_7 cannot be distinguished if the faulty SE is in the first stage of the *level-n* BS and "?" = "ϕ" = 1. The reason is that the two routing tags (and thus the checking data) of the test vectors entering into an SE in the first stage of the *level-n* BS are 1's complements of each other and, therefore, without the constant data field, states S_1 and S_7 produce identical vectors at the outputs of the faulty SE.

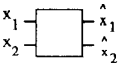
IV. Diagnosis of Link Faults

For convenience, we define the expected path of a test vector to be the path it traverses when the bitonic sorter is free of faults. Furthermore, a test vector is said to be a faulty vector if it is lost (i.e., does not appear at the output) or duplicated (i.e., more than one copy appears at the output). The expected path of a faulty test vector is called a faulty path.

For an s-a-0 fault, one of the test vectors becomes an all-zero vector and thus can be easily detected. In addition, by observing the vectors received by the output ports, one can determine which test vector is lost. From Proposition 4, the faulty link can be located by finding the common link of two faulty paths, one in each phase.

Similarly, for an s-a-1 fault, one of the test vectors becomes an all-one vector. Again, one can obtain a faulty path in phase I and another faulty path in phase II. By intersecting the two faulty paths, one gets the faulty link.

TABLE I
State Names and their Symbolic Representations of Sorting Element



state name	switch element symbol	$\hat{x}_1$	$\hat{x}_2$	state name	switch element symbol	$\hat{x}_1$	$\hat{x}_2$
S_0		?	?	S_8		x_1	?
S_1		x_2	?	S_9		ϕ	?
S_2		?	x_2	S_{10}		x_1	x_2
S_3		x_2	x_2	S_{11}		ϕ	x_2
S_4		?	x_1	S_{12}		x_1	x_1
S_5		x_2	x_1	S_{13}		ϕ	x_1
S_6		?	ϕ	S_{14}		x_1	ϕ
S_7		x_2	ϕ	S_{15}		ϕ	ϕ

V. Diagnosis of Sorting Element Faults

To detect and locate single SE faults, we partition the 16 states listed in Table I into four sets, $B_1 = \{S_3, S_{12}\}$, $B_2 = \{S_5, S_{10}\}$, $E = \{S_6, S_7, S_9, S_{11}, S_{13}, S_{14}, S_{15}\}$, and $U = \{S_0, S_1, S_2, S_4, S_8\}$. Sets B_1 and B_2 contain states that result in only binary faults. Set E contains states that result in at least one erroneous fault. Set U contains states that result in at least one unidentified fault but no erroneous fault.

Note that, using our proposed test vectors, a single SE fault can always be detected. If the faulty state is in B_1, then at the output, one test vector is lost and one is duplicated and thus the fault can be detected. According to Proposition 2, an SE fault can be detected if the faulty state is in B_2 unless the faulty SE is in the last stage of a *level-i* ($i \leq n-1$) BS. For this case, the two additional tests using $\{c_i\}_{i=1}^{N}$ and $\{d_i\}_{i=1}^{N}$ as routing tags guarantee that the fault can be detected. If the faulty state is in E, then there is at least one lost test vector and one unexpected vector (whose constant data field is 01) appears at the output and thus the fault can be detected. Finally, if

the faulty state is in U, then at least one unexpected all-zero or all-one vector appears at the output. Based on the above results, one not only can detect the fault but also determine to which set the faulty state belongs. This information can be used to identify the fault type [6].

We now discuss how to locate the faulty SE. There are three cases to consider: 1) phases I and II both result in faulty outputs, 2) only phase I or phase II results in a faulty output, and 3) neither phase I nor phase II results in a faulty output.

Case 1: Phases I and II both result in faulty outputs. For Case 1, we have the following theorems. In these theorems, s_1 and s_2 represent the states of the faulty SE when its valid state is S_{10} or S_5, respectively. For ease of description, we assume s_1 is the state of the faulty SE in phase I and s_2 is that of the faulty SE in phase II. In each theorem, we select only one case to prove because the other cases can be proved similarly.

Theorem 1: If $s_1 \in B_1, s_2 \in B_1 \bigcup E \bigcup U$ or $s_1 \in B_1 \bigcup E \bigcup U, s_2 \in B_1$, then the faulty SE can be located.

Proof: Consider the case where $s_1 \in B_1$ and $s_2 \in E$. Since $s_1 \in B_1$, there are two faulty paths, path 1 and path 2, in phase I. According to Proposition 5, the faulty SE is in an ambiguity set which contains the common SE's traversed by paths 1 and 2. Note that every SE in the ambiguity set can result in the observed output and no two SEs can be in the same BS. Similarly, $s_2 \in E$ implies there is at least one faulty path, path 3, in phase II. By intersecting paths 1 and 3, one gets a common link. The element in the ambiguity set to which the common link is connected is the faulty SE.

Theorem 2: If $s_1, s_2 \in B_2$, then the faulty BS can be identified.

Proof: Suppose the faulty SE is in stage k of a *level-i* BS. Let $A\ (B)$ denote the ambiguity set of all the SE's that can result in an output undistinguishable (i.e., same value of m in Proposition 2) from that observed in phase I (phase II). Also, let $A_l\ (B_l)$ be a subset of $A\ (B)$ that contains all the SE's in $A\ (B)$ that belong to a *level-l* BS. According to Proposition 2, we know that $A_i = B_i$, because the faulty SE is assumed to be in a *level-i* BS. Define $R_1\ (R_2)$ to be the set of paths that pass through any SE in $A_i\ (B_i)$ in phase I (phase II). The faulty BS can be identified if one can prove $A_l \bigcap B_l = \emptyset$ for $l \neq i$.

Suppose $k > 1$ and consider $l = i - 1$. The paths in sets R_1 and R_2 can only pass through two *level-*$(i-1)$ BSs, BS_1 and BS_2. Since the routing tags of the applied test vectors are increasing in phase I and decreasing in phase II, all the paths in R_1 (or R_2) must pass through the same *level-*$(i-1)$ BS, BS_1 or BS_2. Moreover, if the paths in R_1 pass through BS_1, then the paths in R_2 must pass through BS_2 and vice versa. Therefore, we have $A_{i-1} \bigcap B_{i-1} = \emptyset$. Before level $i-1$, the two sets of paths are in disjoint BSs and thus $A_l \bigcap B_l = \emptyset$ for $l \leq i-1$. From these results, one can conclude that $A_l \bigcap B_l = \emptyset$ for $l > i$ because if $A_{i+j} \bigcap B_{i+j} \neq \emptyset$ for some j, $1 \leq j \leq n-i$, then, according to Proposition 2, one should have $A_{i+j} = B_{i+j}$ and thus $A_l \bigcap B_l = \emptyset$ for $l \leq i+j-1$, which contradicts $A_i = B_i$.

Consider the case of $k = 1$. From Proposition 2, we know that $A_l = B_l = \emptyset$ for $l \leq i-1$. Therefore, the proof is completed for $i = n$. Assume $i < n$. Suppose $A_{i+j} \bigcap B_{i+j} \neq \emptyset$ for some j, $1 \leq j \leq n-i$. Since the SE's in A_{i+j} are not in the first stage of a *level-*$(i+j)$ BS, we know from the arguments provided in the last paragraph that $A_l \bigcap B_l = \emptyset$ for $l \leq i+j-1$, which again contradicts $A_i = B_i$. This completes the proof of Theorem 2.

To locate the faulty SE after identifying the faulty BS for the case where $s_1, s_2 \in B_2$, one can apply a set of special test vectors to the inputs. The routing tags of the special test vectors are all identical so that all the SEs are supposed to be in state S_{10}. However, the checking data of the special test vector applied to input i is chosen to be i-1. If the bitonic sorter is free of faults, then output port i should receive the test vector applied to input i. Under a single fault, two of the output ports will exchange the test vectors they should receive when the bitonic sorter is fault-free. Therefore, the faulty SE can be located by finding the common SE in the faulty BS that is traversed by the expected paths of the two exchanged test vectors.

Theorem 3: If $s_1 \in B_1, s_2 \in B_2$ or $s_1 \in B_2, s_2 \in B_1$, then the faulty SE can be located.

Proof: Consider the case where $s_1 \in B_1$ and $s_2 \in B_2$. Assume the faulty SE is in stage k of a *level-i* BS. If $k = 1$ and $i > 1$, then, according to Proposition 6, the faulty SE can be located by intersecting the two faulty paths obtained in phase I.

Suppose $k > 1$ or $i = 1$. In this case, the faulty SE can be located once the faulty BS is identified because $s_1 \in B_1$ gives two faulty paths and the stages the faulty SE can possibly be in (see Proposition 5). Let $C\ (B)$ denote the set of all the SE's which can result in an output undistinguishable from that observed in phase I (phase II). Also, let $C_l\ (B_l)$ be a subset of $C\ (B)$ which contains all the SE's in $C\ (B)$ that belong to a *level-l* BS. Then $C\ (C_l)$ is a subset of $A\ (A_l)$ defined in the proof of Theorem 2 and thus the arguments provided there can be directly applied here to identify the faulty BS and complete the proof.

Theorem 4: If $s_1 \in B_2, s_2 \in E \bigcup U$ or $s_1 \in E \bigcup U, s_2 \in B_2$, then the faulty SE can be located.

Proof: Let us consider the case where $s_1 \in B_2$ and $s_2 \in E$. If the faulty SE is in the first stage of a *level-i* $(i \geq 2)$ BS, then from the faulty path, the routing tag, and the checking data of an unexpected vector observed in phase II, one can immediately locate the faulty SE (see Propositions 6 and 7).

Suppose the faulty SE is not in the first stage of a *level-i* $(i \geq 2)$ BS. In this case, the faulty SE can be located once the faulty BS is identified because there is a faulty path in phase II and, from $s_1 \in B_2$, one knows the stages the faulty SE can possibly be in. Again, the same arguments provided in the proof of Theorem 2 can be applied here to complete the proof.

Theorem 5: If $s_1 \in E, s_2 \in E \bigcup U$ or $s_1 \in E \bigcup U, s_2 \in E$, then the faulty SE can be located.

Proof: Since $s_1 \in E$, one knows from Proposition 5 that the faulty SE is in an ambiguity set which contains at most n SEs traversed by a faulty path in phase I. Also, $s_2 \in U$ provides another faulty path in phase II. From the two faulty paths, one can find a common link. The SE in the ambiguity set to which the common link is connected is the faulty SE. In fact, the faulty SE can be located in phase I if it is in the first stage of a *level-i* $(i \geq 2)$ BS (see Propositions 6 and 7).

TABLE II
LOCATING PROCESS FOR CASE 1

locating process s_1 \ s_2	B_1	B_2	E	U
B_1	Theorem 1	Theorem 3	Theorem 1	Theorem 1
B_2	Theorem 3	Theorem 2 + special test vectors	Theorem 4	Theorem 4
E	Theorem 1	Theorem 4	Theorem 5	Theorem 5
U	Theorem 1	Theorem 4	Theorem 5	Identification

TABLE III
LOCATING PROCESS FOR CASE 2 WHEN THE ADDITIONAL TEST RESULTS IN A CORRECT OUTPUT

s	Locating process
B_1	Binary Search (BSearch1) if the faulty SE is not in the first stage of a BS
B_2	Binary Searches BSearch1 + BSearch2
E	Binary Search (BSearch1) if the faulty SE is not in the first stage of a BS
U	Binary Searches BSearch1 + BSearch3

TABLE IV
LOCATING PROCESS FOR CASE 3

Condition	Locating process
Both additional tests (applying $\{c_i\}_i^N = 1$ and $\{d_i\}_i^N = 1$) result in faulty outputs	Special test vectors
Only one additional test results in faulty output	Binary Search BSearch4

The remaining case is $s_1, s_2 \in U$. If s_1 (s_2) is state S_0, then the situation is similar to $s_1 \in B_1$ and $s_2 \in U$ ($s_1 \in U$ and $s_2 \in B_1$). Thus the faulty SE can be located. Suppose neither s_1 nor s_2 is state S_0. In this case, one can still obtain a common link of the faulty paths observed in phase I and phase II. The ambiguity set contains two SE's connected by the common link. To determine which one is faulty, one needs to identify the fault type. The identification process is described in [6]. We summarize the locating process for Case 1 in Table II.

Case 2: Only phase I or phase II results in a faulty output. Suppose phase I results in a faulty output and phase II does not. (The faulty SE can be located similarly for the other case.) In this case, we need to perform an additional test by applying $\{d\}$ to the inputs.

If the output is faulty, then $\{c\}$ is further applied. Consider the output when $\{c\}$ is applied. If the output is correct or faulty and the states of the faulty SE are in different sets (B_1, B_2, E, or U when $\{c\}$ and $\{d\}$ are applied, respectively, then the faulty SE must be in the last stage of a BS and s_1 must be in $B_1 \bigcup E \bigcup U$ (because in this case no faulty output can be observed in phase I if $s_1 \in B_2$). According to Proposition 3, one knows the level number of the faulty BS. Therefore, the faulty SE can be located because there is at least one faulty path in phase I.

If the output is faulty and the states of the faulty SE are in the same set when $\{c\}$ and $\{d\}$ are applied, respectively, then the faulty SE is not in the last stage of any BS. The locating procedure for this case is identical to that described below for the case where the output is correct when $\{d\}$ is applied.

If the output is correct when $\{d_i\}_{i=1}^N$ is applied, then the faulty SE is not in the last stage of a BS and s_2 is the desired valid state. In this case, one or two binary searches are required to locate the faulty SE. We summarize the locating process for this case in Table III. In this table, s is either s_1 or s_2. Different types of binary searches are listed in Table III. If $s \in B_1 \bigcup E$, then the faulty SE can be located once the faulty BS is identified. Therefore, binary search BSearch1

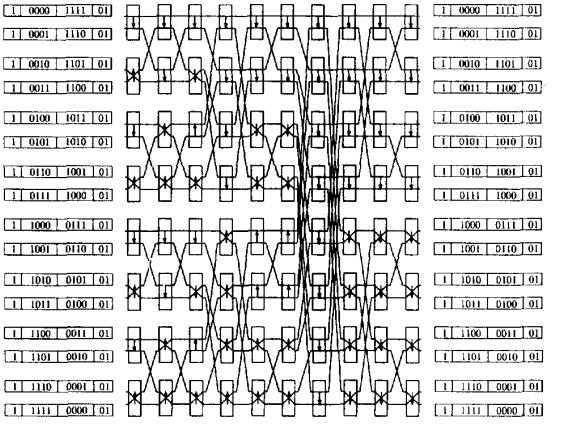


Fig. 8. Result of applying $\{a_i\}_{i=1}^N$ to a fault-free four-level bitonic sorter.

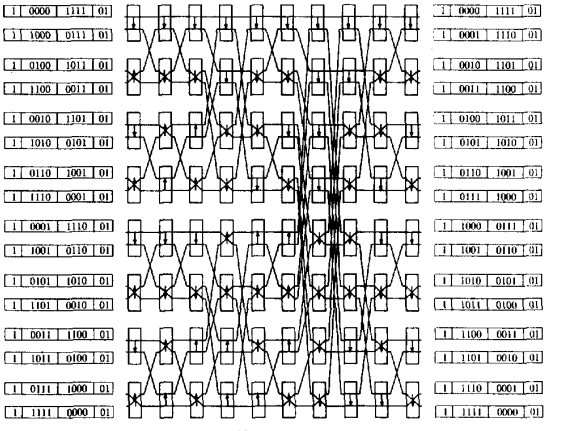
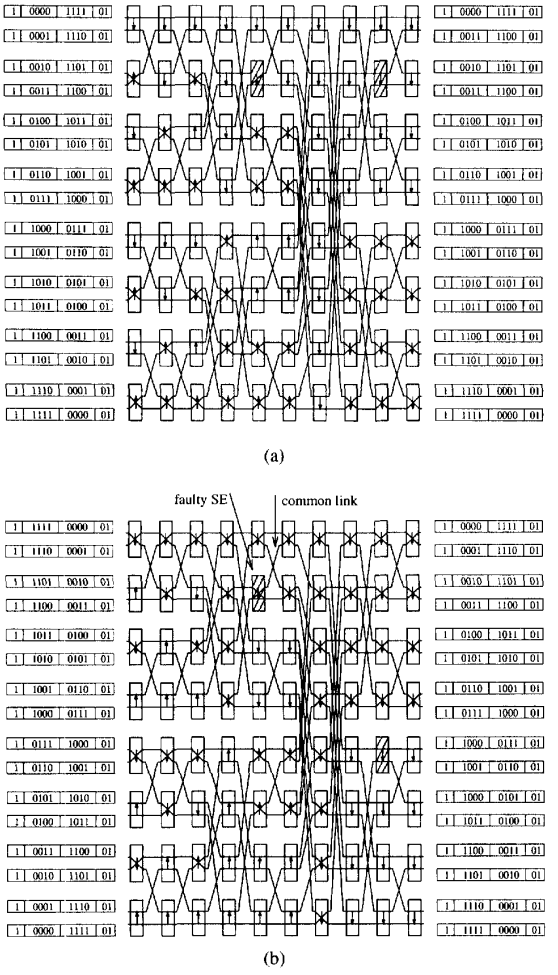


Fig. 9. Result of applying $\{c_i\}_{i=1}^N$ to a fault-free four-level bitonic sorter.



(a)

(b)

Fig. 10. Diagnosis results for Example 1: (a) result of phase I diagnosis and (b) result of phase II diagnosis.

is designed to identify the faulty BS. When $s \in B_2$, one can adopt BSearch1 to identify the faulty BS and then use BSearch2 to locate the faulty SE. BSearch2 is designed to locate the faulty SE in an ambiguity set containing several SE's in the same stage of a BS. When $s \in U$, the faulty BS can be identified using BSearch1. After the faulty BS is identified, another binary search, BSearch3, is needed to locate the faulty SE. BSearch3 is designed to locate the faulty SE in an ambiguity set containing i SE's in a $level$-i BS. We describe in the Appendix how to modify the test vectors applied to the inputs to accomplish a desired search.

Case 3: Neither phase I nor phase II results in a faulty output

For this case, one has to perform additional tests by applying $\{c_i\}_{i=1}^N$ and $\{d_i\}_{i=1}^N$ to the inputs to check the SEs in the last stages of BS's. There are two subcases: 1) both additional tests result in faulty outputs, and 2) only one of the additional tests results in a faulty output. For both subcases, one knows from Proposition 3 the level number of the faulty BS. If both additional tests result in faulty outputs, one can locate the faulty SE by applying the special test vectors with identical routing tags and increasing checking data as we did previously for the case where $s_1, s_2 \in B_2$. For subcase 2, one needs to perform a binary search BSearch4 to identify the faulty BS. Bsearch4 is designed to identify the faulty BS among all the BSs at the same level. The locating process for Case 3 is summarized in Table IV.

VI. Illustrative Examples

In this section, we study some examples of SE faults. A four-level bitonic sorter is used in these examples. For reference, Figs. 8 and 9 illustrate the results of applying $\{a_i\}_{i=1}^N$ and $\{c_i\}_{i=1}^N$, respectively, to the inputs of a fault-free bitonic sorter. The shaded SE's in Figs. 10-13 form the ambiguity set that contains the faulty one. The ambiguity set is finally reduced to contain only one element (i.e. the faulty SE) in each figure.

Example 1: Fig. 10(a) and (b) show an example in which $s_1 \in B_1$ and $s_2 \in E$. The fault can be detected in both phases and the faulty SE can be located according to Theorem 1.

Example 2: Fig. 11(a)-(d) show an example of using BSearch1 and BSearch2 to locate the faulty SE. In this example, we assume both phase II and the additional test applying $\{d_i\}_{i=1}^N$ to the inputs result in correct outputs. Fig. 11(a) shows the result of phase I diagnosis. From the output (the test vectors having a_2 and a_3 as their routing tags switch positions at the output) we know $s_1 \in B_2$. By Proposition 2, we can determine the ambiguity set, as shown in the figure. Fig. 11(b) and (c) illustrate how to identify the faulty BS using BSearch1. In Fig. 11(b), the states of all the SE's in the $level$-4 BS are changed (with respect to the states shown in Fig. 8) and the output is faulty (the test vectors with a_{10} and a_{11} as their routing tags switch positions at the output). Therefore, the faulty SE is in the first $level$-2 BS or the upper $level$-3 BS. In Fig. 11(c), the states of all the SEs in the upper $level$-3 BS and the $level$-4 BS are changed and the output is correct. Thus the faulty SE is in the upper $level$-3 BS. Fig. 11(d) shows the locating process using BSearch2. The state of the second SE in the second stage of the upper $level$-3 BS is changed. Since the output is faulty, we know the faulty SE is the one shown in the figure.

Example 3: Fig. 12(a)-(c) give an example of using BSearch3 to locate the faulty SE. In this example, we assume that the faulty BS (i.e., the $level$-4 BS) has been identified by BSearch1. Fig. 12(a) shows the diagnosis result of phase I, from which we know that the faulty SE is traversed by the expected path of the test vector applied to input 5. Let $\{SE_1, SE_2, SE_3, SE_4\}$ denote the ambiguity set, where SE_i is in stage i. In Fig. 12(b), we change the states of SE_1 and SE_2. Since the output is faulty, the faulty SE is either SE_3 or SE_4. In Fig. 12(c), the states of SE_1, SE_2, and SE_3 are changed and the output is correct. Therefore, SE_3 is the faulty SE.

Example 4: Fig. 13(a)-(b) show an application of BSearch4. In this example, we assume both phase I and phase II result in correct outputs. Moreover, the output is also assumed to be correct when $\{d_i\}_{i=1}^N$ is applied to the

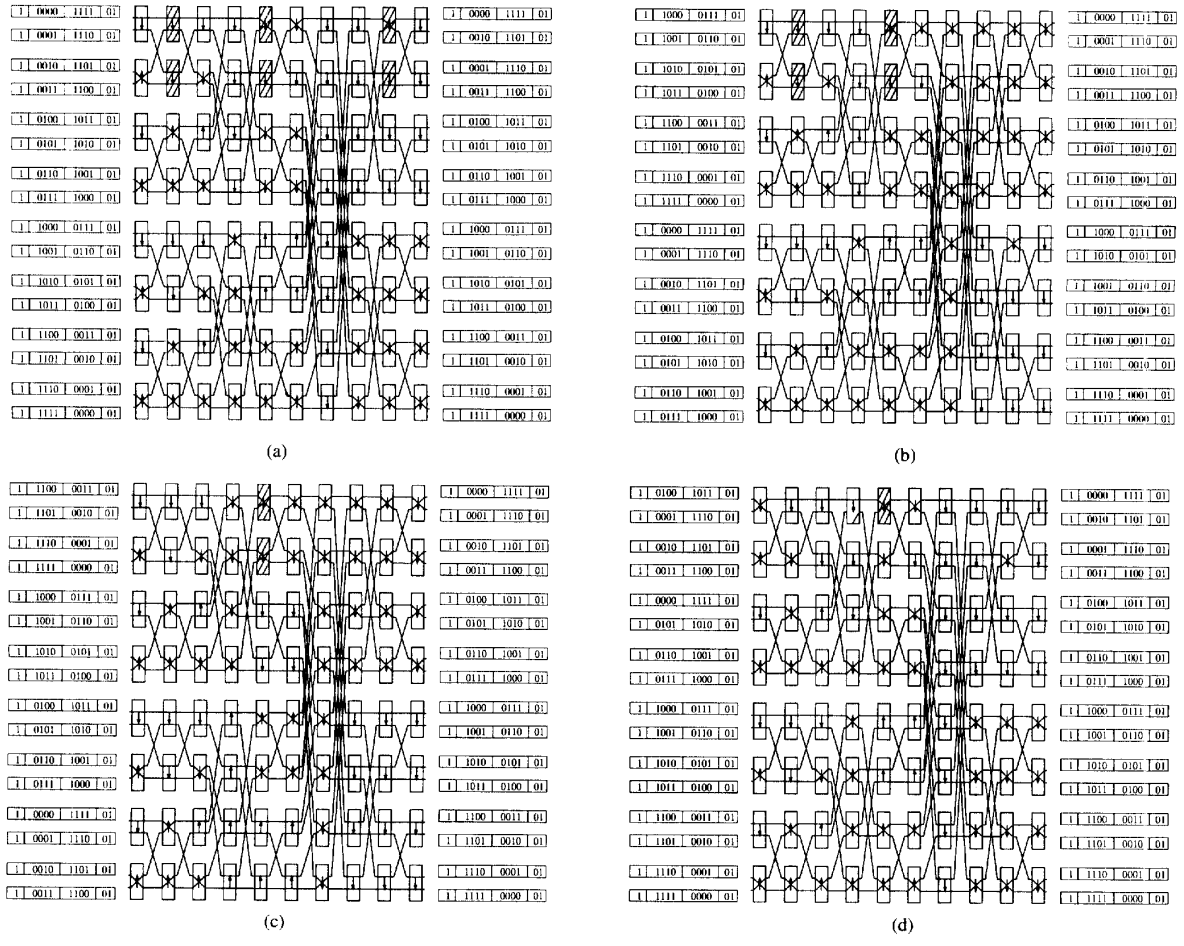


Fig. 11. Diagnosis results for Example 2. (a) Result of phase I diagnosis. (b), (c) Binary search BSearch1, and (d) binary search BSearch2.

inputs. Fig. 13(a) shows the result of applying $\{c_i\}_{i=1}^{N}$ to the inputs. From Proposition 3, we know from the output that the faulty SE is either the upmost one in the last stage of the upper *level*-3 BS or the lowest one in the last stage of the lower *level*-3 BS. In Fig. 13(b), we change the states of all the SEs in the lower *level*-3 BS. Since the output is faulty, we know the upmost SE in the last stage of the upper *level*-3 BS is the faulty one.

VII. Conclusion

We have presented in this paper an efficient fault diagnosis procedure for detecting and locating single faults in a bitonic sorter without state control lines. We showed that at most four tests are required to detect a single fault and most faults require only two tests to be detected. In some cases, one or two binary searches are necessary to locate the faulty SE. For the case where $s_1,\ s_2 \in U - \{S_0\}$, the faulty SE is shown to be in an ambiguity set which contains two elements. To determine which one is faulty, one needs to identify the fault type. The identification process can be found in [6]. Unfortunately, when $s_1,\ s_2 \in U - \{S_0\}$, there are 12 SE faults which cannot be pinpointed at the single SE level and these faults cannot be distinguished from a link stuck fault. This result is also explained in [6]. Further research should focus on diagnosis of multiple faults.

Appendix

In this appendix, we describe the binary searches BSearch1, BSearch2, BSearch3 and BSearch4. We assume that phase I results in a faulty output and phase II does not. The situation for the other case is similar. The faulty SE is assumed to be in stage k of the pth *level-i* BS. Notice that, for a *level-i* BS, there are 2^{k-1} sub-BS from stage k. We assume the faulty SE is in the m^{th} sub-BS from stage k of the faulty BS. In this appendix, we consider routing tags only because they are the only data required to set the state of each SE.

The sequence $\{a_i\}_{i=1}^{N}$ can be partitioned into 2^j groups denoted by $F_1^j, F_2^j, \cdots$, and $F_{2^j}^j$, where $0 \le j \le 2^n$. For example, $\{a_i\}_{i=1}^{N}$ can be partitioned into two groups F_1^1 and F_2^1 if $j = 1$ or four groups F_1^2, F_2^2, F_3^2, and F_4^2 if $j = 2$. It is clear that $F_t^j = F_{2t-1}^{j+1} \oplus F_{2t}^{j+1}$. The numbers entering into the faulty BS are those in F_p^{n-i}. For BSearch2 and

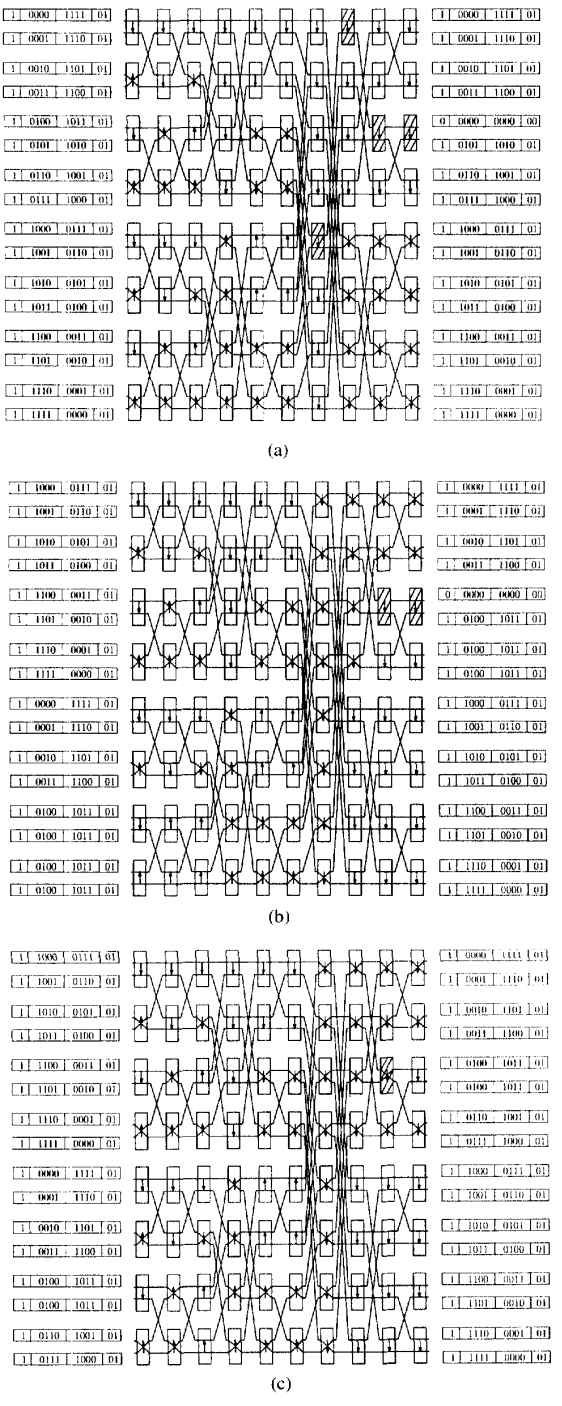


Fig. 12. Diagnosis results for Example 3. (a) Result of phase I diagnosis. (b), (c) Binary search BSearch3.

BSearch3, the group F_p^{n-i} will be further partitioned into 2^l groups $H_1^l, H_2^l, \cdots$, and $H_{2^l}^l$, where l is a variable and $0 \le l \le i$. Owing to space limitations, we only describe how to accomplish a desired search and omit the proofs. Note that the following descriptions are for fault-free bitonic sorters.

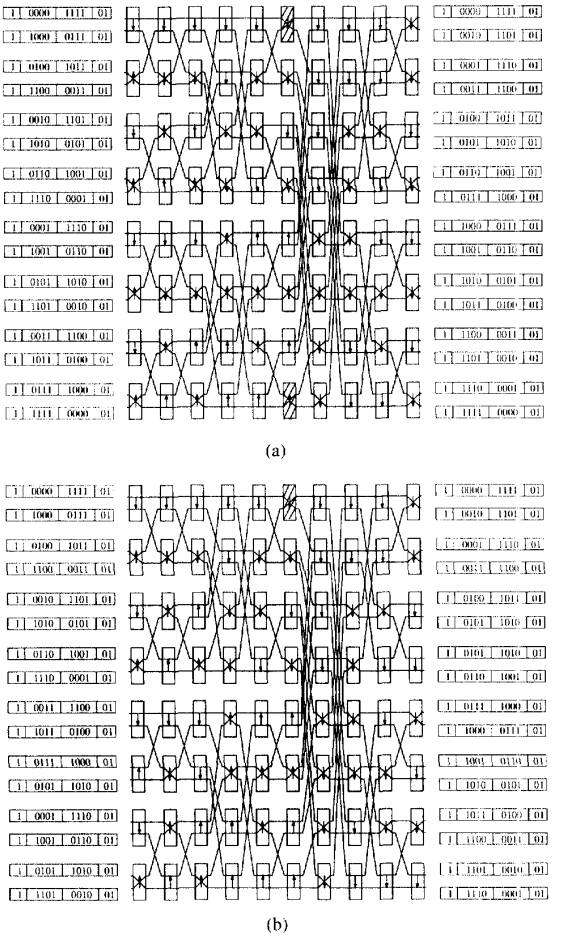


Fig. 13. Diagnosis results for Example 4. (a) Result of applying $\{\sigma_i\}_{i=1}^N$. (b) Additional test to change the states of all the SE's in the lower level-3 BS.

The function of BSearch1 is to identify the faulty BS in an ambiguity set $D_1 = \{BS_{(i-k+1)}, BS_{(i-k+2)}, \cdots, BS_n\}$, where BS_j is a *level-j* BS. Notice that a faulty path passes through all the BS's in D_1. BSearch1 is possible if one can apply an appropriate sequence to change the states (from S_{10} to S_5 or S_5 to S_{10}) of all the SEs in BS_j for all $j \ge n-l$, where $0 \le l \le n-i+k-2$, without changing the state of any SE in BS_r for all $r < n-l$. Suppose BS_j is the f_jth *level-j* BS. It is not hard to see that $f_n = 1$ and $f_{t-1} = 2f_t - 1$ or $2f_t$. The states of all the SEs in the *level-n* BS are changed if the sequence $F_2^1 F_1^1$ is applied to the inputs. To change the states of all the SE's in BS_{n-1} and BS_n, one need only modify the input sequence to become $F_4^2 F_3^2 F_1^1$ for $f_{n-1} = 1$ or $F_2^1 F_2^2 F_1^2$ for $f_{n-1} = 2$. Suppose $f_{n-1} = 2$. The states of all the SE's in BS_{n-2}, BS_{n-1}, and BS_n are changed if the applied sequence becomes $F_2^1 F_4^3 F_3^3 F_1^2$ for $f_{n-2} = 3$ or $F_2^1 F_2^2 F_2^3 F_1^3$ for $f_{n-2} = 4$. This process is continued until we reach BS_{n-l}. Examples of BSearch1 can be found in Fig. 11(b) and (c).

The function of BSearch2 is to identify the faulty SE in an ambiguity set $D_2 = \{SE_1, SE_2, \cdots, SE_{2^{k-1}}\}$. Remember that, from stage k, the faulty BS can be divided into 2^{k-1}

disjoint sub-BS's. The ambiguity set D_2 contains all the SE's in the first stage of the mth sub-BS which contains the faulty SE. Without loss of generality, we assume SE_j is the jth SE in the first stage of this sub-BS. BSearch2 is possible if one can change the states of SE_j for all $j \geq 2^{i-k} - l$, where $0 \leq l \leq 2^{i-k} - 2$, without changing the states of SE_r for all $r < 2^{i-k} - l$. To accomplish this, one need only exchange l numbers in H^k_{2m-1} and H^1_2 if $m \leq 2^{k-2}$ or H^k_{2m} and H^1_1 if $2^{k-2} + 1 \leq m \leq 2^{k-1}$. Fig. 11(d) provides an example of BSearch2.

The function of BSearch3 is to identify the faulty SE in an ambiguity set $D_3 = \{SE_1, SE_2, \cdots, SE_i\}$, where SE_j is an SE in stage j of the faulty BS. Notice that all the SE's in D_3 are traversed by a faulty path. BSearch3 is possible if one can change the states of SE_j for all $j \leq l$, where $1 \leq l \leq i-1$, without changing the states of SE_r for all $r > l$.

Assume $SE_j, 1 \leq j \leq i$, is in the f^{th}_j sub-BS from stage j of the faulty BS. It can be shown that the state of each SE in the upper (lower) sub-BS from stage 2 of the faulty BS is S_{10} (S_5). Consider the case where the faulty path passes through the upper sub-BS from stage 2 (i.e., $f_2 = 1$). One can change the states of SE_j for all $j \leq l$ as follows. Let $t = n-i$. Modify all the numbers in $H^l_{f_{l+1}}$ to be equal to the smallest one in this group and then apply $F^{t+1}_1 F^{t+1}_2 \cdots F^{t+1}_{2p} F^{t+1}_{2p-1} \cdots F^{t+1}_{2^{t+1}}$ to the inputs. Notice that, if $F^{t+1}_1 F^{t+1}_2 \cdots F^{t+1}_{2p} F^{t+1}_{2p-1} \cdots F^{t+1}_{2^{t+1}}$ are applied to the inputs without modifying the numbers in $H^l_{f_{l+1}}$, then the states of all the SE's in D_3 will be changed. With the modification, the states of SE_r are not changed for all $r > l$. For $f_2 = 2$, one can apply the decreasing sequence $\{b_i\}^N_{i=1}$ to the inputs after modifying and changing the positions of the numbers entering into the faulty BS. The technique for this case is similar. Fig. 12(b) shows an example of BSearch3.

For BSearch4, we assume the faulty SE is in the last stage of a *level-i*BS. The function of BSearch4 is to identify the faulty BS in an ambiguity set $D_4 = \{BS_1, BS_2, \cdots, BS_{2^{n-i}}\}$, where BS_j is a *level-i* BS for all $j, 1 \leq j \leq 2^{n-i}$. Since the inputs to these BS's are all different, one can change the states of all the SEs in each BS independently. To change the states of all the SE's in BS_j, one need only partition the test vectors entering into BS_j into two halves and exchange their positions. For example, one can apply the sequence $F^{n-i+1}_1 F^{n-i+1}_2 \cdots F^{n-i+1}_{2j} F^{n-i+1}_{2j-1} \cdots F^{n-i+1}_{2k} F^{n-i+1}_{2k-1} \cdots F^{n-i+1}_{2^{n-i+1}-1} F^{n-i+1}_{2^{n-i+1}}$ to the inputs to change the states of all the SEs in BS_j and BS_k $(k > j)$. Therefore, BSearch4 is possible. Fig. 13(b) gives an example of BSearch4.

REFERENCES

[1] K. E. Batcher, "Sorting networks and their applications," in *Proc. AFIPS 1968 SJCC*, 1968, pp. 307–314.

[2] J. Y. Hui, *Switching and Traffic Theory for Integrated Broadband Networks*. New York: Kluwer Academic, 1990.

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

[4] J. J. Shae, "Fault diagnosis for bitonic sorters," Master's thesis, Nat. Chiao Tung Univ., Taiwan, China.

[5] T. H. Lee, J. J. Chou, and W. T. Wei, "Fault diagnosis for banyan networks without state control lines," submitted for publication.

[6] T. H. Lee and J. J. Chou, "Fault diagnosis of batcher-banyan switching networks," Dep. Commun. Eng., Nat. Chiao Tung Univ., Tech. Rep. NSC82-0404-E009-334, 1993.

[7] J. H. Patel, "Performance of processor-memory interconnection for multiprocessors," *IEEE Trans. Comput.*, vol.C-30, pp. 771–780, Oct. 1981.

Tsern-Huei Lee (M'87) received the B.S. degree from National Taiwan University, Taipei, Taiwan, Republic of China, in 1981, the M.S. degree from the University of California, Santa Barbara, in 1984, and the Ph.D. degree from the University of Southern California, Los Angeles, in 1987, all in electrical engineering.

Since 1987, he has been a member of the faculty of National Chiao Tung University, Hsinchu, Taiwan, China, where he is a Professor of the Department of Communication Engineering and a member of Center for Telecommunications Research. He received an Outstanding Paper Award from the Institute of Chinese Engineers in 1991. His current research interests are in communication protocols, broadband switching networks, and network congestion control.

Jin-Jye Chou receives the B.S. and M.S. degrees in communication engineering from the National Chiao Tung University, Hsinchu, Taiwan, China, in 1987 and 1989, respectively. He is currently pursuing the Ph.D. degree in electronic engineering at National Chiao Tung University, Hsinchu, Taiwan, China.

His principal areas of interest include ATM switch architecture, flow control and fault tolerance of multistage interconnection networks.