The Complexity of Fault Detection Problems for Combinational Logic Circuits

HIDEO FUJIWARA AND SHUNICHI TOIDA

***Abstract*—In this correspondence we analyze the computational complexity of fault detection problems for combinational circuits and propose an approach to design for testability. Although major fault detection problems have been known to be in general NP-complete, they were proven for rather complex circuits. In this correspondence we show that these are still NP-complete even for monotone circuits, and thus for unate circuits. We show that for k-level ($k \geq 3$) monotone/unate circuits these problems are still NP-complete, but that these are solvable in polynomial time for 2-level monotone/unate circuits. A class of circuits for which these fault detection problems are solvable in polynomial time is presented. Ripple-carry adders, decoder circuits, linear circuits, etc., belong to this class. A design approach is also presented in which an arbitrary given circuit is changed to such an easily testable circuit by inserting a few additional test-points.**

***Index Terms*—Combinational circuits, computational complexity, design for testability, fault detection, polynomial algorithms, test generation.**

I. Introduction

Because of the increasing circuit density in LSI/VLSI chips, the fault detection problem is becoming more difficult, and thus an efficient test generation for logic circuits is a matter of prime concern [1]-[3]. Unfortunately, however, major fault detection problems are known to be NP-complete in general [4], [5]. Hence, it appears very unlikely that the fault detection problems can be solved by a polynomial time algorithm. One general approach to this problem is designing easily testable circuits, i.e., design for testability. Many studies have been reported for designing logic circuits for which test sets are easily obtainable [3], [6]-[19].

In this correspondence we show that fault detection problems are NP-complete for k-level ($k \geq 3$) monotone/unate circuits although these circuits are known to be easy to test [15]-[19]. The proof presented here is much simpler than that of [4]. It is also shown that test generation problem for monotone/unate circuits is NP-complete even if the circuits under tests are known to be irredundant. After analyzing the complexity of fault detection problems, we present a class of circuits for which these problems are solvable in polynomial time. Ripple-carry adders, decoder circuits, linear circuits, etc., belong to the above class. A design approach is then presented in which an arbitrary given circuit is changed to such an easily testable circuit by inserting a few additional test-points.

II. Satisfiability Problems

The first NP-complete problem was reported by Cook [20], which is usually referred to as the satisfiability problem (SAT, for short). We give a brief description of this problem in the following since we need it in our discussion of NP-completeness of fault detection problems. For definitions of NP-completeness see [5].

A *literal* is either x or $\bar{x}$ for some variable x, and a *clause* is a sum of literals. A Boolean expression is said to be in *conjunctive normal form* (CNF) if it is a product of clauses. A Boolean expression is *satisfiable* if and only if there exists some assignment of 0's and 1's to the variables that gives the expression the value 1. Then the SAT problem is specified as follows.

Manuscript received July 16, 1981; revised January 7, 1982. This work was supported in part by NSERC of Canada Grant A7396.

H. Fujiwara is with the Department of Electronic Engineering, Osaka University, Osaka, Japan.

S. Toida is with the Department of Systems Design, University of Waterloo, Waterloo, Ont., Canada.

0018-9340/82/0600-0555$00.75 © 1982 IEEE

Satisfiability (*SAT*): Is a Boolean expression satisfiable?

Theorem 1 (*Cook's Theorem* [*20*]): SAT is NP-complete.

An expression is said to be *clause-monotone* if each of its clauses contains either only negated variables or only unnegated variables. The satisfiability for clause-monotone expressions (CM-SAT, for short) is also NP-complete.

Theorem 2: CM-SAT is NP-complete.

For proofs of Theorems 2–4 see [23].

Consider a Boolean expression in conjunctive normal form E with variables $x_1, x_2, \cdots, x_p$ and clauses $C_1, C_2, \cdots, C_q$. A clause C_i is said to *cover* a clause C_j (written $C_i \supseteq C_j$) if all literals in C_j are contained in C_i. An expression E is said to be *reduced* if no clause in the expression covers another. The satisfiability for reduced clause-monotone Boolean expressions (RCM-SAT, for short) is also NP-complete.

Theorem 3: RCM-SAT is NP-complete.

Although SAT, CM-SAT, and RCM-SAT are all NP-complete, the satisfiability problem is solvable in polynomial time if a Boolean expression is monotone or unate. An expression is said to be *monotone* if it contains only unnegated variables. An expression is said to be *unate* if each variable is either only negated or only unnegated. The satisfiability problems for monotone expressions and unate expressions (M-SAT and U-SAT, for short, respectively) have been shown to be solvable in time $0(l)$, where l is the length of an expression.

Theorem 4: M-SAT and U-SAT are solvable in time $0(l)$, where l is the length of an expression.

III. Complexity of Fault Detection

In this correspondence we shall be concerned with multiinput and multioutput combinational circuits composed of AND, OR, NAND, NOR, and NOT gates.

First, we consider the following decision problems.

Fault Detection (*FD, for short*): Can a single stuck fault be detected by input–output experiments?

Irredundancy (*IR*): Is a combinational circuit C irredundant (i.e., can all single stuck faults be detected)?

In general, FD and IR are both known to be NP-complete [4]. In this section we show that these problems are still NP-complete even for k-level ($k \geq 3$) monotone/unate circuits although these are solvable in polynomial time for 2-level monotone/unate circuits. A circuit is said to be of k-level if the maximum number of gates except NOT gates along any path from primary inputs to primary outputs is k.

We shall use the following abbreviations:

kM-FD: FD for k-level monotone circuits,

kU-FD: FD for k-level unate circuits,

kM-IR: IR for k-level monotone circuits,

kU-IR: IR for k-level unate circuits.

We begin with the following theorem.

Theorem 5: 3M-FD and 3U-FD are NP-complete.

Proof: Since monotone circuits are also unate, it is sufficient to prove that 3M-FD is NP-complete.

Obviously, 3M-FD is in NP. Hence, we need to show that some NP-complete problem is polynomially transformable to 3M-FD. We shall transform CM-SAT to 3M-FD.

Given any clause-monotone expression E with variables $x_1, x_2, \cdots, x_p$ and clauses $C_1, C_2, \cdots, C_q$, we construct a 3-level monotone circuit Q_1 as follows (see Fig. 1).

Without loss of generality, we assume that $C_1, C_2, \cdots, C_k$ are the clauses with negated variables and $C_{k+1}, C_{k+2}, \cdots, C_q$ are the clauses with unnegated variables.

Step 1: Construct AND gates $A_1, A_2, \cdots, A_k$ corresponding to the clauses $C_1, C_2, \cdots, C_k$ with negated variables so that each AND gate A_i has the input variables of C_i. for example, suppose a clause $C_i = \bar{a} \vee \bar{b} \vee \bar{c}$, then the output of A_i is $a \cdot b \cdot c$.

Step 2: Connect the above AND gates to OR gates G_1 as shown in Fig. 1.

Step 3: Construct OR gates $O_1, O_2, \cdots, O_{q-k}$ corresponding to the clauses $C_{k+1}, C_{k+2}, \cdots, C_q$ with unnegated variables. For example, the output of O_i is $a \vee b \vee c$ if the clause $C_i = a \vee b \vee c$.

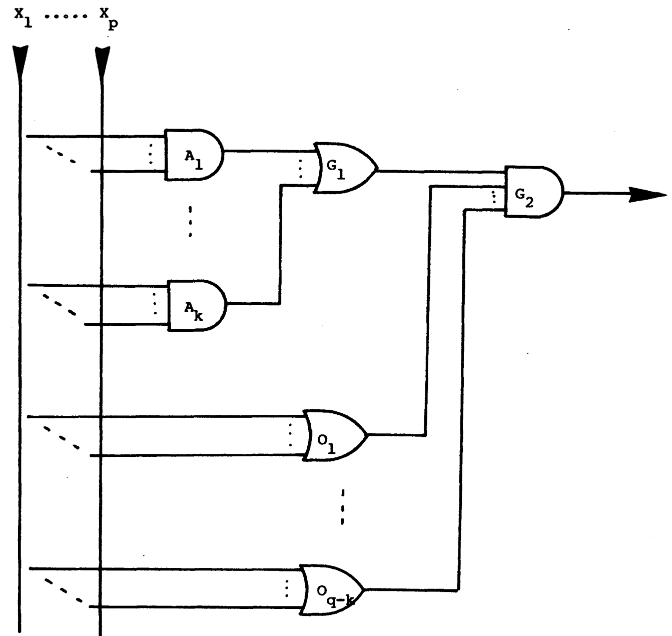

Fig. 1. 3-level monotone circuit Q_1.

Step 4: Connect all the OR gates $O_1, O_2, \cdots, O_{q-k}$, and G_1 to AND gate G_2 as shown in Fig. 1.

In this circuit Q_1, a stuck-at-1 fault at the output of gate G_1, is detectable if and only if there exists a test such that all the outputs of AND gates $A_1, A_2, \cdots, A_k$ are 0 and all the outputs of OR gates $O_1, O_2, \cdots, O_{q-k}$ are 1. Hence, the fault G_1 s-a-1 is detectable if and only if the given expression E is satisfiable.

The above construction can be carried out in an amount of time linear in p and q. Therefore, CM-SAT is polynomially transformable to 3M-FD. Q.E.D.

Theorem 6: 3M-IR and 3U-IR are NP-complete.

Proof: Obviously, 3M-IR is in NP. We show that RCM-SAT is polynomially transformable to 3M-IR.

Given any reduced clause-monotone expression E with variables $x_1, x_2, \cdots, x_p$ and clauses $C_1, C_2, \cdots, C_q$, we construct a 3-level monotone circuit Q_2 as follows (see Fig. 2).

Step 1: Construct the 3-level monotone circuit Q_1 of Fig. 1 in the manner mentioned in the proof of Theorem 5.

Step 2: Add a primary input y and a primary output w to the OR gate G_1 as shown in Fig. 2.

This construction can be performed in time linear in p and q. Hence, there remains to prove that Q_2 is irredundant if and only if E is satisfiable.

Suppose that E is not satisfiable. This implies that there exists no test such that all the outputs of the AND gates $A_1, A_2, \cdots, A_k$ are 0 and all the outputs of the OR gates $O_1, O_2, \cdots, O_{q-k}$ are 1, as shown in the proof of Theorem 5. Hence, a stuck-at-1 fault at line g, which is an input of gate G_2, is not detectable because there exists no test such that $g = 0$ and all other inputs of G_2 are 1. This implies that Q_2 is not irredundant.

Conversely, suppose that E is satisfiable, that is, there exists at least one test such that all the outputs of the AND gates $A_1, A_2, \cdots, A_k$ are 0 and all the outputs of the OR gates $O_1, O_2, \cdots, O_{q-k}$ are 1. We show that Q_2 is irredundant by constructing tests for all single faults.

1) To detect a s-a-1 fault at y, w, and the output of G_1, choose a test such that all the outputs of AND gates $A_1, A_2, \cdots, A_k$ are 0 and $y = 0$.

2) To detect a s-a-0 fault at y, w, and the output of G_1, choose a test such that all the outputs of AND gates $A_1, A_2, \cdots, A_k$ are 0 and $y = 1$.

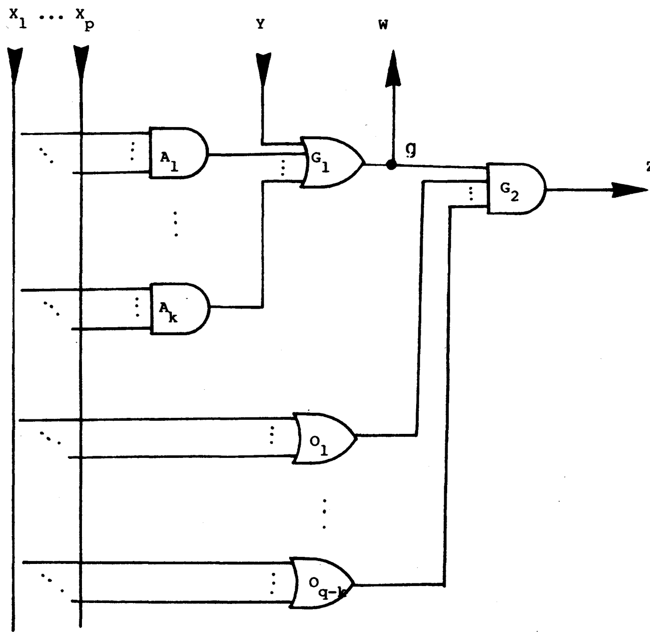

Fig. 2. 3-level monotone circuit Q_2.

3) To detect a s-a-0 fault at g and the output of G_2, choose a test such that all the outputs of OR gates $O_1, O_2, \cdots, O_{q-k}$ are 1 and $y = 1$.

4) To detect a s-a-1 fault at g and the output of G_2, choose a test such that all the outputs of AND gates $A_1, A_2, \cdots, A_k$ are 0, all the outputs of OR gates $O_1, O_2, \cdots, O_{q-k}$ are 1 and $y = 0$. The existence of this test is guaranteed by the hypothesis.

5) To detect a s-a-0 fault at the inputs and the output of AND gate A_i $(1 \leq i \leq k)$, choose a test such that all the inputs of A_i are 1, all the outputs of the other AND gates A_j $(j = 1, 2, \cdots, k; j \neq i)$ are 0 and $y = 0$. This test can be found since E is reduced.

6) To detect a s-a-1 fault at an input h and the output of AND gate A_i $(1 \leq i \leq k)$, choose a test such that $h = 0$, all the other inputs of A_i are 1, all the outputs of the other AND gates A_j $(j = 1, 2, \cdots, k; j \neq i)$ are 0, and $y = 0$. This test can be found since E is reduced.

7) To detect a s-a-1 fault at the inputs and outputs of OR gate O_i $(1 \leq i \leq q - k)$, choose a test such that all the inputs of O_i are 0, all the outputs of other OR gates O_j $(j - 1, 2, \cdots, q - k; j \neq i)$ are 1, and $y = 1$. This test can be found since E is reduced.

8) To detect a s-a-0 fault at an input h and the output of OR gate O_i $(1 \leq i \leq q - k)$, choose a test such that $h = 1$, all the other inputs of O_i are 0, all the outputs of the other OR gates O_j $(j = 1, 2, \cdots, q - k; j\bar{i})$ are 1, and $y = 1$. This test can be found since E is reduced. Q.E.D.

Now we have shown that the fault detection problem and the irredundancy test problem are both NP-complete even for 3-level monotone/unate circuits. These results also imply that the fault detection problem and the irredundancy test problem are in general NP-complete. It was reported earlier that FD and IR are both NP-complete by Ibarra and Sahni [4]. However, the proof in [4] is rather long and complicated. As compared with this, the proof of Theorem 6 in this correspondence is simpler since we only use a simpler circuit Q_2 than the circuit Q in [4].

Next, we consider the following test generation problem with a more stringent condition. Suppose we know that a circuit C is monotone and irredundant and we wish to know how long it will take to find a test for a given fault in C. The following theorem shows that this problem is still NP-complete.

Theorem 7: The problem of finding a test to detect a given fault f in an arbitrary monotone and irredundant circuit C is NP-complete.

Proof: It suffices to show that a polynomial time algorithm for finding a test for f in C can be used to develop a polynomial time algorithm for solving 3M-FD.

Assume that we have a polynomial time algorithm A that finds a test for a given fault f in an arbitrary monotone and irredundant circuit C. Using this algorithm, we can construct a polynomial time algorithm that solves 3M-FD as follows.

Algorithm A

Input: An irredundant monotone circuit C and a single stuck fault f.

Output: A test to detect f in C.

Let $p(\cdot)$ be the polynomial time bound of A.

Algorithm B

Input: A 3-level monotone circuit C' and a single stuck fault f'.

Output: "yes" if there is a test to detect f', and "no" otherwise.

Method:

Step 1: Apply Algorithm A to f' and C'.

Step 2: If A does not halt on f' and C' after $p(\cdot)$ steps, there is no test for f' and the answer is "no."

Step 3: If A halts on f' and C' in less than or equal to $p(\cdot)$ steps, then check whether or not the output is a test for f'. If it is a test for f', then the answer is "yes." Otherwise, there is no test for f', and the answer is "no." Q.E.D.

Although all the above mentioned problems are NP-complete for k-level $(k \geq 3)$ monotone/unate circuits, we can see that such problems are solvable in polynomial time for $k = 2$.

Theorem 8: 2M-FD and 2U-FD are solvable in time complexity $O(m^2)$, where m is the number of lines in a circuit.

Theorem 9: 2M-IR and 2U-IR are solvable in time complexity $O(m^3)$.

For proofs of Theorems 8 and 9, see [23].

A Design for Testability

The result of Theorem 8 indicates that by using 2-level monotone/unate circuits it is possible to convert any circuit into an equivalent circuit for which a test set is obtained in time complexity $O(m^2)$. One realization of a 2-level logic circuit is a programmable logic array (PLA) which is very well suited to LSI/VLSI. A PLA has a structure shown in Fig. 3(a), which is composed of AND array, OR array, and decoders. Fig. 3(a) shows a PLA with 1 bit decoders. The augmented PLA shown in Fig. 3(b) can realize a 2-level monotone circuit by resetting all flip-flops of a shift register. Then the equivalent monotone PLA can be tested in time complexity $O(m^2)$. The Exclusive-OR gates and the shift register can be tested by using extra AND term in time complexity $O(n)$ where n is the number of inputs.

IV. Polynomial Time Class

We have shown that the fault detection problem is still NP-complete even for monotone/unate circuits. However, there are many circuits for which the fault detection problem can be solved in polynomial time, e.g., linear circuits, decoder circuits, parallel adder, etc. In this section we present a class of circuits which contains the above circuits and for which the fault detection problem can be solved in polynomial time of the number of lines in the circuits.

A fan-out-point P is called a *head* fan-out-point if there is a path from a primary input to p without encountering any other fan-out-point.

A combinational circuit C is said to be *k-head-fan-out-bounded* if C can be partitioned into subcircuits, called *blocks*, $B_1, B_2, \cdots, B_t$ such that:

1) there is no reconvergent path in the interconnection of blocks $B_1, B_2, \cdots, B_t$, and

2) for each block B_i $(1 \leq i \leq t)$ the number of head fan-out-points in B_i plus the number of lines coming into B_i from other blocks except primary inputs is at most k.

Note that there may exist parallel lines between blocks (see Fig. 4).

Example 1: Consider a parallel binary adder of p bits constructed by cascading p stages of one-bit full adders called *ripple-carry adder*. Fig. 5(a) shows a p stage ripple-carry adder and Fig. 5(b) shows an implementation for one-bit full adder with three head fan-out-points.

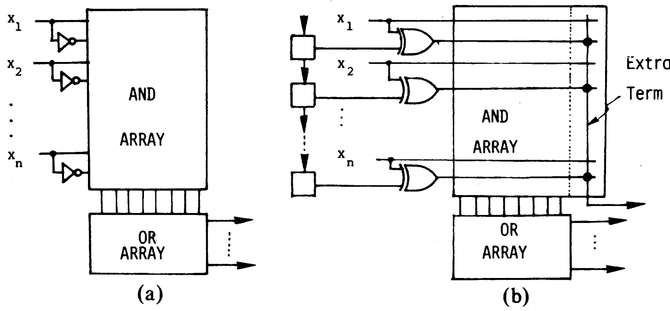

Fig. 3. Easily testable PLA. (a) PLA. (b) Augmented PLA.

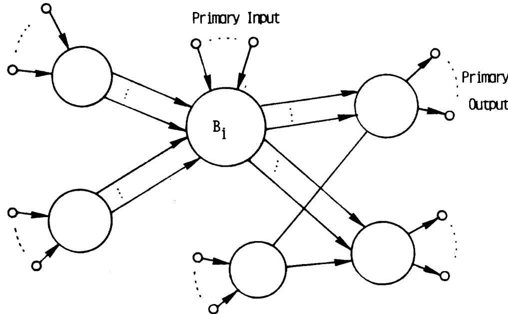

Fig. 4. k-head-fan-out-bounded circuit.

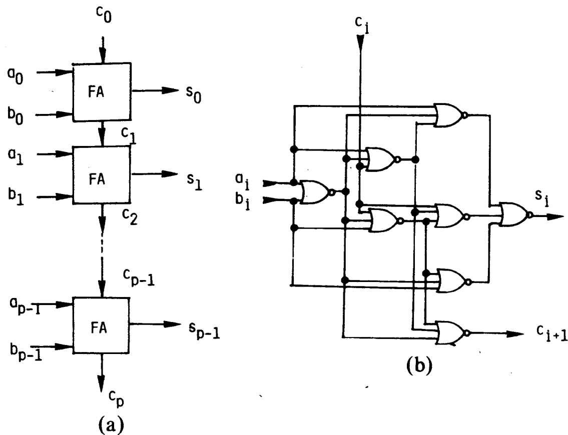

Fig. 5. Ripple-carry adder. (a) Ripple-carry adder. (b) Full adder (FA).

Partition this adder so that each block corresponds to a full adder. Then we see that the ripple-carry adder is a 3-head-fan-out-bounded circuit.

Example 2: Consider a gate-minimum p-bit parallel adder designed by Lai and Muroga [22] whose block diagram is shown in Fig. 6. This circuit is a 6-head-fan-out-bounded circuit.

Example 3: The Exclusive-OR tree realization of a linear function is a 2-head-fan-out-bounded circuit since each Exclusive-OR gate can be realized by two head fan-out-points.

Example 4: A p-bit decoder can be realized by a circuit with 2^p AND gates and p head fan-out-points. Hence, it is a p-head-fan-out-bounded circuit.

Every circuit can be decomposed into subcircuits, called *fan-out-free* circuits, which do not contain fan-out-points. We begin with a lemma for fan-out-free circuits. For proofs of Lemmas 1 and 2, see [23].

Lemma 1: Let C be a fan-out-free circuit with inputs $x_1, x_2, \cdots, x_n$ and output z. Suppose that the values 0, 1, D, and $\overline{D}$ are assigned at Z and some of the inputs of C. Then there is an $O(m)$ algorithm to find an input pattern by assigning values to the remaining inputs which justifies the value of Z, where m is the number of lines in C.

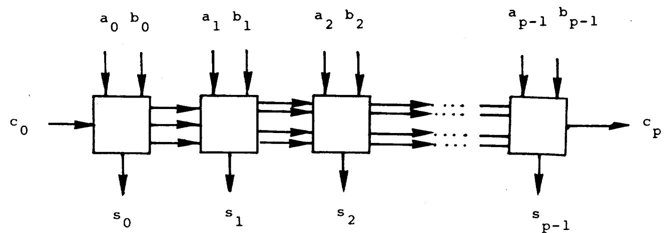

Fig. 6. Gate-minimum p-bit parallel adder.

Lemma 2: Suppose that the values 0, 1, D, and $\overline{D}$ are assigned on all head fan-out-points. Then the values of other nonhead fan-out-points are determined uniquely by the implication operation of time complexity $o(m)$ provided that no inconsistency occurs, where m is the number of lines in a circuit.

Theorem 10: Let C be a circuit with k head fan-out-points. Then there is an algorithm of time complexity $0(4^k \cdot m)$ to find a test for a single stuck fault in C, where m is the number of lines in C.

Proof: The test generation for a fault on line L can be performed in the following steps.

Step 1: Fix a value of L to $D(\overline{D})$ if a fault is stuck-at-0 (1) fault. Assign a value 0, 1, D, or $\overline{D}$ to each head fan-out-point in C to do the following steps for each combination of values on the head fan-out-points. If there is no untried combination remaining and no test has been found, then there exists no test and so stop.

Step 2: For each assignment determine implications, that is, determine all the line values that are implied uniquely by the values assigned on the head fan-out-points. If any inconsistency occurs, then go back to Step 1.

Note that if no inconsistency occurs, the values of all (head and nonhead) fan-out-points are determined uniquely. (See Lemma 2.)

Step 3: for each fan-out-free subcircuit C_i to whose output a value is assigned, find an input assignment to justify the output value of C_i by using the algorithm of Lemma 1. If the justification fails, then go back to Step 1.

Step 4: Check whether or not every D-path starts at the line L under test. If not, go back to Step 1.

Step 5: Check whether or not there is at least one D-path ending at a primary output. If not and if there exists no fan-out-free subcircuit to whose output no value is assigned yet, then go back to Step 1. Otherwise, for each fan-out-free subcircuit C_i to whose output no value is assigned, find an input assignment to propagate either D or $\overline{D}$ to the output of C_i by using the algorithm of Lemma 1. If the propagation fails for every subcircuit, then go back to Step 1. Otherwise, we have found a test, and so stop.

The above algorithm requires the enumeration of at most 4^k combinations of values on head fan-out-points. By Lemma 1, we see that Steps 3 and 5 can be performed in time $0(m_i)$, where m_i is the number of lines in a subcircuit C_i. By Lemma 2, we see that Step 2 can be performed in time $0(m)$. Hence, the above algorithm can be carried out in $0(4^k \cdot m)$ time. Q.E.D.

The algorithm shown in the proof of Theorem 10 generates all the combinations of values 0, 1, D, $\overline{D}$ on all head fan-out-points, i.e., 4^k assignments from the beginning. We can improve this by an implicit enumeration technique. In generating a test, the algorithm creates a decision tree in which a choice is made at each decision node on the value of a line out of a number of possible values. The initial choice is arbitrary, but it may be necessary during the execution of the algorithm to return to the same node and consider another possible choice. This is called a backtrack. In order to guarantee the time complexity $0(4^k \cdot m)$, we have to make sure that backtracks occur only at k head fan-out-points.

We thus have shown that there exists an algorithm of time complexity $0(4^k \cdot m)$ to find a test for a given fault of a circuit. We can easily extend the result to a class of k-head-fan-out-bounded circuits as in the following theorem.

Theorem 11: Let C be a k-head-fan-out-bounded circuit. Then

there is an algorithm of time complexity $0(16^k \cdot m)$ to find a test for a single stuck fault in C, where m is the number of lines in C.

Proof: The proof is similar to that of Theorem 11 in [23].

From Theorem 11 we have the following corollary.

Corollary 1: Let C be a k-head-fan-out-bounded circuit such that $k = \log_2 p(m)$ for some polynomial $p(m)$, where m is the number of lines in C. Then the fault detection problem for C is solvable in time complexity $0(p(m)^4 \cdot m)$.

For a k-head-fan-out-bounded circuit, in order to solve the fault detection problem in polynomial time, it is sufficient that $k = \log p(m)$ for some polynomial $p(m)$.

Theorems 10 and 11 indicate that the complexity or difficulty of fault detection is related to the number of head fan-out-points in the circuit. However, if there is no reconvergent path, a test can be found in time $0(m)$ even though there exist fan-out-points in the circuit. This implies that it is sufficient to consider only reconvergent fan-out-points in order to get a similar result to Theorems 10 and 11. To further improve the time complexity we define a more general set S of points which satisfies the following condition.

Condition 1: For any reconvergent fan-out-point p, either p belongs to S or any path from each primary input to p contains at least one point in S.

Obviously, the set of all head-fan-out-points satisfies Condition 1.

In order to find the smallest set that satisfies Condition 1, we first construct a directed graph $G = (V, E)$ such that V is the set of vertices composed of all fan-out-points plus two new vertices, a source s and a sink t, and E is the set of arcs such that

1) $(s, h) \in E$ for all head fan-out-points h,

2) $(r, t) \in E$ for all reconvergent fan-out-points r, and

3) $(v, u) \in E, v, u \neq s, t$ if there is a path from v to u in the original circuit without encountering other fan-out-points.

For this graph G, we can easily see that a set S of points satisfying Condition 1 corresponds to a set of vertices which cuts s and t. Hence, the problem of finding the smallest set $\hat{S}$ satisfying Condition 1 can be reduced to the problem of finding the minimum vertex cut set.

A Design for Testability

In the above discussion we have shown that the complexity of fault detection is closely related to the number of reconvergent fan-out-points in the circuit. Therefore, if we can reduce the number of reconvergent fan-out-points, finding a test becomes easier. The reduction of the number of reconvergent fan-out-points can be done by placing in reconvergent paths an additional Exclusive-OR gate as shown in Fig. 7. By controlling the value of x, we can always set an arbitrary value to line b, and observe the value of line a. Therefore, placing an Exclusive-OR gate on a line L is equivalent to cutting L logically. The total number of additional inputs and outputs can be reduced to two if we use a scan shift register approach such as LSSD [10].

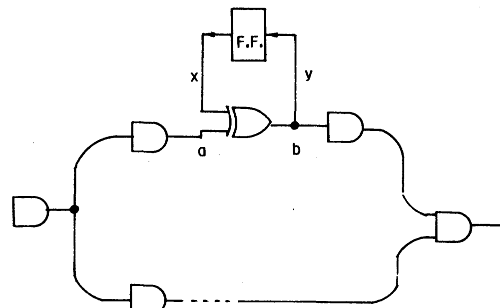

Fig. 7. Reconvergent path.

V. CONCLUSION

In this correspondence we have shown that the fault detection problem and the irredundancy problem are both NP-complete even with a stringent condition such as monotonicity and unateness of circuits. It is shown that for k-level ($k \geq 3$) monotone/unate circuits these problems are NP-complete although these can be solved in polynomial time for 2-level monotone/unate circuits. This implies that by using 2-level monotone/unate circuits it is possible to convert any circuit into an equivalent easily testable circuit. We have also presented an implementation of an easily testable PLA.

We have introduced a class of circuits solvable in polynomial time, called k-head-fan-out-bounded circuits. If K is bounded by $\log_2 p(m)$ for some polynomial $p(m)$, where m is the number of lines in a circuit, then the fault detection problem is solvable in time $0(p(m)^4 \cdot m)$. Parallel adders, decoder circuits, linear circuits, etc., belong to this class. We have also presented a more general class by considering reconvergent fan-out-points. A design approach is then presented in which an arbitrary given circuit is converted to such an easily testable circuit by placing Exclusive OR gates in reconvergent paths.

The algorithm presented in the proof of Theorem 10 is not efficient since it considers all the combinations of values 0, 1, D, $\bar{D}$ at all head fan-out-points in a circuit. However, since the objectives of this correspondence are to clarify the computational complexity of fault detection problems, to present some classes of circuits for which a test can be found in polynomial time, and to give an approach to the design for testability, we have not tried to develop more efficient algorithms in this correspondence. Better and more efficient algorithms for test generation are now being developed in our group.

REFERENCES

[1] J. P. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM J. Res. Develop.*, vol. 10, pp. 278-291, July 1966.

[2[P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-30, pp. 215-222, Mar. 1981.

[3] M. A. Breur and A. D. Friedman, *Diagnosis and Reliable Design of Digital Systems.* Woodland Hills, CA: Computer Science Press, 1976.

[4] P. H. Ibarra and S. K. Sahni, "Polynomially complete fault detection problems," *IEEE Trans. Comput.*, vol. C-24, pp. 242-249, Mar. 1975.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, CA: Freeman, 1979.

[6] M. J. Y. Williams and J. B. Angell, "Enhancing testability of large-scale integrated circuits via test points and additional logic," *IEEE Trans. Comput.*, vol. C-22, pp. 46-60, Jan. 1973.

[7] J. P. Hayes, "On modifying logic networks to improve their diagnosability," *IEEE Trans. Comput.*, vol. C-23, pp. 56-62, Jan. 1974.

[8] K. K. Saluja and S. M. Reddy "On minimally testable logic networks," *IEEE Trans. Comput.*, vol. C-23, pp. 552-554, May 1974.

[9] A. Vogel and W. Coy, "Improving testability by simple modifications of combinational nets," Berichte der Abteilung Inform. der Univ. Dortmund, NR. 54, 1978.

[10] T. W. Williams and K. P. Parker, "Testing logic networks and designing for testability," *Computer*, pp. 9-21 and references, Oct. 1979.

[11] S. J. Hong and D. L. Ostapko, "FITPLA: A programmable logic array for function independent testing," in *Proc. FTCS*, vol. 10, Oct. 1980, pp. 131-136.

[12] H. Fujiwara, and K. Kinoshita, "A design of programmable logic arrays with universal tests," *IEEE Trans. Comput.*, vol. C-30, pp. 823-828, Nov. 1981.

[13] M. A. Breuer, "Generation of fault tests for linear logic networks," *IEEE Trans. Comput.*, vol. C-21, pp. 79-83, Jan. 1972.

[14] S. C. Seth and K. L. Kodandapani, "Diagnosis of faults in linear tree networks," *IEEE Trans. Comput.*, vol. C-26, pp. 29-33, Jan. 1977.

[15] R. Betancourt, "Derivation of minimum test sets for unate logical circuits," *IEEE Trans. Comput.*, vol. C-20, pp. 1264-1269, Nov. 1971.

[16] R. Dandapani, "Derivation of minimal test sets for monotonic logic circuits," *IEEE Trans. Comput.*, vol. C-22, pp. 657-661, July 1973.

[17] S. B. Akers, Jr., "Universal test sets for logic networks," *IEEE Trans. Comput.*, vol. C-22, pp. 835-839, Sept. 1973.

[18] S. M. Reddy, "Complete test sets for logic functions," *IEEE Trans. Comput.*, vol. C-22, pp. 1016-1020, Nov. 1973.

[19] H. Fujiwara, "On closedness and test complexity of logic circuits," *IEEE Trans. Comput.*, vol. C-30, pp. 556-562, Aug. 1981.

[20] S. A. Cook, "The complexity of theorem proving procedures," in *Proc. 3rd ACM Symp. Theory of Comput.*, 1971, pp. 151–158.
[21] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Reading, MA: Addison-Wesley, 1974.
[22] H. C. Lai and S. Muroga, "Minimum parallel binary adders with NOR (NAND) gates," *IEEE Trans. Comput.*, vol. C-28, pp. 648–659, Sept. 1979.
[23] H. Fujiwara and S. Toida, "The complexity of fault detection problems for combinational logic circuits," Dep. Syst. Design, Univ. of Waterloo, Waterloo, Ont., Canada, Tech. Rep. 78-P-HW-150681, June 1981.

0018-9340/82/0600-0560$00.75 © 1982 IEEE