THE COMPLEXITY OF FAULT DETECTION: AN APPROACH TO DESIGN FOR TESTABILITY

Hideo Fujisawa
Department of Electronic Engineering
Osaka University
Yamada-Oka 2-1, Suita
Osaka, 565 JAPAN

Shunichi Toida
Department of Systems Design
University of Waterloo
Waterloo, Ontario
CANADA N2L 3G1

ABSTRACT
In this paper we analyse 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 paper we show that these are still NP-complete even for monotone circuits, and thus for unate circuits. We show that for k-level (k ≥ 3) monotone/unate circuits these problems are still NP-complete but that they 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.

1. INTRODUCTION
Because of the increasing circuit density in LSI/VLSI chips, 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 reported for designing logic circuits for which test sets are easily obtainable [3, 6–19].

In this paper, we show that fault detection problems are NP-complete for k-level (k ≥ 3) monotone/unate circuits though 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 test 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. We show that the fault detection problems for them are solvable in time complexity O(m), where m is the number of lines. 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.

2. 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 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:

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 un-negated 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 [21].

Consider a Boolean expression in conjunctive normal form E with variables x1, x2, . . ., x and clauses C1, C2, . . ., Cn. A clause Cj is said to cover a clause Cj′ (written Cj ⊇ Cj′) if all literals in Cj′ are contained in Cj. 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

* This work was supported in part by NSERC of Canada Grant A7396.
it contains only un-negated variables. An expression is said to be unate if each variable is either only negated or only un-negated. 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 \( O(2^k) \), where \( k \) is the length of an expression.

Theorem 4: M-SAT and U-SAT are solvable in time \( O(2^k) \), where \( k \) is the length of an expression.

3. COMPLEXITY OF FAULT DETECTION

In this paper we shall be concerned with multi-input and multi-output 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 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 though 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.
\( kW-IR \): IR for \( k \)-level monotone circuits.
\( kW-IR \): 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 \( x_1, x_2, \ldots, x_p \) and clauses \( C_1, C_2, \ldots, C_q \), we construct a 3-level monotone circuit \( Q_1 \) as follows (see Fig.1):

Without loss of generality, we assume that \( C_1, C_2, \ldots, C_k \) are the clauses with negated variables and \( C_{k+1}, C_{k+2}, \ldots, C_q \) are the clauses with un-negated variables.

Step 1) Construct AND gates \( A_1, A_2, \ldots, A_k \) corresponding to the clauses \( C_1, C_2, \ldots, C_k \) with negated variables so that each AND gate \( A_i \) has the input variables of \( C_i \). For example, suppose a clause \( C_i = a \lor b \lor c \), then the output of \( A_i \) is \( a \lor b \lor c \).

Step 2) Connect the above AND gates to OR gate \( G_1 \) as shown in Fig.1.

Step 3) Construct OR gates \( G_1, G_2, \ldots, G_{q-k} \) corresponding to the clauses \( C_{k+1}, C_{k+2}, \ldots, C_q \) with un-negated variables. For example, the output of \( G_i \) is \( a \lor b \lor c \) if the clause \( C_i = a \lor b \lor c \).

Step 4) Connect all the OR gates \( G_1, G_2, \ldots, G_{q-k} \) 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 \( G_2 \) is detectable if and only if there exists a test such that all the outputs of AND gates \( A_1, A_2, \ldots, A_k \) are 0 and all the outputs of OR gates \( G_1, G_2, \ldots, G_{q-k} \) are 1. Hence, the fault \( G_1 \) s-a-l 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.

\[ \text{Figure 1. 3-level monotone circuit } Q_1. \]

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 \( x_1, x_2, \ldots, x_p \) and clause \( C_1, C_2, \ldots, C_q \), we construct a 3-level monotone circuit \( Q_2 \) as follows (see Fig.2):

Step 1) Construct the 3-level monotone circuit \( Q_2 \) 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 \( A_1, A_2, \ldots, A_k \) are 0 and all the outputs of the OR gates \( O_1, O_2, \ldots, O_{q-k} \) are 1, as shown in the proof of Theorem 5. Hence, a stuck-at-1 fault at line \( x \), which is an input of gate \( G_2 \), is not detectable because there exists no test such that \( y=0 \) and all other inputs of \( O_2 \) 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 \( A_1, A_2, \ldots, A_k \) are 0 and all the outputs of the OR gates \( O_1, O_2, \ldots, O_{q-k} \) are 1. We show that \( Q_2 \) is irredundant by constructing tests for all single faults.

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

2. To detect a stuck-at-0 fault at \( y \), and the outputs of \( G_2 \), choose a test such that all the outputs of AND gates \( A_1, A_2, \ldots, A_k \) are 0 and \( y=1 \).

3. To detect a stuck-at-0 fault at \( y \) and the output of \( G_2 \), choose a test such that all the outputs of \( G_2 \) are 1 and \( y=1 \).

4. To detect a stuck-at-1 fault at \( y \) and the output of \( G_2 \), choose a test such that all the outputs of \( G_2 \) are 0 and \( y=0 \). The existence of this test is guaranteed by the hypothesis.

5. To detect a stuck-at-0 fault at \( y \) and the outputs of \( G_2 \), choose a test such that all the outputs of \( G_2 \) are 0 and \( y=0 \). This test can be found since \( E \) is reduced.

6. To detect a stuck-at-1 fault at \( y \) and the output of \( G_2 \), choose a test such that all the outputs of \( G_2 \) are 1 and \( y=0 \). This test can be found since \( E \) is reduced.

7. To detect a stuck-at-1 fault at \( y \) and the outputs of \( G_2 \), choose a test such that all the outputs of \( G_2 \) are 0 and \( y=0 \). This test can be found since \( E \) is reduced.

8. To detect a stuck-at-1 fault at \( y \) and the output of \( G_2 \), choose a test such that all the outputs of \( G_2 \) are 1 and \( y=0 \). This test can be found since \( E \) is reduced.

Figure 2. 3-level monotone circuit \( Q_2 \).

9. To detect a stuck-at-1 fault at \( y \) and the outputs of \( G_2 \), choose a test such that all the outputs of \( G_2 \) are 0 and \( y=0 \). This test can be found since \( E \) is reduced.

10. To detect a stuck-at-1 fault at \( y \) and the output of \( G_2 \), choose a test such that all the outputs of \( G_2 \) are 1 and \( y=0 \). This test can be found since \( E \) is reduced.

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 paper 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:**

1. Apply Algorithm A to \( f' \) and \( C' \).
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".
3. If \( A \) halts on \( f' \) and \( C' \) in less...
than or equal to \( p(.) \) 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:** \( ZN-FD \) and \( ZD-FD \) are solvable in time complexity \( O(m^2) \), where \( m \) is the number of lines in a circuit.

**Theorem 9:** \( ZN-IR \) and \( ZD-IR \) are solvable in time complexity \( O(m^3) \).

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

### 4. 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 adders, 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.

First, we introduce an extended operation of implication, called an *extended implication*, in the following.

Consider a combinational circuit \( C \) in which the values \( 0,1,D, \) and \( \bar{D} \) may be assigned at some lines. Let \( V(x_i) \) be a subset of \( \{0,1,D,\bar{D}\} \) with respect to line \( x_i \) in \( C \). We construct each set \( V(x_i) \) working breadth-first from primary inputs to primary outputs as follows:

For each primary input \( x_i \) (1 \( \leq i \leq n \)), we let

\[
V(x_i) = \begin{cases} 
    (a) & \text{if the value } a \text{ is already assigned at } x_i, \text{ where } a \in \{0,1,D,\bar{D}\} \\
    (0,1) & \text{otherwise.} 
\end{cases}
\]

For an AND gate with input \( x_1, x_2, \ldots, x_p \) and output \( Y \), we construct \( V(Y) \) in the following steps:

1. If \( 0 \in V(x_j) \) for some \( x_j \), then \( 0 \in V(Y) \).
2. If \( D \in V(x_k) \) and \( \bar{D} \in V(x_j) \) for some \( x_k \) and \( x_j \), then \( 0 \in V(Y) \).
3. If \( 1 \in V(x_i) \) for all \( x_i \), then \( 1 \in V(Y) \).
4) If \( D \in V(x_j) \) for some \( x_j \) and \( D \) or \( 1 \in V(x_j) \) for all other \( x_j \) (\( j \neq 1 \)), then \( D \in V(Y) \).

5) If \( D \in V(x_j) \) for some \( x_j \) and \( D \) or \( 1 \in V(x_j) \) for all other \( x_j \) (\( j \neq 1 \)), then \( D \in V(Y) \).

Similarly, for other types of gates, OR, NAND, NOR, NOT, we can construct \( V(Y) \).

When we encounter a line \( k \) on which a value \( a \) is already assigned, we check whether or not \( V(k) \) contains \( a \). If not, then an inconsistency occurs. Otherwise, we redefine \( V(k) = \{a\} \) and continue the computation.

When we encounter a line \( L \) with a fault, we check whether or not

\[
1 \in V(L) \quad \text{for a s-a-0 fault, and} \\
0 \in V(L) \quad \text{for a s-a-1 fault.}
\]

If not, then an inconsistency occurs. Otherwise we redefine

\[
V(L) = \begin{cases} 
(b) & \text{for a s-a-0 fault} \\
(D) & \text{for a s-a-1 fault}
\end{cases}
\]

and continue the computation.

Obviously, the above computation requires at most \( O(m) \) time, where \( m \) is the number of lines in \( C \).

**Lemma 1:** Let \( C \) be a reconvergence-free circuit. Suppose that the values 0, 1, \( D \), and \( D \) are assigned at some line in \( C \) and that the extended implication for the assignment is performed without occurring any inconsistency. Let \( a \) be a value in \( V(k) \) for some line \( k \) in \( C \). Then there is an \( O(m) \) algorithm to find an input pattern which justifies the value \( a \).

A set \( S \) of points in a circuit \( C \) is called a head point set of \( C \) if for any reconvergent fanout-point \( p \), either \( p \) belongs to \( S \) or any path from each primary input to \( p \) contains at least one point in \( S \).

In order to find the smallest head point set, we first construct a directed graph \( G(V, E) \) such that \( V \) is the set of vertices composed of all fanout-points plus two new vertices, a source \( s \) and a sink \( t \), and \( E \) is the set of arcs such that

1) \((s, h) \in E \) if there is a path from a primary input to \( h \) in the original circuit without encountering other fanout-points,

2) \((r, t) \in E \) for all reconvergent fanout-points \( r \), and

3) \((v, u) \in E \) if there is a path from \( v \) to \( u \) in the original circuit without encountering other fanout-points.

For this graph \( G \), we can easily see that a head point set \( S \) corresponds to a set of vertices which cuts \( s \) and \( t \). Hence, the problem of finding the smallest head point set can be reduced to the problem of finding the minimum vertex cut set.

**Lemma 2:** Let \( S \) be a head point set of a circuit \( C \). Suppose that the values 0, 1, \( D \), and \( D \) are assigned on all head points in \( S \). Then the values of other non-head reconvergent fanout-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 \( C \).

**Theorem 10:** Let \( C \) be a circuit and let \( S \) be a head point set with \( k \) head points. Then there is an algorithm of time complexity \( O(4^k \cdot m) \) 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:

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

2) For each assignment, determine implications, that is, for all the line values which are implied uniquely by the values assigned on the head points. If any inconsistency occurs, then go back to step 1.

3) Perform extended implications working breadth-first from primary inputs to primary outputs, that is, compute \( V(k) \) for all lines \( k \).

4) If any inconsistency occurs, then go back to step 1.

5) Check whether or not every \( D \)-path starts at the line \( L \) under test, where a \( D \)-path is a path such that \( V(k) \) contains either \( D \) or \( D \) for all lines \( k \) on the path. If not, then go back to step 1.

6) Check whether or not there is at least one \( D \)-path ending at a primary output. If not, 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 points. By Lemma 1, we see that Steps 3 and 5 can be performed in time \( O(m) \), where \( m \) is the number of lines. By Lemma 2, we see that Step 2 can be performed in time \( O(m) \). Hence, the above algorithm can be carried out in \( O(4^k \cdot m) \) time.

The algorithm shown in the proof of Theorem 10 generates all the combinations of values 0, 1, \( D \), \( D \) on all head 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 \( O(4^k \cdot m) \), we have to make sure that backtracks occur only at \( k \) head points.

Next we shall consider a combinational circuit \( C \) which is partitioned into sub-circuits, called blocks, \( B_1, B_2, \ldots, B_r \). A fanout-point \( p \) in a block
B is called a reconvergent fanout-point with respect to \( B_k \) if there exist two paths both of which start from \( p \) and either reconverge within \( B_k \) or arrive at an adjacent block \( B_i \) \((j \neq i)\). A non-primary input of \( B_i \) is a signal line coming into \( B_i \) from another block except a primary input of the circuit C.

A head point set \( S_i \) of a block \( B_k \) is a set of points such that

1) all non-primary inputs of \( B_k \) are contained in \( S_i \), and
2) for any reconvergent fanout-point \( p \) with respect to \( S_i \), either \( p \) belongs to \( S_i \) or any path from each primary input to \( p \) contains at least one point in \( S_i \).

For a combinational circuit C partitioned into blocks, we define a directed graph \( G_c \) with respect to the circuit C and its blocks such that each vertex represents a block and each arc represents connection between blocks (see Fig. 4).

A combinational circuit C is said to be \( k \)-head-bounded if C can be partitioned into blocks \( B_1 \)'s \((i=1,2,\ldots,t)\) such that

1) the graph \( G_c \) with respect to C and \( B_1 \)'s is acyclic and has no reconvergent path, and
2) for each block \( B_i \) \((1 \leq i \leq t)\), there is a head point set \( S_i \) with at most \( k \) head points.

\[ \text{(a) Circuit C partitioned into blocks.} \]

\[ \text{(b) Graph } G_c \]

\[ \text{Figure 4. } k \text{-head-bounded circuit.} \]

\[ \text{(a) Ripple-carry adder} \]

\[ \text{(b) Full adder (FA)} \]

\[ \text{Figure 5. Ripple-carry adder.} \]
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 fanout-points in the circuit. Therefore, if we can reduce the number of reconvergent fanout-points, finding a test becomes easier. The reduction of the number of reconvergent fanout-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 $b$, that is, we can set $b$ to 0 by choosing $x = 1$ and to 1 by choosing $x = 0$. We can also observe the value of line $a$ through $y$, that is, $a = x \oplus y$. Therefore, placing an Exclusive-OR gate on a line $i$ 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].

5. CONCLUSION

In this paper, we have shown that the fault detection problem and the irredundancy problem are both NP-complete even with a stringent condition such as monotone or unateness of circuits. It is shown that for $k$-level $(k>3)$ monotone/unate circuits these problems are NP-complete though 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 as an easily testable PLA.

We have introduced a class of circuits solvable in polynomial time, called $k$-head-bounded circuits. If $k$ is bounded by $\log_{p}(m)$ for a polynomial $p(m)$, where $m$ is the number of inputs in a circuit, then the fault detection problem is solvable in time $O(p(m)^{n} \cdot m)$. Parallel adders such as
ripple-carry adders and gate-minimum adders, linear circuits, decoder circuits, etc. can belong to this class. Indeed they are 3-, 2-, and 0-head-bounded, respectively. 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, 2, 3 at all head points in a circuit. However, since the objectives of this paper 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 algorithm in this paper. Better and more efficient algorithms for test generation are now being developed in our group.

REFERENCES


