COMPUTATIONAL COMPLEXITY OF CONTROLLABILITY/OBSERVABILITY PROBLEMS FOR COMBINATIONAL CIRCUITS

Hideo Fujiwara

Reprinted from PROCEEDINGS OF THE EIGHTEENTH
INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING
Tokyo, Japan, June 27-30, 1988

The Computer Society of the IEEE
1730 Massachusetts Avenue NW
Washington, DC 20036-1903

Washington • Los Alamitos • Brussels

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

IEEE

IEEE
COMPUTER
SOCIETY
PRESS

Computational Complexity of Controllability/Observability Problems for Combinational Circuits

Hideo Fujiwara

Department of Electronics and Communications
Meiji University
Kawasaki 214, Japan

Abstract

In this paper, the computational complexity of fault detection problems and various controllability and observability problems for combinational logic circuits are analyzed. It is shown that the fault detection problem is still NP-complete even for monotone circuits limited in fanout, i.e., the number of signal lines which fanouts from a signal line is limited to three. It is also shown that the observability problem for unate circuits is NP-complete, but that the controllability problem for unate circuits can be solved in time complexity O(m), where m is the number of lines in a circuit. Furthermore, two classes of circuits, called k-binate-bounded circuits and k-bounded circuits, are introduced. For k-binate-bounded circuits the controllability problem is solvable in polynomial time, and for k-bounded circuits the fault detection problem is solvable in polynomial time, when $k \leq \log p(m)$ for some polynomial p(m). The class of k-bounded circuits includes many practical circuits such as decoders, adders, one-dimensional cellular arrays, two-dimensional cellular arrays, etc.

I. Introduction

Testing has two main stages: the generation of tests for a given circuit and the application of these tests to the circuit. Hence, the complexity of testing can be classified into the complexity of test generation and the complexity of test application. The computational complexity of the algorithms used to generate a test is used to estimate the complexity of test generation. The size of a test set or the length of a test sequence is adopted as a measure of the complexity of test application [1][2]. In this paper, we are concerned with the complexity of test generation.

It is well known that major fault-detection problems are NP-complete in general [3], and that they are still NP-complete even for monotone circuits without negated gates such as NOT, NOR, and NAND [4]. It is apparent that the fault-detection problem for reconvergent-free circuits can be solved in O(m), where m is the number of signal lines. On the other hand, for the circuits with reconvergent fanouts, backtracking may occur during test generation. This backtracking due to reconvergency becomes a cause of NP-completeness. To clarify the relation between reconvergency and NP-completeness, we consider in this paper the fault-detection problem for circuits with a stringent condition on fanout of reconvergent paths. The fault detection problem is proved to be still NP-complete even for monotone circuits limited in fanout, i.e., the number of signal lines which fanouts from a signal line is limited to three. This means that even if we limit the number of reconvergent paths from a fanout point to three, the fault-detection problem is NP-complete. However, if we limit the total number of fanout points to a constant, then the fault-detection problem can be solved in linear time [4]. Therefore, we see that the main cause of NP-completeness is not the number of reconvergent paths from a fanout point but the number of fanout points which reconverge.

Fault detection problem for combinational circuits can be divided into two subproblems; controllability and observability problems. The controllability problem is to decide whether there exists an input pattern which produces a specified logical value on a given signal line in the circuit. The observability problem is to decide whether there exists an input pattern which propagates the logical value on a specified signal line to a primary output of the circuit. In this paper, we show that the observability problem for unate circuits is NP-complete, but that the controllability problem for unate circuits can be solved in time complexity O(m), where m is the number of lines in a circuit. Furthermore, we introduce a class of circuits called k-binate-bounded circuits, for which the controllability problem is solvable in polynomial time when k $\leq \log p(m)$, where p(m) is a polynomial in m. After analyzing the complexity of various problems, we present a class of logic circuits for which these fault detection problems are solvable in polynomial time. One-dimensional arrays like ripple-carry adders, two- dimensional arrays, decoder circuits, etc., belong to this class.

II. Various Satisfiability Problems

In this section, we clarify the computational complexity of several satisfiability problems for various classes of Boolean expressions. The analysis of satisfiability problems is important to know the complexity of fault-detection, controllability and observability problems since they are closely related with each other. We give some notations and definitions necessary for our discussion of satisfiability problems. For definitions of NP-completeness see [5].

A *literal* is either x or x' for some variable x, where x' denotes a complement of 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 satisfiability problem is specified as follows:

Satisfiability (SAT, for short): Is a Boolean expression satisfiable?

Theorem 1 (Cook's Theorem [6]): 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. For example, $(x_1+x_2)(x'_2+x'_3)$ is clause-monotone, but $(x_1+x_2)(x_2+x'_3)$ is not. The satisfiability for clause-monotone expressions (CM-SAT, for short) is proved to be NP-complete.

Theorem 2 [4]: CM-SAT is NP-complete.

An expression is said to be *monotone* if it contains only un-negated variables. An expression is said to be *unate* if each variable is either only negated or ony un-negated. A negated (un-negated) variable in a unate expression is called to be *negative (positive)* unate.

Theorem 3 [4]: SAT for unate expressions is solvable in time complexity O(e), where e is the length of an expression.

An expression is said to be in *k-conjunctive normal form* (k-CNF) if it is a product of sums of at most k literals. The k-*satisfiability problem* (k-SAT) is to determine whether an expression in k-CNF is satisfiable. For k=1 or 2 there exist polynomial algorithms to test k-SAT. However, 3-SAT is known to be NP-complete.

Theorem 4 [6]: 2-SAT is solvable in polynomial time, but 3-SAT is NP-complete.

This k-SAT problem is related to the fault-detection problem for circuits limited in fanin, i.e., the number of inputs which fanin to a gate is limited to value k. Similarly, we can define another k-SAT problem that is related to the fault-detection problem for circuits limited in fanout, i.e., the number of signal lines which fanout from a signal line is limited to value k. An expression is said to be *k-fanout-conjunctive normal form* (kF-CNF) if it is a product of sums such that each variable appears at most k times. For example, $(x_1+x_2)(x'_1+x_3)(x_1+x'_2)$ is 2-CNF but 3F-CNF, since variable x_1 (x_1 and x'_1) appears three times. The *k-fanout-satisfiability problem* (kF-SAT, for short) is to determine whether an expression in kF-CNF is satisfiable. Before showing that this SAT problem for kF-CNF is NP-complete, we present two lemmas.

Lemma 1: $x_1=x_2=...=x_m=y'_1=y'_2=...=y'_m$ if and only if $(x_1+y_1)(x_2+y_2)...(x_m+y_m)(x'_1+y'_2)(x'_2+y'_3)...(x'_m+y'_1)=1$.

Lemma 2: Given a CNF of a Boolean expression F where literals x and x' appear p and q times, respectively. Suppose we introduce new 2m variables, $x_1, x_2, ..., x_m, y_1, y_2, ..., y_m$, where m=max{p, q}, and replace p variables of x by $x_1, x_2, ..., x_p$ and q variables of x' by $y_1, y_2, ..., y_q$. Let the replaced expression be F^*. Then, F is satisfiable if and only if $F^\#$ is satisfiable, where $F^\# = F^*((x_1+y_1)(x_2+y_2)...(x_m+y_m)(x'_1+y'_2)(x'_2+y'_3)...(x'_m+y'_1))$.

Lemma 2 can be proved by using Lemma 1. From the above lemmas, we can prove that SAT for Boolean expressions that are kF-CNF ($k \geq 3$) and clause-monotone (CM-kF-SAT, for short) is NP-complete as follows.

Theorem 5: CM-3F-SAT is NP-complete.

Proof: It is easy to see that CM-3F-SAT and 3F-SAT are both in class *NP*. We transform SAT to CM-3F-SAT. Given a CNF of a Boolean expression F. Let $F^\#$ be the Boolean expression derived from F by applying the operation of Lemma 2. It is obvious that $F^\#$ is clause-monotone and each variable appears at most three times. From Lemma 2, F is satisfiable if and only if $F^\#$ is satisfiable. The transform operation of Lemma 2 can be performed in polynomial time of the size of F. Therefore, SAT is polynomially transformable to CM-3F-SAT. *Q.E.D.*

From this theorem, we have the following corollary.

***Corollary 1*:** 3F-SAT is NP-complete.

Theorem 6: 2F-SAT is solvable in time complexity $O(n^2)$ where n is the number of input variables.

Proof: Let F be a 2F-CNF. For each variable x of F, there are two cases; (1) only literal x appears, and (2) both literal x and x' appear. For each case we delete x and x' from F by applying the following operation.

(1) When only literal x appears, F can be expressed as

$$F = (x+p)(x+q)r \quad \text{or} \quad F = (x+p)r$$

where p and q are sums without x and r is a product of sums without x. Then, by deleting (x+p) and (x+q), we have $F^* = r$. Considering the assignment of x=1, we can see that F is satisfiable if and only if F^* is satisfiable.

(2) When both literals x and x' appear, F can be expressed as

$$F = (x+p)(x'+q)r$$

where p and q are sums without x and r is a product of sums without x. Then, by deleting x and x', we have $F^* = (p+q)r$. Obviously, F is satisfiable if and only if F^* is satisfiable.

By applying the above operation for each variable, we can determine whether F is satisfiable or not. The time complexity of this procedure is O(mn) where m is the size of the Boolean expression and n is the number of input variables. Since m is less than 2n, we have $O(n^2)$. *Q.E.D.*

Corollary 2: CM-2F-SAT is solvable in time complexity $O(n^2)$ where n is the number of input variables.

An expression is said to be *binate* with respect to a variable x if both x and x' are contained in it, and the variable x is said to be *binate*. An expression is said to be *k-binate* if it contains k binate variables.

Theorem 7: SAT for k-binate expressions is solvable in time complexity $O(2^k m)$ where m is the size of the expression. Therefore, if $k \leq \log_2 p(m)$, where p(m) is a polynomial in m, the SAT is solvable in polynomial time such as O(p(m).m).

Proof: Let $F(x_1, x_2, ..., x_k, x_{k+1}, ..., x_n)$ be a k-binate expression. Without loss of generality, we assume that $x_1, x_2, ..., x_k$ are binate and $x_{k+1}, ..., x_n$ are unate. For unate variables, let $(a_{k+1}, ..., a_n)$ be an assignment such that $a_i=0$ if x_i is negative unate and $a_i=1$ if x_i is positive unate. Then $F(x_1, x_2, ..., x_k, x_{k+1}, ..., x_n)$ is satisfiable if and only if $F(x_1, x_2, ..., x_k, a_{k+1}, ..., a_n)$ is satisfiable. To check if $F(x_1, x_2, ..., x_k, a_{k+1}, ..., a_n)$ is satisfiable or not, consider all the combinations of values 0 and 1 on all k binate variables. This computation can be performed in $O(2^k m)$ time.

Q.E.D.

III. Fault Detection Problem

Fault detection problem can be defined as follows:

Fault Detection (FD, for short): Is there any input-output pattern which can detect a single stuck-at fault f in a combinational circuit C?

Theorem 8 [3]: FD is NP-complete.

A combinational circuit is said to be *monotone* if it consists of only unnegated gates such as AND or OR. A combinational circuit is said to be *unate* if the number of negated gates (NOT, NAND, or NOR) in any path connecting two points in the circuit has the same parity (odd or even). FD is known to be NP-complete even for monotone or unate circuits [4].

Theorem 9 [4]: FD for monotone or unate circuits is NP-complete.

It is apparent that the fault-detection problem for reconvergent-free circuits can be solved in O(m), where m is the number of signal lines. On the other hand, for the circuits with reconvergent fanouts, backtracking may occur during test generation. This backtracking due to reconvergency becomes a cause of NP-completeness. To clarify the relation between reconvergency and NP-completeness, we consider here the fault-detection problem for monotone circuits limited in fanout. A combinational circuit is called to be *k-fanout-limited* if the number of signal lines which fanouts from a signal line is at most k. Consider the fault-detection problem for monotone and k-fanout-limited circuits (M-kF-FD, for short).

Theorem 10: M-3F-FD is NP-complete.

Proof: Obviously, M-3F-FD is in class *NP*. Hence, it is sufficient to show that some NP-complete problem is polynomially transformable to M-3F-FD. We transform here CM-3F-SAT to M-3F-FD since CM-3F-SAT is shown to be NP-complete by Theorem 5.

Given any clause-monotone CNF F in which each variable appears at most three times. Without loss of generality, we assume that $C_1, C_2, ..., C_p$ are the clauses with unnegated variables and $C_{p+1}, C_{p+2}, ..., C_q$ are the clauses with negated variables. For this expression F, we construct a 3-level monotone circuit as shown in Figure 1.

(1) Construct OR gates $O_1, O_2, ..., O_p$ corresponding to the clauses $C_1, C_2, ..., C_p$ so that each OR gate O_i has the input variables of C_i. For example, suppose a clause C_i=x+y+z, then the input of O_i is x+y+z.

(2) Construct AND gates $A_1, A_2, ..., A_{q-p}$ corresponding to the clauses $C_{p+1}, C_{p+2}, ..., C_q$ so that each AND gate A_j has the input variables of C_j. For example, suppose a clause C_j=x'+y', then the input of A_j is xy.

Since each input variable appears at most three times in F, in this circuit of Figure 1, the number of signal lines which fanouts from a primary input is limited to three. Hence the circuit of Figure 1 is monotone and 3-fanout-limited. A stuck-at-0 fault at input line x_0 is detectable if and only if there exists a test such that all the outputs of OR gates $O_1, O_2, ..., O_p$ are 1 and all the outputs of AND gates $A_1, A_2, ..., A_{q-p}$ are 0. Hence the fault x_0 stuck-at-0 is detectable if and only if the given expression F is satisfiable.

The above construction of the circuit can be carried out in an amount of time linear in the number of inputs. Therefore, CM-3F-SAT is polynomially transformable to M-3F-FD. *Q.E.D.*

From Theorem 10, we can see that the fault detection problem is still NP-complete even for monotone circuits with a restriction such that each signal line fanouts to at most three signal lines. This means that even if we limit the maximum number of reconvergent paths from a fanout point to three, the fault-detection problem is NP-complete. However, it is proved in [4] that if we limit the total number of fanout points to a constant, then the fault-detection problem can be solved in linear time. Therefore, we can see that the main cause of NP-completeness is not the number of reconvergent paths from a fanout point but the total number of fanout points which reconverge.

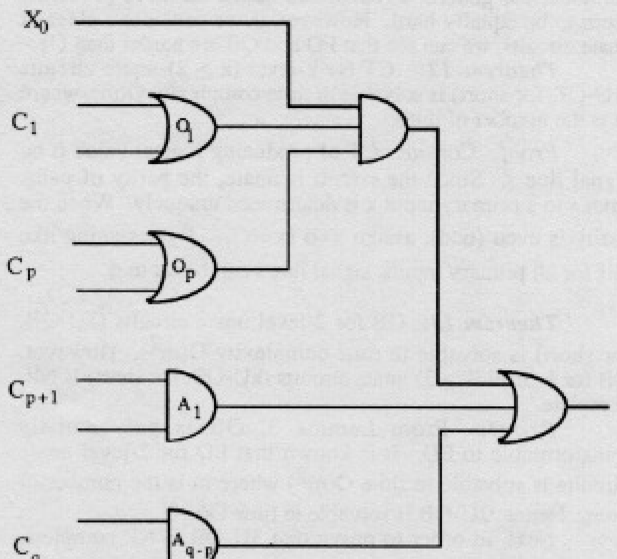

Figure 1. 3-level unate circuit

IV. Controllability/Observability Problems

The process of test generation consists of the tasks of controlling and observing internal logic values. Representatives of controlling and observing tasks in test generation are the consistency and D-drive operations of the D-algorithm, respectively [1]. Hence, fault detection problem for combinational circuits can be divided into two subproblems; controllability and observability problems. The controllability problem is to decide whether there exists an input pattern which produces a specified logical value on a given signal line in the circuit. The observability problem is to decide whether there exists an input pattern which propagates the logical value on a specified signal line to a primary output of the circuit. In this section, we analyze the complexity of those controllability and observability problems.

Controllability Problem (CT, for short): Let C, s, and σ be a circuit, a signal line in C, and a logical value, respectively. Is there any input pattern which produces value σ on line s in C?

Observability Problem (OB, for short): Is there any input pattern which propagates the logical value σ on line s to a primary output of C?

Lemma 3:

(1) SAT is polynomially transformable to CT,
(2) CT is polynomially transformable to OB, and
(3) OB is polynomially transformable to FD.

Theorem 11: Both CT and OB for k-level ($k \geq 2$) combinational circuits are NP-complete.

Proof: Obvious from Lemma 3 and Theorem 1. *Q.E.D.*

From this theorem, FD, CT, and OB are all NP-complete for general circuits, and hence all those problems seem to be equally hard. However, if we consider a class of unate circuits, we can see that FD and OB are harder than CT.

Theorem 12: CT for k-level ($k \geq 2$) unate circuits (kU-CT, for short) is solvable in time complexity O(m), where m is the number of lines.

Proof: Consider CT of producing logical value σ on signal line s. Since the circuit is unate, the parity of paths from s to a primary input x is determined uniquely. When the parity is even (odd), assign $x=\sigma$ ($x=\sigma'$). By assigning like this for all primary inputs, signal line s can be set to σ.

Q.E.D.

***Theorem 13*:** OB for 2-level unate circuits (2U-OB, for short) is solvable in time complexity $O(m^2)$. However, OB for k-level ($k \geq 3$) unate circuits (kU-OB, for short) is NP-complete.

Proof: From Lemma 3, OB is polynomially transformable to FD. It is known that FD for 2-level unate circuits is solvable in time $O(m^2)$ where m is the number of lines. Hence, 2U-OB is solvable in time $O(m^2)$.

Next, in order to prove that 3U-OB is NP-complete, we transform CM-SAT to 3U-OB. Given any clause-monotone CNF F in which each variable appears at most three times. Without loss of generality, we assume that $C_1, C_2, \ldots, C_p$ are the clauses with unnegated variables and $C_{p+1}, C_{p+2}, \ldots, C_q$ are the clauses with negated variables. For this expression F, we construct a 3-level monotone circuit of Figure 1 in the same way as the proof of Theorem 10.

The logical value of input line x_0 is observable at the primary output if and only if there exists an input pattern such that all the outputs of OR gates $O_1, O_2, \ldots, O_p$ are 1 and all the outputs of AND gates $A_1, A_2, \ldots, A_{q-p}$ are 0. Hence The logical value of input line x_0 is observable at the primary output if and only if the given expression F is satisfiable.

The above construction of the circuit can be carried out in an amount of time linear in the number of inputs. Therefore, CM-3F-SAT is polynomially transformable to 3U-FD. *Q.E.D.*

From Theorems 12 and 13, we see that OB is a harder problem than CT. On the other hand, improvement of observability can be achieved more easily than that of controllability. In other word, generally speaking, extra hardware for improving controllability is more expensive than that of observability. Hence, in a view point of design for testability, design methodologies for improving controllability might important than that of observability.

A circuit is said to be *k-binate-bounded* if it can be changed into a unate circuit by cutting at most k signal lines.

Theorem 14: CT for k-binate-bounded circuits is solvable in time $O(2^k m)$, where m is the number of lines in the circuit. Therefore, if $k \leq \log_2 p(m)$, then this controllability problem can be solved in time O(p(m)m).

Proof: Let C be a k-binate-bounded circuit. By cutting signal lines $s_1, s_2, \ldots, s_k$, C is changed into a unate circuit. Assign a value 0 or 1 to every cutted lines. The number of possible assignments for this is 2^k. For each assignment, perform forward and backward implications, i.e., determine all the line values that are implied uniquely by other line values. After the implications, the remaining circuit becomes a unate circuit. Hence we can easily solve CT for the remaining unate circuit in time O(m) from Theorem 12. The above computation requires at most $O(2^k m)$ time. *Q.E.D.*

V. Polynomial Time Class

In this section we introduce a class of circuits for which the fault detection problem can be solved in polynomial time of the number of lines in the circuits. One-dimensional cellular arrays like ripple-carry adders, two-dimensional cellular arrays, decoder circuits, etc., belong to this class.

Consider a *partition* $\Pi = \{C_1, C_2, \ldots, C_t\}$ of a circuit C, where $C_1, C_2, \ldots, C_t$ are sub-circuits of C, called *blocks*, and satisfy $C_i \cap C_j = \phi$ and $C = C_1 \cup C_2 \cup \ldots \cup C_t$. Consider a graph G_Π with respect to Π such that each vertex represents a block and each edge corresponds to each connection line between blocks.

A combinational circuit C is said to be *k-bounded* if there exists a partition $\Pi = \{C_1, C_2, \ldots, C_t\}$ of C such that:

(1) the number of inputs of each block C_i ($1 \leq i \leq t$) is at most k, and

(2) graph G_Π has no cycle.

A combinational circuit C is said to be *(k_1, k_2)-bounded* if it can be changed into a k_1-bounded circuit by cutting at most k_2 lines in C.

Example 1: Figure 2 shows a p stage adder. Suppose a partition such that each block corresponds to a full adder, then we see that the adder is 3-bounded.

Example 2: Consider a two-dimensional cellular array shown in Figure 3. Let N_h and N_v be the numbers of horizontal and vertical inputs of each cell, respectively. Suppose a partition such that each block corresponds to a set of k cells of each column, then we see that the array is $(N_h k+N_v)$-bounded.

Example 3: Consider a two-dimensional cellular array shown in Figure 4. This array is augmented from the array of Figure 3 by adding skew lines. Let N_s be the number of skew lines of each cell. Suppose a partition such that each block corresponds to a set of k cells of each column, then we see that the array is $(N_h k+N_s k-N_s+N_v)$-bounded.

For k-bounded circuits we have the following theorem.

Theorem 15: Let C be a k-bounded circuit. Then there is an algorithm of time complexity $O(16^k m)$ to find a test for a single stuck-at fault in C, where m is the number of lines in C.

Proof: Let C be a k-bounded circuit. Let $\Pi = \{C_1, C_2, \ldots, C_t\}$ be a partition of C. Without loss of generality, we can assume that each primary output constitutes one block individually. They are called *primary output blocks*.

Our test generation procedure consists of two main parts. The first is to construct a graph G_T from C defined later. The second is to find a sub-tree corresponding to a test of a given fault in the graph G_T. Five-valued logic (1, 0, X, D, D') similar to the D-algorithm is used in our procedure.

(1) Construction of graph G_T.

Step 1: For each block C_i ($1 \leq i \leq t$), construct vertices of G_T as follows: Consider all the combinations of values 0, 1, D, D' on all inputs of C_i, and for each input assignment compute the values of internal lines of C_i. If any inconsistency occurs in the computation, then reject the assignment. Note that the value D or D' on any predecessor line of a faulty line L is an inconsistency. Let us represent each of these assignments by a vertex in G_T.

Step 2: For each pair of adjacent blocks C_i and C_j of C, construct edges of G_T as follows: Let $s_1, s_2, ..., s_q$ be the lines connected between C_i and C_j. For a vertex u of C_i and a vertex v of C_j, if the values of $s_1, s_2, ..., s_q$ on u and v are the same, then place an edge between u and v.

Step 3: If there is a vertex v in G_T satisfying the following condition, then delete the vertex v and the edges connected to v.

Condition: Let v be a vertex of C_i. There is no edge between C_i and its adjacent block C_j.

(2) Construction of a test from graph G_T.

Let C_f be a block with a fault. A test is an input assignment such that every assignment between blocks is consistent and there is at least one sensitized path from C_f to a primary output. Hence we can easily see that a test corresponds to a subtree T in G_T satisfying the following:

(a) T contains one vertex for each block C_i $(1 \leq i \leq t)$.

(b) T contains faulty signal D or D' for at least one primary output block.

The computation of steps 1 and 2 of part 1 can be performed in time $O(4^k m)$ and $O(16^k M)$, respectively, where M is the total number of signal lines between blocks. The computation of part 2 can be performed in time O(E), where E is the number of edges of G_T. Hence, the total computation of the above procedure can be carried out in time $O(4^k m) + O(16^k M) + O(16^k M) \leq O(16^k m)$. *Q.E.D.*

Corollary 3: Let C be a k-bounded circuit such that $k \leq \log_2 p(m)$ 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 $O(p(m)^4 m)$.

Corollary 4: Let C be a (k_1, k_2)-bounded circuit. Then there is an algorithm of time complexity $O(16^{k_1} 4^{k_2} m)$ to find a test for a single stuck-at fault in C, where m is the number of lines in C.

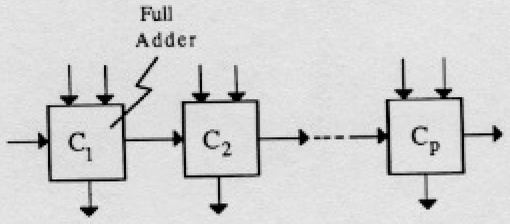

Figure 2. Adder

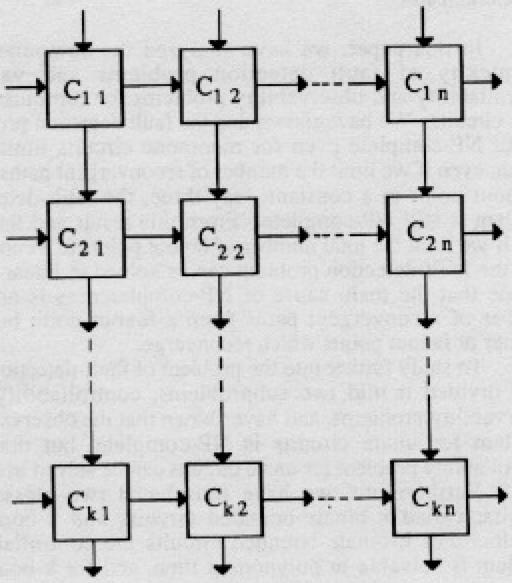

Figure 3. $(N_h k + N_v)$-bounded circuit

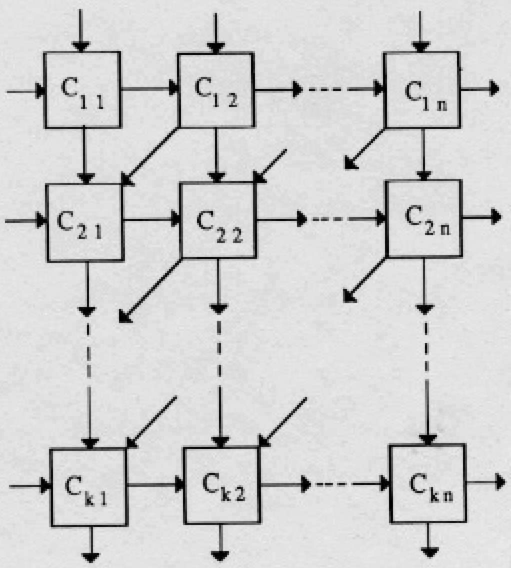

Figure 4. $(N_h k + N_s k - N_s + N_v)$-bounded circuit

VI. Conclusion

In this paper, we have analyzed the computational complexity of fault detection problems and various controllability and observability problems for combinational logic circuits. We have shown that the fault detection problem is still NP-complete even for monotone circuits limited in fanout; even if we limit the number of reconvergent paths from a fanout point to a constant , say three, the fault-detection problem is still NP-complete. From this result and the fact that, if we limit the total number of fanout points to a constant, then the fault-detection problem can be solved in linear time, we see that the main cause of NP-completeness is not the number of reconvergent paths from a fanout point but the number of fanout points which reconverge.

To study further into the problem of fault-detection, we have divided it into two subproblems; controllability and observability problems, and have shown that the observability problem for unate circuits is NP-complete, but that the controllability problem for unate circuits can be solved in linear time. Furthermore, we have introduced two classes of circuits; called k-binate-bounded circuits and k-bounded circuits. For k-binate-bounded circuits the controllability problem is solvable in polynomial time, and for k-bounded circuits the fault detection problem is solvable in polynomial time, when $k \leq \log p(m)$ for some polynomial $p(m)$. The class of k-bounded circuits includes many practical circuits such as decoders, adders, one-dimensional cellular arrays, two-dimensional cellular arrays, etc.

References

[1] H. Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, 1985.

[2] B.Krishnamurthy and S.B.Akers, "On the complexity of estimating the size of a test set," *IEEE Trans. Comput.*, vol. C-33, pp.750-753, August 1984.

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

[4] H.Fujiwara and S.Toida, "The complexity of fault detection problems for combinational logic circuits," *IEEE Trans. Comput.*, vol.C-31, pp.555-560, June 1982.

[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] S.A.Cook, "The complexity of theorem proving procedures," in *Proc. 3rd ACM Symp. Theory of Comput.*, pp.151-158, 1971.