December 1987

UILU-ENG-87-2280
DAC-9

COORDINATED SCIENCE LABORATORY
College of Engineering

THE COMPLEXITY OF TEST GENERATION AT THE TRANSISTOR LEVEL

Farid Najm
Ibrahim Hajj

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-87-2280 (DAC-9)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (*If applicable*) N/A	7a. NAME OF MONITORING ORGANIZATION Semiconductor Research Corporation
6c. ADDRESS (*City, State, and ZIP Code*) 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS (*City, State, and ZIP Code*) Research Triangle Park, NC 27709
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Semiconductor Research Corporation 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER SRC RSCH 86-12-109
8c. ADDRESS (*City, State, and ZIP Code*) Research Triangle Park, NC 27709	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
THE COMPLEXITY OF TEST GENERATION AT THE TRANSISTOR LEVEL

12. PERSONAL AUTHOR(S)
Najm, Farid and Hajj, Ibrahim

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Technical	FROM _____ TO _____	December 1987	27

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	complexity analysis, MOS circuit testing, robust test vectors, sensitization, switch-level fault models, transistor-level test generation

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

It is well known that the gate-level test generation problem for classical stuck-at fault models is *NP - hard*. In this report, we prove that the test generation problem for a *single* general MOS channel-connected subcircuit (logic gate) is also *NP - hard*. More importantly, we prove that this remains true if the subcircuit is constrained to be a series/parallel NMOS gate or a fully complementary series/parallel CMOS gate. This is done by specializing and decomposing the problem into its subproblems whose complexities are derived. Finding a single test vector t_2 for a single MOS channel-connected subcircuit with either a transistor stuck-at-on (or off) or node-stuck-at-1 (or 0) fault is *NP - hard*, and remains so for a series/parallel NMOS gate or a fully complementary series/parallel CMOS gate. For test vectors that require initialization, we prove that finding the initializing vector t_1 is also *NP - hard* and remains so for a fully complementary series/parallel CMOS gate with a stuck-at-off transistor fault. Finally, the problem of ensuring the robustness of

(over)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (*Include Area Code*) / 22c. OFFICE SYMBOL

a given test pair is shown to be *NP - hard* for unconstrained designs and of polynomial complexity for a fully complementary CMOS gate; a linear time algorithm is given. If the subcircuit is restricted to be an NMOS gate or a fully complementary CMOS gate, with a classical stuck-at fault at its output node, then the test generation problem becomes a polynomial complexity. The implications of these complexity results on practical switch-level test generation tools are discussed.

THE COMPLEXITY OF TEST GENERATION AT THE TRANSISTOR LEVEL†

Farid Najm and Ibrahim Hajj

Coordinated Science Laboratory and the
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801

ABSTRACT

It is well known that the gate-level test generation problem for classical stuck-at fault models is $\mathcal{NP}-hard$. In this paper we prove that the test generation problem for a *single* general MOS channel-connected subcircuit (logic gate) is also $\mathcal{NP}-hard$. More importantly, we prove that this remains true if the subcircuit is constrained to be a series/parallel NMOS gate or a fully complementary series/parallel CMOS gate. This is done by specializing and decomposing the problem into its subproblems whose complexities are derived : Finding a single test vector t_2 for a single MOS channel-connected subcircuit with either a transistor stuck-at-on (or off) or node-stuck-at-1 (or 0) fault is $\mathcal{NP}-hard$, and remains so for a series/parallel NMOS gate or a fully complementary series/parallel CMOS gate. For test vectors that require initialization, we prove that finding the initializing vector t_1 is also $\mathcal{NP}-hard$ and remains so for a fully complementary series/parallel CMOS gate with a stuck-at-off transistor fault. Finally the problem of ensuring the robustness of a given test pair is shown to be $\mathcal{NP}-hard$ for unconstrained designs and of polynomial complexity for a fully complementary CMOS gate; a linear time algorithm is given. If the subcircuit is restricted to be an NMOS gate or a fully complementary CMOS gate, with a classical stuck-at fault at its output node, then the test generation problem becomes of polynomial complexity. The implications of these complexity results on practical switch-level test generation tools are discussed.

† This work was supported by the Semiconductor Research Corporation under Contract SRC RSCH 86-12-109.

I. INTRODUCTION

A VLSI circuit consists of an interconnection of primitive modules which we refer to as subcircuits or logic gates. We consider here MOS circuits and define a primitive module to be a channel-connected subcircuit. The problem of automatically generating tests to detect logic faults in VLSI circuits can, in general, be divided into two levels: local and global. At the local level a fault model is constructed which reflects the effects of physical failures in a subcircuit or gate on its logical behavior. A test set is then generated to detect the faults at the local level. The global level then involves the propagation of fault effects forward and backward through the subcircuit interconnections to the accessible external nodes of the circuit.

When only a gate-level description of the circuit is available, classical stuck-at fault models are often used, and the internal structure and technology of the gate are not taken into consideration. In this case the problem of generating test vectors at the local level becomes trivial in the sense that tests can be readily derived; while the problem of global propagation of the tests through the interconnections to the external nodes is $NP-hard$ [1].

Unfortunately, it is now well-established that many transistor-level failures cannot be accurately modeled at the gate level; consequently, test generation techniques derived for a gate-level description fail to detect these failures [6]. As a result, transistor-level test generation has recently attracted the attention of many researchers [7]–[11]. However, many of these techniques are restricted to the detection of faults that require one test vector. In [2] and [3] a method is given which, for a given fault, automatically detects whether one or two test vectors are needed; and automatically generates test pairs that are robust [5].

In this paper we concentrate on the problem of local test generation for a single general MOS channel-connected subcircuit and prove that the problem is $NP - hard$. More importantly, we prove that the problem remains $NP - hard$ even if the subcircuit is restricted to be a single series/parallel NMOS gate or a fully complementary series/parallel CMOS gate. This problem becomes more complex when faults exist that require two test patterns for detection. Such faults are a common occurrence and have been shown to exist in NMOS [2]–[3] as well as CMOS [4]–[6] circuits. The significance of these results is that it is very unlikely that an exact polynomial-time algorithm for local test generation in general MOS circuits will be found.

However, if the problem is restricted to allow only NMOS or fully complementary CMOS gates with only node stuck-at faults allowed at the gate's output node, similar to classical gate-level stuck-at faults, then the problem is trivially solvable in linear-time: a single test vector is required and can be obtained by a depth-first tracing of the transistor graph corresponding to either the N-part or P-part of the gate. Any more generality than this runs into $NP - hard$ problems, as will be shown below.

The careful reader may notice that if all the problems considered below were posed as *decision-problems*, then they would be $NP - complete$, a slightly stronger result. The proofs for $NP - completeness$ would, however, be based on the existence of polynomial-time *circuit simulation* algorithms. While examples of such algorithms abound, we have opted to avoid that approach because it requires one to precisely describe what circuit model is being used and what type of simulation is to be performed, and the results would accordingly be limited by these details. The $NP - hardness$ results to be presented, however, do

not depend on such details and, we feel, highlight the important features of the problem complexity.

The remainder of this paper is divided into six sections. The problem is first formalized and described in section II. Section III then considers the complexity of finding a single test vector, section IV deals with finding an initializing vector, and section V examines the complexity of ensuring robustness. Finally section VI gives some concluding remarks, discusses the implications of these results, and gives suggestions for future work.

II. PROBLEM DESCRIPTION

We rely heavily on the terminology used in complexity theory, such as saying that a problem P is *transformable* in polynomial time to a problem P' (abbreviated $P \propto P'$), a problem is in the class $\mathcal{P}$, a problem is in the class $\mathcal{NP}$, a problem is $\mathcal{NP}-complete$, a problem is $\mathcal{NP}-hard$, etc... . The discussion in [12] is sufficient background to understand the rest of this paper.

A boolean function or expression is said to be *satisfiable* if there exists an assignment of 0's and 1's to its variables that gives it the value 1. We recall the *satisfiability problem* from mathematical logic [12], to be abbreviated SAT, which is defined as follows. A boolean variable or its complement is called a *literal.* Given a boolean expression in *conjunctive normal form* (CNF), ie, it is the product (logical *and*) of a set of sub-expressions called *clauses* where every clause is the sum (logical *or*) of a number of literals. The problem is to decide whether or not the expression is satisfiable. It is well known [13] that SAT is $\mathcal{NP}-complete$. *3-satisfiability* (3SAT) is satisfiability in which the boolean expression in

CNF is allowed to have at most 3 literals per clause; such a representation will be called 3CNF. It is also known [12] that 3SAT is $NP - complete$.

A boolean expression is said to be in *disjunctive normal form* (DNF) if it is a sum (logical *or*) of a set of clauses where each clause is the product (logical *and*) of a number of literals.

Given a boolean function f of n variables $x_1, \ldots, x_n$, the *boolean difference* or boolean *partial derivative* [14] of f with respect to a variable x_i is defined as

$$\frac{\partial f}{\partial x_i} = f(x_1, \ldots, x_i, \ldots, x_n) \oplus f(x_1, \ldots, \overline{x_i}, \ldots, x_n)$$

where $\oplus$ is the *exclusive-or* operation. A boolean function will be said to be *sensitized* to one of its variables x_i if changing x_i while keeping all other x_j's fixed causes f to change. A boolean function is sensitized to one of its variables x_i if and only if its partial derivative with respect to x_i is 1. This notion of sensitization is central to the test generation issue, and is similar to the notion of *observability*. Notice that if $f = gx + h$, where g and h are not functions of x, then $\partial f / \partial x = g\overline{h}$, and f can be sensitized to x if and only if $g\overline{h}$ is satisfiable.

Define the following sensitization problem :

Definition 1. P_0 : *Given a boolean expression* $f(x_1, \ldots, x_n)$ *written in the form* $f = gx_i + h$ *where* $i \in \{1, \ldots, n\}$, *and :*

(i) g *is a boolean expression in CNF that does not depend on* x_i.

(ii) h *is a boolean expression in DNF that does not depend on* x_i.

(iii) g, h, and therefore f, contain no complemented literals.

Decide if f can be sensitized to x_i.

Lemma 1. *P_0 is $\mathcal{NP}$ – complete.*

proof: It is easy to see that $P_0 \in \mathcal{NP}$. We must still show that it is $\mathcal{NP}$ – *hard*, we do so by showing that $3SAT \propto P_0$. Let $E(x_1, \ldots, x_{n-1})$ be a boolean expression in 3CNF. Scan $E(x_1, \ldots, x_{n-1})$ and replace every $\overline{x_i}$ by y_i, get a new expression $E'(x_1, \ldots, x_{n-1}, y_1, \ldots, y_{n-1})$ in linear time. Let $g = \prod_{i=1}^{n-1}(x_i + y_i)$, $h = \sum_{i=1}^{n-1} x_i y_i$, and $f(x_1, \ldots, x_{n-1}, x_n, y_1, \ldots, y_{n-1}) = E'(x_1, \ldots, x_{n-1}, y_1, \ldots, y_{n-1})gx_n + h$. Then f comprises an instance of P_0 for which

$$\partial f / \partial x_n = E'(x_1, \ldots, x_{n-1}, y_1, \ldots, y_{n-1})g\overline{h}$$

Therefore f can be sensitzed to x_n iff $\exists a_1, \ldots, a_{n-1}, b_1, \ldots, b_{n-1}$ such that

$$E'(a_1, \ldots, a_{n-1}, b_1, \ldots, b_{n-1}) = 1,\ g = 1,\ \text{and}\ h = 0$$

But $g = 1$ means that $a_i + b_i = 1, \forall i$. And $h = 0$ means that $a_i b_i = 0, \forall i$. Therefore $b_i = \overline{a_i}, \forall i$, and, equivalently, $E'(a_1, \ldots, a_{n-1}, b_1, \ldots, b_{n-1}) = E(a_1, \ldots, a_{n-1})$. Then $\partial f / \partial x_n$ is satisfiable if and only if $E(x_1, \ldots, x_{n-1})$ is too, and $3SAT \propto P_0$. ■

The fact that sensitizing a boolean function to one of its variables is $\mathcal{NP}$ – *hard* effectively explains why the D-propagation step of the D-algorithm [15] at the global logic level is computationally expensive : D-propagation is nothing but sensitization. It is interesting that sensitizing such a simple function as f above is just as hard.

We denote by *external nodes* all the primary inputs as well as the power supply and ground nodes of an MOS network. All other nodes are called *internal.* A *channel-connected*

subcircuit is defined as a *maximal* subcircuit in which every transistor shares at least a drain or source *internal* node with another transistor in the same subcircuit. In the simplest cases such a subcircuit would be a single NMOS or CMOS logic gate, in which case the gate *output* node is well defined as its unique node that connects to transistors of different types.

A fault is basically an undesired physical modification of the circuit layout that manifests itself at the circuit description level in different ways. Given a channel-connected subcircuit S with a fault f, then a test for f is an assignment or a sequence of assignments of logic values to the subcircuit inputs (transistor gate labels and/or primary inputs) that produces different values at a subcircuit output in the faulty and fault-free circuits. Such assignments are called *vectors*, so that a test is composed of a test vector possibly preceded by one or more initializing vectors.

To establish the complexity of the subcircuit test generation problem in general we will study the complexity of special cases of the problem below. Specifically we will look at faults in the subcircuit that manifest themselves as either transistor *stuck-at-on* (or *off*, also called stuck-at-open) or node *stuck-at-1* (or *0*) faults. Of course the problem in general is no simpler than its special cases.

III. COMPLEXITY OF GENERATING t_2

When a test consists of two vectors the second (test) vector will be called t_2, while the first (initializing) vector will be called t_1. For uniformity of presentation, a test vector will be called t_2 even if it doesn't require initialization. This section is concerned with the complexity of deriving t_2.

Define the following test generation problem :

Definition 2. P_1 : *Given an MOS channel-connected subcircuit S with either a transistor stuck-at-on or stuck-at-off fault f, find a test vector t_2 for f, if one exists.*

The single test vector derived in P_1 may, in general, require initialization. Finding an initializing vector is a totally different problem and will be discussed later on.

Theorem 1. *P_1 is $\mathcal{NP}$ – hard. This remains true even if S is restricted to be either a series/parallel NMOS gate or a fully complementary series/parallel CMOS gate.*

proof: We will prove that $P_0 \propto P_1$ by using an instance f of P_0 to build an instance of P_1. Let $f(x_1, \ldots, x_n) = gx_n + h$ be a boolean expression as defined in Definition 1 above. Construct, in linear time, a series/parallel realization of f using NMOS transistors. Consider this as the driver part of an NMOS gate and complete the construction by adding in a depletion transistor to V_{dd} as shown in Figure 1. Now consider the NMOS transistor with gate label x_n in this gate, and let there be a fault f_n at this transistor (either stuck-at-on or stuck-at-off). This constitutes an instance of P_1. Suppose we find a solution for this involving a test vector t_2. It is evident that the assignment in t_2 to $x_1, \ldots, x_{n-1}$ must sensitize the gate output (and, therefore, f) to x_n, otherwise the outputs in the faulty and fault-free subcircuits would be the same. Conversely, if f is sensitizable to x_n, then a test exists for f_n. So $P_0 \propto P_1$. This completes the proof for the NMOS case, the same can be done for CMOS by replacing the depletion load device by the dual of the NMOS realization using PMOS transistors as shown in Figure 2. ■

The complexity results obtained were for the special case of transistor stuck-at-on/off faults. It should be mentioned that a transistor stuck-at-on fault is equivalent to a *short*

circuit (also called bridging) fault between the drain and source nodes. Similarly, a transistor stuck-at-off fault is equivalent to an *open* circuit fault in its drain or source wire. So an algorithm that generates tests for arbitrary short faults (bridging faults) and open faults inside a subcircuit will almost surely have exponential worst case behavior, and this holds even for NMOS or CMOS gates. Such is the case with the algorithm and program presented by the authors in [3], and by Chen et al. in [21].

The classical node stuck-at-1 and stuck-at-0 faults are special cases of the general class of short circuit faults, they involve a short circuit between a node and V_{dd} or V_{ss}. We now prove that these faults are no easier to test for.

Define the following implication problem :

Definition 3. *P_2 : Given an MOS channel-connected subcircuit S, a value $v \in \{0,1\}$, and a specified internal node Y in S, find an input vector p of S, if it exists, that causes Y to take the logic value v in the steady state.*

Define the following test generation problem :

Definition 4. *P_3 : Given an MOS channel-connected subcircuit S with either a node stuck-at-1 or stuck-at-0 fault f at an internal node Y, find a test vector t_2 for f, if one exists.*

Theorem 2. *P_2 and P_3 are $NP-hard$. This remains true even if S is restricted to be either a series/parallel NMOS gate or a fully complementary series/parallel CMOS gate.*

proof: We will prove that $P_0 \propto P_2 \propto P_3$. Let f be as defined in Lemma 1, construct in linear time a series/parallel realization of g and h using NMOS transistors. Consider

gh to be the driver part of an NMOS gate and complete the construction by adding in a depletion transistor to V_{dd} as shown in Figure 3.

Suppose we would like to make Y take the value 1 in the steady state, this constitutes an instance of P_2. A solution can be found if and only if $g\bar{h}$ is satisfiable; which means that f can be sensitized to x_i, so $P_0 \propto P_2$.

Now suppose there is a fault : node Y stuck-at-0. This constitutes an instance of P_3 for which a solution t_2 can be found if and only if t_2 is also a solution of P_2. So $P_2 \propto P_3$.

The same can be done for CMOS by replacing the depletion load device of S by the dual of the NMOS realization using PMOS transistors as shown in Figure 4. ■

The test generation problem P_3 becomes of polynomial complexity if the subcircuit is either an NMOS gate or a fully complementary CMOS gate and the node at which a stuck-at fault exists is the output node. This is because a path from the output to V_{ss} or V_{dd} in the subcircuit graph can be found in linear time. In this case the test vector does not require initialization, and a valid local test for the fault is indeed derivable in linear time.

IV. COMPLEXITY OF GENERATING t_1

Once a test vector t_2 has been found, it may be necessary to derive an initializing vector t_1. The complexity of the problem of finding such a t_1 will be examined in this section.

Define the following initialization problem :

Definition 5. *P_4 : Given an MOS channel-connected subcircuit S with either a transistor stuck-at-on or stuck-at-off fault f, for which some test vector t_2 is known to require initialization, find a single initializing vector t_1 for t_2.*

Theorem 3. *P_4 is $\mathcal{NP}$ – hard. This remains true even if S is restricted to be a fully complementary series/parallel CMOS gate with a stuck-at-off fault.*

proof: The proof will be based on the need to avoid charge sharing between the output and other subcircuit nodes when the test vector t_2 is applied.

We will prove that $P_0 \propto P_4$. Reconsider the subcircuit built in the proof of Theorem 2 and shown in Figure 4. Assume that $C_Y = C_Z$ and all other internal node capacitances are negligible. Let there be a stuck-at-off fault at an NMOS transistor T_f in one of the parallel branches in h.

Since any t_2 must sensitize the output Z to the gate label of T_f then it must connect Y and Z by at least one path in the g block. Since $C_Y = C_Z$ then an initializing vector must initialize both Y and Z to 1 to avoid charge sharing when t_2 is applied, no other nodes need be initialized because they have negligible capacitances by construction. Therefore an initializing vector t_1 can be found if and only if $g\bar{h}$ is satisfiable, so $P_0 \propto P_4$. ■

The problem of finding an initializing vector for a fully complementary CMOS gate when only the output node needs to be initialized is of linear complexity. This, however, does not mean that a valid test can be found in linear time because initializing vectors are typically required for faults involving transistor stuck-at-off faults, and deriving a t_2 for these faults is $\mathcal{NP}$ – *hard* even for fully complementary CMOS gates as shown above.

The problem of initialization does not arise for NMOS gates. Furthermore, in case of a transistor stuck-at-on fault, P_4 does not arise for either NMOS or CMOS gates. It may, however, arise in general cases when an unconstrained design style is allowed.

Theorem 4. *P_4 is $\mathcal{NP}$ – hard for stuck-at-on faults if an unconstrained design style is allowed.*

proof: We will prove that $P_0 \propto P_4$. As done previously consider f as in Lemma 1 and build a subcircuit in polynomial time using NMOS transistor implementations of g and h as shown in Figure 5. The figure also shows an assumed stuck-at-on fault at one of the transistors with gate label x_f. This constitutes an instance of P_4. It is easy to see that t_2 must make $x_f = 0$ and $gh = 1$ and requires that the output node Z be initialized to 1. Therefore an initializing vector t_1 can be found if and only if $g\overline{h}$ is satisfiable, and $P_0 \propto P_4$.

■

The complexity results obtained for P_4 were for the special case of transistor stuck-at-on/off faults. As was mentioned above, a transistor stuck-at-on (off) fault is equivalent to a short circuit (open circuit) fault between the drain and source nodes (in the drain or source wires). So an algorithm that generates initializing vectors for tests for arbitrary short faults (bridging faults) and open faults inside a subcircuit will almost surely have exponential worst case behavior, and this holds even for CMOS gates.

V. COMPLEXITY OF ENSURING ROBUSTNESS

Suppose a test vector t_2 for a certain fault requires an initializing vector t_1. The object of t_1 is to initialize certain internal nodes in the subcircuit to certain values. The success of t_2 depends on whether or not these values are still there when it is applied,

we will refer to these nodes as *critical precharged nodes.* This in turn depends on the way the transition $t_1 \rightarrow t_2$ takes place. Signal skews at the inputs to the subcircuit can cause certain transistors to switch before others, causing these critical nodes' values to be changed before t_2 is applied. It is of interest, therefore, to devise test pairs (t_1, t_2) that cannot be invalidated no matter how the subcircuit inputs switch; these tests are called *robust* [5]. It is helpful to visualize t_1 and t_2 as *cubes* [20] in the boolean space, a robust test pair then becomes one which cannot be invalidated no matter which path in the boolean space is actually taken to go from t_1 to t_2.

To ensure robustness, a given test pair (t_1, t_2) can be *refined* to become robust by selecting some subcube of each of t_1 and t_2, t_1' and t_2', and requiring that some inputs that do not change in the transition $t_1' \rightarrow t_2'$ be free of glitches (ie, *static-hazard-free*, to be abreviated *shf*). This gives a robust test triplet (t_1', S_{hf}, t_2'), where S_{hf} is the set of entries that need to be shf. The choice of t_1', t_2', and S_{hf} is not unique.

Define the following robustness problem :

Definition 6. *P_5 : Given an MOS channel-connected subcircuit S along with a test pair (t_1, t_2) for a certain fault in S; refine the test pair, if possible, so that it becomes robust.*

Theorem 5. *P_5 is NP − hard if an unconstrained design style is allowed.*

proof: We will prove that $3SAT \propto P_5$. Let $E(x_1, \ldots, x_n)$ be a boolean expression in 3CNF. Construct in linear time a switching function realization of E using NMOS transistors for non-complemented literals and PMOS transistors for complemented ones. Use this to build the subcircuit shown in Figure 6. Let there be a transistor stuck-at-off fault at the NMOS transistor driven by x_1 shown in the figure. The only possible test pair is one that sets

$x_i = 0$ in t_1 and $x_i = 1$ in t_2 for all $i = 1, \ldots, n$, so that Z is initialized to 1 by t_1 and then discharged to 0 by t_2 through the faulty transistor. Furthermore, E has to be 0 when either t_1 or t_2 is applied. This constitues an instance of P_5. Since t_1 and t_2 are both subcubes of dimension 0 in this case, and they are different at all there entries, then $t_1' = t_1$, $t_2' = t_2$ and no refinement for robustness is possible : the pair is either robust or not. Notice that the set of possible intermediate states in the transition is the whole boolean space and that E should be 0 at each of these states to preserve the charge at the output node Z. Therefore a solution can be found (meaning that the pair is robust as is) if and only if E is not satisfiable. This means that $3SAT \propto P_5$. ■

We will now study the complexity of P_5 for the case of a single fully complementary CMOS gate (not necessarily series/parallel) with a transistor stuck-at-off fault. We will prove that the problem becomes of polynomial complexity by giving a linear time algorithm that refines tests to make them robust. Some preliminaries must be made before the actual proof is given.

Given a fully complementary CMOS gate S with a stuck-at-off fault at a transistor T_f whose gate label is x_f. Given also a test pair (t_1, t_2) for this fault. Let the gate output node be Z. A compact representation of t_1 or t_2 is a row-vector of n entries $x_i \in \{0, X, 1\}$, where X means that the value at the corresponding gate input is irrelevant to the test. So, for example, if t_1 requires $x_1 = 0$ and $x_2 = 1$, and doesn't care about x_3 then it is represented by [0 1 X]. A subcube t_i' of t_i has at least the same non-X entries as t_i and may have some more.

In as much as cubes represent special sets of points in the boolean space we will use the set operations $\subseteq$, $\cap$, and $\cup$ to represent the subset relation, and the intersection and

union operations on these sets, respectively. Therefore, if t_i' is a subcube of t_i, we will write $t_i' \subseteq t_i$. The $\cap$ and $\cup$ operations can be performed on the row vector representation of two cubes by doing a *bit-wise and* and a *bit-wise or* operation, respectively, on each entry of the cube using ternary algebra.

We make the reasonable assumption that the capacitance of Z is not negligible compared to other internal nodes of S, therefore it is one of the critical precharged nodes. We also assume, without loss of generality, that T_f is in the N-part of the gate. This means that t_1 joins Z as well as all other critical precharged nodes to V_{dd}, and t_2 attempts to join Z to V_{ss} along a path that goes through T_f.

We will now focus on the N-part of the gate and treat it as a *graph* G where every subcircuit node (transistor) translates to a graph node (edge) of G, however, no edge is inserted in G for the faulty transistor T_f. If t is an input vector to S at a particular time involving no X entries, define $G[t]$ as the subgraph of G induced by the edges turned "on" by t. We will call this subgraph of G the *conduction subgraph.* It is clear (since $T_f \notin G$) that if for some intermediate t in the transition the resulting $G[t]$ joins one of the critical precharged nodes to V_{ss} then the charges will be lost and the test invalidated.

Let t_1^0 (t_2^0) be obtained from t_1 (t_2) by replacing the X's in the row vector representation by 0's. $G[t_1^0]$ contains all the critical precharged nodes in the N-part of S, and therefore contains the output node Z in particular. In fact $G[t_1^0]$ joins the output node Z to every other critical precharged node of G. As for $G[t_2^0]$ it contains both Z and V_{ss}, but does not actually join them by a path because T_f was not included in G ($G[t_2^0]$ is not a connected graph). $G[t_1^0]$ and $G[t_2^0]$ are ,therefore, not disjoint, since they share at least the output node Z. Define $G[t_i] \cup G[t_j]$ to be the graph with node (edge) set equal to the union of the

two node (edge) sets of $G[t_i]$ and $G[t_j]$. We will also say $G[t_i] \subseteq G[t_j]$ if $G[t_i]$ is a *subgraph* of $G[t_j]$.

Let $t_{12} = t_1 \cup t_2$. For example, if

$$t_1 = [011010XXX]$$

and

$$t_2 = [11XX0010X]$$

then

$$t_{12} = [111X101XX]$$

Now let $t_{12}^0 = t_1^0 \cup t_2^0$, it is easy to see that $G[t_{12}^0] = G[t_1^0] \cup G[t_2^0]$, and therefore contains all the critical precharged nodes in the N-part including the output node Z, as well as V_{ss}.

Lemma 3. *The pair (t_1, t_2) can be made robust by refinement if and only if $G[t_{12}^0]$ does not join Z to V_{ss}.*

proof: two parts.

(i) **only if part** : By contradiction. Suppose Z is joined to V_{ss} in $G[t_{12}^0]$. Notice that $G[t_{12}^0] \subseteq G[t]$, $\forall t \subseteq t_{12}$ (provided t has no X entries, so that $G[t]$ is well defined). Suppose certain t_1', t_2', and shf requirements, S_{hf}, have been chosen. Let s_{12}' be the *cube* consisting of all the possible states in the $t_1' \rightarrow t_2'$ transitions obeying the shf requirements. Therefore $t_1' \subseteq s_{12}'$ and $t_2' \subseteq s_{12}'$. Any non-X entries of s_{12}' are shf and must be the same as their corresponding entries in t_1' and t_2', and must, therefore, be subsets of their corresponding entries in t_1 and t_2 ($[0] \subseteq [X]$, $[1] \subseteq [X]$). Therefore $t_{12} \cap s_{12}' \neq \Phi$. Now let $t \subseteq t_{12} \cap s_{12}'$ have no X entries, then t is a possible intermediate

state and $G[t_{12}^0] \subseteq G[t]$, since $t \subseteq t_{12}$. Therefore whatever shf requirement are made for the $t_1' \rightarrow t_2'$ transition, Z and all other critical precharged nodes in the N-part will be joined to V_{ss} for some intermediate vector t. So the test cannot be made robust.

(*ii*) **if part** : Constructive. Suppose $G[t_{12}^0]$ does not join Z to V_{ss}. The 0's in t_{12}^0 correspond to either X's or 0's in t_1 and t_2. Set all these X's to 0's to get t_1' and t_2' (ie, $t_1' = t_1 \cap t_{12}^0$ and $t_2' = t_2 \cap t_{12}^0$), and specify that all the entries corresponding to the 0's of t_{12}^0 be 0-static-hazard-free in the transition $t_1' \rightarrow t_2'$. Therefore if t is any intermediate state in the transition, $G[t] \subseteq G[t_{12}^0]$, so the precharged nodes are never joined to V_{ss}, and the test is robust. ■

Theorem 6. *P_5 becomes of polynomial complexity if S is a fully complementary CMOS gate with a transistor stuck-at-off fault.*

proof: The proof is constructive and gives a linear time algorithm that either decides that a test cannot be made robust, or else refines it to make it robust. Simply stated, the algorithm follows the proof of Lemma 3 : given t_1 and t_2, form t_{12}^0 and check if Z and V_{ss} are connected in $G[t_{12}^0]$. If so then the test cannot be made robust, otherwise form $t_1' = t_1 \cap t_{12}^0$ and $t_2' = t_2 \cap t_{12}^0$, and specify that all the entries corresponding to the 0's of t_{12}^0 be 0-static-hazard-free in the transition $t_1' \rightarrow t_2'$. All the operations described can be easily done in linear time. ■

Even though the algorithm above is very efficient, it does give shf requirements that may be an overkill, the test pair can be made robust using less stringent shf requirements as follows (the assumption below is that Z is not joined to V_{ss} in $G[t_{12}^0]$). Contract all the edges in G that belong to $G[t_{12}^0]$, this can be done in linear time. Call the new node

of which Z is part s, and that of which V_{ss} is part t. Orient the edges of G as follows : all edges incident on Z are oriented away from it, all edges incident on V_{ss} are oriented towards it, and all other edges are duplicated and oriented in both directions. This gives a directed graph G_d, and assigning an edge capacity of 1 to every edge we get a *Network.*

All the edges corresponding to the 0's in t_{12}^0 constitute a *cut*, but this may not be a *minimum cut.* It is in this sense that the shf requirements generated above may be an overkill. By using maxflow-mincut techniques [19] a mincut can be found in $O(|S| \cdot |S|^{1/2})$ time where $|S|$ is the size (number of transistors) of the given subcircuit. Having found such a mincut, make all corresponding entries of t_1 and t_2 0's, this gives t_1' and t_2', and require all these entries to be 0-shf. This gives a robust test pair with minimum requirements in polynomial time.

In a test generation setup the minimum requirements generated here may not be the best choice : they may fail during the justification phase when the test is being propagated at the gate-level description of the circuit. Other alternatives, such as *minimal* cuts become important [2], [7].

VI. CONCLUSIONS

This paper proves that the problem of generating a test for a single fault in one channel-connected MOS subcircuit is, in general, $NP - hard$. This same result is shown to hold in the special cases of series/parallel NMOS or fully complementary CMOS gate designs. The size of the problem is the size of the subcircuit, given by the number of transistors in it. Although the size of many channel-connected subcircuits is small in practice, and tests for them can be derived within reasonable time, there are cases where the size can

grow into the hundreds of transistors, which makes the complexity results derived here particularly important.

The implications of these results are that a transistor level test generation algorithm should use heuristics or approximations in order to perform reasonably. Furthermore, it is important to look for certain restricted design styles and fault types for which efficient, special purpose, polynomial time algorithms exist. In view of this, it is remarkable that the test generation problem for the special case of an NMOS or CMOS gate is still $\mathcal{NP} - hard$.

One way of tackling the algorithmic complexity of the test generation problem in general, aside from using heuristics, is to look for non-deterministic or probabilistic algorithms. Examples in this class are algorithms that perform efficiently on *most* circuits [16], or algorithms that would *probably* perform efficiently on a given circuit [17]. Another way out is to look for approximation algorithms [18] which do not guarantee finding a test whenever one exists, but rather efficiently find only *some* of the tests. These represent research directions that are currently being investigated. The complexity of the problem can be reduced if design for testability concepts [22] are used during the design phase.

REFERENCES

[1] O. H. Ibarra and S. K. Sahni, "Polynomially complete fault detection problems," *IEEE Transactions on Computers*, vol. C-24, no. 3, pp. 242–249, March 1975.

[2] F. N. Najm, "Switch-level test generation for MOS VLSI circuits," Report # UILU-ENG-86-2223, Coordinated Science Lab., Univ. of Illinois at Urbana-Champaign, August 1986.

[3] I. N. Hajj and F. N. Najm, "Test generation for physical faults in MOS VLSI circuits," *IEEE Comp-Euro Conference*, Hamburg, West Germany, pp. 386–389, May 11–15, 1987.

[4] R. L. Wadsack, "Fault modeling and logic simulation of CMOS and MOS integrated circuits,", *The Bell System Technical Journal*, vol. 57, no. 5, pp. 1449–1474, May–June 1978.

[5] S. M. Reddy, M. K. Reddy, and V. D. Agrawal, "Robust tests for stuck-open faults in CMOS combinational logic circuits," *IEEE 14th International Symposium on Fault Tolerant Computing*, Kissimee, FL, pp. 44–49, June 20–22, 1984.

[6] J. A. Abraham and H-C. Shih, "Testing of MOS VLSI circuits," *International Symposium on Circuits and Systems*, Kyoto, Japan, June 5–7, 1985.

[7] M. R. Lightner and G. D. Hachtel, "Implication algorithms for MOS switch-level functional macromodeling, implication and testing," *IEEE 19th Design Automation Conference*, Las Vegas, NV, pp. 691–698, June 1982.

[8] P. Agrawal, "Test generation at switch-level," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, pp. 128–130, Nov. 12–15, 1984.

[9] M. K. Reddy, S. M. Reddy, and P. Agrawal, "Transistor level test generation for MOS circuits," *IEEE 22nd Design Automation Conference*, pp. 825–828, 1985.

[10] S. H. Robinson and J. P. Shen, "Towards a switch-level test pattern generation program," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, pp. 39–41, Nov. 18–21, 1985.

[11] H-C. Shih and J. A. Abraham, "Transistor-level test generation for physical failures in CMOS circuits," *IEEE 23rd Design Automation Conference*, pp. 243–249, 1986.

[12] E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial algorithms, theory and practice*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1977. pp.401–408.

[13] S. A. Cook, "The complexity of theorem-proving procedures," *The 3rd Annual ACM Symposium on Theory of Computing*, Shaker Heights, OH, pp. 151–158, May 3–5, 1971.

[14] A. Thayse and M. Davio, "Boolean differential calculus and its application to switching theory," *IEEE Transactions on Computers*, vol. C-22, no. 4, pp. 409–420, April 1973.

[15] J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits," *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 5, pp. 71–84, Oct. 1967.

[16] R. M. Karp, "The probabilistic analysis of some combinatorial search algorithms," in *Algorithms and Complexity*, ed., J. F. Traub. New York, NY: Academic Press, Inc., pp. 1–19, 1976.

[17] M. O. Rabin, "Probabilistic algorithms," in *Algorithms and Complexity*, ed., J. F. Traub. New York, NY: Academic Press, Inc., pp. 21–39, 1976.

[18] M. R. Garey and D. S. Johnson, "Approximation algorithms for combinatorial problems : an annotated bibliography," in *Algorithms and Complexity*, ed., J. F. Traub. New York, NY: Academic Press, Inc., pp. 41–52, 1976.

[19] S. Even, *Graph algorithms.* Rockville, MD: Computer Science Press, Inc., 1979.

[20] J. P. Roth, *Computer logic, testing, and verification.* Potomac, MD: Computer Science Press, Inc., 1980.

[21] H. H. Chen, R. G. Mathews, and J. A. Newkirk, "An algorithm to generate tests for MOS circuits at the switch level," *Proceedings of the IEEE International Test Conference*, pp. 304–312, 1985.

[22] D. L. Liu and E. J. McCluskey, "Switch-level testability," *IEEE Design and Test of Computers*, vol. 4, no. 4, pp. 42–49, August 1987.

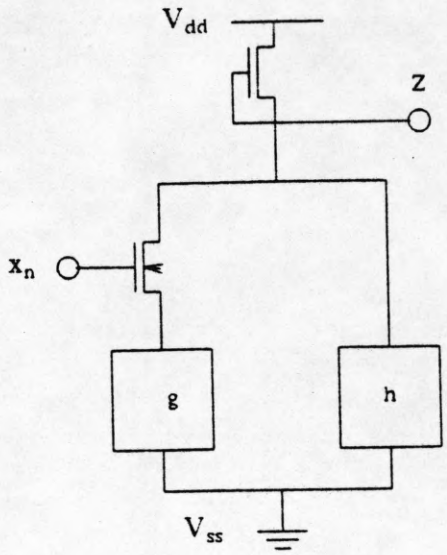

Figure 1. An NMOS gate.

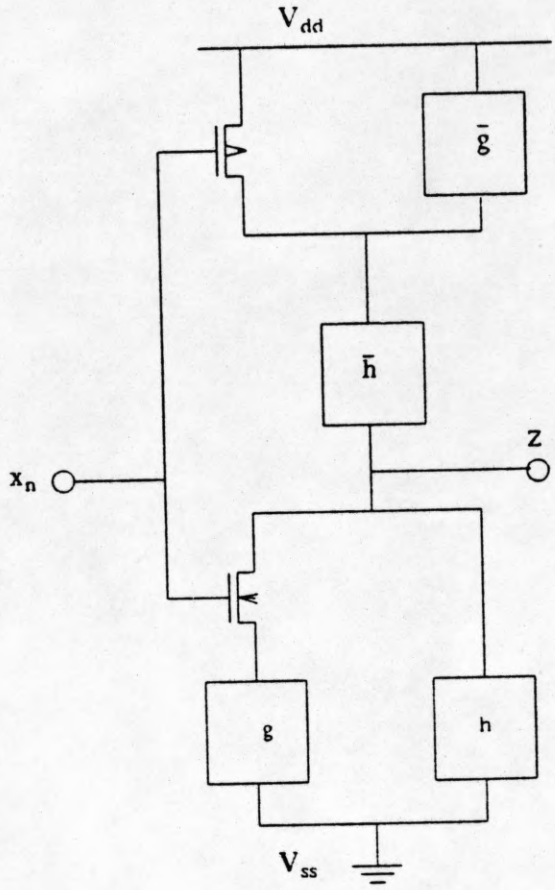

Figure 2. A CMOS gate.

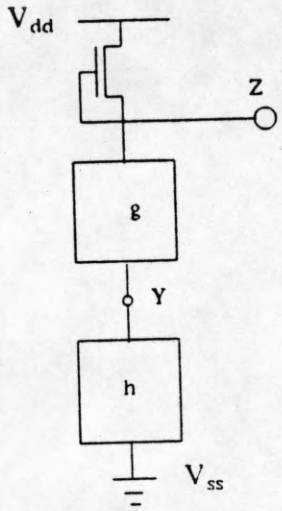

Figure 3. An NMOS gate.

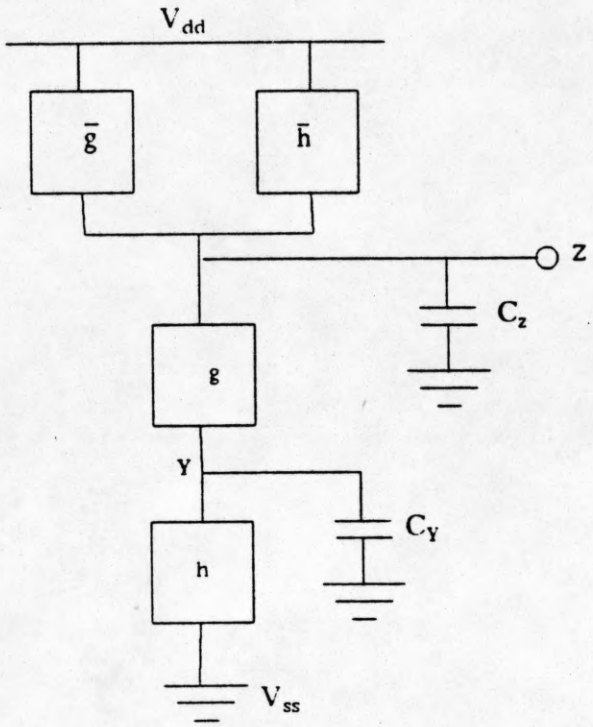

Figure 4. A CMOS gate.

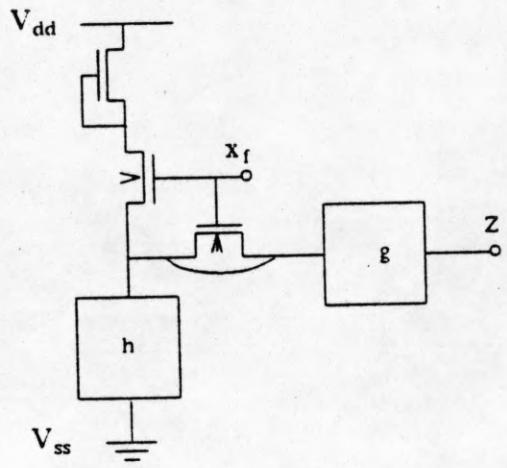

Figure 5. A general subcircuit.

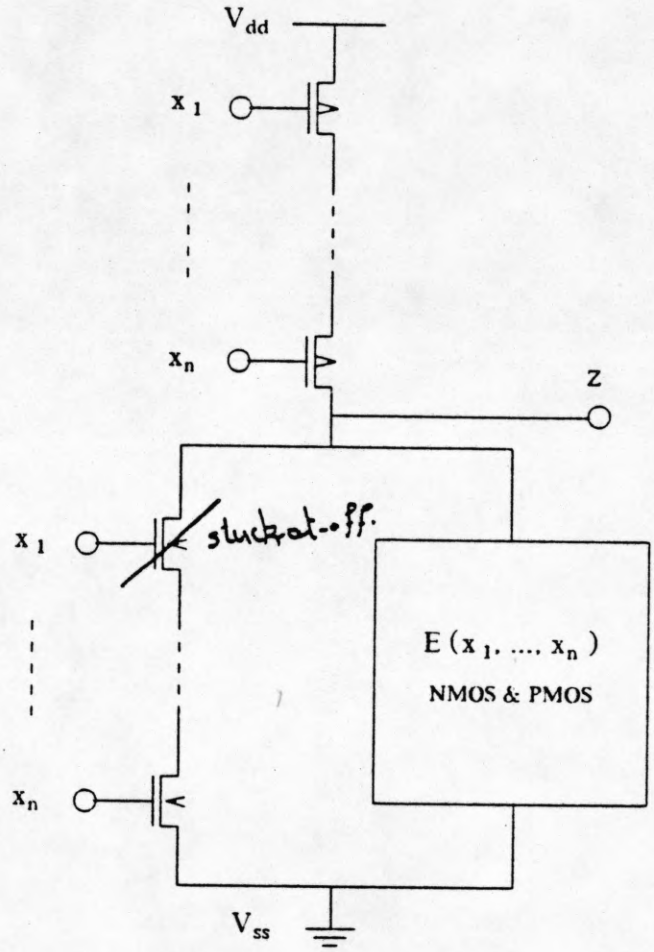

Figure 6. A general subcircuit.