The Complexity of Fault Detection in MOS VLSI Circuits*

Farid N. Najm† and Ibrahim N. Hajj

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

Abstract

This paper considers the fault detection problem for a single fault in a single MOS channel-connected subcircuit. We identify the following three decision sub-problems : *(i)* decide if a test vector exists; *(ii)* decide if an initializing vector exists; and *(iii)* decide if a test pair is robust. We prove that each of these problems is $\mathcal{NP}-complete$. More importantly, we prove that the first two remain $\mathcal{NP}-complete$ for the simplest subcircuit design styles, namely series/parallel nMOS or CMOS *logic* gates. The third subproblem is shown to be of *linear* complexity for a CMOS logic gate with a stuck-open fault. We illustrate that a test pair that is not robust may *contain* a robust sub-test pair, and give a necessary and sufficient condition for this to happen in CMOS logic gates. This leads to a linear-time algorithm for CMOS logic gates which tests for robustness and, if possible, derives a robust test pair from a possibly non-robust pair. The implications of these complexity results on practical transistor-level test generation tools are discussed.

* This work was supported by the Semiconductor Research Corporation under Contract SRC RSCH 84-06-049.

† F. N. Najm is now with the Semiconductor Process and Design Center, Texas Instruments Inc., P.O. Box 655621, MS 369, Dallas, Texas 75265

1 Introduction

The testing of VLSI circuits and systems is a major concern in the electronics industry. Extensive research has been done, and is still underway, in fault simulation and test generation in order to ensure reliable fault-free products. Testing is becoming more critical with the increase in the number of devices on a chip and the corresponding decrease in feature size.

Most test generation tools currently being used in the industry consider gate-level or higher functional-level representation of the design. As a result, faults are restricted to stuck-at fault models (a node shorted to power or ground is said to be stuck-at 1 or 0). The advantage of such representation and fault modeling is that the testing task becomes technology-independent and can be carried out during the system design phase before the actual physical design is completed.

Although gate-level and functional-level testing may be acceptable for many applications, it is not adequate for high performance circuits. Many faults are technology-dependent (they occur at the transistor and layout levels) and should be tested during the physical design phase. Some of these faults are hard, or even impossible, to model by stuck-at fault models at the gate or functional levels [1, 2]. Thus, there is definitely a need to consider technology-dependent and transistor-level representations in order to approach fault-free products.

In this paper we consider the problem of detecting permanent transistor-level faults in MOS digital circuits. These faults are typically modeled as transistor stuck-open or closed faults. They include failures in actual transistors, as well as bridging faults, which can be modeled by inserting artificial transistors between nodes. We consider the circuit to be partitioned into primitive modules consisting of nodes that are connected by transistor channels, along with the transistors forming the connections [3]. Power supply, ground, and primary input nodes always belong to the boundaries of modules. Such modules, which have been called channel-connected subcircuits, static components [3], transistor groups, or circuit blocks, are often simply the MOS implementations of *logic gates*. We will refer to them as *channel-connected subcircuits*, or simply as *subcircuits*. In general, a subcircuit may contain pass-transistors and its behavior cannot be easily predicted by simple boolean models. When it does implement a logic gate, we will refer to a subcircuit as a *logic gate*, or simply a *gate*.

The problem of automatically generating tests to detect faults in VLSI circuits is naturally divided into two stages, which take place at the *local* and *global* levels, as follows. At the local level, the subcircuit in which the fault occurs is first identified, and a *local test(s)* is devised which, if applied at the inputs of that subcircuit, would produce a faulty value at its output. At the global level, this test(s) is propagated backwards to the circuit primary inputs, and the faulty subcircuit output value is propagated forward to the circuit primary outputs. If these propagation steps are successful, then a test has been derived [4].

In classical gate-level test generation schemes, the local test(s) are often readily available from the gate library, and the bottleneck becomes the two forward and backward propagation steps. The classical test generation problem has been extensively studied, and has been shown [5, 6] to be $\mathcal{NP}-hard$. Recently, a number of test generation schemes have been proposed for transistor-level faults [7]–[14]. Perhaps the most significant difference between these and previous, gate-level, test generation schemes is that local tests are no-longer easily available, but must be derived as well.

It has been established [1, 2], and has been our experience [7, 8] that the problem of deriving local tests for transistor-level faults is not trivial. The reason for this is mainly that the circuit response to a transistor-level fault often depends on its *electrical*, rather than simply logical, properties. This is especially true for faults that require two test vectors for detection. Such faults occur frequently [1, 2, 7, 8, 15, 16] in MOS circuits. In such cases, the first vector is used to set-up certain charges at some subcircuit nodes in preparation for the second vector. We will refer to the second vector of a test pair as a *test vector*, represented as t_2, and to the first as an *initializing vector*, represented as t_1. For uniformity of presentation, a test vector will be called t_2 even if it does not require initialization. A given two-vector test sequence may be invalidated in practice due to signal skews at the subcircuit inputs [1, 11, 16]. A test pair that is valid irrespective of signal skews is said to be *robust* [15].

This paper investigates the complexity of the local test generation problem for faults at the transistor-level, and shows that it is $\mathcal{NP}-hard$. To do so, we identify the following three decision sub-problems at the local level : *(i)* decide if a test vector exists; *(ii)* decide if an initializing vector exists; and *(iii)* decide if a test pair is robust. We prove that each of these problems is $\mathcal{NP}-complete$. More importantly, we prove that the first two remain $\mathcal{NP}-complete$ for the simplest subcircuit design styles, namely series/parallel nMOS or

CMOS logic gates. The third subproblem is shown to be of *linear* complexity for CMOS logic gates with single stuck-open faults. We illustrate that a test pair that is not robust may *contain* a robust sub-test pair, and give a necessary and sufficient condition for this to happen in a CMOS logic gate. This leads to a linear-time algorithm for CMOS logic gates which, if possible, derives a robust test pair from a possibly non-robust pair, or else declares the test to be non-robust.

The remainder of this paper is divided into six sections. Section 2 considers the complexity of finding a single test vector, Section 3 deals with finding an initializing vector, and Section 4 examines the complexity of checking for robustness. Section 5 considers the robustness problem in CMOS logic gates. Finally, Section 6 gives some concluding remarks and discusses the practical implications of the results presented in this paper.

2 Complexity of Generating t_2

This section is concerned with the complexity of deriving a single test vector t_2. We begin with some preliminaries from boolean algebra that will be needed in the proofs below.

A boolean 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 [17], 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), i.e., 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 [17, 18] that SAT is $\mathcal{NP}-$ *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. We will represent the fact that a problem P is *transformable in polynomial time* to a problem P' by writing $P \propto P'$.

Definition 1. *Problem P_0 : Let g (h) be a boolean expression in CNF (DNF) with no complemented literals, and let $\overline{h}$ be the complement of h. Is $g\overline{h}$ satisfiable?*

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

proof: By DeMorgan's Laws, $\overline{h}$ is a CNF with only complemented literals. Hence P_0 is an easy equivalent restatement of the satisfiability problem for unate CNF formulas : P_0 is equivalent to the *clause-monotone* problem [6] (also MONOTONE 3SAT [17]) which are $\mathcal{NP}-complete$. ■

Consider the following fault detection problem :

Definition 2. *Problem P_1 : Let S be a series/parallel nMOS or CMOS logic gate with either a transistor stuck-closed or stuck-open fault f, does there exist a test vector t_2 for f?*

Theorem 1. *P_1 is $\mathcal{NP}-complete$.*

proof: P_1 is in $\mathcal{NP}$. To complete the proof we will prove that $P_0 \propto P_1$. Let g and h be an instance of P_0. Construct, in linear time, a series/parallel realization of g and h using nMOS transistors and build the nMOS gate shown in Fig. 1, where x_f does not occur in g or h. Let there be a fault f (either stuck-closed or stuck-open) at transistor T_f. This constitutes an instance of P_1. It is clear that a test vector t_2 can be found if and only if $g\overline{h}$ is satisfiable. Therefore $P_0 \propto P_1$. The same result holds for CMOS by considering the gate in Fig. 2. ■

It is easy to prove, using similar arguments, that the single node stuck-at (0 or 1) fault detection problem for nodes internal to the gate is also $\mathcal{NP}-complete$ (consider a stuck-at-0 fault at node Y in Fig. 3).

3 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. We now examine the complexity of the problem of finding t_1.

To point out some of the issues involved in deriving t_1, and to motivate the formal definition of the problem as given below, consider the CMOS logic gate shown in Fig. 4, where the nMOS transistor driven by input C is stuck-open. Suppose that the two nMOS transistors tied to ground are wide enough so that the parasitic capacitances at X and Y are comparable with that at Z. One possible test vector for this fault is $t_2 = 1010$ ($A = 1, B = 0, C = 1, D = 0$). It works by trying to turn *on* the path "$Z - X - ground$", while keeping *off* the path "$Z - Y - ground$". It is clear that an initializing vector t_1 must precharge Z to 1. However, since t_2 ties Z to X, which has comparable capacitance, and to guarantee that the charge on Z is not lost due to *charge sharing* [1, 16], then t_1 must also precharge

X to 1. Therefore, $t_1 = 1000$ is the required initializing vector. Another valid test vector is $t_2 = 1110$, which ties Z to both X and Y. In this case t_1 must precharge all three nodes $X, Y,$ & Z, hence $t_1 = 1100$. Therefore, the initializing vector t_1 depends not only on the particular fault, but also on the choice of test vector t_2.

Having made this introduction, we are now ready to define the following initialization problem :

Definition 3. *Problem P_2 : Let S be an MOS subcircuit, with either a transistor stuck-closed or stuck-open fault f, for which some test vector t_2 is known to require initialization, does there exist an initializing vector t_1 for t_2?*

Theorem 2. *If S is a series/parallel CMOS logic gate with a stuck-open fault f, then P_2 is $\mathcal{NP}$ − complete.*

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

It is clear that P_2 is in $\mathcal{NP}$, we will prove that $P_0 \propto P_2$. Using an instance of P_0 build (in linear time) the gate in Fig. 5 so that $C_Y = C_Z$ and all other internal node capacitances are negligible. Let there be a stuck-open fault at nMOS transistor T_f.

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 $g\overline{h}$ is satisfiable, and $P_0 \propto P_2$. ∎

Before going on, we should make the following point. If one assumes that the output node capacitance is always much higher than the capacitances at the internal nodes, then charge sharing can be neglected, and the only node to be initialized is the output node. In this case the initialization problem P_2 becomes of linear complexity, since it can be solved by simply finding a path of transistors to be turned on in either the p- or n-block of the gate, depending on the required initial value.

The problem of initialization does not arise for a single nMOS gate. Furthermore, in the case of a transistor stuck-closed fault, P_2 does not arise for either nMOS or CMOS gates. If, however, the design style is not constrained to be either nMOS or CMOS logic gates (in which case we refer to it as an *unconstrained* design style) then initialization may be required even for stuck-closed faults.

As an aside, we should point out that, by unconstrained design style, we *do* mean a completely free style where any interconnection of P or N transistors is allowed. To a MOS circuit designer, such a design style will seem useless, and hence the results in the next two theorems may seem insignificant. However, a tool developer, writing a transistor-level test generation program that accepts circuit descriptions in the form of transistors and nodes, must decide whether to disallow certain strange subcircuit configurations, or else to decide how to handle *all* configurations. It is with this second choice in mind that we include the next two theorems, which are based on the two non-standard subcircuits in Figs. 6 and 8. If nothing else, these two results establish that, if standard design practices are *not* followed, then the problems are no longer of polynomial-time complexity.

Theorem 3. *P_2 is $\mathcal{NP}$ – complete for stuck-closed faults if an unconstrained design style is allowed.*

proof: As above, P_2 is in $\mathcal{NP}$ and we will prove that $P_0 \propto P_2$. Construct a subcircuit in polynomial time using nMOS transistor implementations of g and h as shown in Fig. 6. The figure also shows an assumed stuck-closed fault at one of the transistors with gate label x_f. 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. This constitutes an instance of P_2. An initializing vector t_1 can be found if and only if $g\overline{h}$ is satisfiable, therefore $P_0 \propto P_2$. ∎

4 Complexity of Checking 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.* For instance, in the example in Fig. 4, if the test pair $t_1 = 1100$, $t_2 = 1110$ is used then X, Y, & Z are three critical precharged nodes.

Whether these charges are lost or not depends on the way the transition $t_1 \rightarrow t_2$ takes place. Signal skews at the inputs to the subcircuit [1, 11, 16] can cause certain transistors to switch before others, causing the values of these critical nodes to be changed before t_2

is applied. As an example, consider the CMOS gate in Fig. 7, which has been borrowed from [11], in which a stuck-open fault is assumed at the pMOS transistor driven by input B. In this case, the test vector is $t_2 = 001$, and requires output Z to be precharged to 0. If this charge is lost during the transition then the test will be invalidated. Z is the only critical precharged node in this case. A possible initializing vector is $t_1 = 100$. If, however, input A switches to 0 before C has switched to 1, then the intermediate state 000 will take the output high and invalidate the test.

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* [15]. It is helpful to visualize t_1 and t_2 as *cubes* [4] 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.

Define the following robustness problem :

Definition 4. *Problem P_3 : Given an MOS subcircuit S along with a test pair (t_1, t_2) for a certain fault in S, is the test pair robust?*

Theorem 4. *P_3 is $\mathcal{NP}$ − complete if an unconstrained design style is allowed.*

proof: P_3 is in $\mathcal{NP}$, we still need to show that it is also $\mathcal{NP}$ − *hard*. To do so we will show that a polynomial-time algorithm $\mathcal{A}$ for P_3 can be used to construct a polynomial-time algorithm for SAT. Let $E(x_1, \ldots, x_n)$ be a boolean expression in CNF. The required algorithm is simply as follows:

-1- If $E(0, \ldots, 0) = 1$, then E is satisfiable.

-2- Else if $E(1, \ldots, 1) = 1$, then E is satisfiable.

-3- Else 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 Fig. 8. Let there be a transistor stuck-open 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 (possibly) discharged to 0 by t_2. Since $E(0, \ldots, 0) = 0$ and $E(1, \ldots, 1) = 0$, then the E block is off when either t_1 or t_2 is applied, and therefore (t_1, t_2) is a valid test pair. This constitutes an instance of P_3, and $\mathcal{A}$ can be used to solve it in polynomial-time.

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 the test pair is robust if and only if E is not satisfiable. ■

We will show in the next section that this problem becomes of linear complexity for the special case of a CMOS logic gate.

5 Robustness in CMOS logic gates

We will now study the complexity of P_3 for the case of a single CMOS logic gate (not necessarily series/parallel) with a transistor stuck-open fault. Some remarks are necessary at first to set the stage for the remainder of this section.

A compact representation of t_1 or t_2 is a row-vector, or cube, 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. A cube t' is a subset of a cube t if and only if t' can be obtained from t by replacing some or all of the X entries in t by 0s or 1s. We will use $\subseteq$ to represent the subset (inclusion) relation between two cubes. We also denote by $\cap$ ($\cup$) the bit-wise *and* (*or*) operation between two cubes using ternary logic.

Consider again the example in Fig. 7. As described above, the test pair $t_1 = 100$, $t_2 = 001$ was found to be non-robust because both A and C are supposed to switch, causing a race-hazard situation which can invalidate the test if the state 000 is obtained. In fact, a more efficient description of this initializing vector would be $t_1 = 1XX$, signifying the fact that as long as A is high to drive Z low, the other values at B and C are irrelevant. Of course this t_1 still suffers from the same race-hazard problem. However, if the subcube $t_1' = 1X1$ is chosen, then C no longer has to switch, and the race-hazard is removed, resulting in a robust test pair ($1X1$, 001). However, if a glitch occurs on C, then the state 000 can occur and invalidate the test, this is called a *static hazard* problem. Therefore, an exact description of this robust test pair should carry with it the requirement that C should be *static-hazard-free*, to be abbreviated *shf*, during the transition. In general, a robust test pair comes with shf requirements for one or more input nodes; it will be written as a robust test triplet (t_1, S_{hf}, t_2), where S_{hf} is the set of inputs that need to be shf.

Notice that, as in the above example, a non-robust test pair (t_1, t_2) may be made 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*). The choice of t'_1, t'_2, and S_{hf} is not unique. This process of extracting a robust test pair from a non-robust one will be referred to as *refinement* : (t_1, t_2) is *refined* to produce (t'_1, S_{hf}, t'_2). In the remainder of this section we derive a necessary and sufficient condition for a test pair in a CMOS logic gate to either be robust or to contain a robust sub-test pair. This leads to a linear-time algorithm that either refines a given test pair to make it robust, or else declares it as non-robust. This essentially proves that P_3 becomes of linear complexity for CMOS logic gates.

Given a CMOS logic gate S with a stuck-open fault at a transistor T_f whose gate label is x_f. Given also a test pair $(t_1,\ t_2)$, not necessarily robust, which detects this fault. Let the gate output node be Z. 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 gate 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* associated with t. 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}, or to some other discharged node with high enough capacitance to cause charge sharing, then the charges will be lost and the test invalidated. If this is not the case for a certain $G[t]$, then it will be called *charge-preserving.* We will assume, that, knowing the node capacitances, it is possible to check in linear time whether a certain subgraph $G[t]$ is charge-preserving or not.

Let t_1^0 (t_2^0) be obtained from t_1 (t_2) by replacing the X's in its 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. 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 $t_{12} = t_1 \cup t_2$ and $t_{12}^0 = t_1^0 \cup t_2^0$; 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 2. *The pair (t_1, t_2) can be made robust by refinement if and only if $G[t_{12}^0]$ is charge-preserving.*

proof: Two parts :

(i) **only if part** : By contradiction. Suppose $G[t_{12}^0]$ is not charge-preserving. Notice that if $t \subseteq t_{12}$ has no X entries, then $G[t_{12}^0] \subseteq G[t]$. 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. 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 lose their charges for some intermediate vector t. So the test cannot be made robust.

(ii) **if part** : Constructive. Suppose $G[t_{12}^0]$ is charge-preserving. 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, $G[t] \subseteq G[t_{12}^0]$, and the test is robust. ■

Theorem 5. *If S is a CMOS logic gate with a transistor stuck-open fault, then there exists a linear-time algorithm for P_3.*

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 2 : given t_1 and t_2, form t_{12}^0 and check if $G[t_{12}^0]$ is charge-preserving. If not 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 given above is very efficient, it may give shf requirements that are an overkill; ie, it may be possible to make the test pair robust using less stringent shf requirements. This may be done by posing the problem as a network flow problem [19] and looking for minimum (or minimal) cuts in the resulting network.

6 Conclusion

This paper considers the fault detection problem for a single fault in a single MOS channel-connected subcircuit. We identify the following three decision sub-problems : *(i)* decide if a test vector exists; *(ii)* decide if an initializing vector exists; and *(iii)* decide if a test pair is robust. We prove that each of these problems is $\mathcal{NP}-complete$. More importantly, we prove that the first two remain $\mathcal{NP}-complete$ for the simplest subcircuit design styles, namely series/parallel nMOS or CMOS *logic* gates. The third subproblem is shown to be of *linear* complexity for a CMOS logic gate with a stuck-open fault. We illustrate that a test pair that is not robust may *contain* a robust sub-test pair, and give a necessary and sufficient condition for this to happen in a CMOS logic gate. This leads to a linear-time algorithm for CMOS logic gates that, if possible, derives a robust test pair from a possibly non-robust pair, or else declares the test to be non-robust.

The significance of these results is that it is very unlikely that a polynomial-time algorithm for local test generation in general MOS circuits will be found. We should make the point, however, that our results all refer to the *asymptotic* complexity of this problem. Since "practical" MOS gates have a constant ($\mathcal{O}(1)$) upper limit on their number of transistors, then exponential or exhaustive algorithms may be acceptable for them. Therefore, the formal results derived above may have limited practical importance. Nevertheless, it is important to know whether polynomial-time algorithms for these problems, for general MOS circuits, *exist* or not. Furthermore, test generation tools that work at the switch or transistor levels need to be able to handle not just implementations of logic gates, but also many other configurations, such as pass-transistors in multiplexors, shifters, or data-paths. These subcircuits can grow very large in size, e.g. up to 5000 transistors [3], which gives our asymptotic complexity results significant practical importance in these cases.

Therefore, a transistor-level test generation algorithm should use heuristics or approximations in order to perform reasonably on very large subcircuits. 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 significant that the fault detection problem remains $\mathcal{NP}-complete$ for the special case of a series/parallel nMOS or CMOS logic gate.

Another way around the hardness of this problem may be to preprocess and precharacterize the subcircuits so that tests for them would be readily available for the test generator. Certain design styles, such as standard cells, are especially attractive in this respect because the capacitance at internal nodes, which is crucial to the success of transistor-level tests, is predetermined, and does not depend on the final layout of the whole chip. In previous work [7, 8], the authors describe their implementation of a transistor-level test generation tool, called ITEST, which takes into account the effects of charge-sharing, voltage division, and race and static hazards, and guarantees robust test pairs. It is based on a switch-level model of MOS circuits, and has been used to characterize a variety of MOS subcircuits, ranging from simple gates to more complex circuits with pass-transistors and transmission gates.

References

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

[2] 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.

[3] R. E. Bryant, "A survey of switch-level algorithms," *IEEE Design and Test of Computers*, vol. 4, no. 4, pp. 26–40, August 1987.

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

[5] 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.

[6] H. Fujiwara and S. Toida, "The complexity of fault detection problems for combinational Logic circuits", *IEEE Transactions on Computers*, vol. C-31, no. 6, pp. 555–560, June 1982.

[7] 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.

[8] 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.

[9] 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.

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

[11] S. K. Jain and V. D. Agrawal, "Test generation for MOS circuits using D-algorithm," *IEEE 20th Design Automation Conference*, Miami Beach, FL, pp. 64–70, June 27–29, 1983.

[12] 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.

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

[14] 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.

[15] 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.

[16] S. M. Reddy and M. K. Reddy, "Testable realizations for FET stuck-open faults in CMOS combinational logic circuits," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 742–754, August 1986.

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

[18] 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.

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

Figure Captions

Figure 1: An nMOS gate, used to study the complexity of generating t_2.

Figure 2: A CMOS gate, used to study the complexity of generating t_2.

Figure 3: An nMOS gate.

Figure 4: A CMOS gate, used to illustrate the problems of charge sharing and charge loss.

Figure 5: A CMOS gate, used to study the complexity of generating t_1.

Figure 6: An MOS subcircuit with a transistor stuck-closed fault.

Figure 7: A CMOS gate with a transistor stuck-open fault.

Figure 8: An MOS subcircuit with a transistor stuck-open fault.