TEST GENERATION FOR PHYSICAL FAULTS IN MOS VLSI CIRCUITS†

Ibrahim Hajj and Farid Najm
Coordinated Science Laboratory
and the Department of Electrical and Computer Engineering
University of Illinois
Urbana, IL 61801

ABSTRACT

This paper describes a new approach to automatic test generation for detecting physical faults in MOS digital circuits. The faults could be shorts between lines or nodes, including drain-to-gate and source-to-gate shorts, or opens in lines or in transistor channels. Faults which fail to be tested by existing methods are successfully detected by our approach. This is done by using switch-level models and properly taking into account the possibilities of charge sharing and charge loss. The approach detects the need for setting up appropriate initial conditions when necessary and derives robust test vector pairs that are free from race conditions. The method has been implemented in a computer program called **Itest** *which accepts inputs in the form of transistor interconnections and fault lists, and automatically generates the test vectors.*

I. INTRODUCTION

It is well established that some types of physical faults in digital circuits, especially MOS circuits, cannot be represented by classical fault models (stuck-at-0, stuck-at-1) [1], [2], and thus cannot be detected using the classical test generation techniques [3]. In CMOS circuits, for example, stuck-at-open faults may require two test patterns [4], that cannot be derived using classical techniques. This problem could also occur in NMOS circuits. For example, the simple NMOS circuit of Figure 1, which includes a pass-transistor T3, requires a two-pattern test sequence to test for "T3 stuck at ON" (a SHORT fault). The first test is [X,Y]=[1,1], which is applied to initialize the output node to 0, followed by the test [X,Y]=[0,0]. Notice that if X switches before Y then the output may erroneously go to 1 even if the fault is not there, thus a GOOD (fault-free) circuit would be diagnosed as BAD (faulty). The reason being that the preserved charge (0) is in the GOOD circuit in this case. In the case of an OPEN fault the charge is preserved in the BAD circuit; and if the charge is destroyed, the BAD circuit will be diagnosed as GOOD. In general, it turns out that a test may require initialization of certain charges in either the BAD circuit or the GOOD circuit or both.

The main problems associated with test generation of physical faults are charge-sharing [3] and charge-loss [5]. A test generation algorithm should solve these problems; it should guarantee that the derived tests, if they exist, will not lead to charge sharing or to any ambiguity in the expected output. The tests should also be robust [4]; i.e. the output will not suffer from charge loss.

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

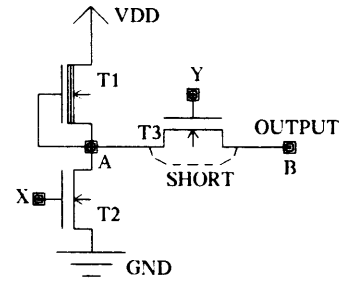


Figure 1. An NMOS circuit that needs two test vectors.

To illustrate these problems consider the circuit in Figure 2, where the fault to be tested for is T stuck at open. This circuit is shown in [3] to demonstrate the inability of existing test generation schemes to derive tests that are free of charge sharing. Charge sharing may occur in the faulty circuit because the application of the test vector t_2, shown in Figure 2, joins nodes 5,6,7 and 8 and causes them to be isolated from the supply nodes. For this reason, the initializing vector t_1 should initialize all of these four nodes to 1 in preparation for t_2. In addition it is important to note that if node 3 suffers a glitch during the transition then these charges may be destroyed and the test invalidated. Thus the circuit suffers from charge loss problems as well.

TEST

	1	2	3	4
t_1	1	1	0	0
t_2	1	1	0	1

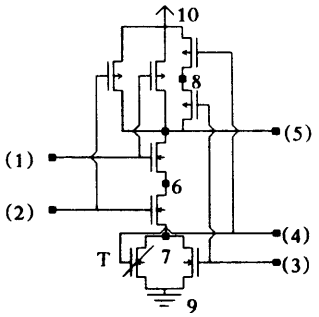


Figure 2. A circuit with charge sharing and charge loss problems.

The rest of this paper is organized as follows. In the next section the proposed test generation technique is outlined. Sections III-V describe the implication, initialization, and propagation phases of the algorithm. Section VI presents examples and implementation issues, and section VII draws some conclusions.

II. PROPOSED SOLUTION

In this paper we describe an automatic approach to deriving tests for faults resulting from either a short between two nodes or lines, or an open in a transistor channel or in a line. Drain-to-gate and source-to-gate shorts, which are not uncommon, and which are hard to detect by existing methods, are included. Thus, local

feedback within a subcircuit or between two adjacent subcircuits is allowed; otherwise the circuits allowed are basically combinational. Although some approaches have been proposed for physical fault test generation [4],[5],[6],[7], these approaches are not general enough to detect all possible faults, especially when the fault creates sequential behavior or causes charge sharing to occur, because they use logical models that cannot accurately describe the behavior of MOS circuits.

At the core of the test generation algorithm is an "implication" step [8]; meaning that the basic question to be answered repeatedly is: what combination(s) of inputs and internal node initial conditions would produce a certain desired output? This "desired" output value is generated by requiring that there be an observable discrepancy between the outputs of the fault-free and the faulty circuits.
If we define

G_0 (G_1) = the set of inputs and initial conditions combinations that would give an output of 0 (1) in the GOOD circuit,

and

B_0 (B_1) = the set of inputs and initial conditions combinations that would give an output of 0 (1) in the BAD circuit (for a certain fault),

then the set of test patterns for the considered fault is:

$$T = (G_0 \cap B_1) \cup (G_1 \cap B_0).$$

For a given fault, the implication step is carried out only on the subcircuit where the fault occurs. The node values are then propagated forward and backward using a modified D-algorithm [13] to reach the external nodes. We consider the circuit to be partitioned into dc-connected subcircuits [9], where every subcircuit contains a set of nodes that are connected by transistor channels. A subcircuit may have more than one output node. For a given fault in a given subcircuit, the test sets are generated using the following steps:

1. Find G_0, G_1, B_0 and B_1 . (implication)
2. Form $T_2 = (G_0 \cap B_1) \cup (G_1 \cap B_0)$
3. For every $t_2 \in T_2$, if no initial conditions are required, then t_2 is a single test vector; otherwise, find T_1={ set of all t_1 initializing input patterns that will produce the initial conditions required by t_2 }.
4. Extract the robust test pairs from those pairs derived in step 3; these tests are still at the subcircuit level.
5. Propagate the node values associated with a certain test or test pair forward and backward to the accessible external nodes of the circuit.

III. IMPLICATION

Implication, in general, is a difficult problem. In our approach, a switch-level model is used [9], and the circuit is first partitioned into dc-connected subcircuits. The implication step is then carried out on the subcircuit where the fault occurs. By using the approach described in [10], where, for a large class of digital circuits, the storage, as well as the input nodes, are assigned to a hierarchy of strength groups determined by path strengths, the implication step becomes easy. This is done by considering the strength groups separately, starting with the strongest, and identifying how the nodes involved can affect the output. For a given subcircuit, the algorithm begins with a graph search starting from the selected output node and discovering all the paths to every other node of the subcircuit that is stronger than the output node. The search algorithm [11] is a variation of the depth-first search [12]. The result of the search gives levels of the hierarchy as well as the paths from the output node to every other node.

IV. INITIALIZATION

Having found T_2, the set T_1 is constructed for every test vector t_2 in T_2 that needs initialization. This is done by using implication where each node, whose initial condition is required, is declared as an output node. In general, some patterns in T_1 may require initialization themselves, so that a test sequence for a given fault may be composed of more than just two test patterns. For clarity, the case when more than two test vectors are needed is not discussed here.

The selection of robust tests is then carried out by constructing a set of input assignments, C, associated with each $t_2 \in T_2$. This is done at the same time when T_1 is being found. C is the set of all allowable states that the inputs may take during the transition from a $t_1 \in T_1$ to t_2 without destroying the charges that were set up by t_1.

The problem now is to select those test pairs t_1 and t_2 which do not create a race condition that involves states outside C. This is done by selecting maximal cubes [13] in C that have non-empty intersection with t_2 and T_1. The two sets resulting from this intersection become the required robust test pair. A further requirement is needed, however, and this is that, depending on the particular maximal cube, some inputs may need to be free of glitches during the transition. Such requirements are directly obtained by looking at the maximal cube and are represented by setting a static hazard free (s.h.f.) flag for these inputs. The result is a set of triplets $<t_1, hf, t_2>$, where hf is a flag indicating the s.h.f. requirements, if any.

V. TEST PROPAGATION

After the test vectors are generated at the subcircuit level, the node values associated with a certain test or test pair are propagated forward and backward to the accessible external nodes of the circuit. We have developed a modification of the classical D-algorithm [13] that can handle multi-input multi-output subcircuits and which takes into account the robustness constraints by propagating two sets of test values simultaneously along with their s.h.f. requirements.

VI. IMPLEMENTATION AND EXAMPLES

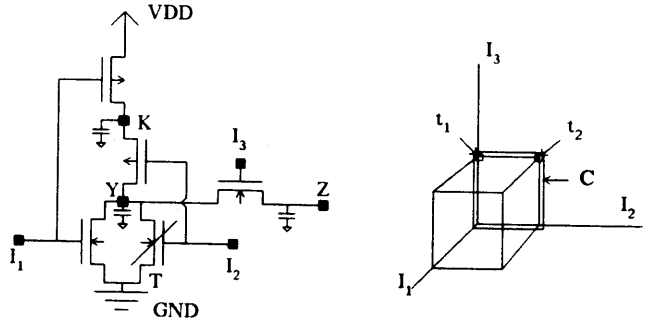
We have implemented the test generation algorithms in a computer program, called **Itest**, which accepts the input in the form of MOS transistor interconnections and fault lists (the program could be made to accept hierarchical circuit descriptions). It then automatically partitions the circuit, sets up an event scheduler and

derives the test sets. The program is written in the language C and is composed of $\simeq$ 5000 lines of code. Test vectors for many circuits with faults for which existing test generation methods failed have been correctly derived by Itest. In the following we present examples on how test generation at the subcircuit level is implemented.

We will represent an input pattern as composed of three fields within brackets:

[... / ... / ...]

The first field contains the values to be assigned to the subcircuit inputs (transistor gate labels and/or primary inputs). The second field shows the required internal node initial conditions (including the output node) to be set up in the GOOD circuit. The third shows these initial conditions required in the BAD circuit. The value "X" will be used to represent a don't care situation.



(a) (b)

Figure 3. CMOS circuit and its boolean input space.

Consider the circuit of Figure 3a . It has three inputs I_1, I_2, and I_3. The three internal (storage) nodes K, Y, and Z (which is the output node) are assumed to have the same node capacitances. The first (strongest) level of the node hierarchy consists of : { GND, VDD }, while the second (and last) level contains : { K, Y, Z }. The fault to be tested for is "transistor T stuck at open." The values inside the fields of an input pattern correspond to the circuit nodes as shown :

[I_1 I_2 I_3 / K Y Z / K Y Z]

By applying implication the following four sets are obtained :

G_0 = { [X11/XXX/XXX], [1X1/XXX/XXX], [XX0/XX0/XXX] }
G_1 = { [001/XXX/XXX], [XX0/XX1/XXX] }

B_0 = { [1X1/XXX/XXX], [011/XXX/X00], [XX0/XXX/XX0] }
B_1 = { [001/XXX/XXX], [011/XXX/X11], [XX0/XXX/XX1] }

T_2 is then constructed by forming the required intersections and unions:

$G_0 \cap B_1$ = { [011/XXX/X11], [XX0/XX0/XX1] }
$G_1 \cap B_0$ = { [XX0/XX1/XX0] }

T_2 = { [011/XXX/X11], [XX0/XX0/XX1], [XX0/XX1/XX0] }

Notice that the second and third patterns in T_2 require that the output node Z be initialized to two different values in the GOOD and BAD circuits. These two patterns are really useless because they require an initializing vector which is the same vector we are trying to derive. So they are automatically deleted from T_2 to give:

T_2 = { [011/XXX/X11] }

The next step is to derive the initializing input pattern to set up nodes Y and Z to "1" in the BAD circuit as required in the last field of t_2. By treating Y and Z as outputs and applying implication we get the initializing set :

T_1 = { [001/XXX/XXX] }

At the same time the allowable set of input states, C, during the transition from t_1 to t_2 is easily derived as a by-product :

C = { [0XX] }

The three dimensional boolean space is shown in Figure 3b, which shows the vertices corresponding to t_1 and t_2, and the set of states corresponding to C. An acceptable maximal cube turns out in this case to be [0XX] and includes all of C. This means that I_3 can have a glitch during the transition irrespective of the time at which I_2 makes the change; on the other hand, I_1 must not get a glitch and should be static hazard free (it is in the dimension in which the maximal cube does not extend). It is easy to see from the circuit that this is indeed required in order to preserve the charges at nodes Y and Z. Therefore the test triplet is (y = yes, n = no):

	I_1	I_2	I_3
t_1	0	0	1
t_2	0	1	1
hf	y	n	n

For another example reconsider the circuit in Figure 2. As shown previously, this circuit, when tested for the fault "T stuck at open," suffers from charge sharing and charge loss problems and cannot be tested by existing techniques. We will present the required Itest input files to describe that circuit and fault and will show the output of the program giving the valid test shown above in Figure 2. Furthermore, Itest reports that this is the only robust test and gives the required hazard free requirement at node 3.

The circuit specification file is as follows :

```
10   2   5   PMOS
10   1   5   PMOS
10   4   8   PMOS
 8   3   5   PMOS
 5   1   6   NMOS
 6   2   7   NMOS
 7   4   9   NMOS  1
 7   3   9   NMOS
V+  10
V-   9
INPUT  1  2  3  4
OUTPUT 5
```

The fault description file in this case is just the line :

OPEN 1

The output from Itest is :

```
*******************************************
TESTS FOR FAULT:
OPEN at element :      1 .

###########################################
# Test:                    <--in.   out  #
# --------------------------------------- #
# Nodes:     4 .  3 .  2 .  1 .     5    #
# --------------------------------------- #
# T1:        0 .  0 .  1 .  1 .          #
# --------------------------------------- #
# T2:        1 .  0 .  1 .  1 .   D_bar  #
# --------------------------------------- #
# HF:        N .  Y .  N .  N .          #
###########################################

*******************************************
```

Where "D_bar" (or $\overline{D}$) means : 0 in the fault-free circuit and 1 in the faulty circuit as in the D-algorithm.

VII. CONCLUSION

A program, Itest, has been presented which uses switch level techniques to derive accurate tests for faults in MOS circuits. The faults tested for include the classical stuck-at faults as well as the (non-classical) transistor stuck-at on or off, drain-to-gate and source-to-gate shorts, and a variety of bridging faults. The tests are guaranteed to be free from charge sharing and charge loss.

Itest is currently run on a VAX 11/780, a GOULD-9050, and a SUN 3/75M work-station. A run time of less than one second is typical for circuits like those given in this paper. For a CMOS 4-bit adder circuit with 200 transistors some faults take less than a second while others may take a minute, depending on the amount of work done in the propagation phase of the program. This computation time could be reduced by using techniques such as PODEM [14].

REFERENCES

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

[2] J. Galiay, Y. Crouzet, and M. Vergniault, "Physical versus logical fault models in MOS LSI circuits, impact on their testability," *IEEE 9th Fault Tolerant Computing Symposium*, pp. 195-202, June 20-22, 1979.

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

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

[5] S. Jain and V. Agrawal,"Test generation for MOS circuits using D-algorithm," *IEEE 20th Design Automation conference*, pp. 64-70, June 27-29, 1983.

[6] S. Robinson and J. Shen, "Towards a switch-level test pattern generation program," *IEEE International Conference on Computer-Aided Design*, pp. 39-41, November 18-21, 1985.

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

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

[9] R. Bryant,"A switch-level model and simulator for MOS digital systems," *IEEE Transactions on Computers*, vol. C-33, no. 2, pp. 160-177, February 1984.

[10] I. N. Hajj and D. Saab, "Fault modeling and logic simulation of MOS VLSI circuits based on logic expression extraction," *IEEE International Conference on CAD*, pp. 99-100, September 1983.

[11] Bernard Carré, *Graphs and Networks*. Oxford : Clarendon Press, 1979, p. 65.

[12] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146-160, June 1972.

[13] 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, October 1967.

[14] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Transactions on Computers*, vol. C-30, no. 3, pp. 215-222, March 1981.