

# TEST GENERATION FOR MOS CIRCUITS USING D-ALGORITHM

Sunil K. Jain  
Vishwani D. Agrawal

Bell Laboratories  
Murray Hill, New Jersey 07974

**ABSTRACT** — An application of the D-algorithm in generating tests for MOS circuit faults is described. The MOS circuits considered are combinational and acyclic but may contain transmission gates and buses. Tests are generated for both, the stuck type faults and the transistor faults (open and short). A logic model is derived for the MOS circuits. In addition to the conventional logic gates, a new type of modeling block is used to represent the "memory" state caused by the "open" transistors. Every fault, whether a stuck type fault or a transistor fault, is represented in the model as a stuck fault at a certain gate input. For generating tests, however, the D-algorithm needs modification. The singular cover and the D-cubes for the new gate include some memory states. To handle the memory state, an initialization procedure has been added to the consistency part of the D-algorithm. The procedure of modeling and test generation is finally extended to transmission gates and buses.

## INTRODUCTION

MOS circuits contain transistors interconnected through wires. A simple fault model should, therefore, include stuck faults (stuck-at-1 and stuck-at-0) on wires and switch faults (stuck-open and stuck-short) in transistors. Some recent workers [1-3] have reported test generation algorithms for transistor faults from the MOS device level description of the combinational circuit. Most of the conventional test generation procedures, however, require a gate-level description of the circuit. Notable among these is the D-algorithm [4].

This paper describes a procedure to convert a transistor structure into an equivalent logic gate structure. In order to model the memory state of a transistor, one new gate is defined. Every fault of the transistor circuit (whether it is a fault in wire or in transistor) is equivalent to a stuck fault in the logic structure. If the circuit is combinational, then the D-algorithm can be used for generating a test for any fault. Certain transistor faults, however, produce memory states. Handling of these faults requires some modifications of the D-algorithm. Basically, an initialization algorithm has been added to the consistency part of the D-algorithm.

An additional motivation for the present work is derived from some recent developments in fault simulation [5]. It is now possible to consider the transistor faults in the fault coverage analysis of tests. Particularly, for CMOS circuits, the coverage of transistor faults has been found to be significantly lower than that of the stuck type faults. One reason for this may be that the tests are often aimed at detecting stuck type faults only while not every transistor fault (e.g., stuck-open fault) can be mapped into a stuck-type fault. A test generation capability for transistor faults is, therefore, of significant value for CMOS technology.

## HOW DOES AN MOS GATE FUNCTION?

Figure 1 shows a simple CMOS gate. It contains two clusters of transistors — one nMOS cluster and the other pMOS cluster. All input signals are connected to the "gate" terminals of transistors. Depending upon the state (logical 0 or 1) of an input signal, the corresponding transistor behaves like an "open" or a "short." If the output is to be set "low" then a conducting path is created between the output and ground while all the paths between the output and VDD are opened. Similarly, the output is set "high" by creating a path to VDD, while all the paths to ground are blocked. This CMOS gate, does not pass on any of its input values but, in fact, regenerates a 1 from VDD or a 0 from ground. This function is represented by the switch model of Fig. 2. The operation of the switches  $S_0$  and  $S_1$  is a function of the inputs A, B and C. Either switch can be open or short. In a CMOS circuit, since  $S_0$  and  $S_1$  are complementary signals, only one switch is short at a time. The open switch, presents a floating (or high impedance) state to the output node. The output node behaves as a bus, on which the high impedance value is overridden by any other logic value. If, however, both inputs are in high impedance state, then the bus will retain its 'past' value. This state results from the charge retention capability of an isolated node in an MOS circuit. If both switches,  $S_0$  and  $S_1$ , are shorted then the output will be a 0 or a 1 depending upon the relative resistances of the load and the driver transistors in their conducting states.

## A MODEL FOR TEST GENERATION

Most test generation algorithms work on a logical description of the circuit. In these cases, the circuit should be modeled using the building blocks whose input/output behavior can be described logically (e.g., by truth tables). We will discuss a logic model for MOS transistor structures in terms of conventional logic gates, AND, OR, NOT, NAND, NOR, and one additional modeling block. The additional block is necessary for handling the high impedance (or memory) state and for eliminating VDD and ground from the logic model [6]. This modeling block is referred to as block 'B'.

The logic gate model for the MOS circuit of Fig. 1 is shown in Fig. 3. Simple rules, given in Table 1, explain how to first replace the transistors of each type (nMOS and pMOS) by logic gates to produce the switch functions  $S_1$  and  $S_0$ . Each of these functions is logic-1 if the switch is shorted and it is logic-0 if the switch is open. The output is then generated by combining  $S_1$  and  $S_0$  in the block B given in Table 1. Notice that the behavior of block B is not symmetric. The signal  $S_1$ , which is the output of the paths from VDD should be connected to the terminal marked as 1 while the output of the

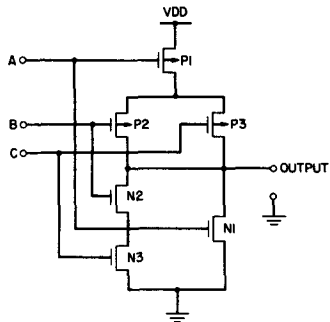


Fig. 1. A CMOS gate.

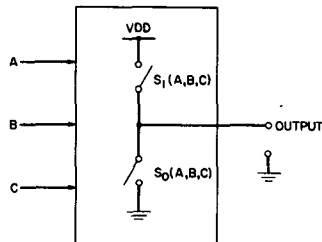


Fig. 2. A switch model for CMOS gate.

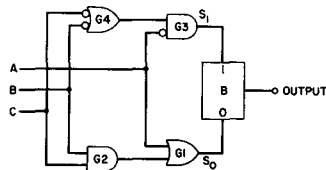


Fig. 3. A logic gate model for the CMOS circuit of Fig. 1.

ground paths,  $S_0$ , should be connected to the 0 terminal of block B. When both the inputs are 0, the output is shown as M. Here M refers to the memory or the previous state (before  $S_1$  and  $S_0$  changed to 0) of the output line. M can be any value among 0, 1 or "unknown." When both the inputs are 1, the output is shown as 0. It is assumed that the path from ground dominates over the path from VDD and hence, the output is pulled to the logic value of 0. If the assumption is not true for any particular technology, this value can be set to 1 or "unknown," as appropriate.

It is easy to verify that the logical behavior of the equivalent logic circuit in Fig. 3 is identical to that of the CMOS circuit in Fig. 1. A similar transformation of a nMOS circuit is shown in Fig. 4. Here the depletion mode transistor D1 is always "on" as it would require a negative voltage at its gate to be in the "off" state.

## REPRESENTATION OF FAULTS

We will consider two types of faults in the MOS circuits. These are the stuck faults (s-a-0 and s-a-1) at the inputs and outputs of the MOS gates and the transistor (short or open) faults. Table 2 gives a list of faults in the MOS circuit of Fig. 1 and their corresponding equivalent faults in logic circuit of Fig. 3. Every fault in the MOS circuit has an equivalent stuck type fault in the logic representation. The input/output stuck faults in MOS circuit correspond to identical stuck faults in the logic model. A transistor fault in the MOS circuit corresponds to a stuck fault on that input lead of the logic gate which is the replacement for the transistor (see Table 1). Thus transistor P1 short

TABLE 1 MOS TO LOGIC TRANSFORMATION	
MOS	LOGIC
GATE TERMINAL INPUT TO NMOS TRANSISTOR	INPUT TO A LOGIC GATE
GATE TERMINAL INPUT TO PMOS TRANSISTOR	INVERTED INPUT TO A LOGIC GATE
DEPLETION LOAD	LOGIC 1
SERIES ELEMENTS	AND GATE
PARALLEL ELEMENTS	OR GATE
MOS GATE OUTPUT	

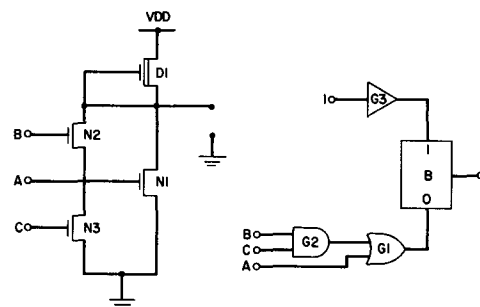


Fig. 4. An nMOS gate and its logic model.

TABLE 2 EQUIVALENT FAULTS	
Fault in MOS circuit (Fig. 1)	Fault in Logic circuit (Fig. 3)
A stuck at j*	A stuck at j
B stuck at j	B stuck at j
C stuck at j	C stuck at j
output stuck at j	output stuck at j
P1 short	G3, input A stuck at 0
P1 open	G3, input A stuck at 1
P2 short	G4, input B stuck at 0
P2 open	G4, input B stuck at 1
P3 short	G4, input C stuck at 0
P3 open	G4, input C stuck at 1
N1 short	G1, input A stuck at 1
N1 open	G1, input A stuck at 0
N2 short	G2, input B stuck at 1
N2 open	G2, input B stuck at 0
N3 short	G2, input C stuck at 1
N3 open	G2, input C stuck at 0

\* j can be 0 or 1.

corresponds to input A of gate G3 stuck-at-0, or transistor P1-OPEN corresponds to the input A of gate G3 stuck-at-1.

The faults in the depletion mode transistors used in nMOS circuits need special mention. In Fig. 4, the fault "transistor D1-SHORT" corresponds to the input of G3 stuck-at-1. Since in our logic model the input of G3 is permanently set to a 1, this fault is undetectable. Indeed this fault is also undetectable in the MOS gate. The fault D1-OPEN, on the other hand, corresponds to the input of G3 stuck-at-0 and this

is detectable in the MOS circuit as well as in our logic model. In general, tests need to be generated for only one fault among those that collapse together.

### APPLICATION OF D-ALGORITHM

The basic D-algorithm as described in the literature [4] requires every gate to be described by its singular covers and the D-cubes. The only additional gate that has been introduced here is the MOS output modeling block B, as shown in Table 1. The singular cover and D-cubes for this block are given in Fig. 5. Singular cover is used for determining the inputs (during consistency operation) or determining the output (during forward implication) when either the output or some of the inputs of a gate are 0 or 1. D-cubes are used for representing the faults at the site of the faulty gate and for propagating the 'D' (during the D-drive).

A D-cube is obtained by combining two rows of the truth table. For example, the first two rows of the truth table of the MOS output gate produce the following D-cube:

$$S_1 = 1, \quad S_0 = D, \quad \text{OUTPUT} = \bar{D}$$

By substituting  $D = 1$ , in this D-cube we get the first row of the truth table and  $D = 0$  leads to the second row. If we remember that the memory state M is the previous state (0 or 1) of the output line, then its value for this D-cube is "don't care." This is shown as the first D-cube in Fig. 5. The first four D-cubes do not involve any M state. The remaining D-cubes have the values of M specified. For example, consider the fifth D-cube which is obtained by combining the following rows of the truth table:

$S_1$	$S_0$	OUTPUT
0	1	0
0	0	M

If the previous state M of the output was 0 then the output will be 0 for any value of  $S_0$ . This state will, therefore, lead to the following cube:

$S_1$	$S_0$	OUTPUT
0	D	0

which does not propagate the D to the output. If, however,  $M = 1$ , then the D on  $S_0$  is propagated to the output as  $\bar{D}$ . This is shown as the fifth D-cube in Fig. 5 where the corresponding value of M is shown in the parenthesis. This assumes that if the output of the gate can be *initialized* to a 1 in both the faulty and the fault-free circuits then a 0 on  $S_1$  and a D on  $S_0$  will produce a  $\bar{D}$  at the output. Other D-cubes in Fig. 5 were obtained in a similar fashion. In our notation,  $D(X)$  or  $\bar{D}(X)$ , (where X in parenthesis means that initialization at the output is not needed) are written simply as D or  $\bar{D}$ .

The test generation for any given fault can now proceed in the conventional manner. First a D or a  $\bar{D}$  is placed at the fault site. D or  $\bar{D}$  assume the logic values 1 or 0, respectively, in the fault-free circuit and the complementary values in the faulty circuit. Consistency, implications and D-drive procedures [4] are repeatedly applied until a D or a  $\bar{D}$  arrives at a primary output. In addition, a back-up (back tracking) procedure in case of an inconsistency or a premature disappearance of all D and  $\bar{D}$  guarantees a test if one is possible. Whenever a D-drive is carried through an MOS output gate (B), the associated memory state of the particular D-cube is checked. If this memory state is 0 or 1 then the following consistency operation includes an additional procedure of *initialization*.

SINGULAR COVER		
$S_1$	$S_0$	OUTPUT
X	1	0
1	0	1
0	0	M

D-CUBES		
$S_1$	$S_0$	OUTPUT
1	D	$\bar{D}$
1	$\bar{D}$	D
D	$\bar{D}$	D
$\bar{D}$	D	$\bar{D}$
0	0	$\bar{D}(1)$
0	$\bar{D}$	$D(1)$
D	0	$D(0)$
$\bar{D}$	0	$\bar{D}(0)$
D	D	$\bar{D}(1)$
$\bar{D}$	$\bar{D}$	$D(1)$

Fig. 5. Singular cover and D-cubes for the MOS output gate B.

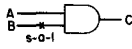
### THE INITIALIZATION ALGORITHM

The initialization algorithm described here has been derived from the D-algorithm [4]. It differs from the D-algorithm in two aspects. First, D-drive is not used and second, a modified singular cover table is used for the faulty gate. To illustrate this modification in the singular cover table, consider an AND gate with one of the inputs (B), stuck-at-1 as shown in Fig. 6. Since the initialization requires that both, the faulty and the fault-free circuits behave identically, the faulty line should not be set to a logic value 0 during the initialization. All singular cubes that attempt to set the input B to a 0 are, therefore, deleted from the singular cover table. In this case, the modified table, as shown in Fig. 6, contains two cubes.

For the initialization, a second copy of the circuit is used so that the states of the circuit on which the D-algorithm is running are not disturbed. The initialization procedure begins by placing unknown or "don't care" values on all lines in this copy of the circuit. The required initialization value (0 or 1) is then placed at the line to be initialized. Consistency and implication procedures are repeatedly applied to all gates until no more changes in line values are necessary. These procedures involve intersection, as defined in [4], of the singular cubes with the existing line-values. Singular cubes are the rows of the singular cover. In the case of an inconsistency, the usual procedure of back-up to the last available choice of an unused singular cube is employed. If the procedure backs up to the very line which needs initialization and no more singular cubes are available, then this initialization is considered impossible and the back up is carried through to the previous D-drive where the memory state was generated. Hence, for a transistor fault, one test pattern is sufficient for initialization and a second test pattern for detecting the fault. These two pattern tests are sufficient to test any transistor fault in a circuit, which does not have memory in the fault-free state. For stuck faults on wires, only one test pattern is sufficient.

### EXAMPLE 1

The circuit shown in Fig. 7 will be used to illustrate the test generation algorithm. This circuit consists of three CMOS gates. It is converted into an equivalent logic gate model by using the rules given in Table 1. The logic gate model for the circuit is shown in Fig. 8. Consider the fault, "transistor T5-OPEN" in the MOS circuit. This fault is equivalent to the fault "line 1b stuck-at-0" in the logic circuit of Fig. 8. The step-by-step procedure of generating a test for this fault is as follows:

		
SINGULAR COVER		
A	B	C
0	X	0
X	0	0
1	1	1

SINGULAR COVER WITH FAULT, B S-A-1		
A	B	C
0	X	0
1	1	1

Fig. 6. Modified singular cover of an AND gate used for initialization.

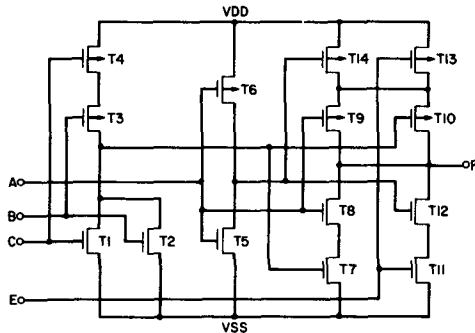


Fig. 7. CMOS circuit for Example 1.

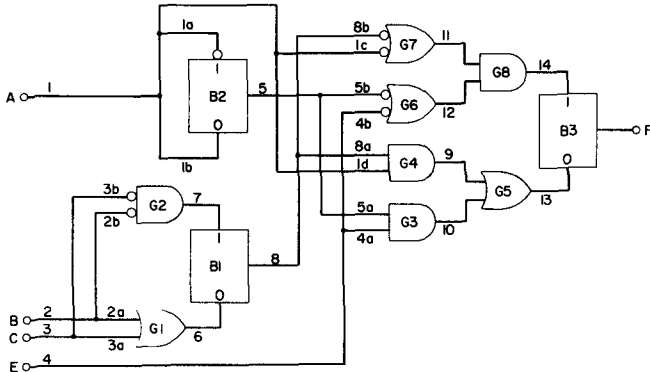


Fig. 8. Logic model for the circuit of Fig. 7.

- Step 1: Since the fault is "line 1b stuck-at-0," this line is set to 1. This sets the primary input  $A = 1$  and line  $1a = 1$ . To represent the fault we rewrite, line  $1b = D$ . No further implications are necessary. All other lines have unknown values represented by X.
- Step 2: D-drive through the output gate B2. Using the fifth D-cube from Fig. 5,  $\bar{1}a = 0$ ,  $1b = D$ ,  $5 = \bar{D}(1)$ , and line 5 is to be initialized to 1.
- Step 3: Initialization. Assume all lines at value X. Place a 1 on line 5. For justifying this value, we need the modified singular cover for gate B2. Due to the fault (stuck at 0) on line  $1b$ , this singular cover consists of only the last two rows of the singular cover given in Fig. 5. Using the second row, we get,  $\bar{1}a = 1$ ,  $1b = 0$ ,  $5 = 1$ . No further implications are necessary. Save initialization pattern,  $A = 0$ ,  $B = C = E = X$  and restore the logic values on various lines to those that existed at the end of Step 2.
- Step 4: D-drive through gate G3.  $5a = \bar{D}$ , set  $4a = 1$ ,  $10 = \bar{D}$ . This sets primary input line E to 1.
- Step 5: D-drive through gate G6. Since  $\bar{4}b = 0$ ,  $5b = D$ ,  $12 = D$ .
- Step 6: D-drive through gate G5. Set line 9 to 0 so that  $13 = \bar{D}$ .
- Step 7: Consistency check for gate G4.  $9 = 0$  and  $1C = 1$  requires  $8a = 0$ . This further requires  $6 = 1$ ,  $7 = 0$ ,  $B = 1$ ,  $11 = 1$ .

Step 8: D-drive through G8. Since  $12 = D$ ,  $11 = 1$ ,  $14 = D$ .

Step 9: D-drive through B3. Using the third D-cube in Fig. 5, we get  $14 = D$ ,  $13 = \bar{D}$ ,  $15 = D$ . Thus the test is  $A = 1$ ,  $B = 1$ ,  $C = X$ ,  $E = 1$ . This test should be preceded by the initialization pattern found in Step 3.

The logic gate model for the CMOS circuit can also be used to generate tests for stuck faults at the inputs and outputs of the gates. As an illustration, we generate a test to detect the fault, "input B stuck-at-1." This fault is equivalent to the "line 2 stuck-at-1" in the logic model of Fig. 8. We proceed as follows:

Step 1: The initial test cube for the fault line 2 stuck at 1 is: line 2 = 0, line 3 = 0 and line 6 =  $\bar{D}$ .

Step 2: To justify the logic value 0 on line 2 and logic value 0 on line 3 requires:  $B = 0$  and  $C = 0$ .

Step 3: The D-drive of  $\bar{D}$  from line 6 to the output line 15 requires the following values on the inputs:  $A = 1$  and  $E = X$ .

Hence, the test requires the logic values 100X on the primary inputs A, B, C and E, respectively.

### EXAMPLE 2

Tests were generated for all stuck and transistor faults in a two-bit adder. A CMOS circuit implementing this two-bit adder is shown in Fig. 9 and the corresponding logic gate model is shown in Fig. 10. A test set for all stuck faults and transistor faults was manually written. Transistor faults were first considered. For the first fault, T17-OPEN, the test consisted of two patterns:

A0	B0	C0	A1	B1
1	1	0	X	X
0	1	0	X	X

Upon fault simulation [5] of the circuit of Fig. 9, it was found that these patterns also detected the faults T13-SHORT, T16-OPEN, T1-SHORT, T9-OPEN and T8-OPEN. The next fault for which a test was then generated was T3-OPEN. In this way 38 patterns were generated to cover the 52 transistor faults. These patterns are shown in Table 3. From the fault simulation of the MOS circuit it was found that all of the 48 stuck type faults were already covered by these patterns. Therefore, no more patterns were generated for them. It should be noted that the number (38) of test patterns is greater than the total number ( $2^5 = 32$ ) of possible input combinations. This is due to the sequential behavior of the circuit in the presence of certain transistor faults.

### MODEL FOR A TRANSMISSION GATE

A transmission gate functions as a switch with a charge retention (memory) capability in its OFF state. Even though the transmission gates are bidirectional devices they are often embedded between unidirectional gates and thus used as unidirectional elements whose function can be represented by a truth table (Fig. 11). Functionally this transmission gate can be modeled using the conventional logic gates and the new output gate "B", introduced in Table 1. The logic model for the transmission gate is shown in Fig. 12. It is easy to verify that the logical behavior of the circuit in Fig. 12 is identical to that of the transmission gate represented in Fig. 11.

In a transmission gate, the transistor -SHORT or -OPEN faults produce the same effects as the CLOCK input stuck-at-1 or stuck-at-0, respectively. Hence, modeling faults at

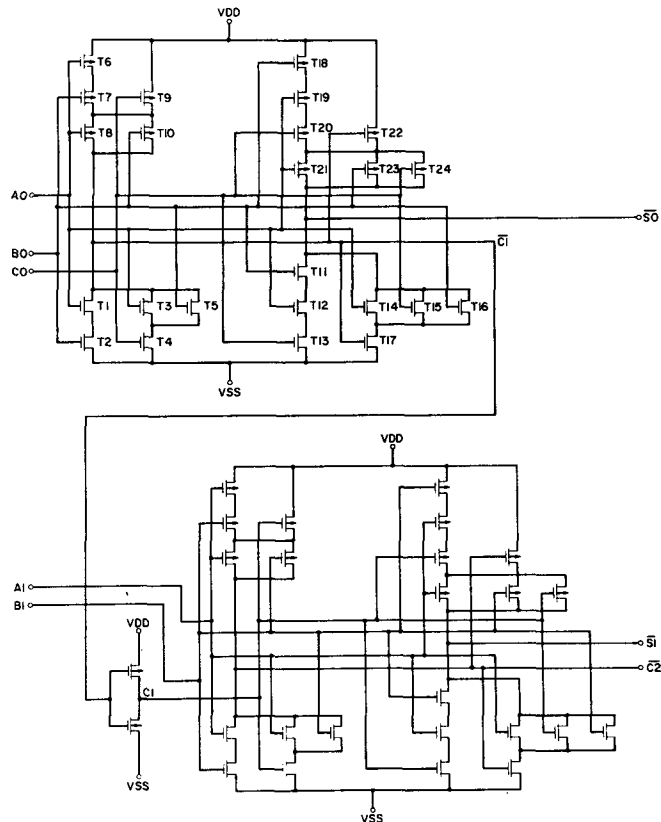


Fig. 9. CMOS implementation of the two-bit adder used in Example 2.

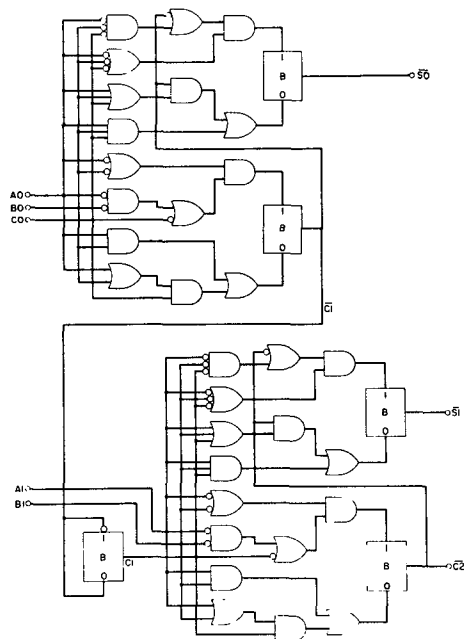


Fig. 10. Logic gate model for the circuit of Fig. 9.

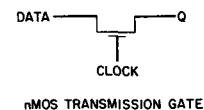
transmission gate inputs and output pins is sufficient. Stuck faults on input and output lines of the transmission gate correspond to stuck faults on the inputs and output lines in the model of Fig. 12. The test generation for any given fault can be done in a manner discussed in the previous section.

#### Example:

The MOS circuit shown in Fig. 13 is used to illustrate the test-generation procedure for a circuit containing a

TABLE 3  
Test Patterns for two-bit adder

Pattern No.	A0	B0	C0	A1	B1
1	1	1	0	X	X
2	0	1	0	X	X
3	1	1	1	X	X
4	1	0	0	X	X
5	1	0	1	X	X
6	0	0	1	X	X
7	0	1	0	X	X
8	1	1	0	X	X
9	0	0	1	X	X
10	0	1	1	X	X
11	0	0	1	X	X
12	0	0	0	X	X
13	0	0	0	X	X
14	0	1	0	X	X
15	1	1	0	X	X
16	1	1	1	X	X
17	0	0	1	X	X
18	0	1	1	1	0
19	0	1	1	X	X
20	0	0	1	1	0
21	1	1	1	1	1
22	1	0	0	0	1
23	1	1	X	0	1
24	1	1	X	0	0
25	0	X	0	0	1
26	0	X	0	1	1
27	1	1	X	0	0
28	0	0	X	0	0
29	X	0	0	1	0
30	1	X	1	1	0
31	0	0	X	0	0
32	X	0	0	1	0
33	X	1	1	0	1
34	0	X	0	0	1
35	0	X	0	1	1
36	1	1	X	0	0
37	1	1	X	1	0
38	1	1	X	1	1



TRUTH TABLE		
DATA	CLOCK	Q
0	1	0
1	1	1
0	0	M
1	0	M

Fig. 11. Transmission gate and its truth table representation as an unidirectional gate.

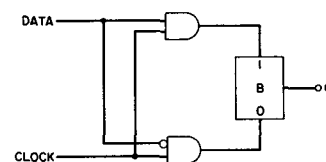


Fig. 12. Logic model for transmission gate.

transmission gate. Let us generate a test to detect the fault "transistor T3-SHORT" (or input line C stuck-at-1). This fault is equivalent to the fault "line 2 stuck-at-1" in the logic model shown in Fig. 14. We proceed as follows:

- Step 1: The initial test cube for fault line 2 stuck-at-1 is: line 2 = 0, line 4b = 1, line 5 = D.
- Step 2: To justify the above values on lines 2 and 4b requires: C = 0 and A = 0
- Step 3: The D-drive of D through output gate B2 is accomplished using the eighth D-cube from Fig. 5.  
line 5 = D line 6 = 0 line 7 = D (0)
- Step 4: Line 7 has to be initialized to 0 in presence of the fault. The initialization requires following values on the inputs: A = 1, C = 1 and B = X.

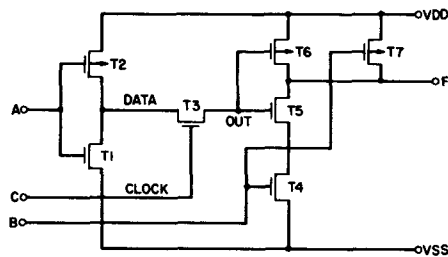


Fig. 13. Circuit used as example of test generation for transmission gate faults.

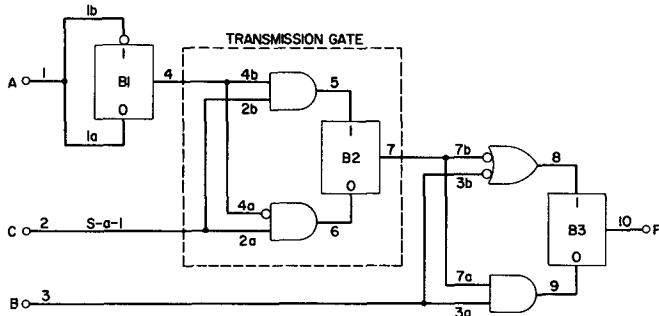


Fig. 14. Logic model for the circuit of Fig. 13.

Step 5: Return to the states at the end of Step 3. Driving  $\bar{D}$  from line 7 to the output F only requires setting the input B to 1.

Hence, the test requires two vectors. The initialization vector sets the values 1X1 on the primary inputs A, B and C, respectively. The second test vector then changes the input values to 010.

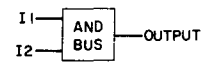
## A MODEL FOR BUS

The model for a "bus" structure is dependent upon the technology. The bus structure in logic circuits is either modeled as Tied-AND or Tied-OR. We will discuss the Tied-AND structure and the reader can easily adapt the model for the Tied-OR structure.

A bus structure is a physical connection of outputs of two or more gates. Hence, all the gates that are tied to a bus should be considered as a family. In order to ascertain the logical value of the bus all the members of this family should be examined. If this is done, then there will be no need to consider the bus separately. The main problem with this approach is that one will have to write D-cubes separately for each bus connection. For simplicity, therefore, we will model the bus as a separate gate.

The inputs to the bus, i.e., the outputs of gates feeding the bus, in a fault-free MOS circuit, can have three states (logical 0, 1 or high impedance). When all the inputs to the bus are in high impedance state, then the bus retains its past value. In addition, we will consider the bus as an unidirectional gate. The high impedance state at an input is represented as "M." Just like logic values 0 and 1 this state can also be justified in certain cases. The truth-table for a two input AND-bus is given in Fig. 15. Singular cover and the D-cubes for the AND-bus are given in Figs. 15 and 16, respectively. Similarly, singular cover and D-cubes can be written for the AND-bus having more than two inputs.

A few things, that are special to the handling of buses, should be noted here. First, if a gate, whose output feeds into a bus, is to be initialized to a certain logic value, then initialization should be done after propagating the D-Drive through the bus. For example, consider the fourth D-cube in



I1	I2	OUTPUT
1	0	0
1	1	1
1	M	1
0	0	0
0	1	0
0	M	0
M	0	0
M	1	1
M	M	M

SINGULAR COVER		
I1	I2	OUTPUT
0	X	0
X	0	0
1	1	1
1	M	1
M	1	1
M	M	M

Fig. 15. Truth table model of a two-input AND-BUS and the corresponding singular cover.

I1	I2	OUTPUT
1	D	D
1	$\bar{D}$	$\bar{D}$
1	$\bar{D}(0)$	$\bar{D}$
1	$\bar{D}(1)$	$\bar{D}$
M	D	D
M	$\bar{D}$	$\bar{D}$
M	D(0)	D(0)
M	D(1)	D(1)
M	$\bar{D}(0)$	D(0)
M	$\bar{D}(1)$	D(1)
D	1	D
D	M	D
$\bar{D}$	1	$\bar{D}$
$\bar{D}$	M	$\bar{D}$
D(0)	M	D(0)
D(1)	M	D(1)
$\bar{D}(0)$	1	$\bar{D}$
$\bar{D}(1)$	1	$\bar{D}$
$\bar{D}(0)$	M	$\bar{D}(0)$
$\bar{D}(1)$	M	$\bar{D}(1)$
D	D	D
$\bar{D}$	$\bar{D}$	$\bar{D}$

Fig. 16. D-cubes for the two-input AND-BUS of Fig. 15.

Fig. 16. Here, the gate whose output is I2 is to be initialized to the value of 1 during the test. Instead of performing the initialization, if we use this D-cube for D-drive, the output of the bus will be set to a  $\bar{D}$  and no initialization will be needed, provided a 1 can be justified for I1. If, however, a 1 is not possible on I1, then the tenth D-cube will be used and the output of the bus will have to be initialized to a 1. Second, some D-cubes are not considered because they will never be needed under the single fault assumption. For example, it is not possible that two gates whose outputs are bused, would require a simultaneous initialization during the test generation.

## Example:

Consider the simple MOS circuit shown in Fig. 17. The logic gate model for test-generation is shown in Fig. 18. Let us generate a test to detect the fault, "transistor T2-OPEN (or control input, B of transmission gate T2 stuck-at-0)." This fault is equivalent to the fault "line 2 stuck-at-0" in the logic gate model of Fig. 18. Test generation proceeds as follows:

Step 1: Primary input B is set to 1; this produces a D on 2a and 2b. By considering the D-cubes for logic gate G2, we get:  $2b = D$ ,  $1b = 0$  and  $6 = D$ . This also sets  $A = 0$ .

Step 2: Forward implication at G1:  $1a = 0$  implies line 5 = 0.

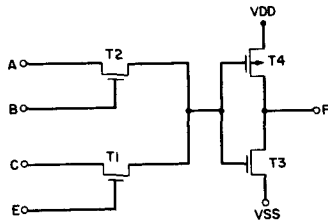


Fig. 17. MOS circuit used in test generation example involving a bus.

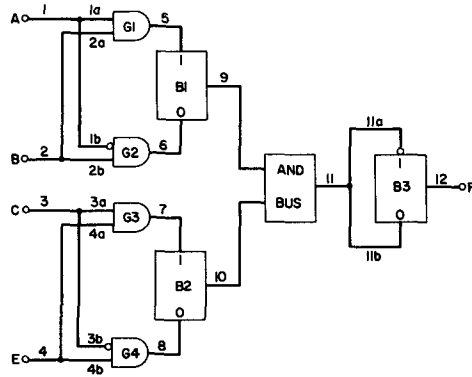


Fig. 18. Logic model for the circuit of Fig. 17.

- Step 3: D-drive through B1: line 5 = 0, line 6 = D, line 9 =  $\bar{D}(1)$ . Since line 9 is an input to bus, the initialization is to be done after D-drive through the bus.
- Step 4: D-drive through AND-bus: line 9 =  $\bar{D}(1)$ , line 10 = 1 and line 11 =  $\bar{D}$ . Thus, no, initialization is needed. [Note: If we had chosen a different D-cube, for example, line 9 =  $\bar{D}(1)$ , line 10 = M, and line 11 =  $\bar{D}$ , then, the initialization at the bus would be necessary.]
- Step 5: Consistency: To justify the value 1 on line 10 we set C = 1 and E = 1. Hence, the test requires the values 0111 on the primary inputs A, B, C and E, respectively.

### EFFECT OF GATE-DELAYS ON TESTS

The test generation procedures described above treat MOS devices as ideal gates, which are described by their logical behavior alone. This description is sufficient for one-pattern, purely combinational, tests. In general, however, transistor faults may require two-pattern tests, where the first pattern is used for initialization. At the transition from the first pattern to the second pattern, the test generator assumes that all logical values on various lines in the circuit change simultaneously. In practice, the delays in devices and lines will cause the line values to change sequentially. This can result in multiple transitions which, in turn, can change the initialization. As an example, consider the fault P2-OPEN in the circuit of Fig. 1. In the logic model of Fig. 3, this fault is represented as "input B of G4 stuck-at-1." Consider the test A = 0, B = 0, C = 1 with the initialization pattern 100. Let us assume that each gate, with the exception of the "B" gate, in the model of Fig. 3 has one unit of delay. Then a transition at the circuit input will propagate to the output in two units of time. In Fig. 19 these transitions are shown for each line as  $i \rightarrow j \rightarrow k \rightarrow \ell$ , where the first transition  $i \rightarrow j$  takes place at time zero, second transition  $j \rightarrow k$  occurs at time = 1 unit and final transition  $k \rightarrow \ell$  occurs at time = 2 units. Notice that at the output of gate G3, two transitions are occurring. The intermediate transition  $0 \rightarrow 1$  causes the output initialization to change to 1 from the original 0 that was required by D(0). Thus the fault will remain undetected. This phenomenon can be traced in the CMOS circuit of Fig. 1 also. The initialization

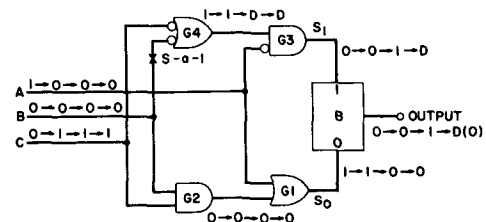


Fig. 19. Propagation of transitions of a two-pattern test.

pattern shorts N1 to set OUTPUT to 0. It also opens P1 and shorts P2 (only in fault-free circuit) and P3. Transition  $1 \rightarrow 0$  at A opens N1 and shorts P1. If the transition  $0 \rightarrow 1$  at C is slightly delayed, OUTPUT will be changed to 1 before P3 is opened. Now whether P2 is short (fault-free) or open (faulty), OUTPUT will be 1. However, if transition  $0 \rightarrow 1$  at C had occurred before the transition  $1 \rightarrow 0$  at A, then OUTPUT remains at 0 and the fault will be detected. It can be easily verified that an initialization pattern 011 would have detected the fault irrespective of the circuit delays. In this case only one input of G3 changes and therefore, the hazard-like transition at its output is avoided. During the generation of a test pattern, the possibilities of such hazards can be reduced by so choosing the D-cubes and singular cubes that the Hamming distance between the individual gate inputs for two-pattern tests is minimized.

### CONCLUSION

A logic model which allows test generation for faults in MOS circuits is described. The algorithms have been illustrated by examples. An implementation of these algorithms should not be much more complex than the conventional D-algorithm as the added initialization phase requires only the basic features of the D-algorithm. Past experience has demonstrated the usefulness of the D-algorithm in generating tests and in detecting redundancies. Also the technology dependent faults (particularly in CMOS circuits) are beginning to cause testing-related concerns. It is believed that the present work will serve a useful purpose.

### ACKNOWLEDGMENT

Authors acknowledge cooperation from A. K. Bose and M. R. Mercer.

### REFERENCES

- [1] Y. M. El-ziq and R. J. Cloutier, "Functional-Level Test Generation for Stuck-Open Faults in CMOS VLSI," IEEE International Test Conference, Philadelphia, PA, October 27-29, 1981, *Digest of Papers*, pp. 536-546.
- [2] K. W. Chiang and Z. G. Vranesic, "Test Generation for MOS Complex Gate Networks," 12th International Symposium on Fault-Tolerant Computing, Santa Monica, CA, June 22-24, 1982, *Digest of Papers*, pp. 149-157.
- [3] Y. H. Levendel and P. R. Menon, private communication.
- [4] 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, October 1967, pp. 547-580.
- [5] A. K. Bose, P. Kozak, C-Y Lo, H. N. Nham, E. Pacas-Skewes, and K. Wu, "A Fault Simulator for MOS LSI Circuits," *Proceedings of 19th Design Automation Conference*, Las Vegas, Nevada, June 14-16, 1982, pp. 400-409.
- [6] Y. H. Levendel, P. R. Menon and C. E. Miller, "Accurate Logic Simulation Models for TTL Totem-pole and MOS Gates and Tristate Devices," *Bell System Technical Journal*, Vol. 60, September 1981, pp. 1271-1287.