YO883-0876 RRS P100 02-800
J. F. Beetem, V. G. Oklobdzija and J. P. Roth

IBM Technical Disclosure Bulletin Vol. 28 No. 2 July 1985

645

TEST GENERATION FOR FET SWITCHING CIRCUITS

85A061645

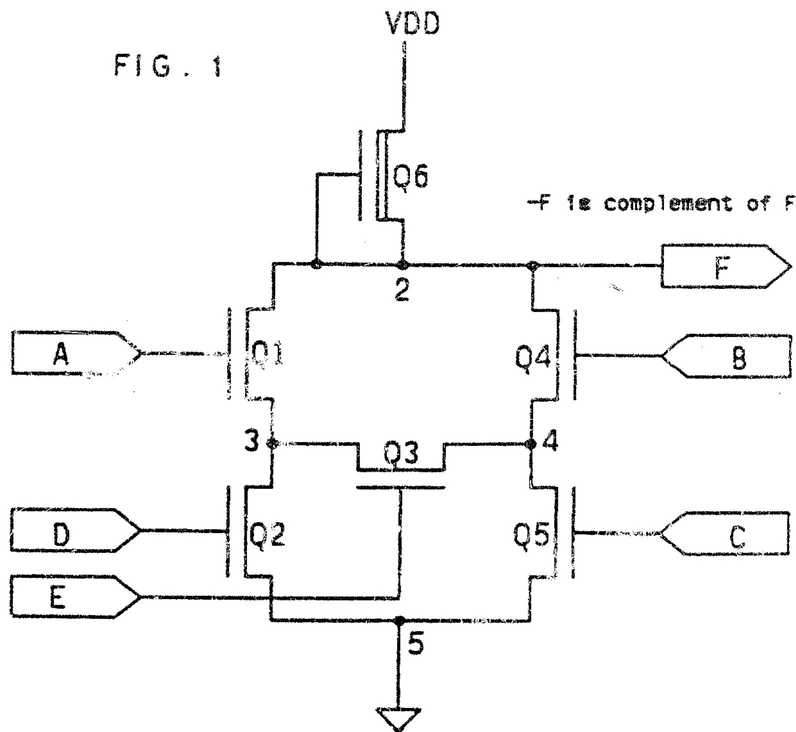

FIG. 2

a	b	c	d	e
D	0	1	1	1
0	D	1	1	1
1	1	D	0	1
1	1	0	D	1
0	1	0	1	D

FIG. 3

a	b	c	d	e
0	0	1	1	1
0	1	0	1	0
0	1	0	1	1
0	1	1	1	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1

As the complexity of VLSI circuits continues to increase, the need to test for failures has become imperative. The problem of generating test to detect failures on FET transistor networks has been unsolved for all but simple cases. In this article we describe an effective method for computing test for failures for FET switching networks. Here we define a function-preserving, failure-preserving transformation

of a switching network, into a logic network. There are efficient means for computing tests for failures in logic networks, specifically, the D-algorithm. Tests so computed for the image logic network are automatically tests for failures in the original switching network. In other words the logic network so generated not only computes the same function; it also has the same failure-structure. Tests for the logic-network failures are computed by efficient test-generation procedures, using the D-algorithm [*]. It is proven for the transformation that a test for a stuck failure in logic network is simultaneously a test for a short (or open) for the corresponding switch in the switching network. Run time for the transformation, from switching to logic network, increases linearly with the complexity of the switching network, so that its run time can be neglected.

Of the means of realization of function, logic circuits and switching circuits, logic circuits are much easier to test, with, e.g., the use of the D-algorithm: direct methods for switching circuits are semi-exhaustive. However, the test for VLSI logic based on the logic representation without regard for the actual FET implementation is proven to be inadequate.

We define a structural mapping of each switching circuit into a logic circuit, performing the same function and having the same failure characteristics. This mapping has linear complexity. We then compute a test assemblage for this constructed logic-circuit image which is guaranteed to detect all of its (stuck) failures. We prove that these same tests cover all transistor failures in the original switching circuits. We use some new techniques to generate a minimal number of tests to cover these failures. Thus, the testing of switching circuits is reduced to the testing of logic circuits.

Definitions

A logic circuit is a directed acyclic graph: on each branch is a binary function of the branches directed toward it; branches with no branches directed toward it are assigned binary variables; branches pointing nowhere are primary outputs; in addition, any branch may be designated as a primary output.

A switching circuit is defined by a graph whose branches are of two types. In one type, there is a switch which is open or closed depending upon the value of a binary variable controlling it. In the other is an inverter, which inverts the value, 1 or 0, of the signal on the input side of the inverter. In general, an inverter is never inserted in parallel with switches, and only at a point where the graph necks down to one branch. The switches are bilateral devices, whereas the inverters are unilateral. The switching circuit has two terminal nodes: ground and input side of the inverter. Therefore, the existence of a path connecting the terminal nodes in the switching circuit means logical zero value at the inverter input, i.e., logical one at the inverter output.

TEST GENERATION FOR FET SWITCHING CIRCUITS

Direct computation of tests for failures in switching circuits has been difficult. Here, an algorithm is given for "failure-equivalent" mapping of switching circuits into logic circuits: it is an iterative and structural mapping. We prove that tests covering stuck failures in the image logic network also constitute tests for shorts and opens in the original switching circuit. Shorts and opens of switches in switching circuits are in one-to-one correspondence with stuck failures in equivalent logic circuits.

Example of Structural Mapping

Consider the bridge circuit consisting of four nodes 2,3,4,5 and five switches interconnected as follows (Fig. 1).

The following notation is used to define a switch:

A switch with vertices a,b controlled by variable v is denoted v v<a,b> . A switching circuit is defined by a collection of such switches in such notation; the description must also specify the (sets of) terminal nodes. The function realized by such a switching circuit is defined by:

a<2,3>,b<2,4>,e<3,4>,c<4,5>,d<3,5>.

Assume that 2 and 5 are terminal pair. We start arbitrarily with 2. The switches contiguous to 2 are a<2,3> and b<2,4>. The first labelled function is

$$F_1 = a<2,3>+b<2,4>$$

The boundaries of this chain are 3 and 4 and their cofaces are

$$D^*(3)=e<3,4>+d<3,5>$$

$$D^*(4)=e<4,3>+c<4,5>$$

Those already in the chain, a<3,2> and b<4,2>, are not in the cofaces. These cofaces are multiplied into the above expression to produce

$$F_2 = a<2,3>(e<3,4>+d<3,5>)+b<2,4>(e<4,3>+c<4,5>)$$

The boundary points of F2 are 4 and 3 (5 being a terminal node). Thus, the terms having 4 and 3 as boundaries are multiplied by their surviving cofaces, namely, c<4,5> and d<3,5>:

$$F_3 = a<2,3>(e<3,4>(c<4,5>)+d<3,5>)+b<2,4>(e<4,3>(d<3,5>)+c<4,5>)$$

This is the final structural chain. The corresponding algebraic chain is:

$$F=a(ec+d)+b(ed+c)$$

a(ec+d)+b(ed+c)

Table I represents the Boolean tree, translated by the procedure from the above switching circuit (e.g., CMOS), in the notation of 'RLD' (regular logic design): the primary inputs E D A B C are given integer labels 2 3 4 5 6; then the variable designated 7 in the Boolean tree formed is specified to be the AND-Invert (AI) of arguments 3 2 5, etc.; the primary output F in the CMOS design is here designated as 12.

TABLE I

N Channel Boolean logic converted from BRIDGE IL1

```
E
1  2=PI
D
1  3=PI
A
1  4=PI
B
1  5=PI
C
1  6=PI
1  7=AI   3  2  5
1  8=AI   6  5
1  9=AI   6  2  4
1 10=AI   3  4
1 11=AND 10  9  8  7
F
1 12=PO  11
```

Computation by the D-algorithm yields the following tests for primary inputs of the generated logic circuit (Fig. 2).

These represent tests both for each primary-input line stuck-at-1 and -0, for the six input lines a,b,c,d,e in order. Clearly, more reduction could be achieved; for example, the test cubes for c and d can be interfaced to produce a single test for c or d stuck-at-1.

Referring to the original switching circuit, we see that D0111 is a test for switch a<2,3> stuck either way, for b=0 blocks either path between the terminals. Likewise 0D111 allows only a path through b between terminals, checking failure both ways on b< 2,4> . Similarly, 11D01 blocks all paths except those through c<4,5>.

It is clear that the number of tests needed to test the switches in the given example is less than what was obtained by using our pro-

cedure for test generation for FET switching circuits via failure-preserving transformation to logic circuits (Fig. 3).

However, our objective is not to produce the minimal test but to assure the coverage of "transistor-faults". In this case the excess test vectors might be beneficial for the coverage of the physical defects not covered by the transistor faults.

Test generation for failures in FET-switching circuits, for small number of FETs, has been possible only by manual methods: -the combinatorics grows exponentially with the size of the circuit for direct approaches. For logic circuits, on the other hand, extant methods are satisfactory and widely practiced. In this article we define a transformation from switching-to-logic-circuit which preserves function and is failure-preserving, i.e., there is a one-to-one correspondence between shorts and opens of the switching circuit and stuck-1 and stuck-0 failures in the corresponding logic circuit.

We demonstrate that this transformation is linear, in that the run-time doubles with doubling the size of the underlying switching circuit. The conversion time is negligible compared to the time used for the test generation. Thus, the complexity of test generation using this method is comparable to that for the logic circuits.

We use the same recent improvements in the D-algorithm which substantially reduces its run-time as well as the number of tests.

Reference

[*] J. P. Roth, Computer Logic, Testing and Verification, Computer Science Press, Rockville, Maryland 20850 (1980).