# TEST GENERATION FOR FET SWITCHING CIRCUITS

J. Paul Roth, Vojin G. Oklobdzija, and John F. Beetem

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

## ABSTRACT

**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 paper 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 ( 1980 ). It is proven for the transformation that a test for a stuck failure in the 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. A program SW2BOOL of the algorithm (written in Pascal) clearly verify the claims as to correctness and speed.**

## INTRODUCTION

Of the means of realization of function, logic circuits and switching circuits, logic circuits are much easier to test, with e.g. 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 the regard for the actual FET implementation is proven to be inadequate ( 1979 ).

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.

## DEFINITION

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.

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 a 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.),

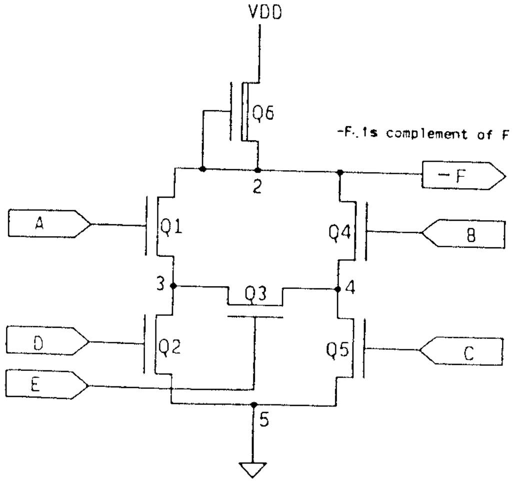

**Fig.1.**

CH0284-2/0000/0059$01.00 © 1984 IEEE

We use the following notation to define a **switch**:

A **switch** with vertices a,b controlled by variable v is denoted 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>, c<3,4>, c<4,5>, d<3,5>.

Assume that 2 and 5 are a 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)

The RLD logic expression ( 1980 ) corresponding to the above circuit is as shown in Fig.2.

*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
```

Fig.2.

## COMPUTATION OF TESTS BY D-ALGORITHM FOR BRIDGE CIRCUIT

Computation by the D-algorithm yields the following tests for primary inputs of the generated logic circuit ( Fig.3.):

| 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.

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 then what was obtained by using our procedure for test generation for FET switching circuits via failure-preserving transformation to logic circuits ( Fig.4. ).

| 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 |

Fig.4.

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 ( 1979 ).

## METHOD AND PROOF

Assume that there is just one pair of terminal nodes. The computation begins with one of these, say $t_1$.

Let switches $a_1 < t_1,u_1 > ,...,a_r < t_1,u_r >$ be all the switches contiguous to $t_1$ . We form the sum of these switches:

$$a_1 < t_1,u_1 > +... + a_r < t_1,u >$$

Having constructed the **algebraic** expression for n-1 nodes, select a boundary node which is not a terminal node; multiply the part of the expression where it appears by the sum of all branches attached thereto which are not repeats, in the same chain of switches.

### Theorem

*The above method correctly computes an assemblage of tests for a* **switching circuit**, *of whatever size, which cover all transistor-shorts and -opens, all* **stuck** *switch failures, of the* **switching circuit**.

### Proof

The proof is by induction. It is evident that the theorem is true for all switching circuits $SW_1$ consisting of exactly one switch - a **switch** is a branch together with a binary variable v **controlling** the connection or disconnection of its vertices (nodes).

Assume then that the theorem holds true for all $SW_s$

having less than n switches. Consider now any **switching** circuit $SW_n$ consisting of exactly n switches. Consider the initial terminal $\alpha$. Let there be k switches attached to $\alpha$. If k=n then $SW_n$ consists of n switches in parallel.

Detach one of these switches $\lambda$ from $\alpha$: what remains has n-1 switches. Call it $SW_{n-1}$ . Therefore, by hypothesis, the above construction - the mapping and the application of test-generation methods, e. g. the D-algorithm interlaced with TestDetect ( 1980 ) to the image-Boolean-graph - yields a set of tests covering the short- and open-failures of $SW_{n-1}$.

Let the functional expression for $SW_{n-1}$ be denoted F($SW_{n-1}$), Consider now the **segment** of the deleted branch: call it $\delta$ . The **segment** of a branch is the subgraph of all branches and nodes connected, directly or indirectly to the branch (or node). Call it $S_\delta$. Clearly $S_\delta$ has less than n branches and therefore, by hypothesis, the method correctly generates a test set covering the assumed failures of $S_\delta$ . Call it $T_\delta$ . Let the set of tests covering $SW_{n-1}$ be denoted $T_{n-1}$ . Let the functional expression for $SW_{n-1}$ be denoted $F_{n-1}$ . Now consider the functional expression $F_n$ for the original switching circuit $SW_n$ . Clearly this may be expressed in the form

$$F_n = F_{n-1} + F_\delta$$

Also clearly therefore the piecewise method of computing a **failure cover** $T_n$ for $SW_n$ :

$$T_n = T_{n-1} + T_\delta$$

is a genuine cover for $SW_n$.

Q.E.D.

### Example of Proof

The following example shows how the proof is put together.

a<2,3>, b<2,4>, c<2,5>, d<3,4>, e<4,5>, f<3,6>, g<4,6>, h<5,6>

It is assumed that 2 and 6 are the terminal nodes.

Instead of forming an expression having three initial terms, for the coface of terminal 2, we shall start with two, and leave the other for the inductive step. Thus our first expression is

$$G_1 = b < 2,4 > +c < 2,5 >$$

The usual multiplication within this expression, but it always remains a sum of the above two terms, thus,

$$G_2 = b < 2,4 > (d < 4,3 > +e < 4,5 > +g < 4,6 > )$$
$$+ c < 2,5 > (e < 5,4 > +h < 5,6 > )$$

$$G_3 = b < 2,4 > (d < 4,3 > f < 3,6 > + e < 4,5 > h < 5,6 >$$
$$+g < 4,6 > ) + c < 2,5 > (e < 5,4 > (d < 4,3 > f < 3,6 >$$
$$+g < 4,6 > ) + h < 5,6 > )$$

On the other hand the expression for the switch left out, a<2,3>, is

$$H_1 = a < 2,3 > (d < 3,4 > +f < 4,5 > )$$

## IMPLEMENTATION AND RESULTS

The algorithm was implemented as a Pascal program called SW2BOOL. SW2BOOL reads an input file consisting of the description of the FET switching circuit. It produces a boolean logic description ( Regular Logic Design RLD ) , ( 1980 ). FET description is similar to the one used in the SPICE ( circuit simulation ) program deck. Both gate-oriented and pass transistor circuits can be handled.

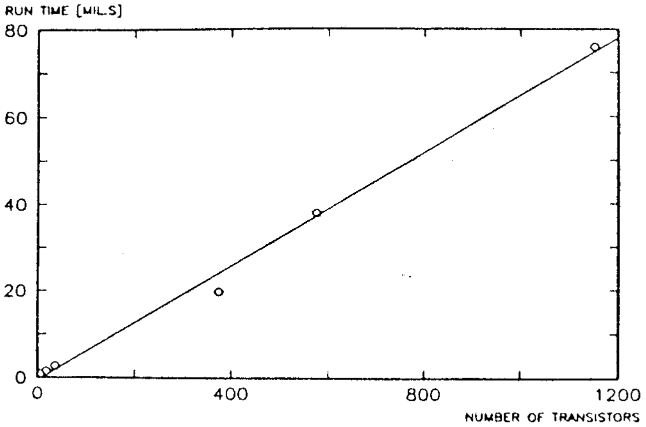

Fig.5.

We used the program to run on numerous examples of various sizes. The largest circuit consisted of an 32-bit wide ALU and other were the circuits of smaller sizes but different in their functions and complexity. The FET to logic conversion time was found to be a linear function of the number of transistors, as shown in Fig. 5.

In general, the conversion time is small compared to the test pattern generation time.

## CONCLUSION

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 size of circuit for direct approaches. For logic circuits, on the other hand, extant methods are satisfactory and widely practiced.

In this paper 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 some recent improvements in the D-algorithm which substantially reduces its run-time as well as the number of tests.

## REFERENCES

1980. Roth, J. Paul, *Computer Logic, Testing and Verification*, Computer Science Press, Rockville, Maryland 20850

1979. Galiay, J. et. al., *Physical versus Logical Fault Models in MOS LSI Circuits, Impact on their Testability*, Digest of Papers FTCS-9, Madison, Wisconsin, June 20-22, 1979.

1977. Bottorff, P.S., et al., *Test Generation for Large Logic Networks*, Proceedings of 14th Annual Design Automation Conference, IEEE Catalog Number 77 CH1216-IC, New Orleans, 1977, p.479-485.