Automatic Test Pattern Generation for Multiple Stuck-At Faults: When Testing for Single Faults is Insufficient

Conrad J. Moore[1], Amir Masoud Gharehbaghi[2], Masahiro Fujita[3]
VLSI Design and Education Center, University of Tokyo
Yayoi 2-11-16, Bunkyo-ku, Tokyo, 113-8656, Japan
E-mail : [1]conrad@cad.t.u-tokyo.ac.jp, [2]amir@cad.t.u-tokyo.ac.jp [3]fujita@ee.t.u-tokyo.ac.jp

Abstract: As fabricated circuitry gets larger and denser, modern industrial ATPG techniques which focus on the detection of single faults become more likely to overlook multiple (simultaneous) faults. Although there are exponentially more multiple faults than single faults, previous works have shown that given an initial set of test patterns for single faults, relatively few additional tests are required in order to cover all multiple faults.[1] The exact situations in which test patterns generated by ATPG for single stuck-at (SSA) faults do not detect multiple stuck-at (MSA) faults will be examined. This will be done by presenting proofs which show the conditions that need to be met such that ATPG for single faults can cover all multiple faults. An analysis is then performed to determine the exact conditions that, when removed from the circuit, violate the assumption that ATPG for single faults will detect all multiple faults. Finally, our proposed ATPG algorithm will be explained.

Keywords—**Automatic Test Pattern Generation, Double fault, Single fault, Combinational logic**

1. Introduction

There are techniques that are used for generating test patterns which can efficiently detect all single faults in a circuit for industrial size designs. However, no practical techniques for the detection of simultaneously occurring multiple faults exist for large designs. It is possible to use similar techniques which are used for single faults on an expanded problem space to include multiple faults, but this means an exponentially greater number of faults must be considered. This method does not work for circuits consisting of millions of gates.

Despite this, it has been shown that, given an initial compact set of test patterns for SSA faults, the number of additional test patterns necessary to cover all MSA faults is not significantly large.[1] Results using the &fftest function of the testing and verification software ABC[2] are shown in Table 1. Note that some smaller circuits do not need additional test patterns to cover multiple faults, and even the largest circuit examined only needs about 13% more test patterns for full coverage. On the other hand, some circuits require a significant amount of extra testing.

This leads to the question about what specific circuit structures tend to cause ATPG algorithms for single faults to overlook multiple faults. Two situations have been discovered manually, but before those are covered, a proof of the necessary conditions under which ATPG for SSA faults is sufficient will be presented.

Table 1. Test patterns required for SSA faults and ABC generated tests including MSA faults.

Circuit	Given Test Patterns for SSA Faults	Generated Test Patterns for MSA Faults	Additional Necessary Test Patterns
s27	5	5	0
s298	28	28	0
s386	64	64	0
s400	28	30	2
s444	25	26	1
s820	99	99	0
s832	101	104	3
s1196	117	117	0
s1238	130	153	23
s1423	25	31	6
s1488	108	108	0
s1494	110	112	2
s5378	102	108	6
s9234	134	297	163
s13207	250	319	69
s15850	116	141	25
s35932	30	103	73
s38417	120	182	62
s38584	174	197	23

2. When ATPG for SSA Faults is Sufficient

2.1 Definitions, Notations, Lemmas

A **tree** is a graph which is connected and contains no cycles.[4] The following characteristics of a tree are implied by its definition:

1. It has no cycles, and a simple cycle is formed if any edge is added.
2. If any edge is removed, it is no longer connected.
3. Any two vertices can be connected, and the path connecting them is unique.
4. If there are n vertices, then there are n-1 edges.

A **binary tree** is a tree in which each node has at most two children nodes.

$\mathbf{f_i}$ is a fault at location i. It is an element of the set of all SSA faults, $\mathbf{F_1}$.

$\mathbf{v_i}$ is the test vector which detects f_i.

$\{\mathbf{f_i}, \mathbf{f_j}\}$ is a double fault which consists of simultaneous faults at locations i and j. It is an element of the set of all double stuck-at (DSA) faults, $\mathbf{F_2}$.

$\mathbf{A(v_i)}$ is the value of the signal at line A, given a non-faulty circuit and input vector v_i.

$\mathbf{A(v_i, f_i)}$ is the value of the signal at line A, given a circuit with fault f_i and input vector v_i.

$\mathbf{A(v_i, \{f_i, f_j\})}$ is the value of the signal at line A, given a circuit with double fault $\{f_i, f_j\}$ and input vector v_i.

Table 2. Truth table of NAND as a masking gate.

In	Faulty In	Out	Faulty Out	Mask?
00	D'D'	1	0	No
01	D'D	1	1	Yes
10	DD'	1	1	Yes
11	DD	0	1	No

Table 3. Truth table of NOR as a masking gate.

In	Faulty In	Out	Faulty Out	Mask?
00	D'D'	1	0	No
01	D'D	0	0	Yes
10	DD'	0	0	Yes
11	DD	0	1	No

D (1/0) represents a faulty signal which has been propagated from some fault, and it is equivalent to the statement, The logic value of the signal at this location should be 1, but due to faults, it is 0."

D' (0/1) is the complement of D. These are the same D and D' which are used in the traditional D-Algorithm.[5]

The term **masking gate** will be used to describe the gate at which, due to MSA fault or SSA fault with multiple propagation paths, both of its inputs may have a faulty value. Depending on the expected values at the gate's inputs, the faulty signals may correct each other, thus "masking" the fault.

Lemma 1: In order for any basic two-input gate (AND/NAND/OR/NOR) to mask a MSA fault, the faulty signals which are propagated to its inputs by the respective faults must both be faulty and of opposite polarity. In other words, one input must be **D**, and the other must be **D'**.

Proof: Consider the truth tables of a NAND and a NOR gate, including faulty signals. As shown in Tables 2 and 3, the output in the faulty case matches that of the non-faulty case only when the inputs are of opposite polarity. Note that the truth table of an AND(OR) gate would be the same as the NAND(NOR) but with inverted values in the "Out" and "Faulty Out" columns. All cases are considered in the truth table. QED.

Note that the XOR gate is not a basic gate. If we use a similar method to determine whether XOR can mask a double fault, it does, in fact, mask every possible combination of double fault. However, the internal structure of the gate violates the constraints which will be used in the proof, so we initially do not consider them.

2.2 DSA Faults in a Tree Circuit

Proposition: Given a redundancy-free combinational circuit in the form of a single-root binary tree graph, assuming basic two-input gates with no fanout, at least one of the test vectors, v_1 or v_2, which detect SSA faults f_1 and f_2 respectively, will also detect the DSA fault $\{f_1, f_2\}$. This statement implies that any set of test vectors which can detect all SSA faults will also detect all DSA faults in the given circuit.

$$\forall S \subset T, \forall \{f_i \in F_1, f_j \in F_1\} \in F_2, Detect(v_i, f_i) \wedge Detect(v_j, f_j) \implies Detect(v_i, \{f_i, f_j\}) \vee Detect(v_j, \{f_i, f_j\})$$

We propose that for all S, some set of test patterns, which is a subset of T, all test patterns, and for all pairs of faults in the circuit, at least one of the tests which detects each SSA fault must also detect the respective DSA fault. $Detect(v, f)$ means test vector v detects fault f.

$$\forall f_i \in F_1, v_i \in S, Detect(v_i, f_i) \implies Complete(S, F_1)$$

For any SSA fault, $f_i \in F_1$, given some test vector v_i in the set of test patterns S, if v_i detects f_i, then this implies that SSA fault coverage is complete, which is represented by $Complete(S, F_1)$.

$$\forall \{f_i \in F_1, f_j \in F_1\} \in F_2, (v_i, v_j) \in S, Detect(v_i, \{f_i, f_j\}) \vee Detect(v_j, \{f_i, f_j\}) \implies Complete(S, F_2)$$

For all DSA faults, $\{f_i \in F_1, f_j \in F_1\} \in F_2$, if v_i or v_j detects the fault, then this implies that DSA fault coverage is complete.

$$\forall S \subset T, Complete(S, F_1) \implies Complete(S, F_2)$$

Therefore, our first statement is equivalent to saying that complete SSA fault coverage implies complete DSA fault coverage.

Consider a circuit made up of NAND gates, without loss of generality, as described in the above proposition. Let us assume that there exists a DSA fault, $\{f_1, f_2\}$, which consists of the two non-equivalent SSA faults, f_1 and f_2. Let there be any pair of test vectors, v_1 and v_2, which detect f_1 and f_2 respectively. Note that while faults f_1 and f_2 are non-equivalent, as a double fault consisting of equivalent single faults is a trivial case, we do not assume anything about the equality or inequality of vectors v_1 and v_2. Let us assume that neither v_1 nor v_2 detects $\{f_1, f_2\}$. Because the circuit is a tree with one root node, it is guaranteed that the propagation path of one fault will intersect with the other and that there is exactly one point of intersection. We can say this with certainty because any two internal signals must somehow propagate to the primary output.

We can ignore any case in which f_2 lies in the path of f_1 (or vice versa), because this is a trivial case in which v_2 (or v_1 in the vice versa case) is sure to detect $\{f_1, f_2\}$. Figure 1 shows this case. If A is neither D nor D', then f_2 is completely unaffected by f_1. Similarly, if the fault effect of f_2 matches the signal at A, fault propagation will continue. The only possible means of blocking fault f_1 is if f_2 and the faulty signal at A don't match, but this would mean that the non-faulty value of A wouldn't have activated fault f_2, which contradicts the assumption that we are using a test vector which detects SSA f_2.

This means that the case which we must consider is when f_1 and f_2 propagate to a masking gate, as shown in Figure 2. By lemmas 1 and 2, in order for this gate to mask the double fault, the inputs of each gate must be D and D' for both v_1 and v_2 when the fault $\{f_1, f_2\}$ is active. Let us refer to the masking gate's inputs as A and B. There are only two

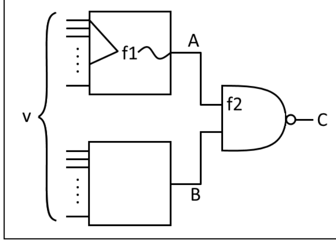

Figure 1. Fault f_2 lies in the propagation path of f_1.

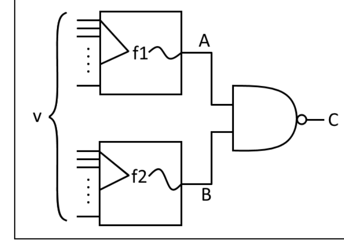

Figure 2. Propagation paths of f_1 and f_2 intersect at a masking gate.

possible cases. Without loss of generality, the fault effect of f_1 at input A is either D or D', assuming it is activated and propagated. Let us consider both cases.

Case 1: The fault effect of f_1 at input A is D (1/0)
1) The case assumption is:

$$A(v_1, f_1) = D$$

2) B is independent of f_1, so its value can be assumed to be non-faulty. Furthermore, a signal of 1 is required for the propagation of SSA f_1.

$$B(v_1) = B(v_1, f_1) = 1$$

3) A is independent of f_2, so when both are active simultaneously, A will act the same as if only f_1 were active.

$$A(v_1, \{f_1, f_2\}) = A(v_1, f_1) = D$$

4) We want to see if it is possible for fault masking to occur, so we set the signal at B given v_1 and the DSA fault to be of opposite polarity to the faulty signal at A, as explained in lemmas 1 and 2.

$$B(v_1, \{f_1, f_2\}) = D'$$

5) Because B is independent of f_1, its value when given the same test vector and only the SSA fault f_2 must be the same as when both faults are simultaneously active. However, this implies that when the signal at $B(v_1)$ is not faulty, it should be 0, by the definition of D' (0/1). This contradicts our second statement that $B(v_1)$ is 1 to continue SSA f_1 propagation. This is a contradiction, and it is impossible for the DSA fault to be undetected in this case.

$$B(v_1, f_2) = B(v_1, \{f_1, f_2\}) = D'$$

Case 2: The fault effect of f_1 at input A is D' (0/1)
1) The case assumption is:

$$A(v_1, f_1) = D'$$

2) Once again, B is independent of f_1, and its value will not be faulty in the SSA f_1 situation. A signal of 1 is required for the propagation of SSA f_1.

$$B(v_1) = B(v_1, f_1) = 1$$

3) A is independent of f_2, therefore:

$$A(v_1, \{f_1, f_2\}) = A(v_1, f_1) = D'$$

4) Assume fault masking is possible and set B given the DSA fault to the opposite polarity of the faulty signal at A.

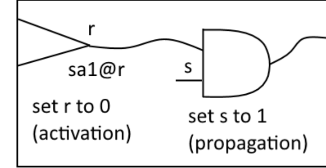

Figure 3. Fault at r is redundant if it is not possible to simultaneously set r to 0 and s to 1.

$$B(v_1, \{f_1, f_2\}) = D$$

5) Because B is independent of f_1:

$$B(v_1, f_2) = B(v_1, \{f_1, f_2\}) = D$$

6) Because A is independent of f_2, and because $A(v_1) = 0$ by the case assumption:

$$A(v_1, f_2) = A(v_1) = 0$$

7) How the circuit should respond given v_1 is now clear, and so far it seems as though it is possible for the DSA fault to be masked for this test vector. Whether or not it is also masked given v_2 must also be determined. Based on step 4, the faulty value at B, should f_2 be activated and propagated, must be D. However, Case 1 shows that if the faulty value is initially D, there will be a contradiction. No matter what, a contradiction occurs.

$$B(v_2, f_2) = D$$

Both cases, one of which must occur, have been considered. In each case, a contradiction occurs if we assume that an undetected DSA fault exists given a complete set of test patterns for SSA faults. Therefore, there must not be any undetected DSA faults. QED.

3. When ATPG for SSA Faults is Insufficient

The essential characteristic of trees which makes the above proof work is the fact that fanout and re-convergence of signals is not permitted. By allowing it, we can have more complicated structures and gates, such as XOR, which make fault masking possible. In addition, redundancies can cause issues because they are "hidden" as SSA faults but "unlocked" as MSA faults, and therefore they are overlooked.

In the example shown in Figure 3, it is assumed that an input pattern which can both activate and propagate the fault exist. However, in the case that simultaneous activation and

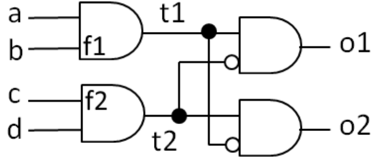

Figure 4. Example circuit in which DSA fault may be overlooked. f_1 and f_2 are "stuck-at 1" faults.

Table 4. Test vector 1001 detects both SSA faults, but it does not detect the DSA fault.

abcd	Fault	t_1	t_2	o_1	o_2
10xx	SSA 1, f_1	D'	0/1	D'/0	0/D
xx01	SSA 1, f_2	0/1	D'	0/D	D'/0
1001	DSA 1, $\{f_1, f_2\}$	D'	D'	0	0
1000	DSA 1, $\{f_1, f_2\}$	D'	0	D'	0
0001	DSA 1, $\{f_1, f_2\}$	0	D'	0	D'

propagation are not possible, the fault at location r becomes redundant. This fault will be ignored by ATPG. For example, assuming only single faults, the activation of sa1@r always results in a value of 0 at s. However, if we also have sa1@s, the previously untested redundant fault, sa1@r, is now irredundant as a DSA fault, {sa1@r, sa1@s}, and it may not be detected by the set of test vectors.

In the previous proof, we assumed a circuit with no fanout. However, many circuits will violate this assumption. One example of a DSA fault which may be undetected due to fanout is shown in Figure 4 and Table 4. ATPG tools may notice that the test vector 1001 will detect both stuck-at 1 faults at locations f1 and f2. However, this test vector will not detect the case when both of the faults are simultaneously active. Furthermore, the DSA fault is not redundant, as either of the tests 1000 or 0001 will detect it. In other words, this DSA fault could potentially slip through the testing phase undetected.

4. Potential Solution

One of the main issues with using "single fault ATPG" for the detection of multiple faults is that a second fault may somehow interfere with the propagation of the first one. Therefore, we propose an algorithm similar to the one used in [3]. The premise is that, given some DSA fault $\{f_1, f_2\}$, as long as f_2 does not interfere with the propagation of f_1, f_1 will be detected, and therefore $\{f_1, f_2\}$ will also be detected. The algorithm focuses on putting path constraints on the circuit such that a fault is guaranteed to propagate to a primary output. Once a test pattern is found for some fault and the constraints are in place, any other fault which may interfere with the constraints can easily be identified and dealt with. Note that this algorithm works for detection, not diagnosis, of DSA faults using ATPG for SSA faults. Furthermore, this method does not work if one of the faults is redundant. Therefore, the target circuit for this algorithm will be used for non-redundant designs. It is currently incomplete, but a brief explanation of the algorithm is as follows:

1. We consider a compacted set of faults, including only primary inputs and fanout wires.
2. Double faults in which both faults are on the same gate are not considered, because there is an equivalent single fault.
3. Generate a list of all eligible SSA faults.
4. Pick a "focus fault", f_i (If possible, give high priority to faults at fanout wires closest to primary outputs and low priority to primary inputs).
5. Generate a test pattern, v_i, for f_i.
6. Based on f_i and v_i, find the necessary path constraints for the fault propagation.
7. Find SSA faults which violate the constraints.
8. Add all DSA faults which consist of the focus fault paired with each of the path constraint violating faults to a list of undetected DSA faults.
9. Remove focus fault from list of remaining SSA faults.
10. If there are remaining SSA faults, repeat from step 4.
11. Else if there is a manageable amount of DSA faults in the list of undetected DSA faults, do simple fault simulation ATPG for each fault.
12. Else quit.

This is a tentative algorithm, and experimentation is still necessary to determine the best implementation.

5. Conclusion and Future Work

The main purpose of this paper was to analyze the necessary conditions in which single fault ATPG is insufficient for multiple fault detection. Further, a potential solution in its early phases is presented. The current focus of this research is the experimentation and implementation of the proposed algorithm. If possible, generating the constraints before the test vector (see steps 5 and 6 above) would be useful because it would allow multiple propagation paths to be activated. However, there may not be a test vector which can satisfy the chosen constraints, and thus it may lead to longer runtime. Also, a DSA fault which is not covered at one point in the algorithm may be covered later. It may be worthwhile to periodically check if any of the remaining DSA faults have been covered before exiting the loop. Finally, there is still the issue of how to deal with redundant (both SSA and DSA) faults.

References

[1] M. Fujita, A. Mishchenko, "Efficient SAT-based ATPG Techniques for All Multiple Stuck-At Faults," *International Test Conference*, Oct. 2014.

[2] R.K. Brayton, A. Mishchenko: ABC: An Academic Industrial-Strength Verification Tool, *22nd International Conference on Computer Aided Verification (CAV 2010)*, pp. 24-40, 2010.

[3] Y. Matsunaga, "A minimum test pattern set generation for large circuits," Inst. of Electron., Inf. and Commun. Engineers, VLD2015-1, 2015.

[4] R. Balakrishnan, K. Ranganathan, Chapter 4: Trees In *A Textbook of Graph Theory* (pp. 73-94). New York: Springer, 2012.

[5] J.P. Roth, "Diagnosis of Automata Failures: A Calculus and a Method," *IBM Journal of Research and Development*, vol. 10, no. 4, pp. 278-291, 1966.