Defect Diagnosis of Digital Circuits Using Surrogate Faults*

Chidambaram Alagappan and Vishwani D. Agrawal

Auburn University
ECE Dept., 200 Broun Hall, Auburn, AL 36849 USA
cza0011@tigermail.auburn.edu,
vagrawal@eng.auburn.edu
http://www.eng.auburn.edu/~vagrawal

Abstract. Classical single stuck-at faults are analyzed as surrogates for any non-classical fault that may have caused an observed failure. Although multiple stuck-at faults are used as an illustrative example of non-classical faults, proposed algorithms are applicable to any other type of fault. Our effect-cause analysis is less complex than existing methods. The diagnostic procedure adds or removes faults from a set of candidate faults based on the observed circuit outputs, using minimal fault simulation, to obtain a small set of suspected faults.

Keywords: Dictionary-less fault diagnosis; fault simulation; multiple stuck-at faults; stuck-at faults; surrogate faults.

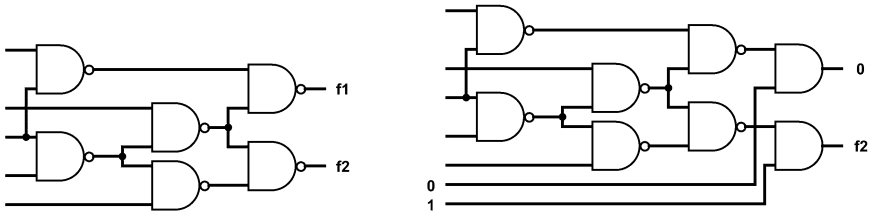
1 Introduction

An ideal fault diagnosis procedure should report true failures with accuracy, i.e., *resolution* (the number of true failures reported among the total number of faults reported) and *diagnosability* (the percentage of correctly identified failures) of the diagnosis result should be high [8]. Previous research on fault diagnosis attempts trade-offs between the resolution, diagnosability and CPU time, but the algorithms become increasingly complex. Two major classes of algorithms are cause-effect and effect-cause types. Cause-effect analysis has a stored simulated response database of modeled faults. The faulty circuit response is compared against this database to find out which fault might have caused the failure [5,7,12,15]. This database, called dictionary, is memory intensive and impractical for large circuits. Effect-cause analysis works on the observed failing signals and searches for the cause by tracing back the error propagation path from the failing primary outputs to identify faults likely to have produced the failure [3,4,9]. Backward implication and forward propagation are used for this purpose [9]. Such procedures use moderate amount of memory.

Although a real defect is rarely a classical single stuck-at fault, diagnostic procedures match observed symptoms to closest single stuck-at faults. This is

* Research supported in part by the National Science Foundation Grant CCF-1116213.

M.S. Gaur et al. (Eds.): VDAT 2013, CCIS 382, pp. 376–386, 2013.



(a) C17 benchmark circuit. **(b)** C17 with output selection.

Fig. 1. Output selection implementation on C17 benchmark circuit

because the available analysis tools only handle single stuck-at faults. The diagnosed single stuck-at faults then are not real but are "surrogates" meaning that they have some, but not all, characteristics of the actual defect in the circuit. The term "surrogate fault" has been used before in the literature [10,13,16].

2 Preliminaries

A fault simulator reports all single stuck-at faults that can be detected by an input pattern on all primary outputs (POs). To use this information for distinguishing among several faults that could have caused the failure we employ *output selection*. AND gates are added in the simulation netlist at each PO, with the other input of the AND gate being a new primary input (PI). The failing test pattern is duplicated as many times as the number of POs, activating exactly one PI at a time. Thus, new PIs that directly go to the added AND gates are all forced to 0 except for one PI to a transparent AND gate to find the detectable faults at the corresponding PO. Consider C17 benchmark circuit of Figure 1a. A test pattern "abcde" produces good circuit responses 'f1' and 'f2'. Assume this circuit has a failure only at the second output. A typical fault simulator may identify detectable faults without associating them to any PO. With output selection of Figure 1b, the test pattern is duplicated as "abcde**10**" and "abcde**01**".

3 Diagnosis Algorithm

The diagnosis algorithm relies on a basic concept that a test pattern fails because a detectable fault is present in the circuit or a test pattern passes because none of the detectable faults is present. For this to be effective, we assume that there is no circular fault masking present in the circuit. Let *'passing_set'* be the set of passing test patterns, *'failing_set'* be the set of failing test patterns, *'sus_flts'* be the suspected fault list, *'set1_can_flts'* be **prime suspect candidate faults** and *'set2_can_flts'* be **surrogate candidate faults**. For simplicity, we will refer to *'set1_can_flts'* as SET1 and *'set2_can_flts'* as SET2.

The algorithm has four phases [6] as shown in the flowchart of Figure 2. Initially, Phase 1 takes the union of all faults detectable by all failing patterns as a list of suspects. Since this set can be large, we need to reduce the list. In Phase 2,

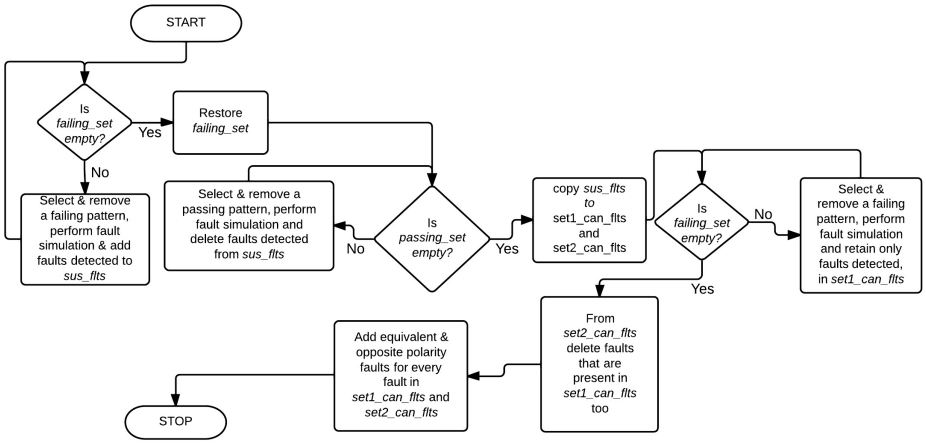


Fig. 2. Flowchart of diagnosis procedure

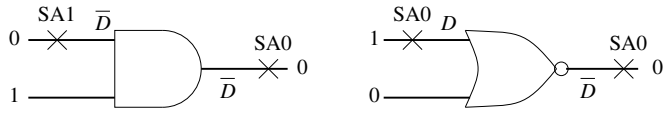


Fig. 3. Opposite polarity fault masking

we take the union of all faults detectable by passing patterns and subtract it from suspect list of Phase 1. Phase 3 takes an intersection of the suspected fault lists of all failing patterns. The resulting faults are called **prime suspects**. These faults are of low priority, but there is a chance that they can be surrogate of an actual fault or one of the actual faults. In Phase 4, equivalent faults of the identified suspected faults are added to the suspect list. To guard against fault masking, we include the opposite polarity faults of the faults that are present in SET1 and SET2, to get the final candidate fault lists. The pseudocode for the entire algorithm is available in a recent thesis [6].

If the actual defect is a single stuck-at fault, the algorithm identifies it as a "prime suspect" in Phase 2. For other defects, it provides a list of surrogate single stuck-at faults "resembling" the actual defect in location or behavior.

Masking. In Figure 3, the top input of an AND gate is stuck-at-1 (SA1) and the output is stuck-at-0 (SA0). To activate the first fault, a '0' must be supplied to the top input and a '1' must be supplied to the bottom input to propagate it. This will produce a $\overline{D}$ on the top input, where $\overline{D} = 1$ if SA1 is present on that input or $\overline{D} = 0$ if the input is fault free. However, $\overline{D}$ will be masked at the output by the SA0 fault. The diagnosis procedure will identify output SA0 as the only suspect. Therefore, SA0s on both inputs are also included as suspects. Similarly, for the NOR gate in Figure 3, when the top input and output have SA0s, the masking occurs. Therefore, Phase 4 enhances the suspect list with all opposite polarity faults for equivalent faults of a suspected single stuck-at fault.

Theorem 1. *If there is only a single stuck-at fault present in a failing circuit under diagnosis (CUD), the diagnosis algorithm will always identify that fault as a prime suspect, irrespective of the detection or diagnostic coverage of the test pattern set.*

Proof. Assume that CUD has a single stuck-at fault that causes $N-k$ out of N test patterns to fail. The remaining k are passing patterns. Because a fault free circuit cannot have any failing test pattern, the presence of failing test patterns indicates the presence of some failure s. In other words, a test pattern can only fail because a fault that it detects is present. Hence all $N-k$ patterns detect the fault s and the remaining k patterns do not detect the fault s. If all $N-k$ patterns detect some fault present in the circuit, it has to be the same fault that all the $N-k$ patterns detect, because there is no more than one fault present in the circuit according to our assumption in the beginning. Moving forward with this revelation, Phase 3 will always come up with one or more prime suspects including the actual fault, as the intersection of the faults detected by all failing patterns. ■

Many possible cases of single stuck-at faults, multiple stuck-at faults without masking, multiple stuck-at faults with masking, and multiple stuck-at faults with interference have been analyzed in detail [6]. Figure 4 shows the comparison of simulation effort between the proposed diagnosis procedure and the traditional fault dictionary diagnosis method. It is plotted for a multiple (two) stuck-at fault case of C432 ISCAS'85 benchmark circuit. This circuit has a total of 1078 single stuck-at faults in the fault list. The test vector set with 100% diagnostic coverage of detectable faults contains 462 test vectors (with output selection implemented). The dictionary method involves simulation of all faults for all test vectors. Hence, the entire area under the straight black line denotes the simulation effort of the fault dictionary method. The considered failure case produced 31 failing vectors and 431 passing vectors. The proposed fault diagnosis procedure performs fault simulation with the failing test vectors first, which is denoted by the solid red line. This line drops down steeply because, as and when the faults are detected, they are dropped. We process fewer faults as we proceed with the simulation. Next, the fault simulation of passing patterns is performed, which is denoted by the dotted blue line. Note that the faults that were detected and dropped during failing pattern simulation are those to be simulated with passing patterns. In this case too, faults are dropped as and when they are detected by the passing patterns, which explains the drop in the line. Once again the number of faults to be simulated keeps reducing throughout simulation. Beyond a certain point, not many remaining faults are detected by the passing patterns, which makes the curve almost flat. After simulating all passing patterns, the remaining faults become the suspects and surrogates. The area under these lines (solid red and dotted blue) denotes the simulation effort of the proposed procedure that is far lower than the traditional dictionary method.

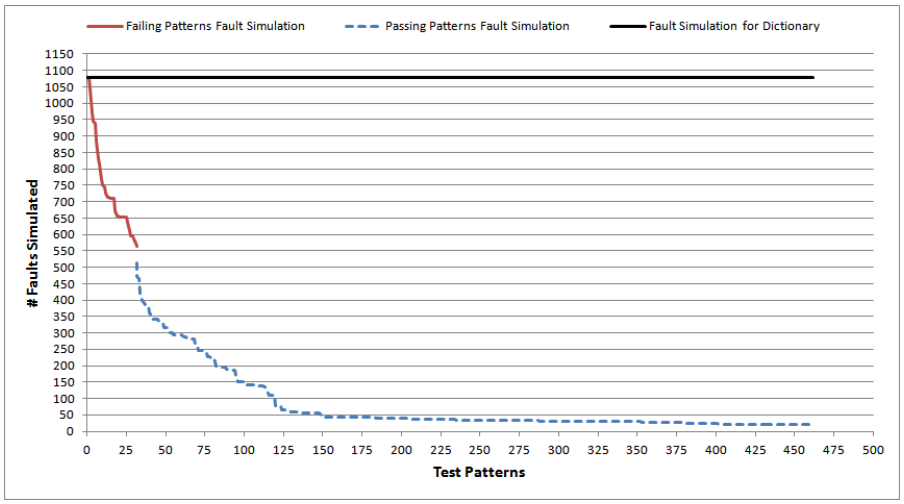


Fig. 4. Simulation effort comparison with dictionary method applied to C432

3.1 Fault Ranking

There is a small probability that the diagnosis procedure comes up with no result, i.e., SET1 and SET2 are both empty after fault simulation. That can happen in a situation like multiple faults with masking and interference such that they together produce faulty output responses that will allow a few of the test patterns detecting them to pass and other test patterns detecting them to fail. For example, consider the case where every fault detectable by failing patters is also detectable by at least one passing pattern. This is a rare phenomenon and only in such cases a ranking procedure is used.

Phase 1 short lists faults in *sus_flts*. For ranking, while carrying out Phase 1 we keep a count of the number of failing patterns that detect each fault. This number is called the weight of the corresponding fault, e.g., if fault F1 is detected by three failing patterns, then the weight of F1 at the end of Phase 1 will be three. Similarly, in Phase 2, which simulates only the faults found detectable in Phase 1, we keep a count of the number of passing patterns for each fault. This number is subtracted from the weight of the fault found in Phase 1. At the end of Phase 2 we get the final weight of every fault. The faults with the highest weight are reported to be prime suspect (SET1) faults and the faults with the second highest weight are reported to be surrogate (SET2) faults.

Note that the final weights can also be negative. This will happen when a fault is detectable by more passing patterns and by fewer failing patterns. Even in this case, the top two highest weights are considered to be suspects. Also, there can be cases where the final weight is zero. This will happen when the fault is detected by the same number of passing and failing patterns.

The fault simulation in ranking is more expensive because it is done without fault dropping. In practice, however, the four phase diagnostic procedure

Table 1. Single fault diagnosis with 1-detect tests

Circuit name	No. of outputs	No. of patterns	DC %	Diagnosis %	CPU* s	Fault ratio SET1	Fault ratio SET2
C17	2	10	95.454	100	0.067	1.100	1.780
C432	7	462	94.038	100	0.189	1.025	6.675
C499	32	2080	98.000	100	0.588	1.029	16.722
C880	26	1664	94.161	100	0.503	1.069	2.248
C1908	25	3625	85.187	100	1.294	1.379	28.290
C2670	140	13300	85.437	100	6.455	1.320	8.207
C3540	22	3520	89.091	100	1.333	1.229	5.200
C5315	123	13899	91.192	100	6.847	1.054	4.204
C6288	32	1056	85.616	100	0.764	1.138	8.255
C7552	108	17064	86.507	100	10.123	1.281	10.765

∗ PC with Intel Core-2 duo 3.06GHz processor and 4GB memory

returning with zero faults is a rare possibility. Out of numerous tests performed on benchmark circuits, the algorithm came up with no suspect only twice, requiring the fault ranking to provide a diagnosis.

4 Experimental Results

The algorithms were applied to ISCAS'85 benchmark circuits using various test pattern sets. The circuit modeling and the entire algorithm were implemented in Python programming language [2], automatically invoking ATPG and fault simulator of Mentor Graphics FASTSCAN [1] software. VBA macros [11] were used to duplicate the test patterns for output selection. The programs were run on a personal computer (PC) with Intel Core-2 duo 3.06GHz processor and 4GB memory. Results for a circuit are averaged over 100 cases, each with a randomly selected fault. C17 has only 22 faults and its results are averaged over 22 cases.

Results of single fault diagnosis using a 1-detect pattern set are shown in Table 1. The first column states the circuit name, the second column contains the number of primary outputs the circuit has. The third column shows the number of patterns (with output selection implemented) used for diagnosis. So the actual number of patterns in the 1-detect pattern set for any circuit will be the number of patterns shown in the third column divided by the number of primary outputs (shown in column 2) of the table.

Diagnostic Coverage (DC) of the test pattern set based on single stuck-at faults, excluding redundant faults, is stated in column 4. It is defined as [17],

$$DC = \frac{Number\ of\ detected\ fault\ groups}{Total\ number\ of\ faults} = \frac{n}{N} \tag{1}$$

Column 5 shows the percentage of cases the single fault was diagnosed. For single stuck-at faults, the algorithm always comes up with the actual fault (100% diagnosis), even if the diagnostic coverage of the pattern set is not as high. Simulation time in seconds is stated in column 6. Ratios of number of candidate

Table 2. Single fault diagnosis with 2-detect tests

Circuit name	No. of outputs	No. of patterns	DC %	Diagnosis %	CPU* s	Fault ratio SET1	Fault ratio SET2
C499	32	3872	98.400	100	1.025	1.029	7.970
C1908	25	6425	86.203	100	2.242	1.379	14.798
C7552	108	27756	86.750	100	16.076	1.281	8.023

∗ PC with Intel Core-2 duo 3.06GHz processor and 4GB memory

faults in SET1 and SET2 are reported in columns 7 and 8, respectively. This is the ratio of the total number of faults reported in each set to the number of faults expected in that set. The expected number of faults includes the actual fault, its equivalent faults and the opposite polarity faults for all equivalent faults, including the actual fault. This ratio denotes the diagnostic resolution of the procedure. The closer the fault ratio is to 1.0, better is the resolution. For single stuck-at faults, the ratio of SET1 faults is almost 1.0 in all cases. Hence, when the faults identified in SET1 are probed (by electron beam or other failure mode analysis procedures), one would locate the actual fault and it will unnecessary to probe the faults in SET2. But in a real situation since we would not know whether the actual fault is a single stuck-at fault or a non-classical fault, the SET2 surrogate faults should not be disregarded.

For circuits C499, C1908 and C7552, the ratio of faults in SET2 is high. This is due to the fact that the diagnostic coverage of the test pattern set is not high enough. To examine the effect of improving diagnostic coverage of the test pattern set on diagnostic resolution, 2-detect test patterns were used to diagnose these three circuits. The results are shown in Table 2. Note that 2-detect patterns provide a marginal, though definite, increase in diagnostic coverage (DC). Most increase occurred for C1908, which is only 1.016%. Still, the resolution improved as SET2 ratio dropped to about 50%. So, for patterns with even higher diagnostic coverage, the resolution will be further improved. An utmost efficiency of the diagnosis algorithm can be expected from higher diagnostic capability test pattern set than from just the detection test pattern set.

To verify the relevance of the reported surrogate faults to actual non-classical faults, we examined multiple stuck-at faults by introducing two stuck-at faults simultaneously. One hundred failure cases were generated for each circuit. In each case, two stuck-at faults were chosen in such a way that they are close to each other in the circuit. The reason for considering only two simultaneous faults is that the probability of fault masking is maximum when there are just two faults and this probability keeps reducing as the number of faults present in the circuit increases. This increased chance of fault masking created a pessimistic environment for the algorithm. All diagnosis results are averaged over 100 cases for each circuit. Table 3 summarizes the multiple fault diagnosis experiment with 1-detect test patterns.

Column 4 of Table 3 shows the percentage of cases where both faults were diagnosed. Column 5 shows the percentage of cases where only one of the actual faults present was diagnosed. The sum of these two percentages subtracted from

Table 3. Multiple (two) fault diagnosis with 1-detect tests

Circuit name	No. of Patterns	DC %	% of cases diagnosed			CPU* s	Fault ratio	
			Both faults	One fault	No fault		SET1	SET2
C17	10	95.454	80.950	19.040	0.000	0.067	0.500	2.091
C432	462	94.038	90.566	7.547	1.886	0.135	0.563	3.516
C499	2080	98.000	49.056	20.754	30.188	0.613	0.371	17.589
C880	1664	94.161	86.792	9.433	3.773	0.502	0.900	3.205
C1908	3625	85.187	90.566	0.000	9.433	0.928	0.488	12.764
C2670	13300	85.437	88.679	3.773	7.547	4.720	0.564	7.046
C3540	3520	89.091	86.792	3.773	9.433	1.547	0.488	5.177
C5315	13899	91.192	98.113	1.886	0.000	7.065	0.422	3.886
C6288	1056	85.616	83.018	0.000	16.981	0.888	0.589	5.536
C7552	17064	86.507	96.226	1.886	1.886	7.539	0.358	7.104

∗ PC with Intel Core-2 duo 3.06GHz processor and 4GB memory

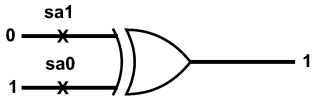


Fig. 5. Fault masking (interference) in XOR gate

100% gives the percentage of cases where both faults were not diagnosed, as shown in column 6. As the results in the table indicate, except for the circuit C499, all other circuits have, at least in 80% cases, a perfect diagnosis of both faults. A point to be noted is that the proposed diagnosis procedure does not assume that fault masking is not present and the reported percentage of diagnosis includes the possible fault masking and interference cases.

The reason for C499 (32-bit single-error-correcting circuit) producing poor multiple fault diagnosis (resulting in irrelevant surrogate faults) even with a test pattern set having very high diagnostic coverage (based on single stuck-at faults) must be examined. We found the presence of circular fault masking in many of the fault cases considered. The circuit has an XOR tree consisting of 104 two-input XOR gates. XOR logic gates are not considered to be elementary logic gates since they are generally constructed from multiple Boolean gates, such that the set of faults depends on its construction. All four test patterns are needed to completely test a 2-input XOR gate, regardless of its construction [14]. Consider the XOR gate shown in Figure 5. The top input has a stuck-at-1 (sa1) fault and the bottom input has a stuck-at-0 (sa0) fault. To propagate a single fault through XOR gate, the other input must be unchanged. But since this is a multiple fault situation, two faults are trying to propagate through the XOR gate at the same time. So a '0' on the top input is required to activate the sa1 fault and a '1' on the bottom input is required to activate the sa0 fault. But since both inputs are changed, the faults mask each other. This phenomenon is called circular masking. Hence the output is '1' which is the same as the good circuit output. Due to this circular masking, the algorithm will not be able to produce the relevant surrogate faults as the actual faults are dropped, since the pattern which should have failed, passes. This is not only for the case where both faults are present on

Table 4. Multiple (two) fault diagnosis with 2-detect tests

Circuit name	No. of patterns	DC (%)	% of cases diagnosed			CPU* s	Fault ratio	
			Both faults	One fault	No fault		SET1	SET2
C499	3872	98.400	49.056	20.754	30.188	0.696	0.371	11.555
C1908	6425	86.203	90.566	0.000	9.433	2.314	0.488	7.232
C7552	27756	86.750	96.226	1.886	1.886	17.291	0.358	5.905

∗ PC with Intel Core-2 duo 3.06GHz processor and 4GB memory

Table 5. Single fault diagnosis with diagnostic patterns

Circuit name	No. of outputs	No. of patterns	DC %	Diagnosis %	CPU* s	Fault ratio	
						SET1	SET2
C17	2	12	100	100	0.067	1.000	1.780

∗ PC with Intel Core-2 duo 3.06GHz processor and 4GB memory

the inputs of the XOR gate, but also for any case where fault effects are being propagated through an XOR gate. The situation will improve while considering more than two faults to be present in the circuit because the probability of a complete circular masking decreases with the increase in the number of faults. In circuit C499, the presence of this huge XOR tree increases circular masking and thereby deteriorates the performance of the diagnosis algorithm. But as discussed before, the algorithm does produce reasonable results even in a highly pessimistic environment, where choices (close neighborhood faults selected) are made in such a way that the probability of masking is high. One other ISCAS'85 benchmark circuit, which has (2-input) XOR gates present, is circuit C432. But it has only 18 XOR gates, which do not form a tree and hence the diagnostic percentage is not hurt significantly.

The ratio of faults in SET1 in Table 3 is less than 1 because in most cases, faults reported in SET1 include one of the actual faults, its equivalent faults and the opposite polarity faults. The other actual fault, its equivalent faults and opposite polarity faults are present in SET2. Hence, the resolution of SET1 faults is mostly closer to 0.5 than being 1.0 when we consider two faults.

The same three circuits show a comparatively poorer SET2 resolution. Hence, 2-detect patterns are used to show that the diagnostic resolution improves upon improving the diagnostic coverage of the test pattern set. The results of this experiment are shown in Table 4. Once again it is seen that, for small increase in diagnostic coverage (DC) of the patterns by 1.016% (maximum) for circuit C1908 the resolution is improved by almost 40%. Other two circuits show a similar trend.

The last experiment was to try the diagnosis procedure on a 100% diagnostic test pattern set. The circuit C17 reports 95.454% of diagnostic coverage (DC) with as few as 5 patterns that have 100% detection coverage. The total number of faults in the circuit is 22. There was only one fault pair that was not distinguished. Adding one more pattern that distinguishes the fault pair yielded 100% diagnostic coverage as expected. The diagnostic algorithm was then run using this test pattern set to yield the results shown in Tables 5 and 6.

Table 6. Multiple (two) fault diagnosis with diagnostic patterns

Circuit name	No. of patterns	DC %	% of cases diagnosed			CPU* s	Fault ratio	
			Both faults	One fault	No fault		SET1	SET2
C17	12	100	80.952	19.047	0.000	0.067	0.489	2.102

∗ PC with Intel Core-2 duo 3.06GHz processor and 4GB memory

Single fault diagnosis with 100% diagnostic coverage vector set produced a perfect diagnostic resolution '1.0' as expected in SET1 and a slightly improved resolution in SET2. Multiple fault diagnosis with this test pattern set improved the resolution in SET1 by a very small amount and decreased the resolution of SET2 by the very same amount. Also, the diagnostic coverage was improved by a very small percentage. Since the 1-detect test pattern set already had a diagnostic coverage of 95.454, there was very little left to improve.

To sum up, the proposed diagnostic procedure, given a failing vector and the cause of failure a single stuck-at fault, will always come up with the actual fault, irrespective of the detection or diagnostic coverage of the test pattern set. If the detection coverage of the test pattern is higher, better will be the resolution of the faults reported. Provided with 100% diagnostic coverage, the maximum resolution can be achieved. If the actual fault is a multiple stuck-at fault without circular fault masking, the diagnostic procedure will come up with surrogate faults that represent the actual faults or the behavior of the actual faults, with higher resolution as the diagnostic coverage of the pattern set increases.

5 Conclusion

We have proposed a lower complexity fault diagnosis algorithm that is based on effect-cause analysis. The algorithm has higher diagnosability and resolution for the surrogate faults identified to represent multiple stuck-at faults without circularly masking, even if provided just with a high detection coverage test pattern set. The same trend is exhibited when the diagnostic coverage of the test pattern set is increased. The algorithm is memory efficient, since it does not require a dictionary and also has reduced diagnostic effort (CPU time), since it works on relatively smaller number of fault suspects and does not require re-running simulations after frequently moving faults to and from the suspected fault list based on heuristics.

In the future, we should examine the performance of the diagnosis algorithm on other non-classical faults by using appropriate fault models and their simulators. Also, redundant faults as one of the interfering fault in fault masking may be examined. Considering that fault simulation tools will always be limited to a few fault models (e.g., single stuck-at or transition faults), we should explore the relationships between non-classical faults (bridging, stuck-open, coupling, path delay, etc.) and the corresponding surrogate classical representatives. For example, some non-classical faults like stuck-open or bridging require an initialization pattern to precede a stuck-at test pattern. Thus, the test result for the non-classical fault agrees with a single stuck-at fault only on a subset of patterns.

Further analysis can establish better correlation between actual faults and their surrogates.

References

1. ATPG and Failure Diagnosis Tools. Mentor Graphics Corp., Wilsonville, OR (2009)
2. Python Tutorial Release 2.6.3. docs@python.org. Python Software Foundation (2009)
3. Abramovici, M., Breuer, M.A.: Fault Diagnosis Based on Effect-Cause Analysis: An Introduction. In: Proc. 17th Design Automation Conf., pp. 69–76 (June 1980)
4. Abramovici, M., Breuer, M.A.: Multiple Fault Diagnosis in Combinational Circuits Based on an Effect-Cause Analysis. IEEE Transactions on Computers C-29(6), 451–460 (1980)
5. Agrawal, V.D., Baik, D.H., Kim, Y.C., Saluja, K.K.: Exclusive Test and Its Applications to Fault Diagnosis. In: Proc. 16th International Conf. VLSI Design, pp. 143–148 (2003)
6. Alagappan, C.: Dictionary-Less Defect Diagnosis as Real or Surrogate Single Stuck-At Faults. Master's thesis, Auburn University, Auburn, Alabama (May 2013)
7. Beckler, M., Blanton, R.D.(S.): On-Chip Diagnosis for Early-Life and Wear-Out Failures. In: Proc. International Test. Conf., pp. 1–10 (November 2012)
8. Bushnell, M.L., Agrawal, V.D.: Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits. Springer, Boston (2000)
9. Cox, H., Rajski, J.: A Method of Fault Analysis for Test Generation and Fault Diagnosis. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems 7(7), 813–833 (1988)
10. Grimaila, M.R., Lee, S., Dworak, J., Butler, K.M., Stewart, B., Houchins, B., Mathur, V., Park, J., Wang, L.-C., Mercer, M.R.: REDO - Random Excitation and Deterministic Observation - First Commercial Experiment. In: Proc. 17th IEEE VLSI Test Symp., pp. 268–274 (April 1999)
11. Kofler, M.: Definitive Guide to Excel VBA. Apress, New York (2000)
12. Millman, S.D., McCluskey, E.J., Acken, J.M.: Diagnosing CMOS Bridging Faults With Stuck-At Fault Dictionaries. In: Proc. International Test. Conf., pp. 860–870 (September 1990)
13. Reddy, S.M., Pomeranz, I., Kajihara, S.: On the Effects of Test Compaction on Defect Coverage. In: Proc. 14th IEEE VLSI Test Symp., pp. 430–435 (April 1996)
14. Stroud, C.E.: A Designer's Guide to Built-in Self-Test. Springer, Boston (2002)
15. Venkataraman, S., Drummonds, S.B.: POIROT: A Logic Fault Diagnosis Tool and Its Applications. In: Proc. International Test Conf., pp. 253–262 (2000)
16. Wang, L.C., Williams, T.W., Mercer, M.R.: On Efficiently and Reliably Achieving Low Defective Part Levels. In: Proc. International Test Conf., pp. 616–625 (October 1995)
17. Zhang, Y., Agrawal, V.D.: An Algorithm for Diagnostic Fault Simulation. In: Proc. 11th Latin-American Test Workshop (LATW), pp. 1–5 (March 2010)