Journal of Engineering Science and Technology
Vol. 15, No. 6 (2020) 3820 - 3835

COMPACT TEST AND DIAGNOSIS PATTERN GENERATION FOR MULTIPLE FAULT PAIRS IN SINGLE RUN

NAVYA MOHAN*, J. P. ANITA

Department of Electronics and Communication Engineering,
Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India
*Corresponding Author: m_navya@cb.amrita.edu

Abstract

In this paper, an efficient algorithm for test and diagnosis pattern generation is proposed. Diagnosis patterns are those which help in distinguishing one fault from the other such that the list of possible defects can be reduced. This algorithm gives a compact set of Test and Diagnosis Patterns (TDPs) in a single ATPG run. Here a new approach is used to generate diagnosis patterns which uses Zero suppressed Binary Decision Diagram (ZBDD). This algorithm contains three main parts, which are ZBDD based diagnosis pattern generation, Fault pair grouping, and equivalent fault pair identification. For the ISCAS'89 and '85 benchmarks circuits considered in this work it could be observed that all the fault pairs were distinguished using a compact set of patterns. None of the fault pairs were aborted in the procedure. The proposed algorithm shows an average of 17% decrease in the number of test and diagnosis patterns as compared to previous works.

Keywords: ATPG, Diagnosis, Fault pair, Stuck-at-faults, ZBDD.

1. Introduction

When a new process technology is introduced, identification of defect locations plays a crucial role to improve the yield. The process of identification of defect location and the type of defect is called diagnosis. In usual diagnosis methods, a Failure log is generated with the help of an Automatic Test Equipment, by taking in Test Patterns (TPs) generated from any commercial ATPG tool as input. Once this step is over, a specialized diagnosis tool is used to generate Defect Candidates (DC). These DCs are then used to arrive at the location where the actual physical defect has actually occurred. This process is called Physical Failure Analysis (PFA). This procedure is shown in Fig. 1(a). The ultimate aim of any diagnosis tool is to reduce the number of DCs for a given faulty response [1]. With the help of Test Patterns (TPs) alone, it is impossible to filter out some of the wrong defect candidates. To diagnose stuck-at faults efficiently, additional patterns are needed because test patterns have low distinguishability. These additional patterns are called the Diagnosis Patterns (DP). Figure 1(b) shows a procedure where an additional pattern generation procedure is used to generate DPs which when used along with TPs produces a smaller FL and thereby a reduced DC. In this paper, a Test and Diagnosis Pattern generation procedure is proposed, which produces both TPs and DPs in a single run of ATPG. Thus, this algorithm helps in identifying the actual defect location with highly compact set of vectors. The proposed workflow is shown in Fig. 1(c).

Different methods for generating diagnosis patterns have been reported in the literature. Most of the existing methods use additional functionalities to be incorporated into the existing ATPGs to generate diagnosis patterns [2-4] proposed a method in which the Circuit Under Diagnosis (CUD) is modified such that existing ATPG is sufficient in generating the DPs. The drawback of the first method is that it requires modification of standard ATPGs and that of the second method is that it requires modification in the circuit model. The circuit modification method is more popular as it is less complicated as compared to ATPG modification.

Gruning et al. [5] proposed a method where the ATPG tool itself has to be modified to incorporate the abilities to generate DPs. This method is complicated and demands modifications in commercial ATPG tools, which may not be always feasible. Another approach to this problem is to modify the CUD such that DPs can be generated easily. This method was found to be more feasible and different methods to implement the same was proposed in [1, 6, 7].

Agrawal et al. [4] considered only 2 faults at a time and uses two copies of the CUD and the output ports are connected using additional XOR gates. There are two drawbacks for this method. Firstly, as it considers only one fault pair in one iteration of the algorithm, the Diagnosis Pattern Generation (DPG) method will take more time and will give a greater number of patterns. Secondly, it requires two replicas of the circuit for pattern generation, so it is not suitable for bigger designs. Pomeranz and Reddy [8] proposed another circuit modification method for DPG which does not require circuit replication. But this method does not consider more than one fault pair simultaneously. Thus, the DPG will take more time and generate less compact DPs. A fast DPG is proposed by Zhang and Agrawal [9] where an initial set of Test Patterns (TP) is considered and the faults which could be diagnosed using these TPs are dropped. This method also considers only one fault pair in one iteration.

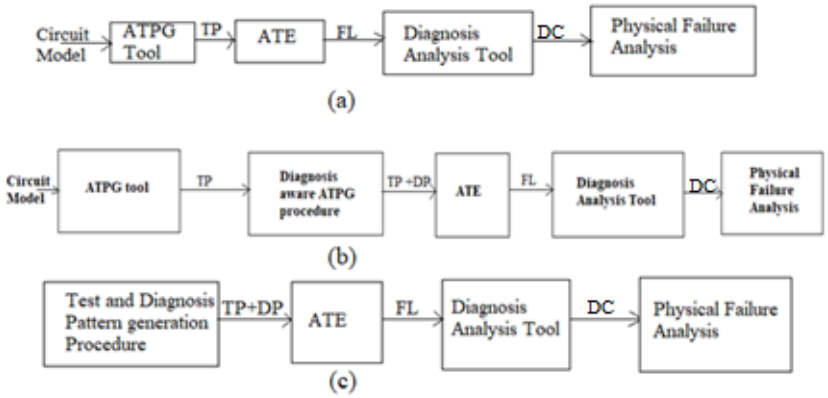


Fig. 1. (a) Diagnosis procedure using only test patterns (b) Diagnosis procedure using additional DPs and (c) Proposed diagnosis procedure using TDPs.

Most recent work in this field is proposed by Wu et al. [7]. It is a three-step procedure that generates highly compact diagnosis patterns. In this procedure, all possible fault pairs are considered simultaneously. For those fault pairs which cannot be considered with in the first step, a circuit model modification method is proposed. It was mentioned that if a fault cannot be distinguished by the first two methods then they are equivalent faults and an efficient method for identifying equivalent faults is also discussed. The preliminary results of this work are also presented in [6]. This method also uses a circuit modification approach. Whenever a fault pair cannot be distinguished using test patterns then they are connected using a 2×1 multiplexer. This generates an additional input, which is the control input of the multiplexer for each pair under consideration and thus instead of detecting the fault pair the fault at the control input has to be detected. This method of changing the fault type is defined by the authors as a User Defined Fault Model (UDFM) based approach as it reduces the of distinguishing a fault pair from one another by converting it into a problem of detection of the stuck-at fault at the select line input of the additional multiplexers. The purpose of these multiplexers is to make sure that only one fault in a pair gets activated at a time. The focus of the proposed work is to deactivate a fault without adding extra circuitry.

The basic flow for generating DPs mentioned in [4, 6, 8] followed the methodology shown in Fig. 1(b). Here an ATPG is initialized first for the generation of test patterns and then for diagnosis patterns. Also, all the methods discussed above have opted for a circuit modification approach in the diagnosis aware ATPG procedure. Generally, a good ATPG procedure is expected to accomplish three basic objectives: 1) lesser defect candidates; 2) small DP volume; 3) Low DPG time. This paper proposes a procedure where the test and diagnosis process are combined hence it is expected to complete the process in less time and will be able to create a smaller set of patterns to diagnose all distinguishable fault pairs in the circuit. The proposed work generates patterns without adding any additional circuitry to the (Circuit Under Test) CUT with the help of ZBDD which is another important advantage as compared to previous works reported in this area.

In the proposed algorithm a fault deactivation method using ZBDD is used to generate DPs. The algorithm has four main sections as follows:

- Generation of test and Diagnosis Patterns using ZBDD XOR operation. Here as soon as a test pattern is generated it is also checked for its diagnostic capability. This helps in reducing the overall pattern count.
- Selective fault deactivation and fault pair grouping is done such that a maximum number of fault pairs which remain in distinguished using already generated patterns can be distinguished using a minimum number of additional patterns.
- Dynamic pattern compaction methodology is used such that test and diagnosis patterns generation and compaction can be carried out simultaneously.
- Finally, those pairs of faults which are not distinguished from the above method are considered and checked for equivalency and then considered one at a time. In this section, only a small portion of the circuit to which the faults propagate has to be considered.

The remaining sections of this paper are organized as follows. Section 2 is a consolidated literature survey related to this work. Section 3 deals with the proposed Test and Diagnosis Pattern Generation algorithm. Section 4 discusses the experimental setup and results comparison. Section 5 concludes the paper and section 6 explains the future scope of the proposed work.

2. Background

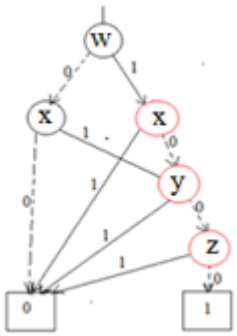
2.1. ZBDD

ZBDD is a compact representation of BDD. It is mentioned in [10] that, a BDD can be reduced to ZBDD without any loss of information by removing all those nodes with a True-edge pointing to a 0-leaf node. This is referred to as the pD-deletion rule [11]. Normally in a BDD if both the edges of a node point to the same node, then the node can be deleted [13]. But this rule is not applicable here. If a node is absent in ZBDD it is to be considered as having a value 0 while finding the cover. The pD-deletion rule is asymmetric in nature i.e., here a 0-edge pointing to a 0-leaf node is not deleted. ZBDD gives very compact representation for functions which are predominantly Zero [14]. However, the maximum size of a ZBDD would be that of its corresponding BDD. This type of BDD is called a zero-suppressed BDDs (ZBDDs). A theorem has been proposed in [12] which states that: For a fixed input order and domain, ZBDD can provide a unique representation of a Boolean function.

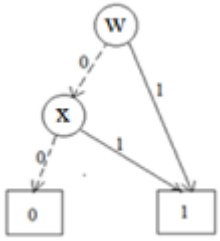
Figure 2(a) shows the BDD for a Boolean function $F(w,x,y,z) = wx'y'z' + w'xy'z'$. Here the nodes *x*, *y*, and *z* which are marked in red indicate the nodes whose True-Edge is pointing to a False-leaf node. Figure 2(b) shows the corresponding ZBDD for the above Boolean function *F*. Here all the nodes marked in red can be removed according to [10]. Hence a very compact representation of F is obtained. By traversing from the root node to 1-leaf node through all possible paths will give patterns that satisfy *F*. Unlike as in BDD, where a node is absent in the path then it means that both true-edge and false-edge are connected to the same next node and can be taken as a don't care bit. When it comes to obtaining patterns from ZBDD, it should be observed that if a node is absent in the trail from the origin node to the 1-leaf node then it indicates that the value of that variable should be a

logic-0. In ZBDD a variable is taken as a don't care only when both 0-edge and 1-edge are pointing to the same next node leading ultimately to 1-leaf node.

As shown in Fig. 2(b) node '*a*' is connected directly to the 1-leaf node and nodes *'x', 'y'* and *'z'* are not present in the path thus the pattern obtained will be 1000. The second path that leads to the 1-leaf node is when *w* is 0 and *x* is 1. In this path *y* and *z* are not present hence the pattern obtained is 0100. The truth table of *F* is shown in Fig. 2(c) for reference.



(a)



(b)

wxyz	F
0000	0
0001	0
0010	0
0011	0
0100	1
0101	0
0110	0
0111	0
1000	1
1001	0
1010	0
1011	0
1100	0
1101	0
1110	0
1111	0

(c)

Fig. 2. (a) BDD (b) ZBDD (c) Truth table for the function *F(w,x,y,z) = wx'y'z' +w'xy'z'*.

2.2. Distinguishing stuck-at fault pairs

Consider the following faults pairs {*n*5 *s-a*-1 (f_1), *n*4 *s-a*-0 (f_2)} and {*PI*1 *s-a*-0 (f_3), *n*5 *s-a*-1 (f_1)} as shown in Fig. 3. In order to generate a pattern to distinguish between two faults in a pair there are two possible conditions with their corresponding constraints:

FP_1: {generate test pattern for f_1; Constraint: *n*4=0,
generate test pattern for f_2; constraint: *n*5=1}
FP_2: {generate test pattern for f_1; constraint: *PI*1=0,
generate test pattern for f_3; constraint: *n*5=1}

It is easy to find a pattern for FP_2 as one of the constraints here is a primary input. Primary inputs have high controllability. But as we proceed into the circuit, the controllability decreases and hence finding patterns that satisfy the constraints becomes more and more difficult. A similar case is shown in FP_1, both the

constraints are internal nodes. An efficient method for generating patterns with constraints is proposed and the procedure is explained in the subsequent sections. In this method, while generating patterns for FP_1 it is also crosschecked whether there is any pattern that could be used in FP_2 also. This helps in reducing the diagnosis pattern volume.

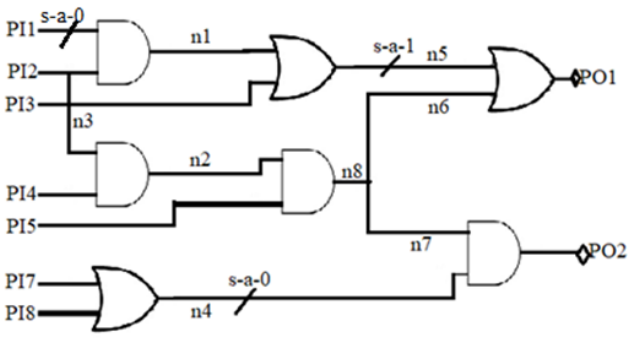


Fig. 3. A circuit with three faults at *PI*1, *n*4 and *n*5 respectively.

2.3. Observation point insertion

Observation points are generally added in a circuit to improve the test coverage. [15] A scan cell is used where an observation point has to be added. A sample scan cell is shown in Fig. 4. The inserted observation point is enabled using the Observation point enable (OPE) Signal. The intermediate signal from the CUD will act as an input here (input D) and is XORed with the scan input (SI). Clock Generator (CG) module is used to activate the entire module when needed [16]

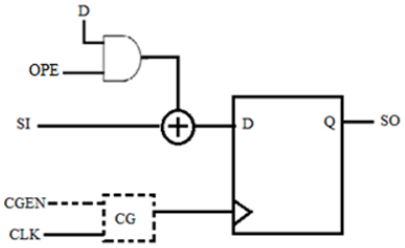


Fig. 4. A scan cell.

3. Proposed Work

3.1. Diagnosis pattern generation using ZBDD

Consider two fault pairs f_1 and f_2 shown in Fig. 5, where f_1 is an internal node n_1 stuck-at-0 and f_2 is an internal node n_2 stuck-at-1 in order to generate a pattern that distinguishes between the two, it should be able to activate f_1 and deactivate f_2 or vise-versa. In order to deactivate f_2 the true value and faulty value at the node n_2 should be the same. In this case, it should be a logic value 1. The pattern which gives a value 1 for node n_2 can be easily obtained by analysing the sub-ZBDD for n_2 and it can be clearly observed from Fig. 5 that only those primary inputs present

in the input projection cone need to be specified for deactivating the fault f_2. Such a pattern is called the Deactivation Pattern for fault f_2 (DAP_{f2}). Now any pattern which detects fault f_1 and is compatible with DAP_{f2} will be a diagnosis pattern.

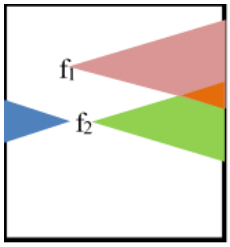


Fig. 5. Observation cones of faults f_1 and f_2.

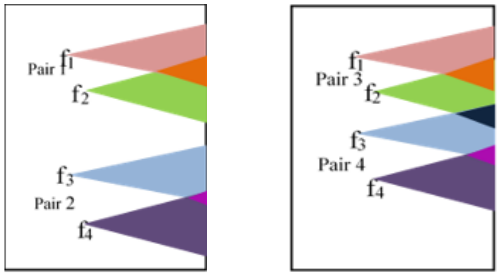
3.2. Fault pair grouping

If more than one fault pairs are considered simultaneously then the diagnosis pattern generation time can be reduced drastically. The fault pairs which satisfy any one of the following two conditions can be considered together.

Condition 1: If the fault pairs have disjointed or nonoverlapping output observation cones then they can be grouped together. As shown in Fig. 6(a) the fault pairs, pair 1 and pair 2 do not have any common outputs, so the propagation constraints are mutually exclusive. Thus, pair 1 and pair 2 can be grouped together.

Condition 2: if the fault pairs have semi overlapped observation cones then the pairs can be grouped together if one fault in a pair does not have any common outputs with the other pair. For example, consider Fig. 6(b) fault f_1 in pair 3 does not propagate to any of the outputs of pair similar is the case with fault f_4 and pair 3. In order to generate the diagnosis pattern f_1 and f_4 can be activated together while deactivating f_2 and f_3, thereby avoiding any possibility of conflicting propagation constraints.

With the help of the above two conditions, more fault pairs can be considered together thereby reducing the diagnosis pattern count drastically. In [1] the pairs which satisfy condition 2 are put in different groups. Even though the number of groups in [2] is less as compared to [1] a 2 × 1 multiplexer has to be inserted for all the fault pairs to be distinguished and an additional circuit is included to activate one group at a time. This adds to the complexity of the procedure.



(a) **(b)**

Fig. 6. (a) Fault pairs with non-overlapping outputs and (b) Fault pairs with overlapping outputs.

3.3. Fault pair equivalency check

With the help of ZBDD XOR operation patterns which activate a fault while deactivating the other is selected as a diagnosis pattern. If the procedure fails to generate a diagnosis pattern it indicates that there is no pattern that can deactivate any one of them while activating the other. This could be due to the following two reasons:

- One of fault in the pair is not being propagated to the output.
- The faults are producing a similar response at the output.

This condition arises frequently when both the faults are in close vicinity. This special case of very closely coupled faults are called Difficult to Distinguish Fault Pairs (DDFP). These faults may be easy to test but very difficult to distinguish. In order to solve this problem, additional observation points are to be added. Such a case is shown in Fig. 7 which shows a portion of a circuit with faults named as f_1 and f_2 on the primary inputs PI2 and PI5 respectively.

As highlighted with red in the Fig. 7, both the faults propagate to the same set of outputs PO1 and PO2. If at least one of the above-mentioned conditions arise for this fault pair then an observation point has to be added to the point just before the two faults interact. In this case, the observation point has to be added to O1. It is the input of the XOR gate which is common to both the faults.

In order to get a pattern that produces different responses at the output, an observation point has to be added at O1. The HDFPs have to be considered one by one. If fault f_1 cannot be observed at O1 then it indicates that the fault cannot be observed at all and the fault pairs are considered equivalent. The net O1 satisfies the above criteria and thus is a suitable node for observation point insertion. If the fault is not observable at the observation point O1, then the pair is considered as equivalent fault pair.

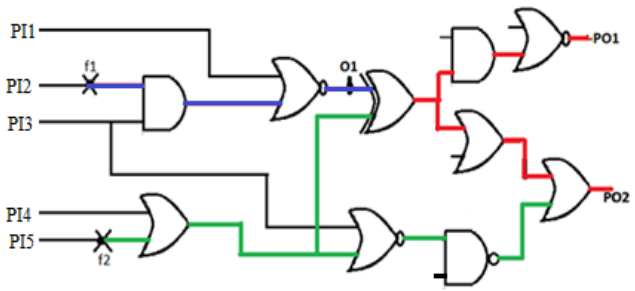


Fig. 7. A case study where faults f_1 and f_2 propagating to the same set of outputs.

The algorithm for selecting a node for the insertion of the observation point is shown in Fig. 8. Inputs to this algorithm are DDFP, list of gates N_1, in the propagation path of fault f_1 and list of gates N_2 in the propagation path of fault f_2. This algorithm is used to select the first gate with fanout which is common in the

propagation path of both faults f_1 and f_2. The next step is to identify the input of this gate which is not in the propagation path of f_2.

Input: DDFP, N_1, N_2;

Output: CUD with observation points

Initialize: X (a set of gates)

1) *consider* $\boldsymbol{g_i \epsilon N_1}$
2) *If* $\boldsymbol{g_i}$ *is a fanout and* $\boldsymbol{g_i \epsilon N_2}$ *goto step 3 else goto step 1*
3) $\boldsymbol{X}$ *<= gates connected to the input of* $\boldsymbol{g_i}$.
4) *If* $\boldsymbol{x_j \notin N_2}$ *got to step 5 else go to step 1*
5) *Insert observation point at the output of* $\boldsymbol{x_j}$

Fig. 8. Proposed algorithm for identifying the nodes for observation point insertion.

Whenever the ZBDD XOR operation and fault pair grouping operation fails to generate a diagnosis pattern this step has to be performed. The advantage of this procedure is that only a small portion of the circuit containing the fault pairs need to be considered for re-evaluation, thereby reducing the complexity further. If we cannot find a suitable point then the fault pair is considered to be indistinguishable and we abort the procedure. This happens when the faults in the pair under consideration are in close proximity (e.g. if the pair is at the output and one of the inputs of the same gate). However, in a collapsed fault list this scenario occurs very rarely.

3.4. Proposed test and diagnosis pattern generation algorithm (TDP generation algorithm)

Inputs to this algorithm are the circuit netlist, collapsed fault-list F, from F a list of fault pairs (FP) is generated, which is the complete list of all fault pairs that can occur because of only the collapsed faults. If there are n collapsed faults in a circuit then there will be ${}^nC_2$ possible fault pairs. Each step in the algorithm shown in Fig. 9 is explained below:

Step1: A test pattern t_i for a fault $f_i \in F$ is generated. Here a ZBDD XOR approach is followed where a True ZBDD and a Faulty ZBDDs are XORed to generate a Test ZBDD. Test patterns are then obtained from the generated Test-ZBDD.

Step2: This generated test vector may be helpful in detecting other faults also. Using a fault simulator all the faults detected by test pattern t_i are found out and removed from the original list F.

Step3: This test vector may also be helpful in diagnosing certain faults by deactivating certain other faults. Such patterns which have additional diagnostic capacity can used to remove some fault pairs from FP. These patterns are called Test and Diagnosis Patterns (TDPs).

Step4: This process is repeated until all faults in *F* are covered.

Step5 and *step6*: if any fault pairs are remaining in FP then group them according to the fault pair grouping procedure already explained. All the faults pairs that do not have conflicting propagation constraints can be injected together to generate their DPs.

Step7: One group is considered at a time. All fault pairs in a group are injected in the circuit simultaneously.

Step8: Supplementary patterns to distinguish the remaining fault pairs are generated. These additional patterns are called Diagnosis Patterns.

Step9: If DPs are generated for the group considered then remove all the corresponding fault pairs from FP If the additional patterns fpr diagnosis could not be generated at this stage then it indicates that the considered fault pair is either an equivalent fault pair or a DDFP.

Step 10: Check whether all the fault pairs are covered then go to *step 7*

Step11: If no diagnosis pattern could be generated by the above steps then perform fault equivalency check. For this, an observation point has to be added to a selected node in order to observe the fault effect at the inserted observation point. If different fault effects could be observed at the observation point and other primary output, then the faults can be distinguished otherwise they are termed as equivalent fault pairs.

Algorithm: Test and diagnosis pattern generation
Input: *Collapsed Fault-list,* ***F***
Circuit Netlist, ***M***
List of all possible fault pairs, ***FP***
Output*:* ***TDP***, ***DP***, *List of indistinguishable fault pairs* ***IFP***, *list of equivalent fault pairs,* ***EFP***.
Initialize: TDP, DP, IFP, EFP, DDFP
1) Consider a fault $f_i \in F$ *and generate a test pattern* t_i *for* f_i *using ZBDD XOR operation*
2) Remove other faults in ***F*** *that can be detected by* t_i.
3) Eliminate the pairs in FP that can be diagnosed by t_i *and add* t_i *to* ***TDP***
4) If all faults are covered go to step 5 else go to step 1.
5) If all fault pairs are covered then ***exit****; else go to step 6.*
6) Group the fault pairs that can be considered simultaneously.
7) If all groups are covered then exit; else go to step 8.
8) Generate additional patterns that can enable only one of the fault in the pair. Add the patterns obtained to ***DP****.*
9) If a diagnosis pattern is obtained then delete the fault pairs from ***FP*** *and go to step 8. Else delete the fault pair from* ***FP*** *and add it to* ***DDFP*** *and go to step 10*
10) If all fault pairs are covered then go to step 7.
11) Check if the faults are equivalent, if yes add the pair to ***EFP*** *else add to* ***IFP*** *and go to step 7.*

Fig. 9. Proposed TDP generation algorithm.

3. Results and Discussion

The proposed algorithm was implemented in C language. All the data structures needed to read the netlist and identifying the nodes for observation point insertion, grouping of fault pairs etc. was implemented in C language. The algorithm was run on a PC running at 1.8 GHz in Linux Operating System. The algorithm uses ATALANTA and HOPE simulators for generating the collapsed fault list and to verify the test and diagnostic coverage. ZBDD package from Colorado University Decision Diagram (CUDD) version 2.5.1 was used. The algorithm was applied to various ISCAS'85 and ISCAS'89 benchmark circuits.

Synopsys Design Compiler was used to synthesize the netlist of the benchmark circuits. A 45 nm standard cell library was used for this purpose.

Table 1 shows the consolidated results attained for ISCAS'85 and '89 benchmark circuits

In Table 1, the % reduction is calculated using the formula shown in Eq. (1):

$$\%reduction\ in\ (TP+DP) = \frac{\#(TP+DP)-\#TDP\ proposed}{\#(TP+DP)} \times 100\% \quad (1)$$

One thing to be noted from the above table is that time required for the circuit c3540 is very much higher than the one mentioned in [7]. This is due to the big size of its ZBDD. As shown in Table 2, column 2 and 3 shows the size of BDD and ZBDD in terms number of nodes. For circuit c3540 the size of its ZBDD is even bigger than its BDD. Due to this the TDP generation time for this circuit was very high. The fourth column in Table 2 shows the CPU time for ZBDD generation for each circuit. This is also evidently high for c3540.

Table 1. Test and diagnosis results of TDP generation algorithm on ISCAS'85 and '89 circuits.

Circuit	#Collapsed Faults	#Fault Pairs	CPU Time (s)	#TDP Proposed algorithm
c432	524	137026	0.734	54
c1355	1574	1237951	5.5	76
c1908	1879	1764381	14.7	114
c2670	2747	2943951	17.18	110
c3540	3428	5873878	132	140
c6288	7744	29980896	24.87	63
c7552	7550	28497475	26	230
s5378	5614	15755691	17.6	130
s9234	4984	12417636	24.2	210
s13207	6460	20862570	27.8	101
s15850	7430	27598735	34.6	133
s35832	18640	173715480	136.2	21
s38417	22379	250398631	144.8	110
s53858	23782	282779871	154.2	148

Table 2. CPU execution time for ZBDD generation.

Circuit	#BDD Nodes	#ZBDD Nodes	CPU Time for ZBDD generation (msec.)
c432	8190	3318	40
c1355	6251	2192	20
c1908	18228	2262	30
c2670	14024	2313	30
c3540	697674	878920	8710
c6288	368201	275395	6500
c7552	25332	10220	100
s5378	6551	2192	20
s9234	103222	3150	10
s13207	36758	11564	123
s15850	44256	9943	70
s35832	56778	10042	78
s38417	49993	13580	130
s53858	74120	23565	5300

Table 3 shows number of pairs remaining after executing step 4 of test and diagnosis pattern generation algorithm. None of the faults were aborted for ISCAS'85 and 89 circuits, which means that all the pairs which could not be distinguished using this approach were found to be equivalent. Last column indicates the number of back tracking used to find the TDP. Maximum backtracking was limited to 10.

Table 3. Number fault pairs for which additional diagnosis patterns have to be generated and maximum backtracking.

Circuit	#Remaining Pairs	Maximum backtracking
c432	161	1
c1355	29	1
c1908	132	2
c2670	221	3
c3540	912	10
c6288	415	6
c7552	62	1
s5378	34	1
s9234	98	1
s13207	198	1
s15850	231	3
s35832	919	4
s38417	362	2
s53858	898	10

The proposed TDP algorithm selects test patterns based on their diagnostic capability. So, the number of fault pairs for which additional patterns are to be generated will be less. This can be seen in Fig. 10 which shows that the number of remaining fault pairs in proposed algorithm is considerably lesser than [7].

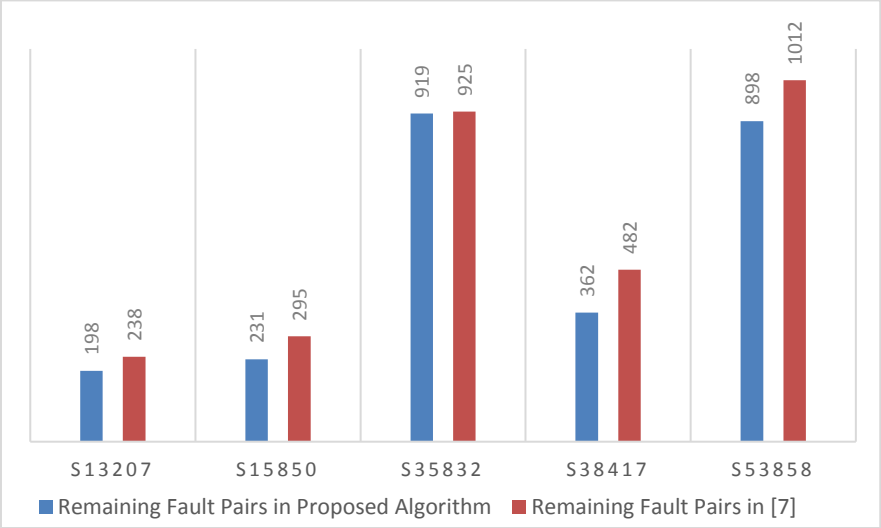


Fig. 10. Comparison of number of fault pairs for which additional diagnosis patterns have to be generated.

Comparison of the proposed work with some earlier work in terms of number of TDP is shows in Table 4. In all cases the proposed work gives a substantial improvement in number of patterns. There is an average 25.28% reduction in the total number of test and diagnosis patterns as compared to [5], 25.01 % as compared to [9] and 33.6% as compared to [17].and its comparison with the most recent work in [7]. The CPU execution time obtained for the proposed algorithm is slightly higher than that in [7]. It is because the processor used by Wu et al. [7] is twice as fast as the one used here. Also, the time reported by them is only the time for generating the DPs whereas here it is total time for test and diagnosis pattern generation. But it can be observed here that the total number of test and diagnosis pattern counter is lesser in the proposed work as compared to [7].

Table 4. Comparison of TDP generation algorithm results for ISCAS'85 circuits with [5, 7, 9, 17, 18].

Circuit	#TDP (Proposed)	#(TP+DP) in [5]	#(TP+DP) in [7]	#(TP+DP) in [9]	#(TP+DP) in [17]	#(TP+DP) in [18]
c1355	76	106	83	87	-	192
c1908	114	134	120	134	121	210
c2670	110	194	115	150	124	242
c3540	140	221	148	174	141	264
c6288	63	72	67	137	195	64
c7522	230	274	234	296	1504	393

Table 5 shows that there is on an average 29.9% reduction in test and diagnosis patterns as compared to [2], 8.56% reduction as compared to [3] and 10.15% reduction as compared to [19]. The proposed algorithm is able to attain 100% diagnostic coverage while achieving 10% reduction in pattern count as compared to [6] and 28% as compared to [7]. Even though the diagnostic coverage reported in [17] is 50%-90%, the pattern count obtained in the proposed method is 66% lesser than the pattern count reported in [17].

Table 5. Comparison of TDP generation results for ISCAS'89 circuits with [2, 3, 6, 7, 17, 19].

Circuit	#TDP (Proposed)	#(TP+ DP) in [2]	#(TP+ DP) in [3]	#(TP+ DP) in [6]	#(TP+ DP) in [7]	#(TP+DP) in [17]	#(TP+DP) in [19]
s5378	130	153	151	149	130	-	144
s13207	101	322	256	118	251	304	278
s15850	133	239	145	146	152	-	172
s35832	21	64	54	34	27	83	29
s38417	110	469	203	119	137	316	137
s53858	148	752	183	150	198	-	168

4. Conclusion

This work proposes an algorithm to produce test patterns and diagnosis patterns simultaneously for all possible stuck-at faults in a circuit. Here a ZBDD based approach is used to generate test and diagnosis patterns. The proposed algorithm does not rely on an initially generated test pattern set to generate the additional patterns needed for fault identification. Here diagnosis patterns are generated during the testing stage itself. Here a 100% distinguishability is observed for all non-equivalent fault pairs in ISCAS'85 and 89 benchmarks circuits considered. From the results obtained it can be observed that only c3540 circuit takes a considerably high amount of time due to its substantially bigger sized ZBDD. Other than this TDPs could be generated in less time and the diagnosis volume is also very less as compared to other diagnosis pattern generation methods.

In order to reduce the pattern generation time multiple fault pairs are considered simultaneously and circuit modification is avoided. This has resulted in an added advantage of reduced pattern count. The results obtained were compared with other existing methods and a substantial improvement in test volume was observed. Here all the fault pairs possible in a circuit are taken into consideration and none of the faults were aborted.

5. Future Scope

Only stuck-at faults were considered in this paper however the methodology can be extended to other types of faults like bridging faults and transition faults as well. The method of deactivating one fault while activating the other fault using ZBDD can be used to identify hybrid fault pairs like stuck-at and bridging fault. Most of the diagnosis analysis tools often misinterpret a bridging fault for a stuck-at fault. But with small modifications, this method can be used to distinguish bridging fault from stuck-at fault. In addition to that, test pattern reordering method can also be applied here to reduce the test power.

Nomenclatures

f_i	Fault in a Circuit
g_i	A gate in a circuit
N_i	A node in a circuit
t_i	A test pattern

Abbreviations

ATPG	Automatic Test Pattern Generator
CPU	Central Processing Unit
CG	Clock Generator
CUD	Circuit Under Diagnosis
CUT	Circuit Under Test
DC	Defect Candidates
DDFP	Difficult to Distinguish Fault Pairs
DP	Diagnosis Patterns
DPG	Diagnosis Pattern Generator
TDP	Test and Diagnosis Pattern
TP	Test Patterns
ISCAS	IEEE International Symposium on Circuits and Systems
OPE	Observation Point Enable
SI	Scan Input
UDFM	User Defined Fault Models
ZBDD	Zero suppressed Binary Decision Diagram

References

1. Pomeranz, I.; and Reddy, S.M. (2009). Selection of a fault model for fault diagnosis based on unique responses. *Institute Electrical and Electronics Engineers (IEEE) Transactions on Very Large Scale Integration (VLSI) Systems*, 18(11), 1533-1543.
2. Pomeranz, I; and Reddy, S.M. (2010). Output-dependent diagnostic test generation. *23rd International Conference on Very Large Scale Integration (VLSI) Design*. Bangalore, India, 3-8.
3. Prabhu, S.; Hsiao, M.S.; Lingappan, L.; and Gangaram V. (2012). A SMT-based diagnostic test generation method for combinational circuits. *Institute Electrical and Electronics Engineers (IEEE) 30th Very Large Scale Integration (VLSI) Test Symposium (VTS)*. Hyatt Maui, USA, 215-220.
4. Agrawal, V. D.; Baik, D. H.; Kim, Y. C.; and Saluja, K. K. (2003). Exclusive test and its applications to fault diagnosis. *Proceedings 16th International Conference on Very Large Scale Integration (VLSI) Design*. New Delhi, India, 143-148.
5. Gruning, T.; Mahlstedt, U.; and Koopmeiners, H. (1991). DIATEST: A fast diagnostic test pattern generator for combinational circuits. *Institute Electrical and Electronics Engineers (IEEE) International Conference on Computer-Aided Design Digest of Technical Papers*. Santa Clara, USA, 194-197.
6. Wu, C. H.; Lee, K. J.; and Lien, W.C. (2014). An efficient diagnosis method to deal with multiple fault-pairs simultaneously using a single circuit model.

Institute Electrical and Electronics Engineers (IEEE) 32nd VLSI Test Symposium (VTS). Napa, USA, 1-6.

7. Wu, C.; Lee, K.J.; and Reddy, S.M. (2019). An efficient diagnosis-aware ATPG procedure to enhance diagnosis resolution and test compaction. *Institute Electrical and Electronics Engineers (IEEE) Transactions on Very Large Scale Integration (VLSI) Systems*, 27(9), 2105-2118.
8. Pomeranz, I.; and Reddy, S.M. (2009). Equivalence, dominance, and similarity relations between fault pairs and a fault pair collapsing process for fault diagnosis. *Institute Electrical and Electronics Engineers (IEEE) Transactions on Computers*, 59(2), 150-158.
9. Zhang, Y.; and Agrawal, V.D. (2010). A diagnostic test generation system. *Institute Electrical and Electronics Engineers (IEEE) International Test Conference*. Austin, USA, 1-9.
10. Minato, S.I. (1993). Zero-suppressed BDDs for set manipulation in combinatorial problems. *Proceedings of the 30th International Design Automation Conference*. Dallas Texas, USA, 272-277.
11. Drechsler, R.; and Sieling, D. (2001). Binary decision diagrams in theory and practice. *International Journal on Software Tools for Technology Transfer*, 3(2), 112-136.
12. Minato, S.I. (2001). Zero-suppressed BDDs and their applications. *International Journal on Software Tools for Technology Transfer*, 3(2),156-170.
13. Zhongliang, P.; Ling, C.; and Guangzhao, Z. (2006), A new method for the detections of multiple faults using binary decision diagrams. *Wuhan University Journal of Natural Sciences*, 11(6), 1943-1946.
14. Mohan, N.; and Anita, J.P. (2016). A zero suppressed binary decision diagram-based test set relaxation for single and multiple stuck-at faults. *International Journal of Mathematical Modelling and Numerical Optimisation*, 7(1), 83-96.
15. Milewski, S.; Mukherjee, N.; Rajski, J.; Solecki, J.; Tyszer, J.; and Zawada, J. (2017). Full-scan LBIST with capture-per-cycle hybrid test points. *Institute Electrical and Electronics Engineers (IEEE) International Test Conference (ITC)*. Tokyo, Japan, 1-9.
16. Sruthi, P. R.; and Devi, M. N. (2012). A modified scheme for simultaneous reduction of test data volume and testing power. *Progress in VLSI Design and Test*. Berlin, Heidelberg, 198-208.
17. Pradhan, M.; and Bhattacharya, B.B. (2017). COMEDI: Combinatorial election of diagnostic vectors from detection test sets for logic circuits. *Institute Electrical and Electronics Engineers (IEEE) Transactions on Very Large Scale Integration (VLSI) Systems*, 25(4), 1467-1476.
18. Veneris, A.; Chang, R.; Abadir, M. S.; and Amiri, M. (2004). Fault equivalence and diagnostic test generation using ATPG. *Institute Electrical and Electronics Engineers (IEEE) International Symposium on Circuits and Systems*. Vancouver, Canada, 221-224.
19. Ye, J.; Zhang, X.; Hu, Y.; and Li, X. (2010). Substantial fault pair at-a-time (SFPAT): An automatic diagnostic pattern generation method. *19th Institute Electrical and Electronics Engineers (IEEE) Asian Test Symposium*. Shanghai, China, 192-197.