A Deterministic Approach for Diagnosis Test Generation - Further Optimizations

Pavlinka Goranova Radoyska

Abstract: **In the paper "A Deterministic Approach for Diagnosis Test Generation" was presented algorithm for diagnosis test pattern generation with polynomial complexity and deterministic nature. In those paper are given some optimizations and experimental results for the subalgorithms 3, 4 and 5, presented in that paper as wall as the method for control the distinguishability.**

Keywords: **algorithm, digital circuits, test pattern generation, fault diagnosis, deterministic, z-set.**

I. INTRODUCTION

The most of the circuit failure diagnostic methods are simulation based. These methods can be classified in two main classes: cause-effect [1], [2], [3], [4], [5] and effect-cause [6], [7], [8]. Well build test patterns are significant for methods efficiency. In the paper "A Deterministic Approach for Diagnosis Test Generation" has proposed an algorithm for diagnosis test pattern generation with deterministic nature. This algorithm is based on single stuck-at fault model. Its aims are to build the better conditions for any of fault diagnosis methods. The algorithm consists of three main steps: (1) build the collection of all test patterns for every group of equivalent faults; (2) minimize the number of test patterns by merging the compatible test patterns; (3) minimize the number of test patterns by extracting the redundant test patterns. They are realized by five sub-algorithms.

The step (1) is performed by subalgorithm1 and subalgorithm2. Subalgorithm1 describes the steps for building the collections of input patterns L_l^v, which can force the line *l* to level *v*. Subalgorithm2 describes the steps for building the collections of test patterns T_a^v, which can detect the fault f_a. The test patterns in L_l^v and T_a^v are in 3-valent logic ('0', '1' and 'x', which means "doesn't matter"). Test pattern reduction is made after any calculation cycle and for every collection. The reduction is made by absorbing. If one test pattern becomes covering for the other, the first one can absorb the second and in the collection rests only the first test pattern. Covering is defined as follows: if there are two test patterns t_i and t_j and if for any bit in t_i and t_j the next statements are true: $t_i[b]=t_j[b]$ or $t_i[b]=$'x'; then the t_i becomes the covering test pattern for t_j and can absorb it.

Subalgorithm3 resolves the task for improving distinguishability between faults with the same observation point and compatible test patterns. The test patterns t_i and t_j are compatible if for any corresponding bit in t_i and t_j, one of the next statements is true: $t_i[b]=t_j[b]$ or $t_i[b]=$'x' or $t_j[b]=$'x', where *b* is a bit index. The idea is to set one of 'x' values in one of compatible pairs in complementary level, according corresponding bit in the other test pattern of the pair.

The step (2) is performed by subalgorithm4. As the first three subalgorithms decide the problem of finding all possible test patterns for detecting every fault in the fault dictionary and to guarantee distinguishability between them, the next two subalgorithms decide the problem of minimizing the final diagnosis test set T_{res}. In subalgorithm4 the pairs of compatible test patterns are replaced with common one. After this the total number of unique test patterns is reduced without disturbing the distinguishability.

The step (3) is performed by subalgorithm5. This subalgorithm builds the minimal test pattern collection for every fault, which guarantee distinguishability and make the final diagnosis test set T_{res} by summarizing these collections and extracting the duplicated test patterns.

The rest of the paper is organized as follows. In Section 2, are given analysis, some experiment results and further optimizations for algorithm 3. In Section 3, are given analysis, some experiment results and further optimizations for algorithm 4. In Section 4, are given analysis, some experiment results and further optimizations for algorithm 5. In Section 4, are given dictionary based method for fault diagnosis. The experiments are performed on benchmark circuits: c17, 74182, 74283 and 74L85.

II. SUBALGORITHM3 ANALYSIS, EXPERIMENTS AND OPTIMIZATIONS

Subalgorithm2 builds the collection *D* of triples $d_i =< f_i, t_i, FO_i >$, for every fault f_i, test patterns t_i, which detect this fault and corresponding fail output FO_i. If fault effect for one test pattern and one fault can be observed on more then one output, for every output are made different triple d_i.

Subalgorithm3. Improve distinguishability for the faults, observing on the same output and having the compatible test patterns. This algorithm follows the next steps:

1. For every primary output FO_x make D collections for every fault pairs f_a and f_b $D_a = \{\forall d_i \in D : d_i =< f_i, t_i, FO_i >, f_i = f_a, FO_i = FO_x\}$ and $D_b = \{\forall d_i \in D : d_i =< f_i, t_i, FO_i >, f_i = f_b, FO_i = FO_x\}$
2. If in D_a and D_b there is at least one incompatible test pattern, f_a and f_b are distinguish, take the other pair. Otherwise for the one of compatible pairs t_i, t_j change one

P.Radoyska is with the College of Energetic and Electronics at Technical University - Sofia, 31 Bulgaria blvd., 2140 Botevgrad, Bulgaria, e-mail: pradoiska@abv.bg, GSM:+359 895 589 981

of 'x' levels to alternative value, so that t_i and t_j becomes incompatible.

There are two points for investigation and optimization at this algorithm: (1) test pattern election for compatible pairs and (2) a bit with 'x' value for change election. For fault election can be proposed three principles:

- pseudo-random (the first test pattern and the first fault in compatible pair);
- the test pattern with maximum number of 'x' values into two faults test pattern collections;
- the test pattern with minimum number of 'x' values into two faults test pattern collections.

For 'x' value for change election can be proposed three principles:

- Pseudo-random (the first 'x' with corresponding '0' or '1' level in the other test pattern in the pair).
- Reducing number of test patterns (the bit, which makes the test pattern suitable for absorption).
- Expanding number of test patterns (the bit, which makes the test pattern unique for the collection of test patterns, detecting this fault).

Table 1. Experimental results for test pattern election

	c17	74182	74283	74L85
Number of Inputs	5	9	9	11
Number of Outputs	2	5	5	3
Fault Dictionary size	22	83	128	105
Number of fault pairs	240	1400	1600	4849
Number of potentially undistinguished pairs	9	97	4	4
Undistinguished pairs percentage	3.75%	6.93%	0.25%	0.08%
Number of diagnosis test patterns				
- pseudo-random	15	48	49	74
- min	16	47	49	73
- max	16	49	48	74

In the table 1 are shown experimental results after applying the mentioned before three types of test pattern election: pseudo-random, maximum number of 'x' values and minimum number of 'x' values. The conclusion that can be made upon this experiment is: the order of test pattern election has not significant effect on the size of final diagnosis test pattern collection. It is due to the quite low percentage of the compatible pairs.

The experimental result on the methods for election the bit with 'x' value, give the similar results, due to the same considerations.

The effect of this algorithm is not so high but it is very important to guarantee the faults distinguishability. To keep low computation complexity, pseudo-random approach is preferred.

III. SUBALGORITHM4 ANALYSIS, EXPERIMENTS AND OPTIMIZATIONS

Subalgorithm 4. Test patterns for faults with different observation points merging. This algorithm follows the next steps:

1) For any primary output make the collection of unique test patterns T_j that can detect any fault on this observation point. If D collection for primary output a is $D_a = \{\forall d_i \in D : d_i =< f_i, t_i, FO_i >, FO_i = FO_a\}$, then $T_a = \{\forall t_i \in D_a\}$, where $a = 1 \div outputs\_number$.

2) Look for compatible test patterns $t_i \in T_a$ and $t_j \in T_b$ and replace them with common test pattern t_{com}, according to the next rules:

0 & x = 0 1 & x = 1 1 & 0 = ? (conflict)
x & 0 = 0 x & 1 = 1 0 & 1 = ? (conflict)

If in any bit in the common test pattern there is a conflict, this pattern discards.

It is important how to choose the compatible patterns so that the resulting number of unique test patterns becomes minimal. Three different functions are written to optimize this sub-algorithm. In the first function are juxtaposed the collections with pseudo-random test pattern order (first come, first compared). In the second function the collections are ascending sort, based on filling (the number of non-'x' levels). In the third function the collections are also sorted, but the first collection is sorted in ascending and the second- in descending manner.

Table 2. Experimental results for test pattern merging

	c17	74182	74283	74L85
Number of Inputs	5	9	9	11
Number of Outputs	2	5	5	3
Fault Dictionary size	22	83	128	105
Number of merged test patterns				
- pseudo-random	149	1793	7261	68342
- ascending sort	149	1891	7261	72333
- ascending-descending sort	110	1469	7292	20806
Number of diagnosis test patterns				
- without merging	15	48	49	74
- pseudo-random	13	44	36	63
- ascending sort	12	40	35	58
- ascending-descending sort	15	43	36	60

In the table 2 are shown experimental results after applying the mentioned before three functions. Any of the function reduces the size of diagnosis test pattern collection. More over it reduce the number of unique $<t_i, FQ>$ pairs, which reduce the operations in the subalgorithm5. The number of operations during the juxtaposition is proportional on n^2, where n is the average number of unique test patterns, which can detect any fault on given output.

From this table it can be seen that the test pattern merging is important procedure for reducing the size of the diagnosis test pattern collection. The best results are received while the incoming tests pattern collections are ascending sort. This is because the possibilities of merging are highest.

IV. SUBALGORITHM5 ANALYSIS, EXPERIMENTS AND OPTIMIZATIONS

Subalgorithm5. Build the final test pattern collection T_{res} by collecting the minimal diagnosis test patterns for every fault. This algorithm follows the next steps:

1) For every $f_a \in F$ make

$D_a = \{\forall d_i \in D : d_i =< f_i, t_i, FO_i >, f_i = f_a\}$.

2) For every unique $<t_i, FQ>$ pair in the D_a make the collections F_i^a of detecting faults.

3) For every fault f_a make an intersection $K = \bigcap F_i^a$, until in *K* remains only f_a. For every F_i^a add t_i to the T_{res}.

4) Minimize the collection T_{res} by extracting the duplicated test patterns.

The critical point of this algorithm is step 3) – the intersection making. The cardinality of final test patterns, that are possible to distinguish the fault f_a, is in strong dependence of the F_i^a collections order.

Let have the fault f_b and four F_i^b collections with the same cardinality: $F_0^b = \{f_a, f_b, f_c\}$, $F_1^b = \{f_a, f_b, f_d\}$, $F_2^b = \{f_a, f_b, f_e\}$ and $F_3^b = \{f_a, f_c, f_e\}$. The intersection makes in three steps:

1) $K = F_0^b \bigcap F_1^b = \{f_a, f_b, f_c\} \bigcap \{f_a, f_b, f_d\} = \{f_a, f_b\}$

2) $K = K \bigcap F_2^b = \{f_a, f_b\} \bigcap \{f_a, f_b, f_e\} = \{f_a, f_b\}$

3) $K = K \bigcap F_3^b = \{f_a, f_b\} \bigcap \{f_a, f_c, f_d\} = \{f_a\}$

It is seen that step 2) don't change the members and cardinality of collection *K*. Hence it is unnecessary to add test pattern for F_2^b collection in the T_{res}.

Let have the fault f_a and three F_i^a collections, respectively: $F_0^a = \{f_a, f_b\}$, $F_1^a = \{f_a, f_b, f_c\}$, $F_2^a = \{f_a, f_c\}$. If the intersection is made in the index order, in the process of intersection must take part all F_i^a collections:

$K = F_0^a \bigcap F_1^a = \{f_a, f_b, f_c\} \bigcap \{f_a, f_b\} = \{f_a, f_b\}$,

$K = K \bigcap F_2^a = \{f_a, f_{bc}\} \bigcap \{f_a, f_c\} = \{f_a\}$.

Respectively in the T_{res} are added three test patterns. But if we change the order and start from collection F_1^a, only two test patterns will be added to the T_{res}:

$K = F_1^a \bigcap F_2^a = \{f_a, f_b\} \bigcap \{f_a, f_c\} = \{f_a\}$.

Hence it is suitable to order the F_i^a collections in ascending order in respect of their cardinality and after that make the intersections.

This is the next question. If there is a test pattern t_j with $F_j^a = \{f_a\}$, when the final test pattern set will be the minimal: when includes the test patterns, such as t_j, which detect only one fault, or when includes more test patterns for every fault, which detect several faults.

In the table 3 are shown experimental results after applying the mentioned before three functions. The best results are registered after descending sort the collections upon them size. For this approach if one test fails, it detects a lot of faults, but if one test passes, it excludes of fault candidate collection a lot of faults. This test pattern collection is suitable for manufacturing testing, because of it compactness and high detectable power.

Table 3. Experimental results for making the intersections

Number of diagnosis test patterns	c17	74182	74283	74L85
- random	12	40	35	58
- sort – descending	11	29	34	53
- sort – ascending	12	35	37	53

V. MODULE FOR FAULT DIAGNOSIS

Test diagnosis dictionary $TDD = \{p_0, p_1, ..., p_n\}$, where $p_i =< test\_pattern, fail\_output >$.

The module for fault diagnosis is written for controlling the effectiveness of generated diagnosis test pattern. This method is dictionary based and performs single stuck-at fault diagnosis. Its aims are to control the prerequisites for fault diagnostic in real processes. Diagnosis algorithm, realized in this module includes the next steps:

1) Build the collection of failing test pairs $Fail = \{p_0, p_1, ..., p_n\}$, where $p_i =< test\_pattern, fail\_output >$

2) Build the collection of pass test pairs $Pass = \{p_0, p_1, ..., p_m\}$ ($Pass = TDD - Fail$, where *TDD* is the full collection og diagnosis pair).

3) For every fault f_i in fault dictionary do

a) *flag_for_Adding*=false, *flag_for_Removing*=false

b) for every test pair tp_j that detects f_i do

i) for every fail pair $p_i \in Fail$ do

if $tp_j \equiv p$ => *flag_for_Adding*=true

ii) for every fail pair $p_i \in Pass$ do

if $tp_j \equiv p$ => *flag_for_Removing* =true

c) if *flag_for_Adding*=true and *flag_for_Removing*=false then add f_i to *Candidates*.

At the and in the *Candidates* collection, for single stuck-at fault solutions, there must be only one fault. This algorithm is applicable for multiple fault solutions, but as a result in the *Candidates* collection there will be more then one candidate fault.

The experimental results show that the fault distinguishability is good. For the used benchmark circuits it varies between 95% and 100%. This means that the algorithm for diagnosis test pattern generation is effective and can improve fault diagnosis on manufacturing stage.

VI. CONCLUSION AND FUTURE WORK

The next conclusions can be made about the subalgorithms after providing the optimizations and experiments.

The effect of subalgorithm3 (improve distinguishability between faults with the same observation point) on the size of final test pattern collection is not high, but it is important for improving distinguishability, checked in section 5. The pseudo-random approach for pattern and 'x' position electing is preferred.

The test pattern merging, done in subalgorithm4, is important procedure for reducing the size of the diagnosis test pattern collection. The best results are received while the incoming tests pattern collections are sort in ascending direction.

For making the minimal diagnosis test pattern collection for every fault, done in subalgorithm5, checking the collection, which are sorted in descending direction on their size, gave the best results.

The experimental results show that the fault distinguishability of the algorithm is in an acceptable level and make good circumstances for manufacturing tests and fault diagnosis.

The main disadvantages of this algorithm are two: it is memory huge and is single stuck-at fault oriented. The first disadvantage can be resolve by storing the temporary collections in the file or data base, which is in process. This algorithm is a first step for the most general algorithm, which aim is diagnosis test pattern generation for multiple faults with masking effect.

Acknowledgements

This work was supported by the TU-Sofia, project number № 091ni009-10.

REFERENCES

[1] Bernardi, P.; Grosso, M.; Rebaudengo, M.; Sonza Reorda, M., "A pattern ordering algorithm for reducing the size of fault dictionaries," *VLSI Test Symposium, 2006. Proceedings. 24th IEEE* , vol., no., pp.6 pp.-391, April 30 2006-May 4 2006

[2] Pomeranz, I.; Reddy, S.M., "A Same/Different Fault Dictionary: An Extended Pass/Fail Fault Dictionary with Improved Diagnostic Resolution," *Design, Automation and Test in Europe, 2008. DATE '08* , vol., no., pp.1474-1479, 10-14 March 2008

[3] Bartenstein, T.; Heaberlin, D.; Huisman, L.; Sliwinski, D., "Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm," *Test Conference, 2001. Proceedings. International* , vol., no., pp.287-296, 2001

[4] Polian, I.; Miyase, K.; Nakamura, Y.; Kajihara, S.; Engelke, P.; Becker, B.; Spinner, S.; Xiaoqing Wen; Diagnosis of Realistic Defects Based on the X-Fault Model, Design and Diagnostics of Electronic Circuits and Systems, 2008. DDECS 2008. 11th IEEE Workshop on 16-18 April 2008 Page(s):1 - 4

[5] Takamatsu, Y.; Seiyama, T.; Takahashi, H.; Higami, Y.; Yamazaki, K., "On the fault diagnosis in the presence of unknown fault models using pass/fail information," *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on* , vol., no., pp. 2987-2990 Vol. 3, 23-26 May 2005

[6] Takahashi, H.; Boateng, K.O.; Saluja, K.K.; Takamatsu, Y., "On diagnosing multiple stuck-at faults using multiple and single fault simulation in combinational circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol.21, no.3, pp.362-368, Mar 2002

[7] Rousset, A.; Bosio, A.; Girard, P.; Landrault, C.; Pravossoudovitch, S.; Virazel, A., "DERRIC: A Tool for Unified Logic Diagnosis," *European Test Symposium, 2007. ETS '07. 12th IEEE* , vol., no., pp.13-20, 20-24 May 2007

[8] Seshadri, B.; Yu, X.; Venkataraman, S.; Accelerating diagnostic fault simulation using z-diagnosis and concurrent equivalence identification, VLSI Test Symposium, 2006. Proceedings. 24th IEEE, April 30 2006-May 4 2006 Page(s):6 pp. - 385