ANNUAL JOURNAL OF ELECTRONICS, 2009, ISSN 1313-1842

A Deterministic Approach for Diagnosis Test Generation

Pavlinka Goranova Radoyska

Abstract: **The main purpose of the diagnosis test pattern generation is to give the enough information for fault diagnosis during the manufacturing time. The most of the test generation methods are simulation based and have probabilistic nature. In this paper I propose the deterministic approach, which guarantees fault distinguishability. Algorithm is single stuck-at fault based. I perform some optimization on algorithms procedures, so that this algorithm obtains polynomial complexity and return the minimal test set.**

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

I. Introduction

Fault diagnosis is used to localize device failures. Followed by failure isolation and analysis, fault diagnosis can provide information about the defect location and defect mechanism present in the circuit and be used to improve the manufacturing yield. The result of fault diagnosis is a small set of faults which can explain an erroneous response. The main approaches for fault diagnosis are two: cause-effect and effect-cause. The cause-effect methods are dictionary based. Algorithms from this group build the simulation-response databases for the modeled faults and compared these databases with the observed failure responses to determine probable cause of the failures. This class of algorithms uses fault simulations on the design time and fills a data base with failing information. They are time consuming while produce the decisions on the manufacturing stage. The both combinational and sequential circuits are handled in the same way. The main disadvantage of this approach is the size of the fault dictionaries – the huge data volume is generated by the fault simulator. There are many decisions for compact dictionary building [1], [2]. The other disadvantage is necessity of accurate description of the fault models, used during fault simulation. Recently are developed some algorithms that not required a detailed fault model description [3], [4], [5].

Effect-cause methods are simulation based [6], [7], [8]. They use actual responses analyzing to determine which faults might have caused the observed failure effect. This class of algorithms uses backtracking from each primary output to determine the error propagation paths for all possible fault candidates. Fault simulations are made for every failing test pattern and any candidate. The aim is to find these candidates, which better explain the failing results. The main advantage of this method is that it does not need to explicitly consider each fault model during the identification of critical lines. This method is memory consuming, but is time greedy. There are developed many algorithms for reducing the diagnosis time, such as Incremental fault diagnosis [9], Z-diagnosis [10], [11], A-diagnosis [12].

Well build test pattern in diagnosis aspect are important for the both of the method types. In the many of the mentioned before papers [2], [8], [9], especially for effect-cause methods, the problem of test pattern generation doesn't treed. In others [4], [10], [13] a standard ATPG tools whit some constrains and added possibilities are used. Some of them [14] use ATPG tools and n-detection principles and hope that if the fault is detected n times, the diagnosis possibilities become high. In this work I propose deterministic approach for test pattern generation with fault diagnosis purpose. My aim is to select optimal set of test patterns that guarantee the faults distinguishability. To achieve this I follow three 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. The rest of the paper is organized as follows. In Section 2, are given some definitions, requisite for further explanations. In Section 3, the algorithm, subalgorithms and some optimizations are presented. In Section 4, is made conclusion and presented future plans.

II. Some Definitions

Collapsed dictionary (***F***) is a fault dictionary without any equivalent faults ($F \subset F_0$). Every collection of equivalent faults is presented by one of them.

Fail output (FO) is the output of the CUD (Circuit Under Detection), which value is different from the expected.

Fail pattern (**Fail test**) is a test pattern for which the values at one ore more outputs of the CUD are different from those of the expected fault-free circuit.

Indistinguishable faults. Two faults f_i and f_j of a given circuit are indistinguishable if for any input sequence they have the same collection of fail outputs. Otherwise, they are **distinguishable**.

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.

Z-set is a collection of primary outputs where one fault could be detected. *Z(f)* for the fault *f* on the line *l* is the set of outputs such that there is a directed path from line *l* to each of them. Z-set is a collection of sensitive outputs. The z-sets are independent of the test set used and the type of the fault. Therefore we can write *Z(f)=Z(l)*.

For illustration, we consider the circuit of figure 1. The Z-sets for faults f_1, f_2 and f_3, shown on figure 1 are $Z(f_1) = \{z_1\}$; $Z(f_2) = \{z_2\}$; $Z(f_3) = \{z_1, z_2\}$. Z-set is a parameter, which

---

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

can be used to distinguish faults. Consider the example on figure 1 the two faults f_1 and f_2 are distinguishable since $Z(f_1) \cap Z(f_3) = \varnothing$.

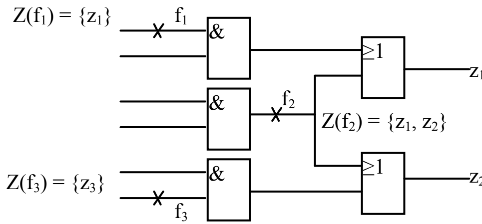

Figure 1. Using z-sets to distinguish faults

Z-set for a line l can described as a vector $Z_l = \{Z_l(0)\ Z_l(1)\ Z_l(2) \ldots\ Z_l(n-1)\}$, where n is a number of outputs, $Z_l(j)=1$ if $z_l \in Z_l$ and $Z_l(j)=0$ if $z_l \notin Z_l$. Consider the example $Z(f_1)=\{10\}$, $Z(f_2)=\{11\}$ and $Z(f_3)=\{01\}$.

Z-set cardinality is the number of sensitive outputs, respectively the number of the digits 1 in Zset. The cardinality of $Z(f_i)$ can be marked as $Cz_i=|Z(f_i)|$. For example considering the figure 1 $Cz_1=1$, $Cz_2=2$ and $Cz_3=1$.

III. SUBALGORITHMS AND OPTIMIZATIONS

Let go back to the classical D-algorithm [15] for single stuck-at fault test pattern generation. According to this algorithm we can build a decision graph and find the entire test patterns $T_a = \{t_1, t_2, ..., t_n\}$, which can detect single stuck-at fault $f_a \in F$. More over we can find fail outputs FO_j for every test pattern $t_j \in T_a$. Then for all faults in the collapsed dictionary and all test patterns we can build the set D of triples $d_i =< f_i, t_i, FO_i >$. For every fault f_a we can extract the subset of D (D_a), such that $D_a = \{\forall d_i \in D : d_i =< f_i, t_i, FO_i >, f_i = f_a\}$. Let label P_a the set of all unique pair $p_i =< t_i, FO_i >$ in D_a. Every pair $p_i \in P_a, i = 1,...,m$ can detects the set of faults $F_i \subseteq F$. Let F^a is the collection of all sets F_i ($F^a = \{F_1, ..., F_m\}$). The fault f_a is distinguishable if the intersection of all fault sets in the F^a is the one element set and this element is f_a ($\bigcap_{i=1}^{m} F_i = \{f_a\}$). If in the F^a there is no fault set F_x for which $\{\bigcap_{i=1}^{x-1} F_i\} \bigcap \{\bigcap_{i=x+1}^{m} F_i\} = \{f_a\}$, the P_a contains the **minimal set** of tests that makes the fault f_a diagnosable.

My purpose is to present the effective deterministic algorithm for finding the minimal set of tests, which make all faults from collapsed dictionary F distinguishable. The proposed algorithm has three main steps:

(1) to build the collection D for every fault $f_a \in F$ and every possible test pattern $t_j \in T$ in 3-valent logic (0, 1 and X), where level 'x' means "doesn't matter";

(2) to minimize the number of test patterns $t_j \in T$ by merging the compatible patterns for triples d_i with equal FO_i;

(3) to minimize the number of test patterns $t_j \in T$ by selecting the minimal test set for every fault.

3.1. D-collection building

Path tracing optimization.

D-algorithm decision graph is a hug and there are many duplicated vertexes and what is more- there are many duplicated graph branches. According to this algorithm every fault has own decision graph. Therefore the redundant calculations become extremely high. To reduce this number it is useful first to calculate the set of input patterns, which can force every line respectively to the level 0 and level 1 and then, to active every fault from the collapsed dictionary and to propagate the fault effect to every observation output. The calculations are made in 3-valent logic - every line take one of three levels:'0', '1' or 'x' (doesn't matter). The two stages use the forward path tracing. Using this approach the redundant calculations are prevented and the decision graph gets a linear instead of a tree structure. The complexity of calculations from exponential becomes polynomial.

Subalgorithm 1. Algorithm for building the collection of input patterns L_l^v, which can force the line l to level v. The calculations are performed for each gate output only if the collections for all inputs are calculated. According to the logic of the gate and the desired output line level, input lines collections $L_i^{v'}$, are juxtaposes and the result is the L_o^v set. The juxtaposing arithmetic has the next rules:

- if it is enough to put only one of the input lines in a level v', then the result collection is a union of all input collections $L_o^v = \bigcup_{i=1}^{n} L_i^{v'}$, where n is the number of gate inputs;
- if it is mandatory to put all input lines in a level v', then the result collection is a Cartesian product from all input collections $L_o^v = \prod_{i=1}^{n} L_i^{v'}$. During this production every two patterns are joined bit by bit according 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 result pattern there is a conflict, this pattern is rejected.

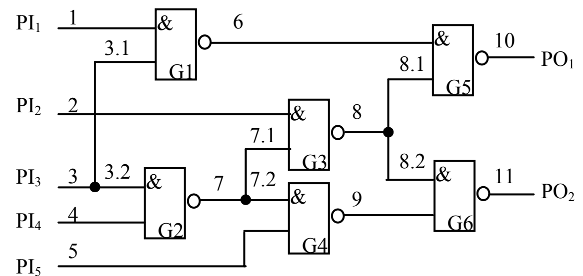

Figure 2. c17 benchmark circuit.

For example let see the circuit, shown on figure 2. To force line 1 to level 0, it is enough to set PI_1=0, therefore $L_1^0 = (0xxxx)$. To force line 6 to level 1, at least one of lines 1 and 3.1 must be forced to level 0, therefore $L_6^1 = L_1^0, L_{3.1}^0 = \{(0xxxx)\}, \{(xx0xx)\} = \{(0xxxx), (xx0xx)\}$. Now let pay attention on calculations for forcing line 10 to level 0.

$L_{10}^{0} = L_{6}^{1} \times L_{8.1}^{1} = \{(0xxxx), (xx0xx)\} \times \{(x0xxx), (xx11x)\} =$
$= \{((0xxxx) \& (x0xxx)), ((0xxxx) \& (xx11x)),$
$((xx0xx)) \& ((x0xxx)), ((xx0xx)) \& ((xx11x))\} =$
$= \{(00xxx), (0x11x), (x00xx), (xx?1x)\} =$
$= \{(00xxx), (0x11x), (x00xx)\}$

In the Cartesian product there are four elements. In the fourth element there is a conflict and it is rejected. The only three patterns remain in the result collection.

Subalgorithm 2. Algorithm for building the collection of test patterns T_l^v , which can detect the fault f_l . This algorithm traces the fault propagation to all observation point and follows two steps: fault activation and fault propagation to every gate output on the path.

(1) ***Fault activation***. Lets have a fault f_l , which is a single stuck-at fault to level *v* on line *l* (l_i / sav). To activate the fault we must force the level $\bar{v}$ on the line l , hence we must extract the collection $L_l^{\bar{v}}$.

(2) ***Fault propagation to every gate output on the path***. To permit fault effect propagation we must force the other gate inputs in such level, so that the gate works as a repeater for fault line. For the AND/NAND gates - $G_{repeater} = \prod_{i=1}^{n-1} L_i^1$, where *n* is the number of gate inputs. For the OR/NOR gates - $G_{repeater} = \prod_{i=1}^{n-1} L_i^0$. For the XOR/NXOR gates - $G_{repeater} = L_i^0 + L_i^1$. The fault propagation collection for gate output can calculate by the equation $T_l^v(G_k) = L_l^{\bar{v}} \times G_{repeater}$, for the first gate and $T_l^v(G_{k+1}) = T_l^v(G_k) \times G_{repeater}$ for the rest gates on fault propagation path.

For example let's return to the circuit, shown on figure 2 and calculate T_9^0 (test patterns for fault sa0 on line 9).

$T_9^0(G_6) = L_9^1 \times G_{repeater} = L_9^1 \times L_{8.2}^1 =$
$= \{(xx11x), (xxxx0)\} \times \{(x0xxx), (xx11x)\}$
$T_9^0(G_6) = \{(x011x), (xx11x), (x0xx0), (xx110)\}$

The fault effect, detected from all the tour test patterns can be observed on output PO_2. The detecting set for this fault is $D_{f_i=9/0} = \{< f_i, (x011x), PO_2 >, < f_i, (xx11x), PO_2 >, < f_i, (x0xx0), PO_2 >, < f_i, (xx110), PO_2 >\}$.

Now let calculate the test pattern for the line 3.2 sa0 fault - $T_{3.2}^0$. There are three fault propagation paths: G2-G3-G5, which can be observed on output PO_1; G2-G4-G6 and G2-G3-G6, which can be observed on output PO_2. For the first path calculations are:

$$T_{3.2}^0(G_2) = L_{3.2}^1 \times L_4^1 = \{(xx1xx)\} \times \{(xxx1x)\} = \{(xx11x)\} \quad (1)$$
$$T_{3.2}^0(G_3) = T_{3.2}^0(G_2) \times L_2^1 = \{(xx11x)\} \times \{(x1xxx)\}$$
$$T_{3.2}^0(G_3) = \{(x111x)\} \quad (2)$$
$$T_{3.2}^0(G_5) = T_{3.2}^0(G_3) \times L_6^1 = \{(x111x)\} \times \{(0xxxx), (xx0xx)\} =$$
$$= \{(0111x), (x1?1x)\} = \{(0111x)\} \quad (3)$$

For the second path we can use the result from equation (1) and calculate

$$T_{3.2}^0(G_4) = T_{3.2}^0(G_2) \times L_5^1 = \{(xx111)\} \quad (4)$$
$$T_{3.2}^0(G_6) = T_{3.2}^0(G_4) \times L_{8.2}^1 = \{(x0111), (xx111)\} \quad (5)$$

For the third path we can use the result from equation (2) and calculate

$$T_{3.2}^0(G_6) = T_{3.2}^0(G_3) \times L_6^1 = \{(x111x)\} \quad (6)$$

From the equations (3), (5) and (6) can make the detecting set for this fault:

$D_{f_j=3.2/0} = \{< f_j, (0111x), FO_1 >, < f_j, (x0111), FO_2 >, < f_j, (xx111), FO_2 >, < f_j, (x111x), FO_2 >\}$

Test pattern collection optimization.

After the algorithm 1 or algorithm 2 calculations it is possible to appear the fully or partially redundant elements in the test pattern collections. Therefore it is useful to optimize every collection after calculation. The optimization can be made by absorption. To explain this algorithm firstly must define the term **covering test pattern**. 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 is 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.

If there is a collection

$T = \{(xx111), (x111x), (0111x), (0x11x), (0111x)\}$ test patterns $t_3 = (0111x)$ and $t_5 = (0111x)$ are identical and t_3 can absorb t_5 (or vice verse). Test pattern $t_4 = (0x11x)$ can absorb $t_3 = (0111x)$. The reduced collection becomes $T = \{(xx111), (x111x), (0x11x)\}$.

Fault dictionary optimization.

Consider the algorithm 2 you can mention that the calculations for the faults of the lines that are close to the outputs are less then these that are far from the output. Therefore it is profitably to put in the collapsed dictionary the closest to the outputs fault from every equivalent group.

3.2. Improving distinguishability

If there are two D elements d_a and d_b, so that $f_a \neq f_b$, $FO_a = FO_b$ and t_a and t_b are compatible, then the two faults are potentially indistinguishable. For example, let have t_a="01xx0" and t_b="x1x00". After optimizations this patterns may become t_a=t_b="01000" and f_a and f_b become indistinguishable. To guarantee the distinguishability at least one of the bits must become explicitly different. To give a possibility for further optimization it is enough to change just one bit. For the example t_a="**0**1xx0" and t_b="**1**1x00".

Subalgorithm 3. Improve distinguishability.

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.

For the one of compatible pairs t_i, t_j change one of 'x' levels to alternative value, so that t_i and t_j becomes incompatible.

3.3. Compatible patterns merging

In the most of the test patterns, generated during the previous step there is a lot of ‘x’ values. This is a preposition for merging the test patterns and reducing the number of unique test patterns. The power of the test pattern is increasing. According to the z-set theory if one test pattern detects two faults, but on different failing outputs, the two faults are distinguishable. Therefore the merging can be done between test patterns that are part of *d* elements with different FO.

Subalgorithm 4. Test pattern merging. This algorithm follows the next steps:

1) Make the $D_a = \{\forall d_i \in D : d_i =< f_i, t_i, FO_i >, t_i = t_a, FO_i = FO_a\}$.

2) For every $d_i \in D_a$ and $d_j \in D_a$ looking for a compatible test patterns $t_i \in d_i$ and $t_j \in d_j$ and replacing them with resulting test pattern, according to the rules, described in algorithm 1.

3.4. Getting the minimal test sets

The main aim of this step is to get the minimal diagnosis test patterns for every fault and to summarize them in the final test pattern collection T_{res}.

Subalgorithm 5. 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 pair $<t_i, FO_i>$ in the D_a make the collections F_i^a of detecting faults.

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

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

IV. Conclusion and Future Work

In this paper is proposed a deterministic algorithm for diagnostic test pattern generation with polynomial complexity. As a result, this algorithm gives a minimal set of test patterns, which allow to distinguish all the faults in fault dictionary as single stuck-at faults. This algorithm can be used for test generation for diagnosis system, based on cause-effect approach for building fault dictionaries. The collected information can be used for accelerating the process of dictionary building. This algorithm can be used as well as in diagnosis system, based on effect-cause approach. It gives appropriate circumstances for candidate faults election and multiple fault analysis.

This algorithm is part of a bigger work that is focused on masking effect analysis. It is based on single stuck-at fault model, but it will be expanded to work with multiple-fault models.

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

[9] Liu, J.B.; Veneris, A., "Incremental fault diagnosis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol.24, no.2, pp. 240-251, Feb. 2005

[10] Pomeranz, I..; Reddy, S.M.; Venkataraman, S.., "z-Diagnosis: A Framework for Diagnostic Fault Simulation and Test Generation Utilizing Subsets of Outputs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol.26, no.9, pp.1700-1712, Sept. 2007

[11] 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

[12] Caruso, A.; Chessa, S.; Maestrini, P.; Santi, P., "Evaluation of a diagnosis algorithm for regular structures," *Computers, IEEE Transactions on* , vol.51, no.7, pp.850-865, Jul 2002

[13] Yung-Chieh Lin; Feng Lu; Kwang-Ting Cheng, "Multiple-Fault Diagnosis Based On Adaptive Diagnostic Test Pattern Generation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol.26, no.5, pp.932-942, May 2007

[14] Zhiyuan Wang; Marek-Sadowska, M.; Tsai, K.-H.; Rajski, J., "Analysis and methodology for multiple-fault diagnosis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol.25, no.3, pp. 558-575, March 2006

[15] ROTH, J. F., Diagnosis of automata failures: A calculus and a method. *ISM J. Res. Deu.,*10, 278-281, 1966