Published in IET Computers & Digital Techniques
Received on 24th August 2011
Revised on 28th February 2012
doi: 10.1049/iet-cdt.2011.0121

ISSN 1751-8601

Diagnostic test-pattern generation targeting open-segment defects and its diagnosis flow

Y.-H. Chen   C.-L. Chang   C.H.-P. Wen

Department of Electrical Engineering, National Chiao Tung University, Hsinchu 300, Taiwan
E-mail: tinger.cm98g@g2.nctu.edu.tw

Abstract: As an open defect occurs in one wire segment of the circuit, different logic values on the coupling wires of the physical layout may result in different faulty behaviours, which are so called the Byzantine effect. Many previous researches focus on the test and diagnosis of open defects but the pattern diagnosability has not properly addressed. Therefore in this study, a high-resolution diagnostic framework for open defects is proposed and consists of a diagnostic test-pattern generation (DTPG) and its diagnosis flow. The branch-and-bound search associated with controllability analysis is incorporated in satisfiability-based DTPG to generate patterns for the target segment. Later, a precise diagnosis flow constructs the list of defect candidates in a dictionary-based fashion followed by an inject-and-evaluate analysis to greatly reduce the number of candidates for silicon inspection. Experimental results show that the proposed framework runs efficiently and deduces nearly one candidate for each open-segment defect on average among all ISCAS'85 benchmark circuits.

1 Introduction

Open defects – the unintended breaks or electrical discontinuities in Integrated Circuit (IC) interconnect lines – are common defects frequently discussed in Very Large Scale Integration (VLSI) testing beyond deep submicron era. One open defect can be classified into: (i) 'intra-gate' opens which are regarded as an open with infinite resistance that disconnects the charge path or discharge path to the gate output and (ii) 'inter-gate' opens which are opens on interconnects that have influences on signal propagation. Typically, intra-gate opens can be regarded as stuck-open faults whereas inter-gate opens can be further classified into two types: (i) resistive opens and (ii) complete opens. Resistive opens are also known as weak opens under which the current still passes through the narrow open defects because of the tunneling effect. Complete opens, on the other hand, are often called interconnect opens where the voltage for the driven gates of the floating interconnect is hard to predict. According to Hawkins *et al.* and Xue *et al.* [1, 2], the majority of open defects are of inter-gate type and a high percentage of them belong to complete opens.

Along with the scaling of the manufacturing process, the distance between interconnects becomes narrower and more neighbours are coupled to a wire. As a result, when an open defect occurs in one segment of a wire, the faulty behaviour because of different coupling condition is complicated. To properly describe the defect behaviours, the Byzantine effect [3, 4] is invented and states that the voltages of driven gates connecting an open segment are determined by comparing their threshold voltages and the floating voltage according to its coupling condition. Fig. 1 shows an example of the Byzantine effect of interconnect opens. In Fig. 1*a*, gate G_1 drives gates G_2, G_3 and G_4 through a logical interconnect. When a defect occurs on the output of G_1, all driven gates, G_2, G_3 and G_4, receive faulty values. However, considering the physical layout shown in Fig. 1*b*, such interconnect can be divided into five wire segments. When the open defect occurs on different segments, the faulty values propagate to different gates and result in different faulty behaviours of the downstream logic.

Many previous works addressed the importance of the open-defect problem and solutions were also proposed for testing. Xue *et al.* [2] showed that ISCAS'85 circuits are likely to have different kinds of open defects. Needham *et al.* [5] also found that open defects can escape under the test of stuck-at fault model. Furthermore, open defects were shown to run correctly in the low frequencies but failed in the high frequencies in [6], and thus its logic testing was difficult. Later, Spinner *et al.* [7] built an aggressor–victim model for both the inter-layer opens and the intra-layer opens, and generated the patterns automatically. Hillebrecht *et al.* [8] developed a flow that integrates the physical information (including the layout and the cell library) and the aggressor–victim model for pattern generation. Devtaprasanna *et al.* [9] gave a solution of testing intra-gate open defects and obtained the complete defect coverage. Gomez *et al.* [10] used the commercial tool to build a flow from layout extraction to Automatic Test Pattern Generation (ATPG) for interconnect open defect.

For diagnosis of open-defects, several previous works took the physical information into account. Huang [3] proposed a diagnosis flow which focuses on single interconnect-defect assumption and uses inject-and-evaluate paradigm for diagnosis to achieve high accuracy. Zou *et al.* [4] further considered the Byzantine effect with physical information

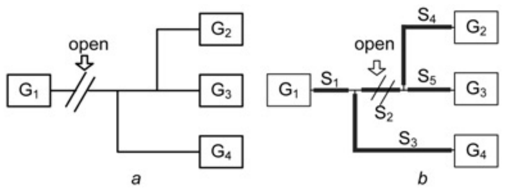


Fig. 1 *Byzantine effect of an interconnect open*

a Logical interconnect view
b Physical layout view

for single open-defect diagnosis. Recently, Kao *et al.* [11] targeted multiple open-segment defects and proposed a diagnosis flow which uses the falling patterns to formulate constraints for Integral Linear Programming (ILP) solving and derives the fault tuples automatically.

Previous researches about open-segment defects mainly focus on the testing and diagnosis perspectives and have not properly addressed the diagnosability of test patterns. The test set with high defect coverage does not necessarily lead to good diagnosability. Besides, from the test-pattern-generation point of view, an interconnect open involves complicated constraints subject to coupling wires from the physical layout and the threshold voltages of driven gates from the design library. From diagnosis point of view, the inefficiency and inaccuracy of previous diagnosis algorithms also can be improved. Therefore we are motivated to consider all these issues and develop a framework including diagnostic test-pattern generation (DTPG) and its diagnosis flow in the paper.

The rest of this paper is organised as follows. In Section 2, we first introduce the fault model of open-segment defects. In Section 3, a high-resolution diagnostic framework is proposed and consists of two stages: (i) DTPG and (ii) the diagnosis flow. Section 3 shows the experimental results including comparison on diagnosis resolution between random patterns, 5-detect stuck-at patterns and our diagnostic patterns on a variety of ISCAS'85 benchmark circuits. Finally, we draw our conclusion and outline future works in Section 5.

2 Fault model of open segments

In this section, we first introduce the fault model of open segments used for later test-pattern generation and diagnosis. Several fault models of open defects were proposed in previous researches [4, 7, 12] and throughout this paper, we use the one in [7] that describes the open-segment fault considering the physical information.

Fig. 2 illustrates this fault model. When a wire segment is open, the node f on the floating side is regarded as an open-segment fault. The voltage of the floating-node f is determined by the coupling capacitances of the neighbouring wires and their logic values. Later, such floating-node voltage is compared with the threshold voltage of one driven gate. If f's voltage is larger than the threshold voltage of a driven gate, the gate receives logic-1 at its input; otherwise, it receives logic-0. Apparently, not all driven gates of an open segment will receive faulty values.

As shown in Fig. 2, the floating-node voltage V_f can be formulated into

$$V_f = V_{dd}\frac{C_1}{C_0 + C_1} + \frac{Q_t}{C_{gnd}} \tag{1}$$

where Q_t is the initial trapped charge of the floating node and C_{gnd} is the capacitance between floating node and ground. C_0 and C_1 are the sums of the capacitances of wires with logic-1 and logic-0, respectively. The values of C_0 and C_1 can be further decomposed into the following equations

$$C_0 = C_{gnd} + C_{n0} + C_{i0} \tag{2}$$

$$C_1 = C_{vdd} + C_{n1} + C_{i1} \tag{3}$$

where C_{vdd} and C_{gnd} are the capacitances between the floating node and the power, and between the floating node and the ground, respectively. C_{n0} and C_{n1} are the lumped capacitances between floating node and its neighbouring wires with logic 0 and logic 1, respectively. C_{i0} and C_{i1} are the internal capacitances inside the driven gate.

However, trapped charge Q_t and internal capacitances, C_{i0} and C_{i1} are typically hard to predict. Moreover, process variation also makes parasitic capacitances extracted from physical layout unpredictable. Therefore we use a simplified model similar to [3, 7, 13] by assuming that the parasitic capacitances between the open segment and its neighbouring segments are the dominant factors to determine the voltage value on the floating node and Q_t can be dropped. Accordingly, the value of C_0 and C_1 can be approximated by

$$C_0 \cong C_{gnd} + \sum_{\text{neighbour } n_i \in L_0} C_{n_i} \tag{4}$$

$$C_1 \cong C_{vdd} + \sum_{\text{neighbour } n_i \in L_1} C_{n_i} \tag{5}$$

where L_0 and L_1 denote logic-0 and logic-1 respectively.

Given a floating-node voltage V_f and related information of one driven gate g, that is, the threshold voltage V_g^{thres}, if $V_f \geq V_g^{thres}$, its logic value v_g is regarded as with logic-1. Otherwise, it is with logic-0, that is

$$v_g = \begin{cases} 1 & \text{if } V_f \geq V_g^{thres} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

To take Fig. 1*b* for example, suppose that V_{t2}, V_{t3} and V_{t4} denote the threshold voltages for G_2, G_3 and G_4, respectively, and the coupling condition comes a floating-node voltage V_f, where $V_{t2} \geq V_f \geq V_{t3} \geq V_{t4}$. If segment S_1 is open, G3 and G_4 will receive logic-1 and G_2 will receive logic-0. If the segment S_2 is open in Fig. 1*b* under the same voltage condition, G_2 and G_3 are with logic-0 and logic-1, respectively, but G_4 is not affected and keeps the correct value as the output of G_1. Therefore for segment S_1, the possible fault combinations include (G_2, G_3, G_4), (G_3, G_4) and (G_4) whereas for segment S_2, its possible fault

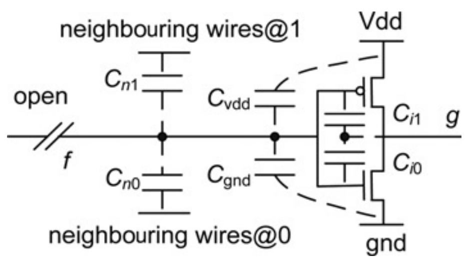


Fig. 2 *Fault model of open defects*

combinations include (G_2, G_3) and (G_3). An open on different segments may result in different faulty behaviours. To correctly model the faulty behaviour, both the coupling condition from the physical layout and the information of threshold voltages of each gate from the cell library should be considered.

3 High-resolution diagnostic framework

A high-resolution diagnostic framework is proposed on the basis of the above open-segment fault model in this paper. Two stages are included: (i) DTPG which applies a modified branch-and-bound search from [13] with a SAT solver to generate unique patterns for each single open-segment for the given circuit and (ii) a dictionary-based diagnosis which is built upon the inject-and-evaluate analysis and can reduce the number of fault candidates greatly.

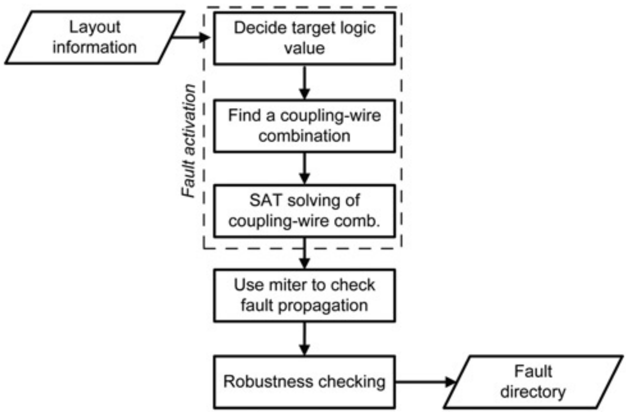
3.1 Diagnostic test-pattern generation

Based on the open-segment fault model, our diagnostic framework first starts the DTPG stage as shown in Fig. 3. Note that, a pre-processing step builds an open-segment list for DTPG in which all structurally untestable defects are removed. Here structurally untestable defects denote those target segments with no neighbouring wires as aggressors in the physical layout so that they remain unknown and cannot be tested.

Traditional ATPG approaches [7, 13] target each single segment as a defect location and aims to generate patterns to ensure the correctness of the given circuit. However, many different defect conditions may result in the same output syndrome and make diagnosis difficult. Therefore we use a different viewpoint and indirectly target each driven gate of the open segment instead of the segment itself directly to avoid ambiguities of repeated output syndromes. Our idea, illustrated in Fig. 4, is not only to reduce the fault number but also to improve the diagnosis resolution. Gate G_1 is connected with three driven gates, G_2, G_3 and G_4 and the logical interconnect can be divided into five segments, S_1, S_2, S_3, S_4 and S_5. Since previous ATPGs in [7, 13] generate patterns targeting each segment, five patterns are generated in this example. However, our DTPG generates only three exclusive patterns targeting G_2, G_3 and G_4, respectively. If $\{G_2, G_3\}$ is the set of gates that capture faulty values, S_2 is the only open segment. If $\{G_2, G_3, G_4\}$ is the faulty-gate set, S_1 is the only open segment.

Since all structurally unstable segments are removed, each remaining segment can be mapped to its driven gate as fault candidate. Our DTPG stage targets the driven gate of open segment instead of the open segment itself. Therefore the complexity of open-segment diagnosis $O(m)$ can be reduced to $O(n)$, where m denotes the number of segments and n denotes the number of gates in the circuits, whereas $m \gg n$ in most circuits. The decreasing number of fault candidate helps to reduce the cost to explore diagnostic test patterns and also improve the resolution as open-segment diagnosis.

In the DTPG stage, besides a fault can activate on the segment, it should be able to propagate towards primary outputs. Therefore our DTPG flow consists of three major components: (i) fault activation, (ii) fault propagation and (iii) robustness checking.



Layout information
Decide target logic value
Find a coupling-wire combination
SAT solving of coupling-wire comb.
Fault activation
Use miter to check fault propagation
Robustness checking
Fault directory

Fig. 3 *DTPG flow*

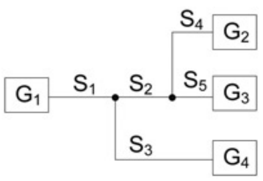


Fig. 4 *Example of affected outputs*

3.1.1 Fault activation: Since the occurrence of open-segment fault depends on its neighbouring wires, a branch-and-bound searching strategy is proposed in [7] to find an input pattern according to different combinations of coupling conditions. Before the branch-and-bound search starts, the logic value of target segment needs to be decided first. Spinner *et al.* [7] chose logic-1 first and then logic-0 to decide the logic value of target segment in DTPG (denoted by $1 \rightarrow 0$). That is, the branch-and-bound method first searches the input pattern by setting the target segment as logic-1. If one input pattern can be found, DTPG continues to process another fault candidate; otherwise, the target segment is set as logic-0 and starts searching. On the other hand, if no input pattern can be found in both rounds, the target fault is marked as redundant.

The search strategy affects the efficiency of DTPG. Since the $1 \rightarrow 0$ search strategy is not necessarily the best way for DTPG, three more strategies are also cross-compared and incorporated: (i) $0 \rightarrow 1$ where logic-0 is first justified and then logic-1; (ii) ‘controllability’ from testability analysis [14] where the controllability values of the coupling wires for the target segment can determines the expected logic value of the target segment to be justified; (iii) ‘observability’ from [14] where the observability value of the target segment determines the logic value to be justified.

Four searching strategies are conducted on four small ISCAS85 circuits and the average runtime of searching the input patterns for all fault candidates are listed in Table 1. The first column shows the name of the circuits and columns 2, 3, 4 and 5 show the runtime in seconds for the $1 \rightarrow 0$, $0 \rightarrow 1$, observability and controllability strategies, respectively. As we can see, both $1 \rightarrow 0$ and $1 \rightarrow 0$ strategies are not necessarily superior whereas the observability strategy and controllability strategy yield comparable results. Especially when the circuit size goes larger, both the observability and controllability strategies obtain better efficiencies over the $1 \rightarrow 0$ strategy. Therefore the controllability strategy is chosen to be incorporated in our DTPG for the simplicity of computation.

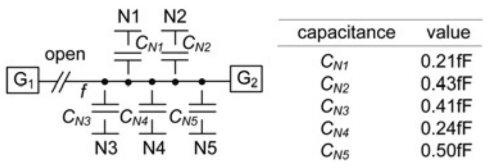
Table 1 Runtime (in seconds) of four searching strategies

Circuit	$1 \rightarrow 0$	$0 \rightarrow 1$	Obserervability	Controllability
*c*432	46	137	75	97
*c*499	497	542	507	470
*c*880	299	330	267	312
*c*1355	1124	993	873	884

After choosing the logic value for the target segment, the branch-and-bound search strategy starts to generate the diagnosis test pattern. It sorts the neighbouring wires by comparing their coupling-capacitance values first, and assigns the logic values in this order. We first intend to decide the logic value of the coupling wire with a large capacitance, which can save time on the exploration of the coupling tree, and then derive the coupling wire combination quickly. Fig. 5 shows such an example. If gate G_1 has five neighbouring wires $N_1 – N_5$ with their coupling-capacitance ranking as $C_{N5} > C_{N2} > C_{N3} > C_{N4} > C_{N1}$, then the coupling tree will have the wire order $N_5 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4 \rightarrow N_1$ to assign logic values for the target segment between G_1 and G_2.

After having the order for logic-value assignments on neighbouring wires, the next step of DTPG aims to find a feasible combination of all coupling wires to induce a proper floating-node voltage. For example, if we use logic-1 to be justified according to controllability analysis, a legal coupling-wire combination should result in a bigger voltage V_f than the threshold voltage of the target driven gate. Given the threshold voltage of G_2 is $V_{G_2}^{\text{thres}} = 0.85\,\text{V}$, two examples for assignments are shown in Fig. 6. As only N5 is assigned logic-1 in the first example, the floating-node voltage is 0.50 V and may not be larger than G_2's threshold voltage. While both N5 and N3 are assigned logic-1 and N2 is assigned logic-0, the floating-node voltage can make G_2 receive faulty logic-1. Under this circumstance, a feasible coupling-wire combination is found.

DTPG checks whether such combination can be justified on the basis on the functionality of the circuit and applies a SAT solver for this purpose. If the solver returns SAT, an input pattern that can result in legal assignments of the coupling wires is found. If UNSAT is returned, the coupling-wire combination is infeasible.



capacitance	value
C_{N1}	0.21fF
C_{N2}	0.43fF
C_{N3}	0.41fF
C_{N4}	0.24fF
C_{N5}	0.50fF

Fig. 5 *Layout extraction and coupling capacitances*

3.1.2 Fault propagation:

Besides a fault can be activated on the segment, a successful DTPG also requires that such fault can propagate and can be observed at one or more outputs of the circuit. After activating the fault on the target segment, we then check the propagation constraint by SAT solving of the found coupling-wire combination over a miter circuit that combines the fault-free and faulty circuits. If SAT is returned, a legal pattern is found and successfully makes the faulty effect propagate to at least one output of the circuit. If UNSAT is returned, the faulty effect is blocked during the propagation and cannot be observed at any output. The next coupling-wire combination is then checked for fault activation.

At most $2n$ rounds of fault-activation and fault-propagation checking can be performed to find the set of legal input assignments, where n denotes the total number of neighbouring wires. Once $2n$ is reached without finding any legal pattern, this defect is claimed as 'pseudo-untestable'. Fig. 7 illustrates the exploration of the coupling tree for the same example used in Fig. 5. Since there are five neighbouring wires, $2n$ is 10. A successful search is obtained during the sixth search where N5, N2 and N3 are assigned logic-0, logic-1 and logic-1, respectively. In other words, the coupling capacitance C_{N5}, C_{N2}, and C_{N3} are taken into consideration to generate diagnostic pattern for the target segment between G_1 and G_2.

3.1.3 Robustness checking:

When a pattern is successfully generated, we need to check its robustness with silicon result. We use the pattern to run fault simulation, inject different defects and then check whether output responses match the expected values under such pattern. If yes, the pattern can be used for diagnosis. Otherwise, the pattern is removed. Fig. 8 illustrates an example. S_1, S_2 and S_4 are injected an open and fault simulation runs with the pattern P for faulty G_2, respectively. Assume that O_2 and

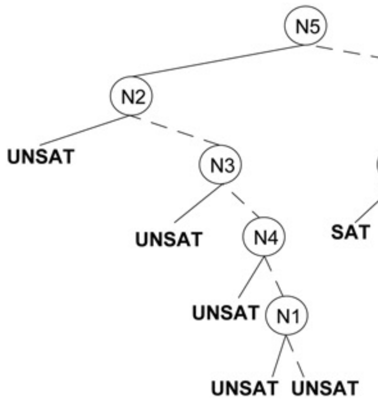


Fig. 7 *Exploration of different coupling-wire combinations in the coupling tree*

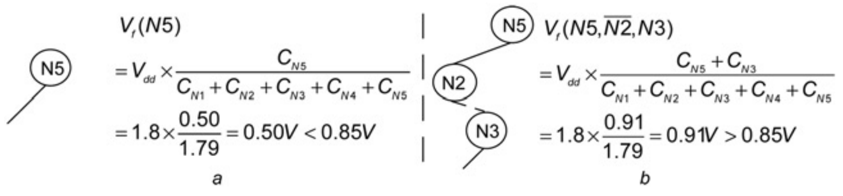

N5
$V_f(N5)$
$= V_{dd} \times \frac{C_{N5}}{C_{N1}+C_{N2}+C_{N3}+C_{N4}+C_{N5}}$
$= 1.8 \times \frac{0.50}{1.79} = 0.50V < 0.85V$
a
N5
N2
N3
$V_f(N5,\overline{N2},N3)$
$= V_{dd} \times \frac{C_{N5}+C_{N3}}{C_{N1}+C_{N2}+C_{N3}+C_{N4}+C_{N5}}$
$= 1.8 \times \frac{0.91}{1.79} = 0.91V > 0.85V$
b

Fig. 6 *Possible coupling-wire assignments to be justified*

a Threshold derivation for N5 = 1
b Threshold derivation for N5 = 1, N2 = 0 and N3 = 1

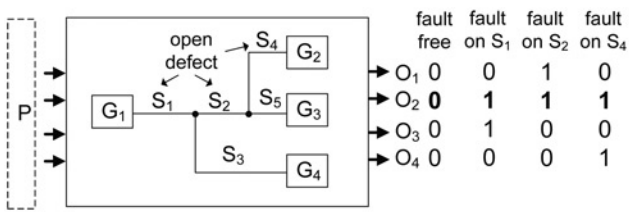


Fig. 8 *Inject and evaluate a defect to derive its output syndrome*

O_3 are the affected output of G_2 under P. If P is robust, three faulty values need to be observed during simulations with different injected defects. In this example, since we observe three faulty values at O_2 in a row, P is robust. Pattern P and its output syndrome at O_2 are saved in the fault dictionary for later diagnosis.

3.2 Proposed diagnostic flow

After DTPG, the information about patterns and their output syndromes are collected. Under the single defect assumption, we can diagnose the faulty circuit by a diagnosis flow as shown in Fig. 9 to achieve high resolution. This flow mainly consists of two stages: (i) a 'dictionary'-based matching and (ii) an 'inject-and-evaluate' pruning. The first stage uses the fault dictionary to build a list of candidates whose output syndromes match the output responses on silicon. In the second stage, each candidate takes turn to be open and runs fault simulation over all patterns to see any inconsistency.

3.2.1 Dictionary-based matching: The first stage starts from constructing a list of open-segment candidates for a dictionary-based matching. All faulty gate candidates can be obtained by matching the output responses from silicon and the output syndromes stored in the dictionary. As the proposed DTPG targets the faulty gate instead of the faulty segment directly, the faulty gate candidates are used to assemble the corresponding faulty segments.

Fig. 10 shows an example where an open occurs on segment S_2 driven by gate G_1. Assume pattern P_2, P_3 and P_4 target faulty gates G_2, G_3 and G_4 according to the silicon results as shown Figs. 10*a*–*c*, respectively. As

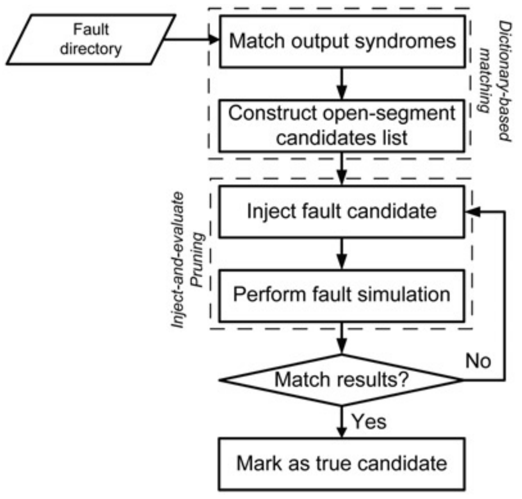


Fig. 9 *Proposed two-stage diagnosis flow*

shown in Fig. 10*d*, since faults appear on G_2 and G_3, S_2 is the only defect candidate among all segments.

3.2.2 Inject-and-evaluate pruning: Multiple open-segment candidates can be obtained from stage one and then an inject-and-evaluate pruning is performed in stage two. During this stage, one defect candidate is chosen at a time for injection and run simulation with the physical information including the cell library and extracted RC values. At the end, whether a candidate is a true or not depends on the simulation result and the corresponding silicon output responses. If they are not mismatched, such defect candidate is eliminated. Otherwise, it remains in the defect pool for silicon inspection in the future.

4 Experimental results

Our experiments are conducted on the ISCAS'85 benchmark circuits where the RTL codes in Verilog, layouts and coupling capacitance information are available from TAMU websites [15]. All ISCAS'85 benchmark circuits are synthesised with a five-metal-layer TSMC 180 nm CMOS technology. The threshold voltage of each type of gate is determined through SPICE simulation. For DTPG, we use an open-source

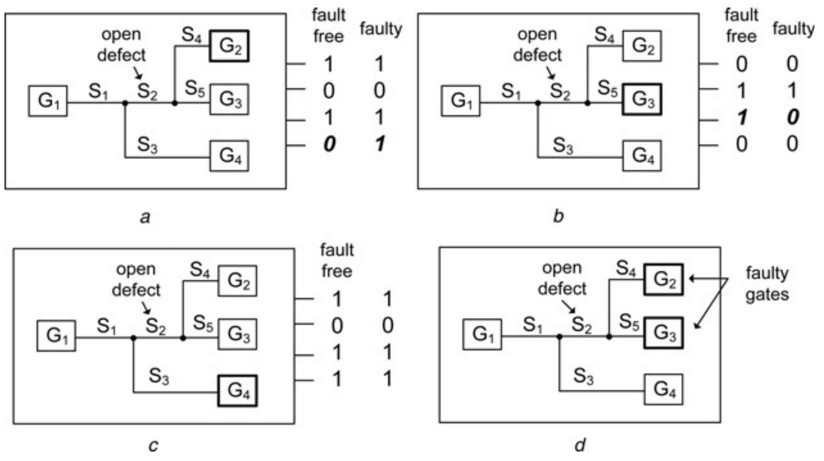


Fig. 10 *Pattern set to target open-segment S_2*

a Pattern P_2 and its output syndrome
b Pattern P_3 and its output syndrome
c Pattern P_4 and its output syndrome
d Assemble the defect based on faulty gates

complete SAT solver, MiniSat 2.0 [16], which is one of the best SAT solvers in practice.

Table 2 shows the basic information of the gate level and physical benchmark circuits where the first column denotes the name of the circuits; the second column denotes the total number of nets; the third column denotes the number of nets with multiple fan-out stems from the total number of nets; the fourth column denotes the total number of segments according to the physical layouts of the circuits.

After the DTPG stage completes, every defect can be further classified into different categories and Table 3 shows the statistics. The first column shows the names of the circuits. The second column shows the number of 'structurally untestable' faults including those segments with no coupling wires according to the physical layout. The third column shows 'TPG-untestable' defects including those cases what no pattern can be generated after trying the $2n$ (pseudo-untestable) or all possible combinations (truly-untestable) of the coupling wires for the target segment. The fourth column shows the number of aborted defects which are defined as the generated patterns resulting mismatch output syndromes compared with those from the silicon output responses. The last column shows the numbers for successful defects where patterns can be successfully generated and assembled after our DTPG.

After generating diagnostic patterns and collecting the corresponding output syndromes, the diagnosis stage proceeds and Table 4 shows the experimental result. The first column represents the total runtime used for diagnosis. The second column shows the DTPG time to generate diagnostic patterns for all testable faults and the third column records the number of diagnostic patterns. The fourth column represents the total number of detected defects and the total number of candidate defects reported by our diagnosis algorithm. In our experimental settings, 100 random defects are injected onto all circuits but not all of them are testable. To take *c*432 for example, only 91 defects are testable and exact 91 candidate defects are reported by our diagnosis algorithm. The fifth column represents the diagnosis resolution which is computed as the total number of found candidates divided by the total number of injected testable defects. For example, the case with the resolution 1.00 means that our algorithm can find the exact defect on each defective sample of the circuit *c*432. The last column represents the total number of generated diagnostic patterns. Note that we have not yet applied any compaction or compression technique on the pattern set and thus pattern reduction can be one of the future directions of this work.

To fairly compare our diagnostic patterns and the diagnosis flow with other conventional random and 5-detect stuck-at patterns, a dictionary-based diagnosis flow consists of two steps: (i) the construction of fault dictionary and (ii) dictionary comparison. The first step applies the pattern set for simulation with the assistance of physical information against all possible open-segment defects. All the falling patterns (i.e. patterns that can result in erroneous outputs) and the related information including faulty driven gates, faulty segments and output syndromes are recorded to build a dictionary for diagnosis. The second step of the flow is to diagnose open defects with the built dictionary. We generate 100 sample circuits with random injection of defects. After defect injection, we run the simulation with physical information. Then we validate whether the defect dictionary matches the output responses from the physical simulation on each pattern. If yes, the count of defects increases by one. Otherwise, we skip and run the next pattern. If the count for one defect is equal to the expected value from the dictionary, such defect is saved as a true defect. Otherwise, it should be removed.

Table 5 compares the dictionary sizes for random patterns, 5-detect patterns and our DTPG patterns and includes all information related to defects, patterns and their output syndromes. The total numbers of random patterns used in our experiments are 1000 while the 5-detect patterns are generated by a commercial tool TetraMax [17]. As a result, our framework yields a much smaller fault dictionary only 0.4 and 0.99% on average of dictionary sizes from random patterns and 5-detect stuck-at patterns, respectively.

After applying the reference diagnosis flow, we can further derive the two tables. Table 6 shows the diagnosis results for 5-detect stuck-at patterns whereas Table 7 shows the results for random patterns. Column 1 shows the circuit name and column 2 shows the runtime required by the diagnosis process. Column 3 reports the numbers of reported

Table 2 Circuit information

Circuit	#Net	#*m*-f net	#Segment
*c*432	196	89	339
*c*499	243	59	437
*c*880	442	125	746
*c*1355	587	259	1069
*c*1908	913	385	1490
*c*2670	1502	454	2273
*c*3540	1719	579	2906
*c*6288	2448	1456	4232
*c*7552	3720	1300	5872

Table 3 Fault classification

Circuit	Structurally untestable	TPG-untest able	Aborted	Successsful
*c*432	28	15	4	292
*c*499	36	79	19	303
*c*880	98	30	3	615
*c*1355	138	170	32	729
*c*1908	195	293	2	1000
*c*2670	198	243	31	1801
*c*3540	269	453	14	2170
*c*6288	305	466	17	3444
*c*7552	705	754	69	4344

Table 4 Diagnosis results of our diagnostic patterns

Circuit	DTPG time, s	#Patterns	#Candidate/ #detected	Resolution	Diagnosis time, s
*c*432	5.1	160	91/91	1.00	11.6
*c*499	21.3	202	74/73	1.01	15.1
*c*880	25.4	383	83/84	1.01	43.2
*c*1355	289.4	546	70/70	1.00	165.8
*c*1908	113.5	880	68/68	1.00	91.9
*c*2670	419.8	1269	73/70	1.04	297.8
*c*3540	583.2	1669	79/79	1.00	445.2
*c*6288	4509.6	2416	79/80	1.01	1563.7
*c*7552	4739.8	3513	84/84	1.00	2424.8

Table 5 Compare fault dictionaries of random, 5-detect and our DTPG patterns

Circuit	Dictionary size, KB			Ratio, %	
	(a) Random	(b) 5-detect	(c) Our	(c)/(a)	(c)/(b)
*c*432	2155	1465	15	0.7	1.0
*c*499	7434	8464	24	0.3	2.8
*c*880	15 760	8614	57	0.4	0.7
*c*1355	15 862	34 881	59	0.4	0.2
*c*1908	19 105	42 342	66	0.3	0.2
*c*2670	190 734	156 652	672	0.4	0.4
*c*3540	34 469	51 238	175	0.5	0.3
*c*6288	180 392	34 624	250	0.1	0.7
*c*7552	416 568	58 369	1497	0.4	2.6

Table 6 Diagnosis result of 5-detect patterns

Circuit	#Patterns	#Candidate/ #detected	Resolution	Diagnosis time, s
*c*432	617	148/92	1.60	107.9
*c*499	794	262/77	3.40	613.4
*c*880	476	123/86	1.43	340.5
*c*1355	1272	232/80	2.90	4093.4
*c*1908	1725	240/75	3.20	7565.4
*c*2670	786	182/75	2.43	3126.1
*c*3540	1465	208/83	2.51	6899.0
*c*6288	191	110/91	1.21	858.3
*c*7552	1255	98/90	1.09	6534.5

Table 7 Diagnosis result of random patterns

Circuit	#Patterns	#Candidate/ #detected	Resolution	Diagnosis time, s
*c*432	1000	148/92	1.60	265.4
*c*499	1000	396/75	5.28	792.9
*c*880	1000	125/85	1.47	1260.5
*c*1355	1000	291/80	5.74	1480.2
*c*1908	1000	419/70	5.99	1942.9
*c*2670	1000	267/64	4.17	4624.6
*c*3540	1000	392/70	5.60	3070.2
*c*6288	1000	103/89	1.16	3543.1
*c*7552	1000	194/65	3.00	4724.8

candidates and detected defects whereas resolution in column 4 is defined as the number of reported candidates divided by that of detected defects. Column 5 shows the total number of patterns used for diagnosis.

By comparing Tables 6 and 7 with Table 5, we can observe that the proposed diagnosis with the diagnostic patterns run much faster than the dictionary-based diagnosis using random and/or 5-detect stuck-at patterns. The numbers of detected defects in Table 4 is smaller than the other ones in Tables 6 and 7 on some cases. This is because of the limitation of the failure of SAT solving during the DTPG stage. Comparing the fourth column of Table 4 with those of Tables 6 and 7, our diagnostic patterns with the proposed diagnostic flow yields better resolutions near one candidate for defective samples of overall benchmark circuits.

5 Conclusions

When an open defect occurs on a segment of the circuit, physical characteristics of the circuit such as the layout and the cell library cause dynamic faulty behaviours under different input patterns. Such phenomenon is called the Byzantine effect which makes open-segment defects difficult to be detected. Previous researches about open-segment defects mainly focused on testing and the diagnosis techniques and did not address the diagnosability of test patterns properly. The test set with high defect coverage does not necessarily result in good diagnosability. Therefore we are motivated to develop a two-stage high-resolution diagnosis framework including diagnostic TPG and its diagnosis flow in the paper.

The first stage for diagnostic TPG aims to generate patterns for all open-segment defects and consists of three steps: (i) finding a feasible coupling-wire combination, (ii) justifying a pattern conforming to the coupling-wire combination and (iii) validating output responses through physical simulation. Branch-and-bound search, controllability analysis and SAT solving techniques are integrated during this stage. Particularly, our indirect diagnostic TPG targets each driven gate of the open segment instead of the segment itself, and thus greatly reduces the pattern size as well as the runtime. Inject-and-evaluate pruning is also applied to remove those candidates that fail to match output responses on silicon.

Experiments are conducted on ISCAS'85 benchmark circuits and the result explains the effectiveness and efficiency of the proposed algorithm. For all benchmark circuits, only one candidate that exactly matches the injected defect is reported for most defective samples. Moreover, diagnosis runtime of our DTPG patterns is about 10× faster than that from conventional patterns. However, some aborted defects may escape from our diagnostic pattern set and thus become one focus of our future work. Other future directions may include (i) extending our DTPG algorithm to handle multiple defects, (ii) applying compaction and compression to further reduce the pattern size and (iii) considering the process variation in the open-segment model for the DTPG flow.

6 References

1 Hawkins, C.F., Soden, J.M., Righter, A.W., Ferguson, F.J.: 'Defect classes – an overdue paradigm for CMOS IC testing'. Proc. Int. Test Conf. (ITC), 1994, pp. 413–425

2 Xue, H., Di, C., Jess, J.A.G.: 'Probability analysis for CMOS floating gate faults'. Proc. European Design Automation Conf. (EDAC), 1994, pp. 443–448

3 Huang, S.Y.: 'Diagnosis of byzantine open-segment faults'. Proc. VLSI Test Symp. (VTS), 2002, pp. 248–253

4 Zou, W., Cheng, W.T., Reddy, S.M.: 'Interconnect open defect diagnosis with physical information'. Proc. Asian Test Symp. (ATS), 2006, pp. 203–209

5 Needham, W., Prunty, C., Yeoh, E.H.: 'High volume microprocessor test escapes, an analysis of defects our tests are missing'. Proc. Int. Test Conf. (ITC), 1998, pp. 25–34

6 Henderson, C.L., Soden, J.M., Hawkins, C.F.: 'The behavior and testing implications of CMOS IC logic gate open circuits'. Proc. Int. Test Conf. (ITC), 1991, pp. 302–310

7 Spinner, S., Polian, I., Engelke, P., Becker, B.: 'Automatic test pattern generation for interconnect open defects'. Proc. VLSI Test Symp. (VTS), 2008, pp. 181–186

8 Hillebrecht, S., Polian, I., Engelke, P., Becker, B.: 'Extraction, simulation and test generation for interconnect open defects based on enhanced aggressor-victim model'. Proc. Int. Test Conf. (ITC), 2008, pp. 1–10

9 Devtaprasanna, N., Gunda, A., Krishnamurthy, P., Reddy, S.M., Pomeranz, I.: 'Test generation for open defects in CMOS circuits'. Proc. Defect and Fault Tolerance in VLSI Systems (DFT), 2006, pp. 41–49

10 Gomez, R., Giron, A., Champac, V.H.: 'A test generation methodology for interconnection opens considering signals at the coupled lines', *J. Electron. Test. (JETTA)*, 2008, **26**, (6), pp. 529–538

11 Kao, C.Y., Liao, C.H., Wen, H.P.: 'An ILP-based diagnosis framework for multiple open-segment defects'. Proc. Microprocessor Test and Verification (MTV), 2009, pp. 69–72

12 Renovell, M., Cambon, G.: 'Electrical analysis and modeling of floating-gate fault', *IEEE Comput. Aided Des. (TCAD)*, 1992, **11**, (11), pp. 1450–1458

13 Lin, X., Rajski, J.: 'Test generation for interconnect opens'. Proc. Int. Test Conf. (ITC), 2008, pp. 1–7

14 Wang, L.T., Wu, C.W., Wen, X.: 'VLSI test principles and architectures: design for testability' (Elsevier Morgan Kaufmann Publishers, Boston, 2006)

15 Lu, X., Shi, W.: 'Layout and parasitic information for ISCAS circuits'. Available at http://dropzone.tamu.edu/~xiang/iscas.html, 2004

16 Eén, N., Sörensson, N.: 'An extensible SAT solver'. Sixth Int. Conf. on Theory and Applications of Satisfiability Testing, 2003, (*LNCS*, **2919**), pp. 502–518

17 TetraMAX ATPG: Available at http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Test/Pages/TetraMAXATPG.aspx