J Electron Test (2011) 27:723–739
DOI 10.1007/s10836-011-5265-0

Diagnosing Multiple Byzantine Open-Segment Defects Using Integer Linear Programming

Chen-Yuan Kao · Chien-Hui Liao · Charles H.-P. Wen

Received: 25 February 2011 / Accepted: 12 October 2011 / Published online: 22 October 2011
© Springer Science+Business Media, LLC 2011

Abstract Faulty behaviors of open-segment defects are non-deterministic due to the Byzantine effect induced by the physical circuit layout. It is the test pattern and difficult for traditional ATPGs to manifest the corresponding faulty effect. Therefore, we propose a three-stage diagnosis approach for finding multiple open-segment defects. Stage one applies path tracing to help extract candidate fault sites from error outputs of failing patterns. An ILP solver in stage two effectively enumerates all fault combinations when considering fault candidates and simulation responses simultaneously. During stage three, fault simulation with support of physical information is responsible for identifying true open-segment defects by pruning false cases. Experimental results show good resolutions (only $1.7X$ and $1.5X$ total numbers of segments on average under 1,000 random and 5-detect patterns, respectively) for all ISCAS'85 circuits with 2–5 randomly-injected open-segment defects.

Keywords Open defect · Diagnosis · Open segment · Byzantine effect · Integer linear programming

Responsible Editor: R. D. Blanton

C.-Y. Kao · C.-H. Liao (✉) · C. H.-P. Wen
Department of Electrical Engineering,
National Chiao Tung University,
Hsinchu, Taiwan
e-mail: liangel.cm97g@g2.nctu.edu.tw

C.-Y. Kao
e-mail: cy.kao@globalunichip.com

C. H.-P. Wen
e-mail: opwen@g2.nctu.edu.tw

1 Introduction

Failure analysis that collects and analyzes data to determine the cause of failures is critical for yield ramp-up of manufacturing integrated circuits (ICs). For typical failure analysis, *diagnosis* is the debugging process of identifying faults as possible defects and provides their locations to be inspected on silicon for later physical repair. However, as the process technology scales down, failure mechanisms such as electromigration and stress voiding collectively result in intricate and dynamic fault behaviors on ICs and make deterministic fault models such as stuck-at faults no longer effective for IC testing and diagnosis. As a result, many advanced fault models such as open faults and bridge faults are developed to address such issues.

Open defects, the unintended breaks or electrical discontinuities in interconnects, are one of the most important production defects and are vulnerable to the failure mechanisms in the deep submicron regime. They expose greater behavioral complexity than bridge defects and become more challenging to IC testing and diagnosis [6, 19].

Open defects can be usually classified into *intra-gate opens* and *inter-gate opens*: inter-gate opens come with an infinite resistance that disconnects the charge path or discharge path to the gate output whereas intra-gate opens are typically regarded as stuck-open faults.

Particularly, inter-gate opens have significant impacts on signal propagation and consist of: (1) *resistive opens* and (2) *complete opens*. Figure 1 illustrates these two categories. *Resistive opens* are also known as *weak opens* under which the current still passes through the narrow open defects on the net due to the tunneling effect [19]. *Complete opens*, on the other hand, are

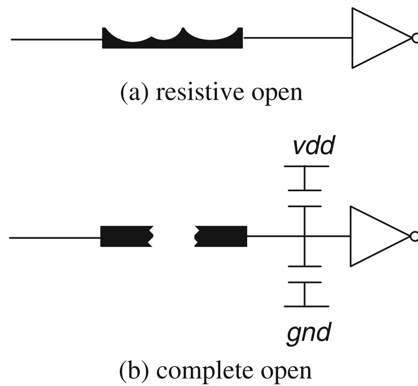


Fig. 1 Two types of inter-gate open defects

often termed as *strong opens* which make the driven gates of the net floating permanently. According to [4, 22], the majority of open defects are of the inter-gate type and a high percentage of the known open defects in metal lines belong to strong opens. Therefore, complete opens are the defects of interest and modeled as faults for diagnosis in this paper.

However, inspecting an entire net on silicon to isolate defect locations can be time-consuming. Moreover, from a logical perspective, when a net that drives multiple gates is open, all of the driven gates have faulty values. However, from a physical perspective, such a logical net can be further divided into multiple segments on the layout where each segment only drives a subset of total gates. Figure 2 shows an example of a net connecting one driver gate and three driven gates from two perspectives. In Fig. 2a, gate G1 drives gate G2, G3 and G4 through a logical net. If an open occurs on the net and then a fault is generated, all G2, G3 and G4 receive faulty values. However, considering the layout as shown as Fig. 2b, such logical net can be divided into five physical segments: segment A drives G2, G3 and G4; segment B drives G2 and G3; segment C drives G2; segment D drives G3; segment E drives G4. Faulty values do not necessarily appear on all of the gates. For example, if segment B is open, only G2 and G3 can receive faulty values.

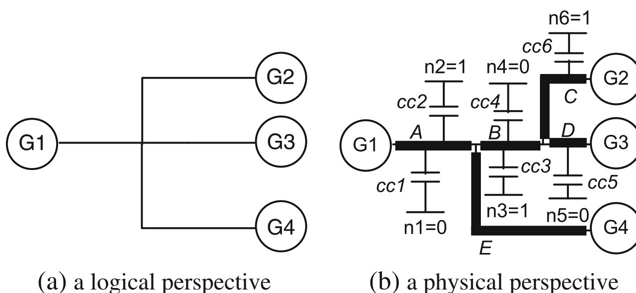


Fig. 2 A net connecting one driver gate with three driven gates

Even worse, along with the lower power supply and closer wires in the deep submicron regime, parasitic capacitances induce more complex circuit behaviors. Driven gates of one open segment are greatly influenced by the *Byzantine effect* [6, 29]. The Byzantine effect denotes the type of failures in which components of a system fail in arbitrary ways and manifests that different coupling condition (i.e. logic values on all coupling nets) of one floating net determines different faulty effects: logic values received by individual driven gates are decided by comparing the voltage value on the floating net with the thresholds of respective gates. If the floating-net voltage value is larger, the driven gate receives *logic 1* at its input. Otherwise, it receives *logic 0*. A open-segment fault appears when the logic values of the driver gate and the driven gate are different.

To take Fig. 2b for example again, if segment A is open and six aggressors, $n1$ to $n6$, as well as their coupling capacitances, $cc1$ to $cc6$, collectively result in a floating-net voltage greater than the threshold voltages of G2 and G4 but smaller than that of G3. Then G2, G3 and G4 receive *logic 1*, *logic 0* and *logic 1*, respectively. If G1 is with *logic 0*, then such open generates two faults on G2 and G4. A different coupling condition may only result in *logic 1* on G4 and thus makes only G4 receive the faulty value. Considering both the circuit layout and the Byzantine effect, an open that occurs on different segments under different coupling conditions result in various fault combinations. Therefore, open-segment defects are often regarded as *dynamic* faults and accurate diagnosis of Byzantine open-segment faults can hardly ignore related physical information such as the circuit layout and the cell library.

As a result, traditional physically-independent approaches can no longer work effectively on diagnosing Byzantine open-segment defects. Moreover, as multiple defects coexist, single-location-at-a-time (SLAT) patterns [11, 12] have difficulty on differentiating the output responses under the single defect assumption from the ones under the multiple, simultaneously-active defect assumption. Therefore, to propose a better diagnosis approach, physical information needs to be incorporated during data analysis of test patterns and output responses.

In this paper, for diagnosing multiple open-segment defects, we propose a three-stage approach which consist of (1) the failing patterns are used to identify the possible faulty nets that can be expanded into multiple segments as defects; (2) the passing patterns facilitate eliminating logically false candidates; (3) physical circuit simulation verifies each combination against all test patterns and output responses. One of the core problems in the proposed approach is how to precisely

formulate the relationships between test patterns and output responses. Accordingly, an enhanced integer-linear-programming (ILP) based approach is developed based on [9] with the objective to identify combinations of the least segments that can correctly explain output responses for all passing and failing patterns. As a result, the proposed approach demonstrates its effectiveness through good resolutions (<5 combinations) for all ISCAS'85 benchmark circuits with the injection of 2–5 open defects under random patterns and 5-detect stuck-at patterns. The total numbers of segments on average to be inspected on silicon are also greatly reduced and less than $2X$ of total numbers of injected open defects.

The rest of the paper is organized as follows. In Section 2, the diagnosis problem is formulated and different approaches of previous researches are reviewed. The fault model for an open-segment defect is elaborated in Section 3. Section 4 first outlines our ILP-based diagnosis and then details three stages including *faulty-net candidate identification*, *k-net fault generation* and *k-segment fault composition*. In Section 5, ISCAS85 benchmark circuits are used for experiments and provide results to demonstrate the effectiveness of the proposed approach on the basis of random patterns and 5-detect stuck-at patterns, respectively. Last, conclusions and future work are presented in Section 6.

2 Previous Research

The review of researches on open defects first starts with several important works devoted to ATPG and diagnosis. To address the problem of the Byzantine effect, related works are further presented to brief its impact and the proposed techniques. Later, various state-of-the-art techniques for multiple-defect/fault diagnosis with or without fault models are also compared and contrasted in this paper. Finally, the requirements of this research are summarized on the basis of the previous works.

Studies on open defects are first dedicated to logical testing and diagnosis [6, 7, 16, 25, 29]. Venkataraman et al. from [25] proposed a diagnostic fault model to capture the faulty behavior of an open defect as well as a path-tracing procedure to limit the interconnect size for analysis. In [16], Reddy et al. proposed a gate-level fault model for *interconnect opens* whose number grows exponentially in terms of the fanout size. Information of fanout branches and primary output was used to efficiently reduce the list of faults to be considered. Desineni et al. from [2] proposed DIAGNOSIX, which automatically extracts fault model using limited layout information. The fault model extracted from DIAGNOSIX is accurate and provides much information of the defect behavior as well as its location. Huang from [6, 7] narrowed down the fault locations to the segment level and proposed a logic diagnosis approach for single *open-segment faults* where fault candidates are collected from structural analysis. Zou et al. considered the routing topology and coupling capacitance to reflect the Byzantine effect during diagnosis of open-segment defects in [29]. Enhancing the diagnosis of full open defects in interconnect is proposed in [18] where vias and metal lines are considered. In addition, the information of quiescent current consumption is also used to rank diagnosis candidates.

Later, the research direction of open defects was extended to test generation and several works were published. Spinner et al. from [21] proposed an algorithm to generate test patterns for a single open defect considering the physical information and the Byzantine effect simultaneously. A selection scheme was developed to force specific values on aggressors for fault activation and later propagation of a fault was verified under the generated pattern. Lin et al. presented three different strategies of test generation for interconnect opens in [13] and demonstrated their patterns could help improve the diagnosis resolution. Hillbert et al. proposed the segment stuck-at fault model to generate test patterns for interconnect opens in [5] and identified the testability for each interconnect open defect without accurate technology information.

On the other hand, diagnosis of failing ICs for multiple faults or defects has also been extensively studied and become more important with the ever increasing number of gates and the density of circuits. Current CMOS ICs which has six or more metal interconnect layers along with numerous vias and contacts are susceptible to open fault [3, 10, 25]. In addition, according to Semiconductor Industry Association (SIA) roadmap, the total wire length grows exponentially [20]. Above reasons increase the likelihood of occurrence of open faults. Therefore, several works [14, 15, 25] have been published to target multiple open defect diagnosis.

Veneris et al. demonstrated an incremental diagnosis for multiple stuck-at faults in [26] where one single location modeled as a fault iteratively resembled the final combination. Bartenstein et al. from [1] further proposed the use of single location at-a-time (SLAT) patterns to diagnose multiple stuck-at faults. By observing simulation results of SLAT patterns, multiplets are collected as the potential faults. Later, in [11, 12], SLAT patterns were also extended into a special diagnostic patterns named SO-SLAT patterns to achieve higher resolutions for multiple faults.

However, the presence of multiple defects (or faults) makes SLAT patterns no longer guarantee the activation and propagation of faults. Moreover, efficient SLAT-based diagnosis often requires an *equivalent fault* evaluation in advance which is hard to be performed on open-segment defects.

Rather than diagnosing one defect (or fault) at a time iteratively, a X-fault model for physical diagnosis of unknown behaviors was proposed by Wen et al. in [27]. An efficient X-fault simulation and diagnostic reasoning were also presented to improve resolutions. Later, Yu et al. proposed a multiple defect diagnosis for arbitrary defect behaviors by only analyzing failing pattern characteristics in [28]. Path-based site elimination was shown to effectively reduce the number of candidates and capable of capturing 65% of the defective sites. To bring together interconnect opens and multiple defects, Liu et al. from [14] proposed a two-stage incremental diagnosis in which a list of fault tuples of multiple open defects was first identified to correctly explain all output responses followed by a generalized fault simulation to enumerate all possible faulty behaviors from the first stage.

However, for open-defect analysis at the logic level, the Byzantine effect makes previous defect/fault reasoning not accurate enough. All static diagnosis techniques fail to perfectly explain those multiple-fanout nets in which some fanout gates receive correct values and others receive faulty values. Therefore, in this work, we intend to design a dynamic approach which considers the Byzantine effect for multiple open-defect diagnosis. Furthermore, for the ease of future silicon inspection, defects are set to be reported at the segment level to provide a better resolution. Based on the analysis of previous works, it is necessary to incorporate the related physical information such as the circuit layout and the cell library into our diagnosis approach.

3 Fault Model of an Open Segment

Fault models for open defects have already been discussed in previous researches [17, 21, 29]. In this paper, we target the fault model that describes an open on one segment of the net according to the physical layout. When a segment of one net is open, the node f on the floating side is regarded as an ***open-segment*** fault. The logic value of each driven gate is interpreted by comparing the floating-net voltage on f and the threshold voltage of the driven gate. If the floating-net voltage is larger, the logic value is interpreted as a *logic-1* to the input of the driven gate; otherwise, it is interpreted as a *logic-0*. A fault is generated and received by the driven gate if the logic value from the driving side is different from that received by the driven side. As a result, not all driven gates of a floating net will receive faulty values.

As shown in Fig. 3, the floating-net voltage V_f can be formulated into the following equation:

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

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 sum of the capacitances with logic 1 and logic 0, respectively. Furthermore, the values of C_0 and C_1 can be decomposed into:

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

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

where C_{vdd} and C_{gnd} are the capacitances between the floating net and the power, and between the floating net and the ground, respectively. C_{i0} and C_{i1} are the internal capacitances residing inside the driven gate. C_{n0} and C_{n1} are the total capacitance between the floating net and its coupling nets with *logic* 0 and *logic* 1, respectively.

In reality, it is difficult to measure trapped charge Q_t and internal capacitances, C_{i0} and C_{i1}. Moreover, as [19] indicates, C_{n0} and C_{n1} typically dominate the total capacitance. Therefore, similar to [13, 21], the computation of the floating-net voltage in this paper only considers the parasitic capacitances between the floating net and its coupling nets C_{n0} and C_{n1} without loss of generality. The ratio of Q_t to C_{gnd} can also be defined as a controlling parameter α decided by the user. Therefore, the computation of the floating-net voltage V_f is

$$V_f = V_{dd} \times \frac{C_{n1}}{C_{n0} + C_{n1}} + \alpha \tag{4}$$

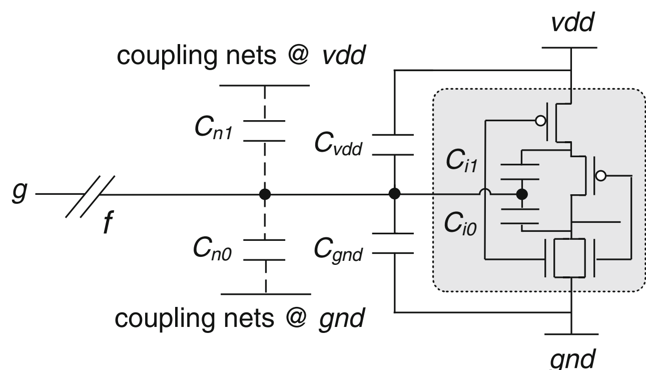


Fig. 3 The fault model for an open defect

Given the voltage V_f of a floating net f and the threshold voltage Vth of one driven gate G, if $V_f > V$th, G receives *logic* 1 from f; otherwise, G receives *logic* 0. Whether a fault will appear on the driven gate G or not further depends on the comparison of the logic values between the driven gate G and the driver gate R. Suppose that l_R and l_G denote the logic values at the output of the driver gate R and at the input of the driven gate G. If $l_R \neq l_G$, then G receives a faulty value.

Figure 4 illustrates the Byzantine effect on two different segments being open. Suppose that V_{t2}, V_{t3} and V_{t4} denote the threshold voltages for G2, G3 and G4, respectively and $V_{t2} < V_{t3} < V_{t4}$ according to the cell library. Considering the physical layout, an open on segment A can only result in one of faulty-gate combinations, {G2, G3, G4}, {G3, G4}, {G3} and {G4}, while in contrast, an open on segment B can only result in one of the combinations, {G2, G3} and {G3}. Note that faulty-gate combinations only mean that each gate in one combination can but not necessarily receive a fault.

The faulty behaviors further depend on the knowledge of the logic value on the driver gate, that is, l_{G1} in Fig. 4. Given $V_{t2} < V_f < V_{t3}$, $l_{G2} = 1$, $l_{G3} = 0$ and $l_{G4} = 0$ in Fig. 4a and $l_{G2} = 1$, $l_{G3} = 0$ and $l_{G4} = l_{G1}$ in Fig. 4b. If $l_{G1} = 1$, then $G3$ and $G4$ receive faults simultaneously in Fig. 4a while only $G3$ is faulty in in Fig. 4b.

Under the multiple-defect assumption, if one open-segment defect is activated under a pattern, the fault can be masked due to the logic value on one of coupling nets is replaced by a faulty value. Vice versa, it is also likely that an inactive open-segment defect is activated by another fault passing through its coupling net. Fault masking effect is highly complicated and will be detailed in Section 5. Because the activation of an open defect requires specific assignments on coupling nets, its fault equivalence requires a more robust and rigorous definition. Therefore, for simplicity, we only treat each open-segment defect as an independent fault assuming no equivalent fault throughout this paper. As a result, the total number of open-segment defects is the total number of segments in the circuit.

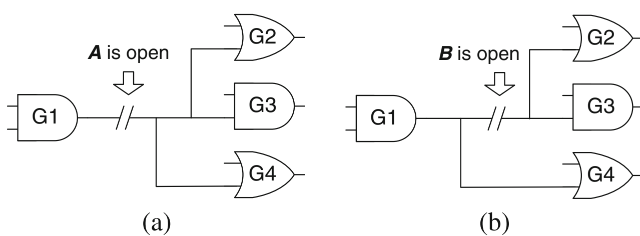


Fig. 4 The Byzantine effect on different segments

4 Three-stage Integer-Linear-Programming (ILP) Based Diagnosis

Based on the open-segment fault model in Section 3, the next step is to find the set of faults that matches the simulation results with the given patterns. Therefore, a three-stage diagnosis approach is proposed to determine the number of faults as well as the corresponding segment combinations. Figure 5 shows the overall flow of the proposed approach consisting of three stages: (1) *faulty-net candidate identification*, (2) *k-net fault generation* and (3) *k-segment fault composition*. In stage 1, faulty-net candidate identification is developed and based on traditional logical filtering of candidate sites [6, 14, 27]. Under Byzantine effect, it is hardly to result in the same faulty behaviors for two different fault combinations because the activation of each open defect requires specific assignments on coupling nets. Fault equivalence of multiple open-segment faults requires a more robust and rigorous definition. Besides, we follow the spirit of Occam's Razor, often summarized as "the simplest explanation is most likely the correct one" to design our diagnosis flow. As a result, we only find the smallest set from all fault combinations to explain failing POs. By encoding the candidate nets as a binary-integer-programming (BIP) problem, a ILP solver is able to find net combinations as k-net faults incrementally where k starts from 1. If no net combination can be found to correctly explain the patterns expressed by the ILP constraints, k increments by 1 until a feasible solution is found. Logical pruning by X simulation is also performed to reduce the size of k-net faults before the end of stage 2. In stage 3, the injection of an open on segments with the aid of physical information ensures that the remaining k-segment faults can result in the expected behaviors on the circuit under all patterns and thus requires further inspection on silicon.

4.1 Faulty-net Candidate Identification

In stage 1, the proposed diagnosis flow generates a list of net candidates as defect locations according to failing patterns. Each candidate in the list is called a ***faulty-net candidate***. The initial *fault-net candidate* list is computed by *path tracing*. Path tracing starts from an erroneous primary output (hereafter, EPO) of one failing pattern. Nets are identified as defect locations and stored into a list of candidates. This process iterates to update the faulty-net candidate list for each EPO until failing patterns are completely explored.

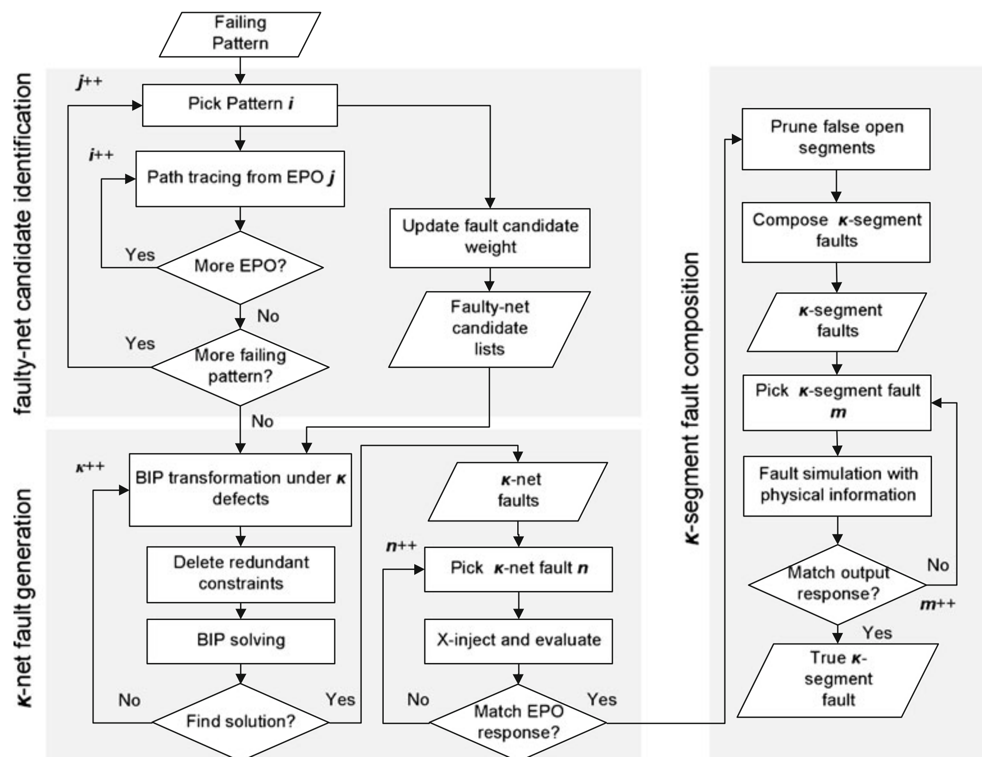


Fig. 5 Three-stage ILP-based diagnosis flow

Backtracking algorithm is applied to every EPO from failing patterns during path tracing. For each EPO, it traces the circuit backward to find the nets on which an open can explain the output mismatch. If multiple fanins of a gate have controlling values, only those controlling-fanin nets are considered and collected as faulty-net candidates. If all of fanins are non-controlling, all fanin nets are collected and the backtracking continues to search each fanin net. Figure 6 shows an example where path tracing starts from net H. Considering the controlling values on both fanins of gate 5, net F and net G are faulty-net candidates and collected into the list. For net G, both fanins are also collected because net B and net E are non-controlling simultaneously.

For a net i, $w_i = 1$ is labeled if an open occurring on net i can **fully** explain one EPO. But if $0 < w_i < 1$ is labeled, an open on net i can **partially** explain one EPO. When path tracing starts from one EPO, it is assigned with the full weight 1. Given a specific weight w on the output of a gate, if all fanins of the gate have non-controlling values, all fanin nets are assigned weight w. If the gate has N simultaneous controlling-value fanins, the weight is divided and each fanin receives w/N. In summary, when a net receives multiple weight assignments from different stems, all the weights are summed up as its final weight.

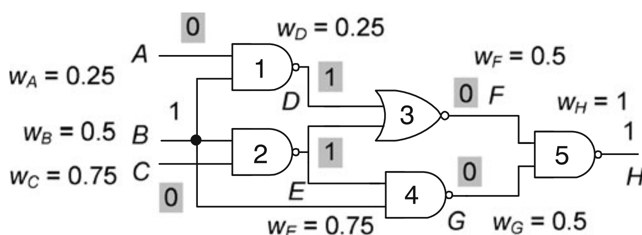


Fig. 6 An illustrative example for path tracing and weight assignment

To give an example, a circuit under test is shown in Fig. 6 with the logic-value assignments on all gates. Path tracing starts from net H with weight $w_H = 1$. Both net F and net G have controlling values (as highlighted) to the gate connecting net H, $w_F = w_G = 1/2 = 0.5$. Similarly for net F, considering that both net D and net E have controlling values (as highlighted), $w_D = w_{E_1} = w_F/2 = 0.25$. However, $w_{E_2} = w_B = w_G = 0.5$ since both net E and net B have non-controlling values to the gate connecting net G. Therefore, $w_E = w_{E_1} + w_{E_2} = 0.5 + 0.25 = 0.75$. As a result, $w_A = 0.25$, $w_B = 0.5$, $w_C = 0.75$, $w_D = 0.25$, $w_E = 0.75$, $w_F = 0.5$, $w_G = 0.5$ and $w_H = 1$.

4.2 k-net Fault Generation

The candidate list extracted from stage 1 only collects the faulty-net candidates labeled with non-zero weights

as the capability of correctly explaining EPOs under one failing pattern. In stage 2, we further explore the combinations of faulty-net candidates that can fully explain every EPO under all failing patterns satisfactorily. Note that a combination of k faulty-net candidates is termed as a ***k*-net fault** hereafter.

The weight assignments for candidates are used to transform the search of combinations of net faults into a binary integer-programming (BIP) problem. For example, given three EPOs under one failing pattern with the candidate list L_1, L_2 and L_3 extracted from stage 1:

$$L_1 = \{A, B, C, E\} \qquad \text{for } EPO_1$$
$$L_2 = \{B, E\} \qquad \text{for } EPO_2$$
$$L_3 = \{A, D\} \qquad \text{for } EPO_3$$

where A, B, C, D and E are nets of the circuit.

To further decide the size k and the set of k-net faults that can fully explain all EPOs, the corresponding BIP problem can be expressed into:

$$n_A + n_B + n_C + n_E \geq 1$$
$$n_B + n_E \geq 1$$
$$n_A + n_D \geq 1$$

where all net variables, n_A, n_B, n_C, n_D and n_E, are binary and represent if an open occurs on net A, B, C, D and E, respectively. Another constraint equation denoting the assumption for the size of k is also added as follows.

$$n_A + n_B + n_C + n_D + n_E = k$$

The above equation enforces that exact k opens can occur on net A to net E simultaneously.

The BIP problem is solvable and has at least one k-net fault if some of the net variables are 1. These variables jointly form a combination of a fault that can correctly explain EPOs under all failing patterns. The ILP solver starts to find solutions from $k = 1$. If no feasible solution can be found, k increments by 1 until a solution is found. As a result, four 2-net faults, (A, B), (A, E), (B, D) and (D, E) are found.

To further eliminate the faults that cannot satisfactorily explain all failing patterns, the weights of the candidates are added as the additional constraint equations during the BIP solving. Therefore, given the set S of k-net faults for all EPOs, the BIP problem can be updated as follows:

$$\sum_{n_i \in S} w_i \times n_i \geq 1 \tag{5}$$

$$\sum_{n_i \in S} n_i = k \tag{6}$$

where each EPO under one failing pattern corresponds to one constraint equation represented by Eq. 5. After applying the ILP solver, all k-net faults that can correctly explain all failing patterns are reported. Note that repeated constraints are first removed from the constraint equations to reduce the runtime of the solution generation.

For example, according to the result of path tracing in Fig. 6, the boundary inequality equation for such pattern can be expressed into:

$$w_A n_A + w_B n_B + w_C n_C + w_D n_D$$
$$+ w_E n_E + w_F n_F + w_G n_G + w_H n_H \geq 1$$

where n_A, n_B, n_C, n_D, n_E, n_F, n_G, n_H are all binary and $w_A = 0.25$, $w_B = 0.5$, $w_C = 0.75$, $w_D = 0.25$, $w_E = 0.75$, $w_F = 0.5$, $w_G = 0.5$ and $w_H = 1$. More inequality equations can be added if other failing patterns are provided. At last, the equation denoting the size k for fault combinations is also added:

$$n_A + n_B + n_C + n_D + n_E + n_F + n_G + n_H = k$$

where exact k opens can occur among net A, B, C, D, E, F, G and H under all failing patterns.

These equations and weight assignments effectively limit the total number of solutions reported by the ILP solver. However, when reconvergences of multiple faults occurs in the circuit with respect to one failing pattern, the k-net fault found by the ILP solver may no longer correctly explain the reconvergent scenario. Taking Fig. 6 for example, based on the previous constraint, (B, E) has the weighted sum 1.25 and can be reported as one 2-net faults by the ILP solver. However, under the failing pattern, an EPO occurring on net H depends on the propagation of multiple faulty values through the nets connecting inputs of gate 3 and gate 5. An open on (B, E) fails to create a faulty value on net D connecting the input of gate 3 and hence no fault can be propagated to net F. Thus, net H will not be affected by the (B, E) combination.

To avoid generating such **fake** faults, constraints called *fault-propagation trees* (f.p.t.) are further added for better guidance on our BIP solving.

Definition A *fault-propagation tree* t^i_j is a tree for traces of signal propagations and traverses backwards in the circuit topology under one failing pattern. Its root locates the net j connecting the input of a *controlling-reconvergent* gate i and path tracing finds nets to construct the fault propagation tree in a backward manner. Each of its leaves locates a PI or a net connecting the output of another controlling-reconvergent gate.

In the definition, a *controlling-reconvergent* gate denotes the gate with multiple inputs of simultaneously controlling values under the pattern. Taking Fig. 7 for example, both net F and net G have controlling values (as highlighted). Hence, gate 5 is one of *controlling-reconvergent* gates. Similarly, gate 3 is also a *controlling-reconvergent* gate since both net D and net E have controlling values. For gate 5, two fault propagation trees t_F^5 and t_G^5 are generated on net F and G. Similarly, for gate 3, two fault propagation trees t_D^3 and t_E^3 are generated on net D and E. Later, each fault-propagation tree is described as an individual constraint and formulated into:

$$\sum_{n_r \in t_j^i} n_r \leq k - 1 \tag{7}$$

where net r is one net of the fault-propagation tree t_j^i. Note that if t_j^i consists of only one net, the constraint is of no use and does not need to be added in the BIP problem. Fault propagation trees t_F^5 in Fig. 7 is such example since gate 3 is another *controlling-reconvergent* gate. Therefore, fault propagation tree t_F^5 will not be added in the BIP formulation. As to the derivations of other fault propagation trees t_G^5, t_D^3 and t_E^3 details will be provided soon.

Each of the above constraints means that at most $(k-1)$ defects can occur in t_j^i and leaves one defect in another fault-propagation tree of gate i. Besides, in our diagnosis flow, the number of faults is a sufficient condition. Changing $(k-1)$ in Equation 7 to $(k-2)$ is over-constrained. For example, in Fig. 8, gate 2 is a three-input gate where two inputs are stemmed from gate 1. Therefore, at most $(k-1)$ defects (2 in this case) is specified in t_A^2, and we need to leave at least one defect for another fault-propagation tree t_D^2 of gate 2. As a result, for opens on (A, B), the net E will be affected and receive three faulty inputs. No other fault combination can affect net E because it can only change one of the controlling inputs of the EPO gate.

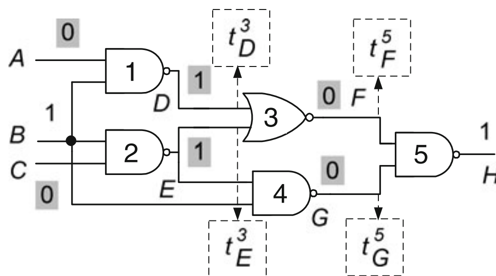


Fig. 7 An illustrative example for fault propagation trees

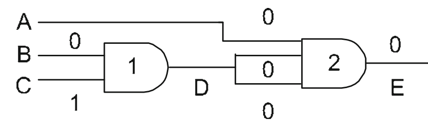


Fig. 8 An illustrative example for fault-propagation tree constraints

To take Fig. 6 for example again, gate 5 is one controlling-reconvergent gate and faulty values need to propagate through both net F and net G to result in an EPO on net H. Therefore, path tracing ends up with finding constraints for t_F^5 and t_G^5. In Fig. 9a, since gate 3 is also one controlling-reconvergent gate, the constraint for t_F^5 with only net F need not to be generated but two following constraints for t_D^3 and t_E^3 are added accordingly. In Fig. 9c, path tracing finds net B, C, E and G, for t_G^5 in a backward manner. As result, the equations for all f.p.t. constraints are formulated as:

$$n_A + n_D \leq k - 1$$

$$n_C + n_E \leq k - 1$$

$$n_B + n_C + n_E + n_G \leq k - 1$$

After adding these f.p.t. constraints, fake faults such as $\{B, E\}$ will not be reported by the ILP solver.

Considering the physical layout, each net in one k-net fault may consist of multiple segments and thus the size of k-segment faults can grow exponentially. For example, if a 3-net fault $\{A, B, C\}$ that can be physically divided into segment set $\{A_1, A_2, A_3\}$, $\{B_1, B_2\}$ and $\{C_1, C_2, C_3\}$, respectively, the total number of 3-segment faults corresponding to this fault is $3 \times 2 \times 3 = 18$. To avoid the exponential growth on the size of k-segment faults, a *X-inject-and-evaluate* approach is applied and logically prunes the false cases of k-net faults.

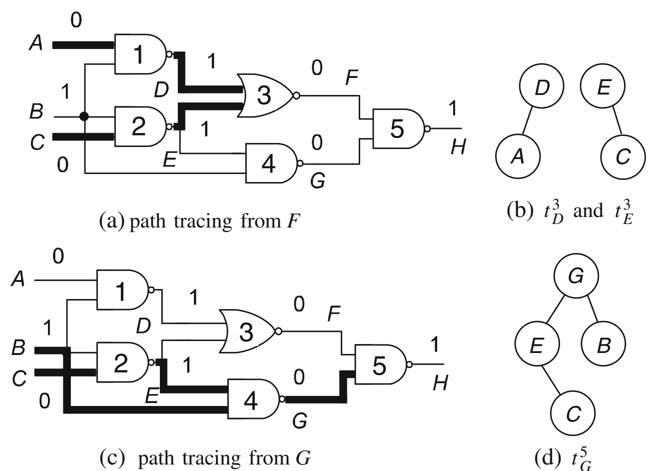


Fig. 9 Path tracing of fault-propagation trees

X simulation is a common technique used in fault diagnosis; the proposed X-inject-and-evaluate approach can be viewed as an extension of X simulation. X's are assigned to each net in one k-net fault and propagate towards the outputs simultaneously. If X's cannot be observed at all EPOs under one failing pattern, then this k-net fault is a false case and should be removed. Figure 10 illustrates two examples for the X-inject-and-evaluate approach. In Fig. 10a, suppose that 2-net fault (E,G) is the target. After X's are injected on E and G, one X is blocked at gate 3 due to the controlling-value side-input connecting D. Therefore, X cannot be observed on the only EPO (net H) and thus (E,G) is removed. In Fig. 10b, X's are injected on B and F and can successfully result in one X on net H. Therefore, 2-net fault (B,F) is reserved.

4.3 k-segment Fault Composition

During this stage, k-net faults are further verified and expanded into k-segment faults via two steps. In step 1, nets in k-net faults remained from stage 2 are expanded into individual segment lists to compose the corresponding k-segment faults and physical pruning is also performed to eliminate k-segment faults that cannot satisfactorily explain output responses of all patterns through fault simulation with the aid of physical information.

k-segment fault composition first starts from enumerating a list of segments for each net remained in the k-net faults and composes the k-segment faults from the segment lists. Take the circuit layout shown in Fig. 2b for example. Suppose two faulty values appear on G3 and G4 under one failing pattern. Since an open on segment B, C or D can never result in a faulty value on G4 and an open on segment E can never result in a faulty value on G3, segment A is the only candidate on which an open can result in faulty values on G3 and G4 simultaneously. Therefore, the segment list for this net only contains $\{A\}$.

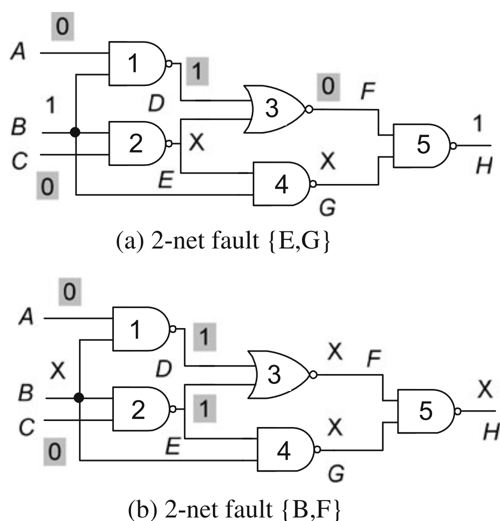


Fig. 10 Two examples for X-inject-and-evaluate

Later, step 2 verifies if each k-segment fault can result in correct output responses under both failing and passing patterns. Such verification is done by the fault simulation in aid of physical information such as the circuit layout and the cell library. One k-segment fault will be removed if injecting opens on all segments in this fault cannot match the trace of signal propagation under one pattern (either failing or passing). Finally, the remaining segments in k-segment faults are treated as the true defect locations for future silicon inspection.

When an open is injected to one segment in the circuit, information about the physical routing of the net, the related coupling capacitances, and threshold voltages are gathered from the circuit topology and the cell library. After one pattern is assigned at the primary inputs, the logic values can be derived on each net accordingly. The voltage on the open segment can be thus computed by Eq. 1 followed by comparing this value with the threshold voltage of one driven gate to decide if a fault is generated. For example, in Fig. 2, given that an open is injected on segment A and the coupling capacitances for $n1$ to $n6$ are 2, 3, 1, 2, 1 and 4 fF, respectively. $n2$, $n3$, and $n6$ are with *logic* 1 whereas $n1$, $n4$, and $n5$ are with *logic* 0. Therefore, $C_1 = cc2 + cc3 + cc6 = 8$, $C_0 = cc1 + cc4 + cc5 = 5$ and the floating-net voltage on segment B is $V_B = 8/(5+8) = 0.615Vdd$. Suppose that the threshold voltages for $G1$, $G2$ and $G3$ are $0.4Vdd$, $0.5Vdd$, $0.7Vdd$, respectively. Both $G1$ and $G2$ receive *logic* 1 while $G3$ is unchanged. After the driven-gate responses are computed and propagated toward the primary outputs, the simulated and real output responses are compared. Any mismatch will remove such k-segment fault from the fault list.

4.4 Comparison of Applied Constraints

At the end of this section, a preliminary comparison of applying the different constraints to the BIP solving is presented. Table 1 shows the results on three small ISCAS benchmark circuits under the random injection of two open defects. The first column shows the circuit name and the second column shows the types of applied constraints: *or*, *wa* and *fp* denotes the original, weight-assignment and fault-propagation-tree constraints, respectively. The third (N_{knf1}), forth (N_{knf2}) and fifth (N_{ksf}) columns show the total number of k-net faults

Table 1 Comparison of applied constraints to the BIP solving

Circuit	Const.	N_{knf1}	N_{knf2}	N_{sgf}	Time (s)
c432	or	272.65	140.14	591.64	1.27
	or+wa	139.07	77.43	424.39	0.69
	or+wa+fp	92.10	62.01	310.99	0.74
c499	or	2528.41	292.59	2726.64	11.17
	or+wa	487.54	256.65	2157.73	2.45
	or+wa+fp	436.66	182.13	1504.51	1.94
c880	or	418.57	136.60	1353.36	2.00
	or+wa	221.20	130.47	1303.53	1.62
	or+wa+fp	198.90	108.94	821.64	1.53

that a ILP solver CPLEX generates, the total number of k-net faults after applying X-inject-and-evaluate, and the total number of k-segment faults at the end, respectively. The last column shows the time that the BIP solving uses for k-net fault generation.

According to Table 1, both weight-assignment and fault-propagation-tree constraints effectively reduce the total number of k-net faults as well as that of k-segment faults. Moreover, both constraints also jointly facilitate k-net fault generation by using less time on the BIP solving. Note that on c432, adding weight-assignment and fault-propagation-tree constraints to the BIP solving takes slight more time than adding weight-assignment constraints alone does. This is because the proposed approach extracts more fault-propagation-tree constraints than weight-assignment constraints during analysis and requires extra time on parsing the fault-propagation-tree constraints. However, for the cases of c499 and c880, adding the fault-propagation-tree constraints can still result in additional saving on the total runtime.

5 Experimental Results

The experiments were conducted on the ISCAS'85 benchmark circuits where the corresponding layouts and coupling capacitance information are available on the TAMU website [24]. Each ISCAS'85 benchmark circuit was synthesized using a 5-metal-layer TSMC 180 nm CMOS technology. Threshold voltages for each type of gates were obtained from SPICE simulations with the assumption of 0 trapped charge, i.e. $\alpha = 0$ in Eq. 4. For the BIP solving, we used the commercial mixed integer programming solvers named CPLEX [8] which particularly is capable of populating exhaustive solutions. One hundred samples with injections of different defect sizes (from 1 to 5) under 1,000 patterns are generated for each ISCAS 85 benchmark circuit. Two different types of test patterns are applied to investigate the impact of pattern quality on diagnosis as well. Table 2 shows the gate-level and physical information of ISCAS'85 benchmark circuits: the first column indicates the circuit names; the second column (N_{net}) indicates the total number of nets; the third column (N_{stem}) indicates the total number of nets with multiple-fanout stems; the forth column (N_{seg}) indicates the total number of segments contained in the circuits.

5.1 Applying Random Patterns

First, experiments on the 100 sample circuits under 1,000 random patterns were conducted. Tables 3, 4, 5 and 6 show the results with the injection of 2, 3, 4 and 5 random defects: the total number of failing patterns is shown in each second column (N_{fpttn}); each third column (N_{wa}) denotes the number of constraints transformed from Eq. 5; each forth column (N_{fp}) denotes the number of fault propagation tree constraints; each fifth column (N_{fnet}) denotes the number of net candidates collected from *faulty-net candidate identification*; each sixth column (N_{knf}) denotes the number of k-net faults generated from *k-net fault generation*; each seventh column (N_{ksf}) denotes the number of k-segment faults composed from *k-segment fault composition*; each eighth column denotes the stage 2 runtime and total runtime required by our diagnosis approach.

Although hundreds of random patterns were failing patterns, only few constraints were extracted and transformed into Eq. 5. During the stage of k-net fault generation, a large number of fault-propagation-tree (f.p.t.) constraints was generated and caused extra timing overhead on performing the reduction of fake f.p.t constraints and the BIP solving. The physical simulation effectively pruned the impossible segments and successfully reduced the total number of k-segment faults. As a result, an average of 5.2 k-segment faults was reported under different sizes of defects and run within less than an hour for most of the benchmark cir-

Table 2 Circuit information of ISCAS85 circuits

Circuit	N_{net}	N_{stem}	N_{seg}
c432	203	89	443
c499	275	59	566
c880	468	125	979
c1355	619	259	1,404
c1908	938	385	1,893
c2670	1,642	454	2,821
c3540	1,741	579	3,781
c5315	2,608	806	5,878
c6288	2,480	1,456	6,252
c7552	3,828	1,300	7,990

Table 3 $k = 2$ defects under 1,000 random patterns

Circuit	N_{fpttn}	N_{wa}	N_{fp}	N_{fnet}	N_{knf}	N_{ksf}	Time (s) stage2/total
c432	323.7	25.9	15025.2	137.8	186.7	2.6	2.0/2.9
c499	218.4	10.4	74704.2	151.2	307.2	3.6	8.8/12.6
c880	394.6	14.8	233257.8	128.6	133.4	3.2	2.6/3.8
c1355	436.8	8.6	33911.2	288.8	476.1	7.7	37.4/53.5
c1908	385.8	10.2	75374.6	538.5	1431.2	6.4	184.7/263.9
c2670	255.1	6.5	23609.8	227.6	181.5	4.5	40.2/57.5
c3540	205.8	12.1	103454.8	726.6	301.1	3.0	47.7/68.2
c5315	115.7	35.1	88578.8	387.9	768.6	2.2	110.7/158.2
c6288	270.5	33.3	949998.1	1743.7	1850.3	1.4	245.2/350.4
c7552	440.2	9.4	48506.1	586.3	1497.7	8.2	166.7/238.2
Avg	**304.7**	**16.4**	**791420.5**	**491.7**	**713.4**	**4.3**	**84.6/120.9**

Table 4 $k = 3$ defects under 1,000 random patterns

Circuit	N_{fpttn}	N_{wa}	N_{fp}	N_{fnet}	N_{knf}	N_{ksf}	Time (s) stage2/total
c432	408.1	27.9	16187.8	148.6	89.8	2.2	2.0/2.8
c499	381.4	16.1	3137.6	140.7	81.6	3.3	20.1/28.7
c880	698.2	35.2	13337.3	202.3	155.8	7.0	16.9/24.2
c1355	308.3	13.1	2021.4	391.4	1014.8	5.2	111.8/159.5
c1908	381.3	15.2	14505.7	516.3	260.4	1.3	254.0/362.9
c2670	254.3	5.5	6839.5	284.8	779.9	1.5	258.8/369.7
c3540	199.5	13.2	18544.3	895.8	818.3	2.0	286.1/408.7
c5315	324.7	42.7	6921.6	310.8	4363.3	8.5	506.4/723.4
c6288	187.0	30.1	63512.0	1153.1	1178.5	1.6	862.4/1232.0
c7552	259.6	10.1	7843.2	421.0	8430.3	5.5	520.5/743.6
Avg	**340.3**	**20.9**	**15285.0**	**446.5**	**1717.3**	**3.8**	**283.5/405.5**

Table 5 $k = 4$ defects under 1,000 random patterns

Circuit	N_{fpttn}	N_{wa}	N_{fp}	N_{fnet}	N_{knf}	N_{ksf}	Time (s) stage2/total
c432	414.4	360.5	29856.5	195.1	37386.5	3.6	198.0/253.9
c499	372.4	255.1	5282.3	172.2	41977.6	5.7	429.2/572.2
c880	634.3	448.3	4062.0	189.2	4852.4	4.2	332.3/443.4
c1355	322.1	96.3	3194.7	429.9	4293.2	4.7	441/0588.0
c1908	390.0	24.1	30151.1	533.4	11360.0	3.2	1143.0/1523.9
c2670	281.3	117.7	7218.3	305.3	2139.5	4.1	732.9/977.2
c3540	124.0	24.7	16383.6	1051.7	13303.4	3.0	791.3/1055.5
c5315	391.7	159.4	5120.3	492.5	37237.8	6.0	1703.3/2271.4
c6288	403.5	26.1	43103.5	1612.4	56130.3	2.8	3577.0/4769.3
c7552	312.3	89.1	11357.0	566.3	30246.9	5.5	2018.7/2691.6
Avg	**364.6**	**160.1**	**11573.0**	**464.8**	**23892.7**	**4.3**	**1136.0/1514.6**

Table 6 $k = 5$ defects under 1,000 random patterns

Circuit	N_{fpttn}	N_{wa}	N_{fp}	N_{fnet}	N_{knf}	N_{ksf}	Time (s) stage2/total
c432	422.1	388.2	17327.6	188.3	2303.0	5.3	418.4/523.4
c499	382.7	213.6	6248.6	192.4	51296.6	13.7	908.0/1135.3
c880	658.3	401.1	8732.0	200.7	7892.3	6.2	497.8/622.2
c1355	331.6	133.6	2937.4	473.0	3376.4	6.5	586.4/733.6
c1908	362.5	37.5	22171.2	557.4	112374.1	10.3	9432.0/11790.0
c2670	240.1	155.3	6293.8	352.2	8237.5	7.4	1224.8/1531.6
c3540	239.2	52.3	10936.1	1167.9	32562.2	6.0	1537.6/1922.1
c5315	417.2	157.0	4092.3	479.0	76237.2	7.3	3120.8/3901.3
c6288	385.5	23.2	24967.8	1663.6	213968.3	15.3	12487.2/15609.9
c7552	339.0	105.2	3825.6	592.2	137274.7	6.3	7065.6/8832.6
Avg	**377.8**	**166.7**	**10753.2**	**586.7**	**64552.2**	**8.4**	**3088.0/3860.2**

Table 7 5-detect stuck-at-fault patterns

Circuit	N_{5d}	Circuit	N_{5d}
c432	225	c2670	286
c499	267	c3540	528
c880	182	c5315	297
c1355	432	c6288	81
c1908	591	c7552	460

cuits. Note that random patterns that failed to provide sufficient information to diagnose injected open defects will demonstrate a low diagnosability which is defined as the ratio of the number of detected defects to the number of injected defects. Such issue will be explored on random patterns in details together with 5-detect patterns in the later section.

5.2 Applying 5-detect Stuck-at-Fault Patterns

To achieve a higher diagnosability, we tried to apply 5-detect stuck-at-fault patterns (if its total number is less than 1,000, random patterns are appended) to conduct the same experiments. The total number of 5-detect stuck-at-fault patterns generated from Mentor Graphics FastScan [23] for each ISCAS'85 circuits is presented in Table 7. Similarly, Tables 8, 9, 10, and 11 shows the diagnosis results with the injection of 2 to 5 random defects, respectively.

The results under 5-detect patterns reveal that more failing patterns are observed and thus more constraints provide useful information when diagnosing open defects. As a result, the average number of k-segment faults from 1,000 5-detect patterns are 4.0, less than 5.2 from 1,000 random patterns. More specifically, most numbers of the reported k-segment faults under 5-detect patterns are less than the corresponding numbers under random patterns (with 10 exceptions).

5.3 More Comparisons on Resolution and Diagnosability

Detailed comparisons between 5-detect patterns and random patterns of four circuits are presented in this section. *Resolution* denotes the ratio the number of reported faults to the number of detected faults. In advance, *net resolution* and *segment resolution* are defined separately and indicate the fault numbers for reported nets and reported segments, respectively. Figures 11 and 12 show the results for the net resolution and the segment resolution. Again, both the net and segment resolutions under random patterns are worse than those under 5-detect patterns for the most of the circuits. Such result meanwhile states that 5-detect patterns can reduce the number of net/segment candidates to be inspected on silicon in the future.

Based on Tables 3–11, as the circuit size or defect multiplicity increases, diagnosis time increases proportionally, which indicates that the applicability of our ILP method is still limited. The proposed constraints including fault-propagation trees in this paper only help to lower the difficulty of scalability.

Furthermore, the diagnosability are also compared in Fig. 13 and several findings can be concluded. First, diagnosability under random patterns decreases quickly when open defect size k increases. With fewer failing patterns, diagnosability from random patterns is inferior to that from 5-detect stuck-at-fault patterns.

Table 8 $k = 2$ defects under 1,000 5-detect stuck-at based patterns

Circuit	N_{fpttn}	N_{wa}	N_{fp}	N_{fnet}	N_{knf}	N_{ksf}	Time (s) stage2/total
c432	187.0	343.5	10751.0	180.4	186.5	2.4	4.9/7.0
c499	263.4	174.9	877.5	196.7	1817.4	2.3	49.1/70.2
c880	336.1	181.9	4391.5	167.6	98.9	2.1	2.7/3.8
c1355	742.0	672.8	3302.2	532.7	2817.5	1.9	148.7/212.4
c1908	864.0	1345.2	49030.6	767.9	3678.7	3.0	79.4/113.4
c2670	212.2	193.4	4911.4	707.8	1374.6	3.3	279.7/399.5
c3540	349.3	530.2	40906.3	1388.6	1309.4	1.7	259.0/370.0
c5315	227.1	265.9	3532.9	791.9	1093.6	1.8	124.7/178.1
c6288	71.6	52.3	7080.9	2216.6	428.2	1.3	1227.8/1754.2
c7552	723.1	534.9	15016.3	732.5	8243.4	5.5	1227.5/1753.6
Avg	**397.6**	**429.5**	**13980.0**	**768.3**	**2104.8**	**2.5**	**340.3/486.2**

Table 9 $k = 3$ defects under 1,000 5-detect stuck-at based patterns

Circuit	N_{fpttn}	N_{wa}	N_{fp}	N_{fnet}	N_{knf}	N_{ksf}	Time (s) stage2/total
c432	396.5	847.5	26064.0	195.0	5237.2	4.5	68.4/97.7
c499	398.7	336.1	2188.0	199.1	37501.7	3.3	127.9/182.8
c880	457.9	271.9	6887.6	219.6	1303.3	3.0	110.0/157.2
c1355	792.8	589.7	5698.9	570.8	9132.4	5.5	736.4/1052.7
c1908	235.5	313.5	11383.3	706.7	1770.4	2.9	334.0/477.2
c2670	253.5	196.4	6839.6	1096.4	799.5	2.6	263.9/377.0
c3540	97.5	56.4	7994.2	1399.3	54.2	1.6	573.9/819.9
c5315	342.7	649.7	6921.0	1080.1	5131.5	2.2	751.8/1074.3
c6288	133.8	32.6	14618.4	1687.9	1707.4	1.4	911.7/1302.4
c7552	259.6	109.2	3921.6	420.1	5226.1	5.4	798.4/1140.2
Avg	**336.8**	**340.3**	**9251.7**	**757.5**	**6786.3**	**3.2**	**485.8/694.8**

Table 10 $k = 4$ defects under 1,000 5-detect stuck-at based patterns

Circuit	N_{fpttn}	N_{wa}	N_{fp}	N_{fnet}	N_{knf}	N_{ksf}	Time (s) stage2/total
c432	413.3	249.1	16230.4	201.6	3910.0	2.1	152.0/202.7
c499	421.5	383.6	5135.5	221.7	13060.6	2.8	420.8/561.1
c880	523.5	632.1	23413.1	306.4	11342.8	7.3	656.3/875.1
c1355	801.1	611.7	9922.0	562.5	10982.3	5.2	1523.3/2031.8
c1908	351.4	492.3	6732.9	746.1	6089.1	3.2	848.6/1131.4
c2670	302.9	518.2	7671.7	1503.0	5266.2	4.0	715.0/953.3
c3540	114.7	203.1	62329.5	1529.3	4865.1	6.2	1292.0/1722.6
c5315	486.2	492.0	18721.3	1277.1	6844.5	6.6	1663.5/2218.0
c6288	522.1	63.7	187831.6	1913.2	29131.3	5.9	2897.2/3862.9
c7552	430.1	183.5	37288.2	593.2	37079.4	8.5	2244.8/2993.1
Avg	**436.7**	**382.9**	**37527.6**	**885.4**	**12857.1**	**5.2**	**1241.4/1655.2**

Table 11 $k = 5$ defects under 1,000 5-detect stuck-at based patterns

Circuit	N_{fpttn}	N_{wa}	N_{fp}	N_{fnet}	N_{knf}	N_{ksf}	Time (s) stage2/total
c432	462.3	230.1	9235.3	187.1	1205.5	3.2	681.6/852.3
c499	452.2	329.3	6266.3	237.1	7264.1	3.7	1068.8/1336.6
c880	544.3	583.2	22398.5	293.5	8883.2	4.4	741.8/927.3
c1355	793.2	620.2	10344.8	525.3	13532.6	5.2	3673.6/4592.8
c1908	334.5	408.8	7295.5	662.6	3021.0	5.8	13625.6/17032.6
c2670	329.0	493.3	7327.6	1341.3	1793.2	3.1	3455.2/4319.3
c3540	122.2	221.6	43106.0	1472.4	2395.1	4.6	4153.6/5192.2
c5315	493.2	526.4	13681.1	1362.3	5927.0	6.6	4028.8/5036.3
c6288	502.3	48.0	73266.4	1966.3	39138.2	8.1	18521.6/23152.5
c7552	428.9	200.3	36207.3	475.6	23962.2	6.9	16165.6/20207.4
Avg	**436.7**	**366.1**	**22912.8**	**848.6**	**10712.2**	**5.2**	**6611.2/8264.9**

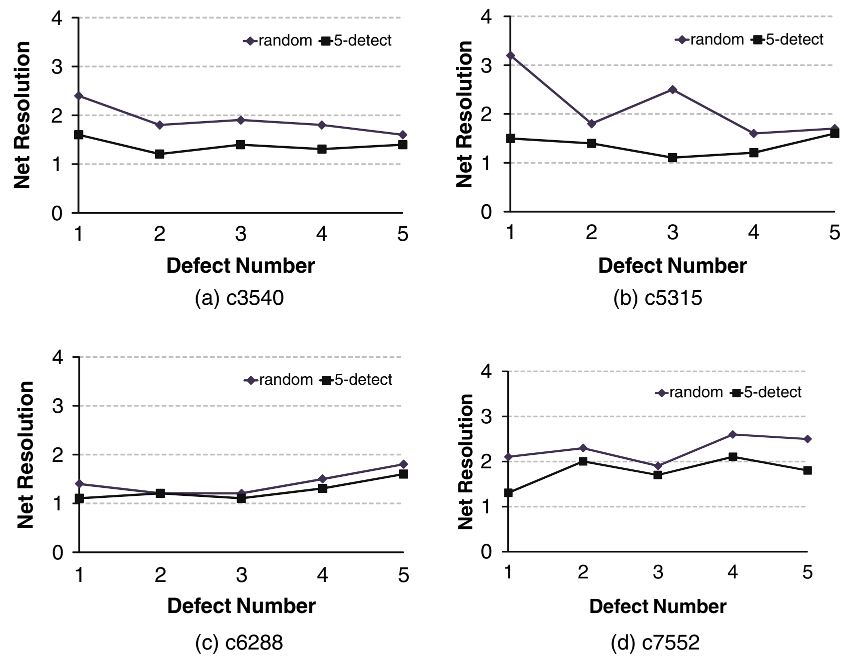


Fig. 11 Comparison of net resolution on four ISCAS circuits

Therefore, *pattern quality* (in terms of the number of failing patterns) is highly correlated to its diagnosability. Second, the decrease in diagnosability stems from how difficult the multiple open-segment defects can be activated or propagated. Intuitively, if the given pattern set fails to activate or propagate the faulty effect of some defects, only a subset of total defects can be identified on the basis of the limited information. For example, in Fig. 14, assume that four open-segment defects, f_A, f_B, f_C and f_D, are injected. Only f_A can explain EPO_1 while EPO_2 can be explained by either one of f_A, f_B and f_D. Moreover, only f_B can explain

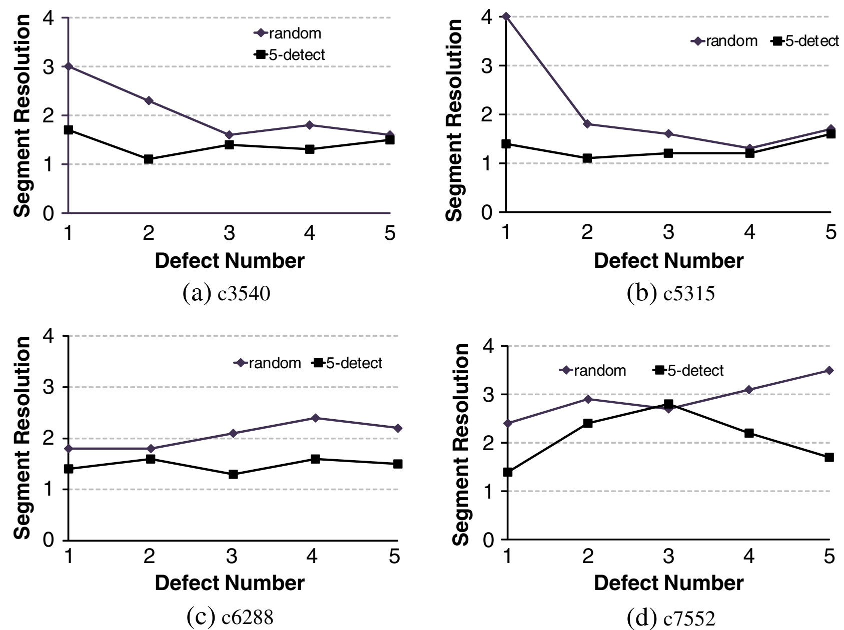


Fig. 12 Comparison of segment resolution on four ISCAS circuits

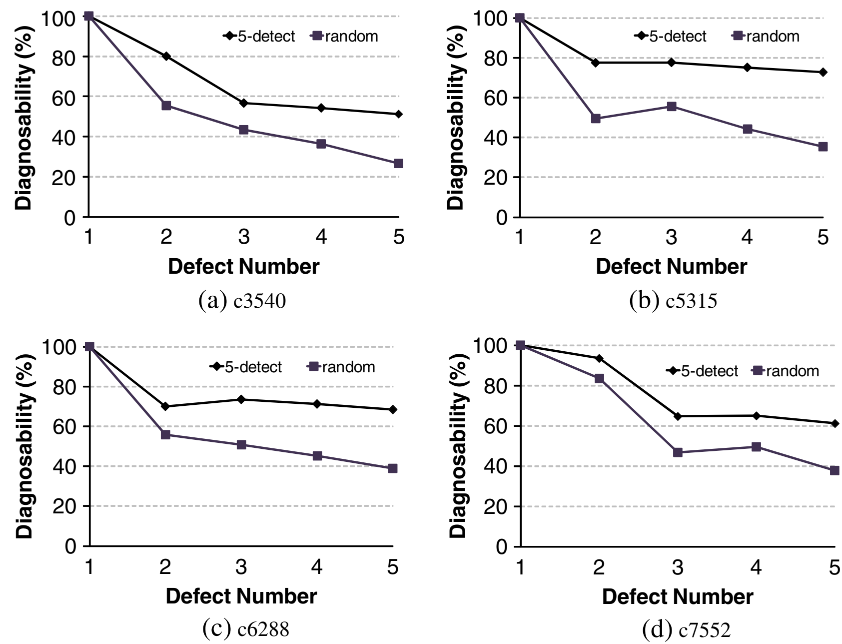


Fig. 13 Comparison of diagnosability on four ISCAS circuits

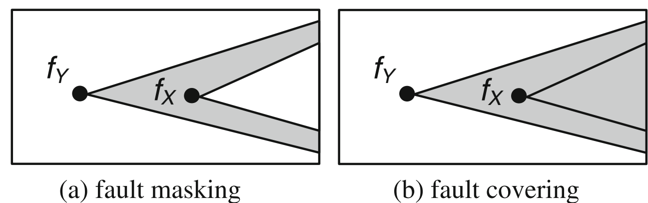
EPO_3 and EPO_4. Consequently, f_C does not lead to any EPO and thus is unlikely to be detected. In this case, this pattern set misses the faulty effect from f_C, and only f_A, f_B and f_D are detectable defects for diagnosis. Therefore, if one pattern set can expose failing EPOs for more defects, our approach will capture more defects precisely.

An open-segment defect that cannot be observed may stem from two effects: *fault masking* or *fault covering*. *Fault masking* means that the faulty effect generated from one defect is masked by another defect during propagation and Fig. 15a shows an example. f_X represents the defect being masked whereas f_Y masks the faulty effect generated from f_X. As a result, EPO tracing cannot locate f_X in this case. On the other hand, *fault covering* represents the EPO set induced by one defect covers the EPO set induced by another defect. Figure 15b illustrates the concept of fault covering where f_Y has a larger fault-propagation cone and its EPO set covers that induced by f_X. That is, $EPO_X \subseteq EPO_Y$.

Fault masking is a common effect on testing and often caused by logical conflicts during propagation of multiple stuck-at faults. However, for open-segment defects, fault masking may even worsen the fault activation when the activation condition required by one defect results in a conflict on one or more common coupling nets with another defect. To avoid non-covered EPO sets between two defects is important for diagnostic test pattern generation of multiple open-segment defects and deserves further investigation and experimentation.

	f_A	f_B	f_C	f_D
EPO_1	×		never activated	
EPO_2	×	×		×
EPO_3		×		
EPO_4		×		

Fig. 14 Association between EPOs and defects



(a) fault masking (b) fault covering

Fig. 15 Two effects that make one defect undiagnosable

6 Conclusion

Opens in metal lines are one type of the most important defects for failure mechanisms in the deep submicron regime. Open-segment defects belong to the majority of the open-defect types where the Byzantine effect makes faulty behavior of an open segment *nondeterministic* and thus diagnosing open-segment defects is more challenging. Fault activation and propagation depends on both input patterns and the physical information. In this paper, a three-stage diagnosis approach is proposed to generate k-segment faults as defects and consists of three stages including *faulty-net candidate identification*, *k-net fault generation* and *k-segment fault composition*.

For ISCAS'85 circuits, experiments were conducted on 100 different sample circuits with randomly injecting 2–5 opens on segments under random patterns and 5-detect patterns, respectively. Results show that the proposed approach can effectively generate 5.2 and 4.0 k-segment faults on average for 1,000 random patterns and 5-detect patterns, respectively. Moreover, we also demonstrated that 5-detect patterns reach higher diagnosability than random patterns do which also suggests higher pattern quality leads to higher diagnosability. Therefore, diagnostic test-pattern generation becomes a legitimate topic for future research.

References

1. Bartenstein T, Heaberlin D, Huisman L, Sliwinski D (2001) Diagnosing combinational logic designs using single location at-a-time (SLAT) paradigm. In: Proc Int'l Test Conf (ITC), pp 287–296
2. Desineni R, Poku O, Blanton RD (2006) A logic diagnosis methodology for improved localization and extraction of accurate defect behavior. In: Proc International Test Conf (ITC), pp 1–10
3. Di C, Jess JAG (1993) On accurate modeling and efficient simulation of CMOS opens. In: Proc International Test Conf (ITC), pp 875–882
4. Hawkins CF, Soden JM, Righter AW, Ferguson FJ (1994) Defect classes—an overdue paradigm for CMOS IC testing. In: Proc Int'l Test Conf (ITC), pp 413–425
5. Hillebrecht S, Polian I, Engelke P, Becker B, Keim M, Cheng W-T (2008) Extraction, simulation and test generation for interconnect open defects based on enhanced aggressor-victim model. In: Proc Int'l Test Conf (ITC), pp 1–10
6. Huang SY (2002) Diagnosis of byzantine open-segment faults. In: Proc Asian Test Symp (ATS), pp 248–253
7. Huang SY (2003) A symbolic inject-and-evaluate paradigm for byzantine fault diagnosis. J Electron Test Theory Appl 19:161–172
8. ILOG CPLEX Optimizer®, IBM™ (2011) http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/
9. Kao CY, Liao CH, Charles Wen HP (2009) An ILP-based diagnosis framework for multiple open defects. In: Proc int'l Microprocessor Test and Verification workshop (MTV), pp 69–72
10. Konuk H (1997) Fault simulation of interconnect opens in digital cmos circuits. In: Proc IEEE International Conference on Computer-Aided Design (ICCAD), pp 548–554
11. Lin YC, Cheng KT (2006) Multiple-fault diagnosis based on single-fault activation and single-output observation. In: Proc Design, Automation and Test in Europe conf (DATE), pp 1–6
12. Lin YC, Lu F, Cheng KT (2007) Multiple-fault diagnosis based on adaptive diagnostic test pattern generation. In: IEEE Tran Computer Aided Design of integrated circuits and systems (TCAD), vol 26, pp 932–942
13. Lin X, Rajski J (2008) Test generation for interconnect opens. In: Proc Int'l Test Conf (ITC), pp 1–7
14. Liu JB, Veneris A, Takahashi H (2002) Incremental diagnosis of multiple open-interconnects. In: Proc Int'l Test Conf (ITC), pp 1085–1092
15. Mark D, Fan J (2004) Localizing open interconnect defects using targeted routing in FPGA's. In: Proc International Test Conf (ITC), pp 627–634
16. Reddy SM, Pomeranz I, Tang H, Kajihara S, Kinoshita, K (2002) On testing of interconnect open defects in combinational logic circuits with stems of large fanout. In: Proc Int'l Test Conf (ITC), pp 83–89
17. Renovell M, Cambon G (1992) Electrical analysis and modeling of floating-gate fault. In: IEEE Tran Computer Aided Design of integrated circuits and systems (TCAD), vol 11, pp 1450–1458
18. Rodriguez-Montanes R, Arumi D, Figueras J, Einchenberger S, Hora C, Kruseman B, Lousberg M, Majhi AK (2006) Diagnosis of full open defects in interconnecting lines. In: Proc 25th IEEE VLSI Test Symp (VTS), pp 158–166
19. Segura J, Hawkins CF (2004) CMOS electronics: how it works, how it fails. IEEE, Wiley
20. Semiconductor Industry Association, ISA (2011) http://www.sia-online.org/
21. Spinner S, Polian I, Engelke P, Becker B (2008) Automatic test pattern generation for interconnect open defects. In: Proc VLSI Test Symp (VTS), pp 181–186
22. Sue H, Di C, Jess JAG (1994) Probability analysis for CMOS floating gate faults. In: Proc Europe Design Test Conf (EDTC), pp 443–448
23. Tessent FastScan®, Mentor Graphics™ (2011) http://www.com/products/silicon-yield/products/fastscan/mentor.
24. The TAMU Website (2011) http://dropzone.tamu.edu/~xiang/iscas.html
25. Venkataraman S, Drummonds SB (2000) A technique for logic fault diagnosis of interconnect open defects. In: Proc VLSI Test Symp (VTS), pp 313–318
26. Veneris A, Liu JB, Amiri M, Abadir M (2002) Incremental diagnosis and correction of multiple faults and errors. In: Proc Design, Automation and Test in Europe conf (DATE), pp 716–721
27. Wen X, Tamamoto H, Saluja KK, Kinoshita K (2003) Fault diagnosis for physical defects of unknown behaviors. In: Proc Asian Test Symp (ATS), pp 236–241
28. Yu X, Blanton RD (2008) Multiple defect diagnosis using no assumption on failing pattern characteristics. In: Proc Design Automation Conf (DAC), pp 361–366

29. Zou W, Cheng WT, Reddy SM (2006) Interconnect open defect diagnosis with physical information. In: Proc Asian Test Symp (ATS), pp 203–209

Chen-Yuan Kao is a DFT engineer in Global Unichip Ltd, currently working on memory testing. He has received the master degree from National Chiao Tung University and developed diagnosis techniques for open defects. His research interests include memory testing and diagnosis.

Chien-Hui Liao graduated from National Cheng Kung University and majored in Engineering Science from 2004 to 2008. Currently, she is a Ph.D student and majors in Communication Engineering at National Chiao Tung University. Her research interests include task scheduling, test and diagnosis on 3D architectures.

Charles H.-P. Wen a Professor at National Chiao Tung University, Hsinchu, Taiwan, is a specialist in VLSI verification and test with a background in the field of computer science and engineering. He has a PhD in the area of functional test and verification for embedded processors. Over the past ten years, his work has focused on applying data mining and machine learning techniques to SoC design and analysis especially on statistical soft error rates, circuit diagnosability against various advanced defect models and design for testability on 3d integrated circuits (ICs).