An Improved Method of Per-Test *X*-Fault Diagnosis for Deep-Submicron LSI Circuits

Xiaoqing Wen [1], Yuta Yamato [1], Kohei Miyase [2], Seiji Kajihara [1], Hiroshi Furukawa [1]
Laung-Terng Wang [3], Kewal K. Saluja [4], and Kozo Kinoshita [5]

[1] Dept. of CSE, Kyushu Institute of Technology, Iizuka 820-8502, Japan
[2] Innovation Plaza Fukuoka, Japan Science and Technology Agency, Fukuoka 814-0001, Japan
[3] SynTest Technologies, Inc., 505 S. Pastoria Avenue, Suite 101, Sunnyvale, CA 94086, USA
[4] Dept. of ECE, 1415 Engineering Drive, University of Wisconsin - Madison, Madison, WI 53706, USA
[5] Faculty of Informatics, Osaka Gakuin University, Suita 564-8511, Japan

Abstract

Per-test diagnosis based on the X-fault model is an effective approach for a circuit with physical defects of non-deterministic logic behavior. However, the extensive use of vias and the unpredictable order relation among threshold voltages at fanout branches, both being typical phenomena in a deep-submicron circuit, have not been fully addressed by conventional per-test X-fault diagnosis. To solve these problems, this paper proposes an improved per-test X-fault diagnosis method, featuring (1) an extended X-fault model to handle vias and (2) occurrence probabilities of logic behavior for a physical defect to handle the unpredictable relation among threshold voltages. Experimental results show the effectiveness of the proposed method.

1. INTRODUCTION

Fault diagnosis is the most widely used approach to help localize physical defects in a failing LSI circuit [1]. In fault diagnosis, a *fault model* in an abstract circuit model (usually a gate-level netlist) is used to represent the logical behavior of physical defects in an actual LSI circuit. A fault is considered *responsible* if the simulated response of the fault in the circuit model matches the observed response of the failing circuit under certain criteria used in a *fault diagnosis procedure* [2]. The locations of physical defects are then identified with the help of the information on such responsible faults. Clearly, a good fault model and a good diagnosis procedure are needed in order to obtain an acceptable resolution in fault diagnosis.

A good fault model for fault diagnosis needs to closely resemble underlying physical defects from two attributes: *location* and *logical behavior*. In a gate-level circuit model, the first attribute means one or more nets or pins, and the second attribute means one or more logic values. *Fault modeling* defines these attributes in a general manner. On the other hand, physical defects can be characterized from three aspects: *complexity* (simple or complex), *temporality* (static or dynamic), and *cardinality* (single or multiple), as described bellow:

A. Defect Complexity Issue

The defect complexity issue has been addressed by using a set of simple fault models or by using a realistic fault model. For example, [3] uses four fault models to cover various defects. On the other hand, various realistic fault models, such as stuck-open [3], bridging [4, 5], transistor leakage [6], and Byzantine [7, 8], better reflect actual defect mechanisms.

Recently, a new realistic fault model, called the *X-fault* model [9, 10], has been proposed for modeling complex defects, especially those with unpredictable and non-deterministic logic behavior. The *X*-fault model represents all possible faulty logic behavior of a physical defect or defects in a gate and/or on its fanout branches by using different *X* symbols assigned onto the fanout branches. This makes the *X*-fault model highly accurate since no defect information is lost in fault modeling. In addition, partial symbolic fault simulation, instead of full symbolic fault simulation [8], is used in *X*-fault simulation in order to achieve high time efficiency. Since an ever-increasing portion of physical defects in deep-submicron LSI circuits manifest themselves by unpredictable and non-deterministic logic behavior [11], using the *X*-fault model in fault diagnosis is becoming more and more advantageous.

B. Defect Cardinality and Temporality Issues

Per-test fault diagnosis is gaining popularity as an effective approach to handle the cardinality and temporality issues of complex defects [12]. The basic idea is to process failing vectors separately, one at a time, in fault diagnosis, based on the observation that only one of the multiple defects in an LSI circuit may be activated by one failing vector in some cases. This allows a single fault model to be assumed for the activated defect and a relatively easy fault diagnosis procedure based on single fault simulation to be used for multiple and/or dynamic defects.

Several per-test fault diagnosis methods have already been proposed [3, 9, 12-14]. The single stuck-at fault model is used in [12-14]; while a combination of stuck-at, stuck-open, net, and bridging faults is used in [3]. These methods attempt to find a minimal set of faults that explains as many failing vectors as possible. Such a fault set is called a *multiplet* in [12]. In addition, [3] calculates a score for a fault depending on the number of failing vectors explained by the fault, and [12] further extracts diagnostic information from multiplets, while [14] scores each multiplet based on probability functions.

Recently, a per-test fault diagnosis method based on the *X*-fault model has been proposed [9]. On top of the accuracy achieved by *X*-fault modeling, this fault diagnosis procedure employs a flexible matching criterion that takes matching details into consideration. The detailed diagnostic information extracted from the relation between an observed response and a simulated response is expressed as a *diagnosis value* for each *X*-fault and each failing test vector, and all diagnosis values form a *diagnosis table*, from which multiplets are obtained and ordered. It has been shown that such per-test *X*-fault diagnosis can achieve high diagnostic resolution for complex, multiple, and/or dynamic defects.

However, the per-test *X*-fault diagnosis method of [9] faces two serious problems described bellow:

Problem 1: *The existence of vias is not considered in relation to fanout branches.*

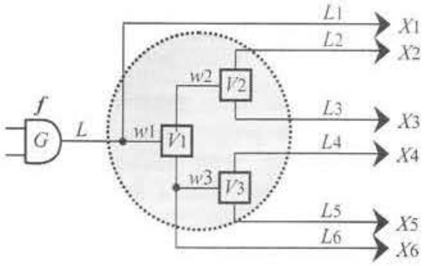


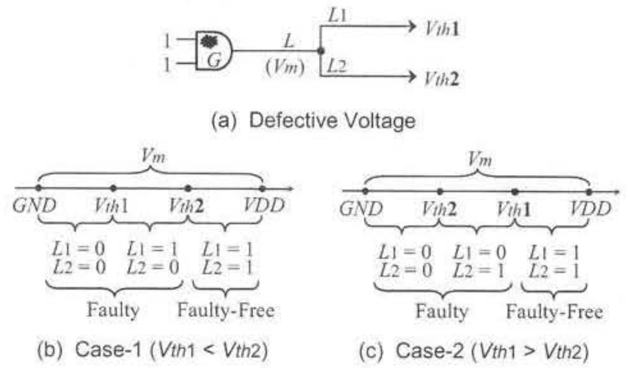
Fig. 1 Vias, Fanout Branches, and *X*-Fault

In the *X*-fault model proposed in [9], the existence of vias is ignored. This is illustrated in Fig. 1, where the gate *G* is considered to have 6 fanout branches, $L_1 \sim L_6$. That is, 6 *X* symbols, $X_1 \sim X_6$, need to be assigned to $L_1 \sim L_6$, respectively. In reality, however, 3 vias, $V_1 \sim V_3$, may exist as shown in Fig. 1. Since *X*-fault diagnosis cannot identify defective fanout branches, treating many lines as fanout branches from the same gate may reduce diagnostic resolution.

Problem 2: *Difference in occurrence probabilities of possible logic behavior of a physical defect is ignored.*

Again, in the conventional *X*-fault diagnosis procedure of [9], all possible logic combinations at the fanout branches of a gate are considered to be equally likely. In reality, however, this is usually not true. This is illustrated in Fig. 2, where the gate *G* has a defective voltage V_m, which may be intermediate. In Fig. 2 (a), the gate *G* has two fanout branches, L_1 and L_2, corresponding to threshold voltages V_{th1} and V_{th2}, respectively. In Case-1 ($V_{th1} < V_{th2}$), <00>, <10>, and <11> may occur at L_1 and L_2. However, in Case-2 ($V_{th1} > V_{th2}$), <00>, <01>, and <11> may occur at L_1 and L_2. Since in a deep-submicron LSI circuit, especially with low-voltage design, the order relation between V_{th1} and V_{th2} may be unpredictable due to process variation. That is, both Case-1 and Case-2 may show up. Under this condition, it is clear that the occurrence probabilities of <00>, <10>, <01>, and <11> are 2/6, 1/6, 1/6, and 2/6, respectively, which are clearly not equal. Therefore, treating the occurrence probabilities of all logic combinations at the fanout branches of a gate as equal in an *X*-fault diagnosis procedure may either reduce diagnostic resolution or produce misleading diagnostic results.

Therefore, there is a need to solve Problem 1 and Problem 2 in order to improve diagnostic resolution and diagnosis efficiency, and this is the focus of this paper.



(a) Defective Voltage

(b) Case-1 ($V_{th1} < V_{th2}$)

(c) Case-2 ($V_{th1} > V_{th2}$)

Fig. 2 Defective Voltage and Possible Logic Combinations

2. Previous Per-Test X-Fault Diagnosis Method

2.1 *X*-Fault Model

The conventional *X*-fault model [9] is defined as follows:

Definition 1: A fanout gate has one *X-fault*, corresponding to any physical defect or defects in the gate or on its *n* fanout branches. The *X*-fault assumes *n* different *X* symbols on the *n* fanout branches to represent all possible faulty logic values in fault simulation.

Fig. 3 shows the *X*-fault for an AND gate with two fanout branches, where X_1 and X_2 denote two arbitrary faulty logic values. Clearly, <X_1, X_2> represents any possible faulty logic combination that may appear on the fanout branches. Note that the conventional *X*-fault model treats all fanout branches as directly connected signal lines, without considering that vias may exist at fanout branches.

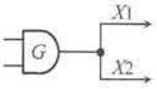
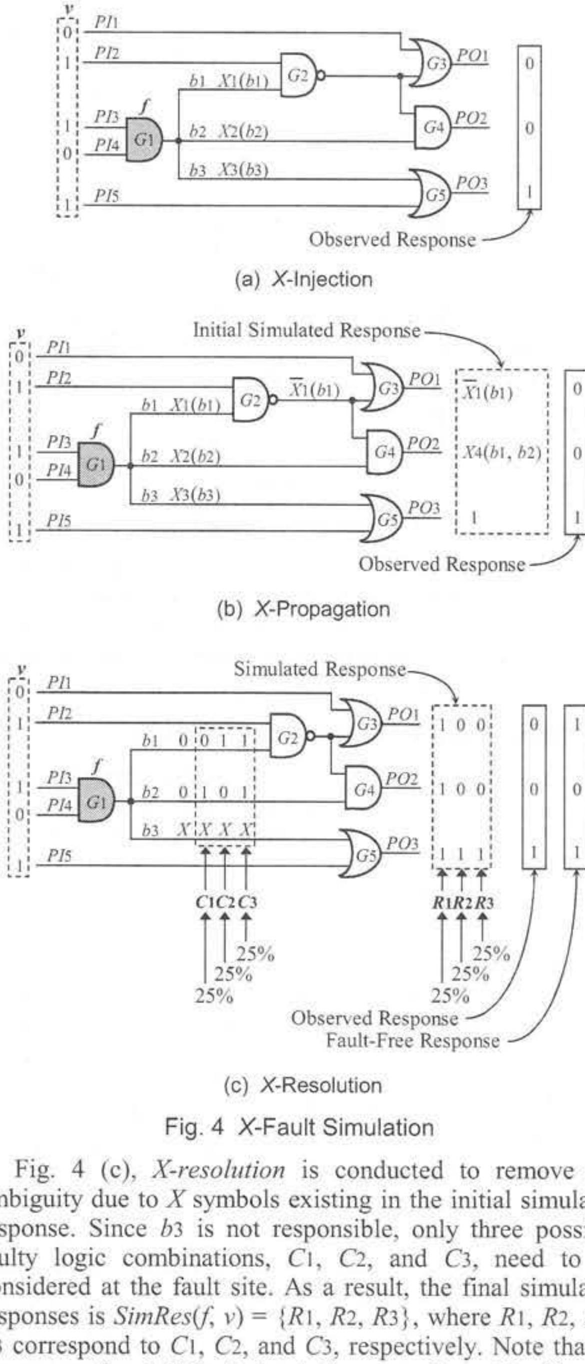


Fig. 3 Conventional *X*-Fault Model

2.2 *X*-Fault Simulation

Given an *X*-fault *f* and an input vector *v*, *X*-fault simulation is to obtain the simulated response of *f* under *v*, denoted by $SimRes(f, v) = \{R_1, R_2, ..., R_k\}$, where $R_1, R_2, ..., R_k$ ($k \geq 1$) are logic combinations at primary outputs, corresponding to *k* possible faulty logic combinations, $C_1, C_2, ..., C_k$, at the site of *f*, respectively. Generally, *X*-fault simulation uses a partial-symbolic procedure, consisting of three steps: (1) ***X-injection*** for assigning different *X* symbols to the fanout branches of a gate, (2) ***X-propagation*** for propagating *X* symbols to primary outputs, and (3) ***X-resolution*** for resolving all *X* symbols at primary outputs to obtain a final simulation result [9]. An example is shown in Fig. 4.

In Fig. 4 (a), *X-injection* assigns 3 initial *X* symbols, $X_1(b_1)$, $X_2(b_2)$, and $X_3(b_3)$, to the fanout branches, b_1, b_2, and b_3 of the gate G_1, respectively. In Fig. 4 (b), *X-propagation* is conducted by keeping inversion function but ignoring all other logic functions. For example, the output of the gate G_4 is the AND function of $\overline{X_1}(b_1)$ and $X_2(b_2)$. This functional information is ignored and the result is a new *X* symbol $X_4(b_1, b_2)$, in which (b_1, b_2) is used to indicate that the output of G_4 only comes from branches b_1 and b_2.



(a) *X*-Injection

(b) *X*-Propagation

(c) *X*-Resolution

Fig. 4 *X*-Fault Simulation

In Fig. 4 (c), *X-resolution* is conducted to remove the ambiguity due to *X* symbols existing in the initial simulated response. Since *b*3 is not responsible, only three possible faulty logic combinations, *C*1, *C*2, and *C*3, need to be considered at the fault site. As a result, the final simulated responses is *SimRes*(*f*, *v*) = {*R*1, *R*2, *R*3}, where *R*1, *R*2, and *R*3 correspond to *C*1, *C*2, and *C*3, respectively. Note that in the conventional *X*-fault simulation procedure, *C*1, *C*2, and *C*3 are assumed to be equally likely, i.e. each of them having an occurrence probability of 25%. In reality, this assumption is usually not true. This problem will be addressed in this paper by calculating the occurrence probability of each possible faulty logic combination at fanout branches, and by using this probabilistic information in *X*-fault diagnosis.

2.3 Diagnosis Value Calculation

After conducting *X*-fault simulation for an *X*-fault *f* under a failing vector *v*, the simulated response *SimRes*(*f*, *v*) = {*R*1, *R*2, ..., *Rk*} needs to be compared with the observed response *ObvRes*(*v*) to extract diagnostic information. The comparison result is represented by a so-called *diagnosis value* under *v* and *f*, denoted by *d*(*f*, *v*), and the method for calculating *d*(*f*, *v*) is as follows [9]:

$$d(f, v, Ri) = \frac{Level(f)}{L_{max}} \times \frac{|\,Error\_PO(v) \cap Reach\_PO(f)\,|}{|\,Reach\_PO(f)\,|}$$

if *Ri* is the same as *ObvRes*(*v*) on *Reach_PO*(*f*); otherwise, *d*(*f*, *v*, *Ri*) = 0. And,

$$d(f, v) = (\sum_{i=1}^{k} d(f, v, Ri)\,)/k$$

Here, *Error_PO*(*v*) is the set of all primary outputs on which an observed response has errors, and *Reach_PO*(*f*) is the set of primary outputs that is reachable from the gate with the *X*-fault *f*. In Fig. 4 (c), *Error_PO*(*v*) = {*PO*1} and *Reach_PO*(*f*) = {*PO*1, *PO*2, *PO*3}. Moreover, *Level*(*f*) is the level of the output of the gate with the *X*-fault *f*, and *Lmax* is the maximum level in the circuit, assuming that all primary outputs have level 1. In Fig. 4, *Level*(*f*) = 3 and *Lmax* = 3.

For the *X*-fault simulation result shown in Fig. 4 (c), *SimRes*(*f*, *v*) = {*R*1, *R*2, *R*3} = {<111>, <001>, <001>}, *ObvRes*(*v*) = <001>, and the fault-free simulation result is <101>. Clearly, *d*(*f*, *v*, *R*1) = 0 and *d*(*f*, *v*, *R*2) = *d*(*f*, *v*, *R*3) = (3/3) × (1/3) = 0.33. Therefore, *d*(*f*, *v*) = (0 + 0.33 + 0.33) / 3 = 0.22.

Table 1 Fault Diagnosis Table

	*f*1	*f*2	*f*3	*f*4	*f*5
*v*1	0.81	0.65	0	0	0
*v*2	0	0	0.61	0.17	0
*v*3	0.26	0	0.83	0	0
*v*4	0	0	0	0	0.55
ave	0.27	0.16	0.36	0.04	0.14

Diagnosis values are calculated for all failing vectors and faults, and they are stored in a table called a *fault diagnosis table*, as illustrated in Table 1. Clearly, compared with a normal fault dictionary with only 0 and 1 entries, a fault diagnosis table contains more diagnostic information.

It is obvious that diagnosis values are calculated with unique matching criteria, which take the reachable range of primary outputs, the number of matched errors, and the depth of a fault into consideration [9]. Note that, this calculation method follows the assumption that all possible faulty logic combinations at the fanout branches of a gate are equally likely, which may not be true in reality.

2.4 Per-Test *X*-Fault Diagnosis Flow

Fig. 5 shows the per-test *X*-fault diagnosis flow [9], which consists of two stages, one for collecting diagnostic information and the other for drawing diagnostic conclusion.

In Stage-1 (*Information-Collecting*), all *X*-faults are simulated for each failing vector, the simulated responses are compared with observed responses, and a diagnosis table is created. In Stage-2 (*Diagnostic-Reasoning*), a diagnosis result is produced from the fault diagnosis table. The basic processing is to find a minimal set of *X*-faults that cover all failing vectors corresponding to non-all-zero rows in a fault diagnosis table. Such a fault set is called a *multiplet*. Then, the score of each multiplet is calculated by adding up the diagnosis values of all composing *X*-faults. These scores are then used to determine the order of multiplets, and the *X*-faults in top multiplets form the final fault diagnosis result.

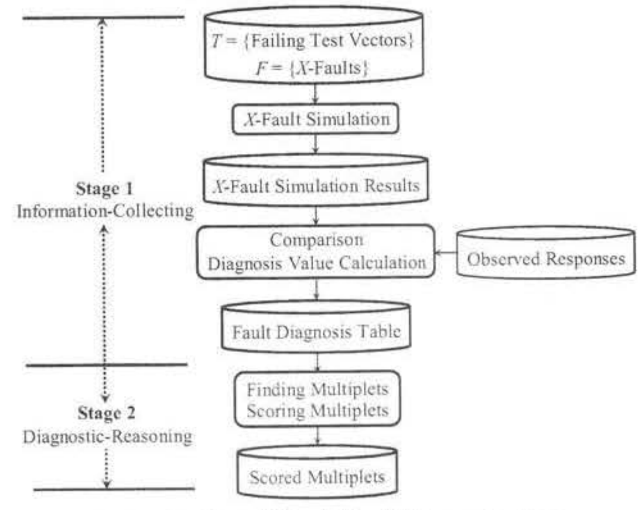


Fig. 5 Per-Test X-Fault Fault Diagnosis Flow

3. IMPROVED PER-TEST *X*-FAULT DIAGNOSIS

As described above, there are two major problems with the conventional per-test *X*-fault diagnosis: (1) treating all fanout branches of a gate as directly connected signal lines without considering the existence of vias, and (2) assuming the all possible logic combinations at the fanout branches of a gate have equal occurrence probabilities. The first problem is addressed in Section 3.1 with an extended *X*-fault model, and for the second problem, a method for calculating the occurrence probability of each possible logic combination is described in Section 3.2. Section 3.3 defines a new diagnosis value based on occurrence probability.

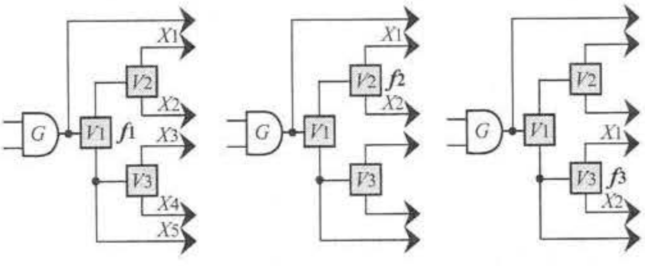
3.1 Extended *X*-Fault Model

The *X*-fault model [9] assumes one *X*-fault for each gate that may have any number of fanout branches. This is illustrated in Fig. 1, where all fanout branches, $L_1 \sim L_6$, from the gate G are treated without any distinction from each other. Note that, when the *X*-fault f is assumed at G, as shown in Fig. 1, 6 X symbols, $X_1 \sim X_6$, are assigned to the branches. The disadvantage of this *X*-fault model is that it can only lead to the finding that a fault and its fanout branches may be defective, without sufficient information to identify which fanout branch is more likely to be defective.

In reality, the layout of a deep-submicron LSI circuit usually involves multiple layers, which means that vias are extensively used. In order to handle such via information, the *X*-fault model [9] is extended as follows:

Definition 2: A fanout element (gate or via) has one *X-fault*, corresponding to any physical defect or defects in the element or on its fanout branches. The *X*-fault assumes different X symbols on the fanout branches of the element to represent non-deterministic faulty logic values in fault simulation.

Examples of the extended *X*-fault model are shown in Fig. 6 (a) ~ (c). Since there are three vias $V_1 \sim V_3$, 3 extended *X*-faults, $f_1 \sim f_3$, are added, in addition to the conventional *X*-fault f shown in Fig. 1. 6, 2, and 2 different X symbols are assumed for the three extended *X*-faults, f_1, f_2, and f_3, respectively. Obviously, these new added *X*-faults are all different from the conventional *X*-fault f shown in Fig. 1.



(a) X-Fault f1 (b) X-Fault f2 (c) X-Fault f3

Fig. 6 Extended *X*-Fault Model

The most significant advantage of the extended *X*-fault model is that it can locate defects to the via level, which greatly improves the diagnostic resolution. In the case of Fig. 6, diagnostic results now can be obtained with respect to V_1, V_2, and V_3, instead of only G.

3.2 Occurrence Probability Calculation

Suppose that a gate or via has n fanout branches, L_1, L_2, ..., and L_n, whose corresponding threshold voltages are V_{th1}, V_{th2}, ..., and V_{thn}, respectively. If the order of V_{th1}, V_{th2}, ..., and V_{thn} is fixed and known, there will be exactly n+1 possible logic combinations at the fanout branches [15].

In a real LSI circuit, however, process variation in the deep sub-micron era and shrinking difference among threshold voltages in low-voltage design increasingly make it difficult to deterministically know the order of threshold voltages corresponding to the fanout branches of a gate or via. In order words, it is necessary to consider all possible orders of threshold voltages at the fanout branches. As illustrated in Fig. 2, this results in different occurrence probabilities of logic combinations at the fanout branches of a gate or via. This phenomenon can be quantitatively expressed by the following theorem.

Theorem 1: For a gate or via with n fanout branches, the total number of possible orders of threshold voltages at the fanout branches is $n!$. In addition, the probability that the fanout branches have a logic combination with p 0's ($0 \leq p \leq n$) is $(p! \times (n-p)!) / (n+1)!$

Proof: Consider the general case shown in Fig. 7. Here, the n fanout branches, $L_1 \sim L_n$, whose corresponding threshold voltages are $V_{th1} \sim V_{thn}$, respectively. In addition, it is assumed that the stem L has a non-deterministic voltage V_m. Depending on the relations of L with $V_{th1} \sim V_{thn}$, different logic combinations may appear on $L_1 \sim L_n$ [15].

1st Half: When n threshold voltages are ordered, there are n choices for the first threshold voltage, (n-1) choices for the second threshold voltage, ..., and 1 choice for the n-th threshold voltage. As a result, there are a total of $n!$ possible orders of n threshold voltages.

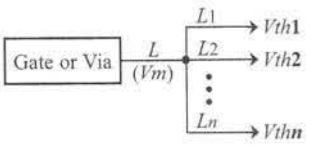


Fig. 7 Fanout Branches

2nd Half: First, for each order of n threshold voltages, there are $(n+1)$ possible logic combinations [15], depending on which of the $(n+1)$ voltage intervals in the order the corresponding intermediate voltage V_m falls in. That is, the total number of occurrences of possible logic combinations is $(n+1) \times n! = (n+1)!$ In addition, a logic combination having p 0's ($0 \leq p \leq n$) means that p threshold voltages are lower than V_m and $(n\text{-}p)$ threshold voltages are higher than V_m, where V_m is the corresponding intermediate voltage. As a result, there are $p! \times (n\text{-}p)!$ logic combination with p 0's. Therefore, the occurrence probability of such a logic combination is $(p! \times (n\text{-}p)!) / (n+1)!$ □

The occurrence probabilities of logic combinations at n fanout branches are summarized in Table 2 (a), and the special case of $n = 3$ is shown in Table 2 (b). Clearly, logic combinations with different numbers of 0's may have different occurrence probabilities.

Table 2 Occurrence Probabilities of Logic Combinations

(a) Case of *n* Fanout Branches

Number of 0's in logic combination	Number of Occurrences	Probability of Occurrence
0	$n!$	$1 / (n+1)$
1	$1! \times (n\text{-}1)!$	$1 / (\mathrm{n} \times (n+1))$
2	$2! \times (n\text{-}2)!$	$2 / ((n\text{-}1) \times n \times (n+1))$
⋮	⋮	⋮
p	$p! \times (n\text{-}p)!$	$(p! \times (n\text{-}p)!) / (n+1)!$
⋮	⋮	⋮
n	$n!$	$1 / (n+1)$

(b) Case of 3 Fanout Branches

Number of 0's in logic combination	Number of Occurrences	Probability of Occurrence
0	6	25%
1	2	≈ 8%
2	2	≈ 8%
3	6	25%

3.3 Use of Occurrence Probabilities

3.3.1 New X-Resolution

Based on Theorem 1, one can determine the occurrence probabilities for all possible faulty logic combinations at the fanout branches of a gate or via. This information is used in *X*-resolution during *X*-fault simulation, in order to better reflect the reality in deep-submicron LSI circuits.

Consider the *X*-resolution example shown in Fig. 8 for simulating the *X*-fault at the gate G_1. This gate has 3 fanout branches, b_1, b_2, and b_3, and there are 3 possible faulty logic combinations, C_1 = <01*X*>, C_2 = <10*X*>, and C_3 = <11*X*>. Since C_1 represents <010> and <011> that both have the occurrence probability of 8%, the occurrence probability of C_1 is 16%. Similarly, the occurrence probabilities of C_2 and C_3 can be obtained as 16% and 32%, respectively. As a result, the occurrence probabilities of C_1, C_2, and C_3 are 16%, 16%, and 32%, respectively, which are different from the conventional assumption that all of them have the same occurrence probability of 25% as shown in Fig. 4 (c).

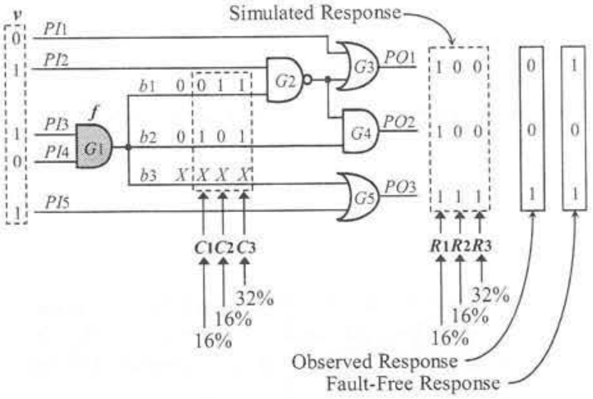


Fig. 8 *X*-Fault Simulation

Suppose that the possible logic combinations at the fanout branches for *X*-fault *f* under test vector *v* are $C_1 \sim C_n$, whose occurrence probabilities are $p(C_1)$, $p(C_2)$ ..., and $p(C_n)$, respectively. Note that $p(C_1)$, $p(C_2)$ ..., and $p(C_n)$ can be readily calculated by using Theorem 1. Also suppose that the simulated response $SimRes(f, v) = \{R_1, R_2, ..., R_n\}$, where R_1, R_2, ..., and R_n are resulting logic combinations at primary outputs, corresponding to $C_1 \sim C_n$, respectively. Clearly, the occurrence probabilities of R_1, R_2, ..., and R_n, denoted by $p(R_1)$, $p(R_2)$ ..., and $p(R_n)$, respectively, are equal to $p(C_1)$, $p(C_2)$ ..., and $p(C_n)$, respectively.

For example, in Fig. 8, since the occurrence probabilities of logic combinations at fanout branches, C_1, C_2, and C_3, are 16%, 16%, and 32%, respectively, the occurrence probabilities of logic combinations at primary outputs, R_1, R_2, and R_3 are also 16%, 16%, and 32%, respectively.

3.3.2 New Diagnosis Value Calculation

In the conventional definition of diagnosis value [9] described in 2.3, it is assumed that all possible faulty logic combinations at the fanout branches of a gate are equally likely, which may not be true in reality. In the following, a new definition of diagnosis value is presented to take the difference in occurrence probabilities into consideration.

Generally, the simulated response $SimRes(f, v) = \{R_1, R_2, ..., R_k\}$ is compared with the observed response $ObvRes(v)$ to extract diagnostic information, and the comparison result is represented by a *diagnosis value* under *v* and *f*, denoted by $d(f, v)$, and the new method to calculate $d(f, v)$ is as follows:

$$d(f, v, Ri) = \frac{Level(f)}{L_{max}} \times \frac{|\,Error\_PO(v) \cap Reach\_PO(f)\,|}{|\,Reach\_PO(f)\,|}$$

if Ri is the same as $ObvRes(v)$ on $Reach\_PO(f)$; otherwise, $d(f, v, Ri) = 0$. And,

$$d(f, v) = \sum_{i=1}^{k} (d(f, v, Ri) \times p(Ri))$$

Here, *Error_PO*(v), *Reach_PO*(f), *Level*(f), and *Lmax* are all as defined in 2.3.

In Fig. 8, for example, $SimRes(f, v) = \{R_1, R_2, R_3\} = \{<111>, <001>, <001>\}$, $ObvRes(v) = <001>$, and the fault-free simulation result is <101>. Thus, $d(f, v, R_1) = 0$, $d(f, v, R_2) = d(f, v, R_3) = (3/3) \times (1/3) = 0.33$. Therefore, $d(f, v) = 0 \times 16\% + 0.33 \times 16\% + 0.33 \times 25\% = 0.14$. Clearly, this diagnosis value is different from the diagnosis value calculated in Section 2.3 for the example of Fig. 4 (c).

3.4 Improved Per-Test *X*-Fault Diagnosis Flow

The general flow of the improved per-test *X*-fault diagnosis is basically the same as shown in Fig. 5. The differences are as follows:

- The extended *X*-fault model described in Section 3.1 is used. Since via information is utilized, it becomes possible to locate defects to the via level, greatly improving the diagnostic resolution. In addition, since the number of fanout branches from a gate or a via becomes smaller, the time for *X*-resolution also becomes shorter.
- The new diagnosis value described in Section 3.3 is used. Since the occurrence probabilities of possible faulty logic combinations are taken into consideration, the reality in a deep-submicron LSI circuit is better reflected, which contributes to the improvement of diagnostic resolution.

4. EXPERIMENTAL RESULTS

Table 3 summarizes the experimental results. The number of input vectors for each circuit [16] is shown under "Vector". In each experiment, a via-open defect was randomly inserted at a fanout gate to imitate a defective chip, and the defect was assumed to cause faulty effects at multiple branches of the gate. 10 experiments were conducted for each circuit, and the average number of faulty fanout branches is 2.0.

Table 3 Fault Diagnosis Results

Circuit	Vector (#)				FH(#)			New-B
		SLAT (#)	MPLT (#)	Exact (%)	Old	New-A	New-B	Time (Sec.)
C432	28	2.6	9.0	11.1	2.8	2.5	2.7	0.01
C499	52	4.1	35.4	2.8	27.3	2.2	1.0	0.01
C880	21	4.7	4.3	23.3	1.2	1.4	1.0	0.01
C1355	84	8.0	40.1	2.5	9.0	1.2	1.0	0.15
C1908	106	2.3	43.8	2.3	16.7	6.5	7.0	0.08
C2670	45	21.4	5.4	18.5	1.3	3.6	1.9	0.34
C3540	93	5.6	14.0	7.1	1.1	5.6	6.1	0.37
C5315	46	9.9	19.2	4.7	1.5	4.9	5.4	0.60
C6288	14	7.2	11.3	8.8	2.5	5.6	4.0	2.99
C7552	75	20.2	48.2	2.1	4.8	5.4	3.9	2.45
——					**6.8**	**3.7**	**3.4**	—

For each sample, three per-test fault diagnosis programs were run: "OLD" uses the conventional *X*-fault model [9], "New-A" uses the extended *X*-fault model but the conventional method for diagnosis value calculation [9], and "New-B" uses the extended *X*-fault model and the proposed method for diagnosis value calculation. "SLAT" shows the average number of SLAT vectors for 10 samples. "MPLT" shows the average number of multiplets; and "Exact" shows the average percentage of exact diagnosis, i.e. a multiplet containing all inserted defects, both for 10 samples. "FH" (First Hit) shows the average position of the first exact-diagnosis-producing multiplet in a multiplet list. "Time" shows the average run time of "New-B" (CPU: 2.6GHz).

From the average FH results, it is clear that the use of extended *X*-fault model is indeed effective. Furthermore, the use of occurrence probability in diagnosis value calculation can further improve FH.

5. CONCLUSIONS

This paper proposed a new per-test fault diagnosis method based on (1) the use of the extended *X*-fault model for handling vias and (2) the calculation of occurrence probabilities of possible faulty logic combinations at the fanout branches of a gate or a via. As demonstrated by experimental results, the improved per-test *X*-fault diagnosis method can achieve better diagnostic resolution. More experiments on larger circuits are under way to further evaluate and improve the proposed method.

REFERENCES

[1] L.-T. Wang, C.-W. Wu, and X. Wen, (Editors), *VLSI Test Principles and Architectures: Design for Testability*, Elsevier, 2006.

[2] H. Takahashi, K. O. Boateng, K. K. Saluja, and Y. Takamatsu, "On Diagnosing Multiple Stuck-At Faults Using Multiple and Single Fault Simulation", *IEEE Trans. on Computer-Aided Design*, vol. 21, no. 5, pp. 362-368, 2002.

[3] S. Venkataraman and S. Drummonds, "POIROT: A Logic Fault Diagnosis Tool and Its Applications", *Proc. Intl. Test Conf.*, pp. 253-262, 2000.

[4] S. D. Millman, E. J. McCluskey, and J. M. Acken, "Diagnosing CMOS Bridging Faults with Stuck-At Fault Dictionaries", *Proc. Intl. Test Conf.*, pp. 860-870, 1990.

[5] P. Maxwell and R. Aiken, "Biased Voting: A Method for Simulating CMOS Bridging Faults in the Presence of Variable Gate Logic Thresholds", *Proc. Intl. Test Conf.*, pp. 63-72, 1993.

[6] W. Mao and R. K. Gulati, "QUIETEST: A Quiescent Current Testing Methodology for Detecting Short Faults", *Proc. ICCAD'90*, pp. 280-283, 1990.

[7] D. Lavo, T. Larrabee, and B. Chess, "Beyond the Byzantine Generals: Unexpected Behavior and Bridging Fault Diagnosis", *Proc. Intl. Test Conf.*, pp. 611-619, 1996.

[8] S. Huang, "Speeding Up the Byzantine Fault Diagnosis Using Symbolic Simulation", *Proc. VLSI Test Symp.*, pp. 193-198, 2002.

[9] X. Wen, T. Miyoshi, S. Kajiihara, L.-T. Wang, K. K. Saluja, and K. Kinoshita, "On Per-Test Fault Diagnosis Using the X-Fault Model", *Proc. Int'l Conf. on Computer-Aided Design*, pp. 633-640, 2004.

[10] X. Wen, H. Tamamoto, K. K. Saluja, and K. Kinoshita, "Fault Diagnosis for Physical Defects of Unknown Behaviors", *Proc. Asian Test Symp.*, pp. 236-241, 2003.

[11] *ITRS Roadmap: Test & Test Equipment*, 2004 Edition, http://public.itrs.net/, 2004.

[12] T. Bartenstein, D. Heaberlin, L. Huisman, and D. Sliwinski, "Diagnosing Combinational Logic Designs Using the Single Location At-a-Time (SLAT) Paradigm", *Proc. Intl. Test Conf.*, pp. 287-296, 2001.

[13] J. Waicukauski and E. Lindbloom, "Failure Diagnosis of Structured Circuits", *IEEE Design and Test of Comp.*, vol. 6, no. 4, pp. 49-60, 1989.

[14] D. Lavo, I. Hartanto, and T. Larrabee, "Multiplets, Models and the Search for Meaning", *Proc. Intl. Test Conf.*, pp. 250-259, 2002.

[15] X. Wen, H. Tamamoto, K. K. Saluja, and K. Kinoshita, "Fault Diagnosis for Static CMOS Circuits", *Proc. Asian Test Symp.*, pp. 282-287, 1997.

[16] F. Brglez and H. Fujiwara, "A Neutral Netlist of Combinational Benchmark Circuits and a Target Translator in FORTRAN", *Proc. Intl. Symp. on Circuits and Systems*, pp. 663-698, 1985.