2009 27th IEEE VLSI Test Symposium

Multiple-Fault Diagnosis Using Faulty-Region Identification

Meng-Jai Tasi, Mango C.-T. Chao, Jing-Yang Jou, Meng-Chen Wu

Dept. of Electronics Engineering, National Chiao-Tung University, Hsinchu, Taiwan
{mjtasi@eda.ee.nctu.edu.tw, mango@faculty.nctu.edu.tw,
jyjou@faculty.nctu.edu.tw, mcwu@eda.ee.nctu.edu.tw}

***Abstract*—The fault diagnosis has become an increasing portion of today's IC-design cycle and significantly determines product's time-to-market. However, the failure behaviors from the defective chips may not be fully represented by the single fault model. In this paper, we propose a fault-diagnosis framework targeting multiple stuck-at faults. This framework first reports a minimal suspect region, in which all real faults are topologically covered. Next, a proposed ranking method is applied to sieve out the real faults from the candidates within the suspect region. The experimental results show that the proposed diagnosis framework can effectively locate the multiple stuck-at faults within a neighborhood, which may generate erroneous signals cancelling one another and are difficult to be diagnosed based on a single-fault-model method.**

I. INTRODUCTION

Due to the high complexity, high variation, and incomplete characterization of advanced process technologies, the first couple silicons of today's ICs usually fail or suffer an unacceptable low yield. Based on the collected responses of failed chips, the *failure analysis* is applied to identify the root causes of the failure, which are then used to correct the design, improve the process, or enhance the yield for the next silicon. However, the nonstop increase of today's design complexity and process complexity has made the failure analysis more and more difficult and time-consuming. Therefore, the time, cost, and effectiveness of the failure analysis significantly affects IC's time-to-market and total cost [1].

In the failure analysis, a process called *fault diagnosis* is used to find out the possible candidates of real defects represented by a given fault model. Based on the reported fault candidates, the designers could infer the locations of real defects and then physically examine those possibly defective locations through the focused-ion-beam (FIB) or decapsulation techniques [2]. Those physical-examination techniques are still slow and expensive in current industry. Thus, a robust fault-diagnosis framework with high accuracy and efficiency is highly desired to speedup and cost down the entire process of the failure analysis.

The simplest method for fault diagnosis is to build a *fault dictionary*, in which the faulty response generated by each fault is stored. Then we try to match the collected responses of the failed chips with the faulty responses stored in the fault dictionary. The size of the fault dictionary is directly proportional to the number of modeled faults in a design, which can easily reach to millions today. Most recent researches in the dictionary-based diagnosis focus on the compression techniques reducing the size of the fault dictionary [3]–[7]. Thus, the application of this dictionary-based method is limited to a single-fault model. The dictionary size for a multiple-fault model would be prohibitively huge [8]. However, one defect in a chip may affect multiple transistors and multiple defects may simultaneously exist in a chip. The behavior of failed chips may not be fully represented by a single-fault model [4][5][7].

Several methods for multiple-fault diagnosis have been published in the past. One type of multiple-fault-diagnosis methods are region-based methods [9] [10] [11], in which all real defects are assumed to locate within a neighborhood. The region-based methods utilize the X (unknown) model to approximately represent the possible faulty behaviors of a defective region in the simulation and then rank the defective regions based on how their X-simulation results can match the failing responses. However, these region-based methods assume that the size of defective region is fixed, but this size may not be known before the diagnosis. In addition, their computation complexity increases exponentially with the increase of region's radius. The experimental results reported in [9] [10] [11] are limited to a small radius (such as 1).

[12] proposes the concept of SLAT (single-location-at-a-time) patterns, which can generate a faulty response fully explained by a single stuck-at fault. With the SLAT patterns, individual stuck-at faults can be identified and then used to derive the possible combinations of the real multiple faults (also called as *multiplets*). [13], [14] and [15] utilize the concept of SLAT patterns to diagnose complex faults, such as Byzantine faults or bridging faults. [8] can further handle some non-SLAT patterns by dividing the faulty outputs into independent fanout regions. However, all above SLAT-based methods depend on the existence of enough SLAT patterns, which may not be always true in reality. Moreover, it is also difficult for SLAT-based methods to diagnose defects generating erroneous signals masking or interacting one another [8][12][13]. In this case, the number of reported multiplets would be large [12].

In order to reduce the number of possible multiplets or fault candidates, additional test patterns are applied to increase the diagnosis resolution [14][16]. [16] prunes the false candidates by generating the SO-SLAT (single-observation SLAT) patterns, which detect the target fault at a single observation point and guarantee that the faulty outputs of the target fault cannot be masked by other faults in the candidate list. In reality, the diagnosis patterns are usually the same as the test patterns (mostly stuck-at-fault patterns). Adding additional test patterns may not be always possible in current diagnosis flows.

In this paper, we propose a diagnosis framework targeting multiple stuck-at faults. This framework consists of two main components, the *faulty-region identification* followed by the *fault-candidate ranking*. In the faulty-region identification, we utilize the X-simulation technique and the bit-flipping technique to gradually shrink the suspect region covering all real faults based on both failing and passing patterns. Unlike the region-based methods, our framework requires no assumption for the radius of the suspect region and hence is more flexible. In the fault-candidate ranking, we classify and rank the fault candidates based on the information collected from the faulty-region identification. The experimental results show that the proposed diagnosis framework can efficiently and effectively minimize the suspect region and sieve out the real faults from the region even when the failing-pattern percentage and the number of SLAT patterns are both low for the circuits under diagnosis (CUDs). Those are the difficult cases to be diagnosed by the traditional SLAT-based and region-based methods.

II. Proposed Fault-Diagnosis Framework

A. Overall Diagnosis Flow

Figure 1 shows the overall flow of the proposed fault-diagnosis framework. Given the CUD, the test patterns, and the corresponding test responses obtained from the CUD, we apply the procedure, *faulty-region identification*, to report a minimal faulty region which covers all real faults. Then, the procedure, *fault-candidate ranking*, ranks the fault candidates within the faulty region based on the information collected during faulty-region identification. Next, we use the FIB technique to physically repair the fault candidate with the highest rank and then apply the same test patterns to the CUD. If the responses obtained from the repaired CUD is different from the test responses obtained before the FIB repair, the repaired fault candidate is a real fault. Then we rerun the faulty-region identification and fault-candidate ranking based on the new obtained responses. Otherwise, the repaired fault candidate is not a real fault. Then we use the FIB technique to physically repair the next ranked fault candidate until a real fault is detected.

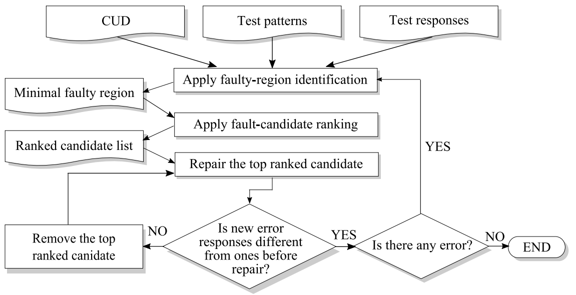
CUD
Test patterns
Test responses
Apply faulty-region identification
Minimal faulty region
Apply fault-candidate ranking
Ranked candidate list
Repair the top ranked candidate
YES
Remove the top ranked canidate
NO
Is new error responses different from ones before repair?
YES
Is there any error?
NO
END

Fig. 1. **Overall flow of the proposed diagnosis framework.**

B. Key Concepts

In this subsection, we describe three key concepts used in our diagnosis framework: *X-region*, *not-stuck-at-0* (or *not-stuck-at-1*) signals, and *value-flipping* technique.

1) X-region: An *X-region* is a set of connected gates, whose logic values are set to unknown. When simulating failing patterns along with the unknown values in the X-region, we can determine whether the X-region covers all the real faults by comparing the simulated value on each failing output with the collected erroneous response. If the simulated value on any failing output is opposite to the value of its erroneous response, it means that the value on the failing output cannot be corrected no matter how the unknown values in the X-region are assigned. In such a case, this failing output must result from a fault outside the X-region and hence such an X-region does not cover all the real faults.

In our diagnosis framework, we use an X-region to represent a possible suspect region which covers all real faults. Figure 2 shows a simple circuit with stuck-at-1 faults at $net18$ and $G1$. We use this faulty circuit as an example throughout the entire paper. Column 1 to Column 4 at Table I list the test patterns, the responses obtained from the good-circuit simulation, the responses obtained from the faulty-circuit simulation (or from the actual defective circuit), and the corresponding failing outputs, respectively, for the exemplary circuit. If we give an X-region covering the two stuck-at faults as shown by the shadow background in Figure 2, the simulated value of all the failing outputs is set to unknown for each failing pattern. The simulation results associated with the given X-region are listed in Column 5 at Table I.

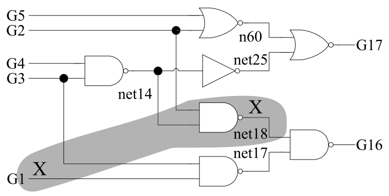

Fig. 2. **An example of X-region.**

Test Pats.	Sim. Resp.	Test Resp.	Failing PO	X-Sim. Results
G1 G2 G3 G4 G5	G16 G17	G16 G17		G16 G17
11010	11	01	G16	X1
11001	11	01	G16	X1
11011	11	01	G16	X1
10001	01	01	–	X1
01111	00	10	G16	X0
00010	00	00	–	X0
10101	11	11	–	X1
00110	00	10	G16	X0

TABLE I
Test patterns, simulated good-circuit responses, and collected responses for the exemplary circuit.

2) Not-stuck-at-0 & not-stuck-at-1: Unlike other dictionary-based or region-based diagnosis methods, which attempt to identify a modeled fault or region matching the actual faulty syndrome, our diagnosis framework attempts to identify the signals which cannot be a fault, i.e., to identify the *not-stuck-at-0* or *not-stuck-at-1* signals. If a candidate signal can be proved that its value on the defective chip is 0 (1) for any pattern, the candidate signal is then proved as a not-stuck-at-1 (not-stuck-at-0) signal. A candidate signal is guaranteed to be fault-free if this signal is proved to be both not-stuck-at-0 and not-stuck-at-1 [17][18]. For convenience, we use *NSA0* and *NSA1* to represent *not-stuck-at-0* and *not-stuck-at-1*, respectively. We also use *NSAv* to represent a signal which can be either not-stuck-at-0 or not-stuck-at-1.

3) Value flipping on X-region's boundary signals: For each test pattern, we check whether a target signal on the X-region's boundary is NSA0 or NSA1. A signal g is said to be on the *boundary* of an X-region if $g \in$ X-region and there exists at least one g's fan-out signal p, where $p \notin$ X-region. For example, G2, net14, and G16 are on the boundary of the X-region shown in Figure 3. We first calculate the good-circuit value v of the target signal through simulation. Then, we flip the value on the target signal to v' and simulate the pattern again with all other boundary signals remaining the unknown value. If the simulated value on any output is different from its collected response, then the target signal cannot be a stuck-at-v' fault (or is NSAv'). Otherwise, setting a value v' on the target signal contradicts with the collected response. If the target faulty signal may interact with other faults within the X-region, the contradicted outputs must have an unknown value and hence no contradiction can be sustained.

Figure 3 shows an example of the above value-flipping simulation. The X-region is highlighted by the shadow background. The value of $G2$ is originally 1 and then flipped to 0. Then a mismatch is observed on $G17$. Therefore, the value of $G2$ must be 1. In other words, $G2$ is *NSA0*.

III. Faulty-Region Identification

The following two subsections detail the procedures of the *X-region initialization* and the iterative *X-region shrinking*. We also use the defective circuit shown in Figure 2 and the test patterns in Table I as example to illustrate the two procedures.

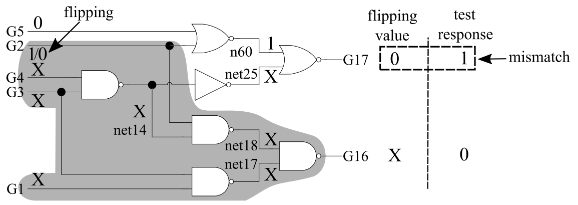

Fig. 3. **A mismatch occurs on $G17$ at pattern $(G1, G2, G3, G4, G5) = (11010)$ when applying the value-flipping on $G2$.**

A. Procedure of X-region Initialization

The inputs of the X-region initialization are the test patterns, the collected responses from the defective chip, and the CUD's netlist. This initial X-region becomes the starting point of the X-region shrinking to obtain a minimal X-region. Therefore, this identified initial X-region may be larger than it could necessarily be, but it has to cover all the real faults.

We first perform the active-path tracing from each failing output for each failing pattern [19]. Then we use the union of all the traced signals for all the patterns as the initial X-region, which is a conservative estimation of the region covering all real faults.

B. Procedure of X-region Shrinking

In X-region shrinking, we use the initial X-region obtained from the X-region initialization as a starting point. All the signals within the X-region are set to unknown to represent the potential implicated values of the real faults during simulation. Based on the X-region concept (introduced in Section II-B1), we apply the value-flipping technique (introduced in Section II-B3) to identify the NSAv signals on X-region's boundary and then shrink the X-region by removing the signals which are both NSA0 and NSA1.

In the following discuss, we use the *simulated good-circuit value* and *simulated value-flipping value* to represent a signal's logic value computed through a good-circuit simulation and a simulation with the value-flipping on a target signal (also called *value-flipping simulation*), respectively. We also use the *response value* to represent the logic value observed at an CUD's output. We say that a signal has the *good value* when its deduced value obtained from the value-flipping simulation is consistent with the simulated good-circuit value.

Figure 4 shows the X-region shrinking's algorithm, and the algorithm attempts to identify the NSAv signals based on both passing and failing patterns. For each pattern, we first run good-circuit simulation to obtain the simulated good-circuit value for each signal (Line2). Second, we replace the value of signals within the X-region with the unknown value (Line3-6). Then, the sub-procedure `IGV` utilizes the value-flipping technique to check whether each signal on X-region's boundary is NSAv' if the simulated good-circuit value of the signal is v (Line9). During `IGV`, some boundary signals are identified to have the good value. Next, the sub-procedure `IUI` performs the backward and forward implication based on the new identified good values to further derive more good values on the boundary (Line11). We repeatedly perform `IGV` and `IUI` until no more good value can be found for the given pattern.

The detail steps of the sub-procedure `IGV` are listed in Figure 5. For each signal on X-region's boundary with the unknown value, we first assign the signal's value opposite to its simulated good-circuit value v (Line2), and then obtain the value-flipping value for each output through the 3-value simulation (Line3). Next, if there exists a simulated value-flipping value on any output different from its response value, then the target signal is NSAv' and guaranteed to have good value for the given pattern (Line5-7). We also check if the target signal has been recognized as a NSAv for previous patterns. If yes, then we remove it from the X-region (Line8-10). Last, we assign the value of the target boundary signal as the good

Procedure 1 X-region Shrinking

```
1: for all {$p : p \in T, T = Test\ Set$} do
2:    Good_Circuit_Simulation()
3:    for all {$g : g \in X\text{-}region$} do
4:       Value($g$) ← $X$
5:    for all {$g : g \in Boundary$} do
6:       Value($g$) ← Good_Circuit_Value($g$)
7:    repeat
8:       //Identify_Good_Values_Using_Gate_Values
9:       IGV(Boundary)
10:      //Identify_Gate_Values_Using_Input_Information
11:      IUI(Boundary)
12:   until no good value identified in IUI
```

Fig. 4. **Procedure of X-region shrinking.**

value v instead of its original unknown value for any later value-flipping simulation of the current pattern (Line11). This assignment can help to identify more potential mismatches between output's response value and value-flipping value later on.

In addition, this assignment is only valid for the current pattern. Even though a signal is identified NSAv' and its good-circuit value for a later pattern is v, we cannot directly assign v at the signal since we only know that this signal is not stuck-at-v' and cannot prove that its value is v. We have to apply the value-flipping technique again to check whether the target signal has the good value for the current pattern, and then we can assign the good value at the target signal. Therefore, when we perform the X-region shrinking to the next pattern, the boundary signals of the X-region are still set to unknown initially.

Sub-procedure 1 IGV //Identify_Good_Values_Using_Gate_Values

```
1: for all {$g : g \in Boundary$ and Value($g$) $= X$} do
2:    Flip($g$)
3:    3-Value_Simulation()
4:    for all {$z : z \in PO, PO = Primary\ Output$} do
5:       if Mismatch($z$) then
6:          $v$ ← Good_Circuit_Value
7:          $g$ is NSA$v'$ and has Good_Circuit_Value
8:          if $g$ is also NSA$v$ then
9:             X-region ← X-region − {$g$}
10:            update Boundary
11:         Value($g$) ← Good_Circuit_Value
12:         break
```

Fig. 5. **Identify good values using gate values**

Figure 6(a) and Figure 6(b) show an example how the signal $net14$ is proved as NSA0 and NSA1 at different patterns, respectively. Since being both NSA0 and NSA1, $net14$ can be proved fault-free and then removed from the X-region.

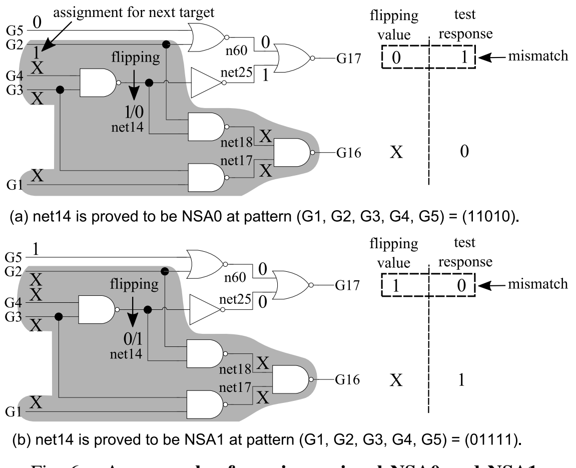

Fig. 6. **An example of proving a signal NSA0 and NSA1.**

Figure 7 shows the detail steps of the sub-procedure `IUI`, which applies the implication technique and the value-flipping technique on the fanins of the boundary signals to further identify more boundary signals having the good value. Those new identified good values can then help the sub-procedure `IGV` in the next iteration to find more NSA0 and NSA1 signals.

For each boundary signal with the unknown value, the sub-procedure `IUI` attempts to prove that this signal has the good value v, where v is the simulated good-circuit value for the given pattern. If the target signal has been proved NSAv' and its fanins' good values can propagate to the target signal, then the target signal has the good value. Two cases are discussed in the sub-procedure `IUI` to determine whether the signal's fanins have the good value and can propagate to the signal. If the simulated good-circuit value of any its fanin is the controlling value, then the target signal has the good value if any of its controlling fanin has the good value (Line4-8). Otherwise, the target signal has the good value if all of its non-controlling fanins have the good value (Line10-16).

Sub-procedure 2 IUI //Identify_Gate_Values_Using_Input_Information

```
1:  for all {$g : g \in Boundary$ and Value($g$) $= X$} do
2:    if $g$ is NSA$v$' then
3:      if $g$ has controlling fan-ins then
4:        $FI \leftarrow$ ControllingFanIn($g$)
5:        for all {$g_{fi} : g_{fi} \in FI$} do
6:          if Is_Input_Good($g_{fi}$) = true then
7:            $g$ has GoodValue
8:            break
9:      else
10:       $FI \leftarrow$ NonControllingFanIn($g$)
11:       for all {$g_{fi} : g_{fi} \in FI$} do
12:         if Is_Input_Good($g_{fi}$) = false then
13:           break
14:       if all $g_{fi} \in FI$ are good then
15:         $g$ has GoodValue
16:         break
```

Fig. 7. **Sub-procedure of** IUI**.**

In the sub-procedure `IUI`, we use the function `Is_Input_Good(`g`)` to determine whether the fanin g has the good value. The fanins of the target signal are classified into three types: on the X-region's boundary, outside the X-region, and inside the X-region. The examination rule for each type of fanins is listed as follows:

1) If the fanin is on the boundary, we already know whether this fanin has the good value during the sub-procedure `IGV` or the earlier stage of the sub-procedure `IUI`.
2) If the fanin is outside the X-region, the fanin is fault-free. However, the fanin's value on the CUD is not necessarily the same as the simulated good-circuit value if this fanin is on the error-propagation path of real faults. To obtain this fanin's actual value, we apply the value-flipping technique to check if any output mismatch exists. If the mismatch exists, the fanin is guaranteed to have the good value. Otherwise, we recursively check its fanins until a good value can be found or no good value can be deduced. An example is shown in Fig.8.
3) If the fanin is inside the X-region, we cannot know its value since it is set to unknown.

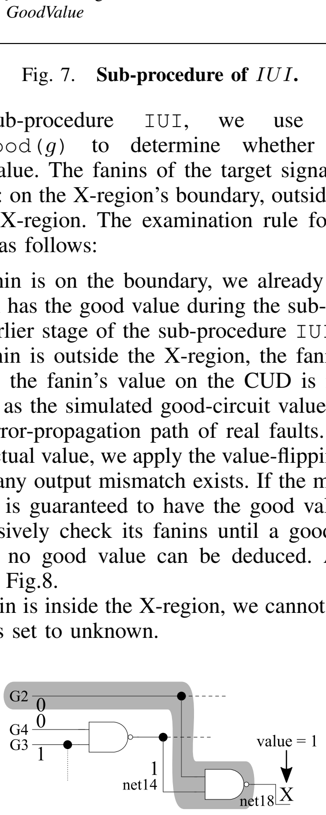

Fig. 8. $net18$ **can be derived as 1 when** $net18$ **is** *NSA0* **and** $G2$ **is 0.**

Figure 9 shows an example that the sub-procedure `IGV` can observe more output mismatch after knowing some additional good values in the sub-procedure `IUI`. It also means that more NSAv signals can be found and hence the X-region can be further shrunk.

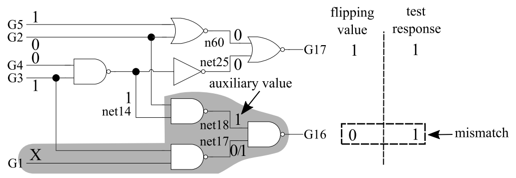

Fig. 9. **The derived value of** $net18$ **helps the faulty value of** $net17$ **propagating to POs in IGV, and** $net17$ **can be proved as** *NSA1***.**

IV. Fault-Candidate Ranking

After a minimal suspect region covering all real faults is reported by the faulty-region identification, we then rank the fault candidates within the reported region to specifically identify the real faults. In our fault-candidate ranking, we utilize two cost functions, $Match\_Sum(c)$ and $FM\_Det(c)$, to evaluate a fault candidate c by comparing the fault simulation result of c with the CUD's response. Only the failing patterns are used in this ranking process.

The notations, for calculating $Match\_Sum(c)$, $EPO_{fsim}(c,p)$ denotes the set of erroneous outputs obtained from the fault simulation of fault c at pattern p. $EPO_{test}(p)$ denotes the set of erroneous outputs obtained from the CUD's response at pattern p. $|EPO_{fsim}(c,p) \cap EPO_{test}(p)|$ denotes the number of erroneous outputs in the intersection of $EPO_{fsim}(c,p)$ and $EPO_{test}(p)$, meaning the number of erroneous outputs produced by the fault c and are also in the CUD's response at pattern p.

Equation 1 shows the definition of $Match\_Sum(c)$, which represents the total number of matched erroneous outputs for the fault c over all failing patterns. Basically, a higher value of $Match\_Sum(c)$ means a higher possibility that the candidate c is a real fault in our ranking method, and equation 2 shows the definition of $FM\_Det(c)$.

$$Match\_Sum(c) = \sum_{p=1}^{|\text{failing pat.}|} |EPO_{fsim}(c,p) \cap EPO_{test}(p)| \quad (1)$$

$$FM\_Det(c) = \sum_{p=1}^{|\text{failing pat.}|} isFM(c,p), \quad (2)$$

where

$$isFM(c,p) = \begin{cases} 0 & \text{if } EPO_{fsim}(c,p) \neq EPO_{test}(p) \\ 1 & \text{if } EPO_{fsim}(c,p) = EPO_{test}(p). \end{cases}$$

The function $isFM(c,p)$ returns whether the fault c can *full-match* the response of the failing pattern p. The function $FM\_Det(c)$ represents the total number of the full-matched patterns by the fault c.

Figure 10 summarizes our ranking method. A fault c with at least one full-matched pattern is more likely to be a real fault than a fault without any full-matched pattern even if the fault c has less $Match\_Sum(c)$ [8][12][13][15][20]. Therefore, our ranking method generally ranks the faults with a full-matched pattern higher than the faults without a full-matched pattern. We further divide the full-matching faults into three types, (1) the faults on the X-region's boundary and the fault site is identified as a NSA0 or NSA1 signal during the X-region shrinking, (2) the faults inside the X-region, and (3) the faults on the X-region's boundary and the fault site cannot be identified as a NSA0 or NSA1 signal during the X-region shrinking. A 1st-type fault ranks higher than the other two types because (i) a NSAv signal shows the possibility of generating a mis-match output from the fault site, and (ii) the existence of the 1st-type fault may be the reason why this fault site cannot be proved NSAv'. A 3rd-type fault ranks lower than the other two types because if the 3rd-type fault is the real fault which explains a certain failing pattern, there must be propagation paths to propagate its faulty value to outputs. Then its opposite faulty value could likely be propagated to outputs through similar paths. Since its fault site cannot be proved as a NSAv signal, a reasonable explanation is that the real fault is not on that fault site.

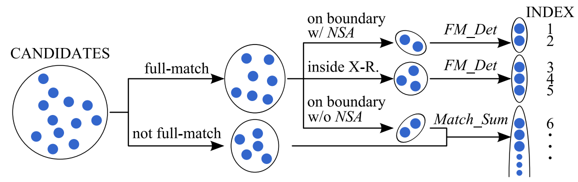

Fig. 10. **Illustration of the ranking ordering.**

V. Experimental Results

The proposed diagnosis framework is implemented in C++ and the experiments are conducted on a workstation with a 2.0GHz CPU and 16G memory. The benchmark circuits are ISCAS'89 benchmark circuits, where all circuits are full-scanned such that all flip-flops can be directly observed and controlled. For each benchmark circuit, the test patterns are generated by a PODEM-based ATPG [21] and the coverage for single stuck-at fault is 100%.

In the following experiment, we inject 3 nearby stuck-at faults at a time to sample a CUD, and then apply our diagnosis framework. Table II lists the average results after applying the proposed method over 100 sampled CUDs. Column 2 shows the number of test patterns used in the experiments. Column 3 and 4 list the percentage of failing patterns and the *fault diameter* associated with the injected faults, respectively. The definition of the fault diameter is the maximum length of all the shortest paths on netlist between any two injected faults, which is used to measure the locality of the injected faults. Column 5 and 6 list the size of the X-region, that is the number of signals covered by the X-region, before and after applying the X-region-shrinking procedure, respectively. Column 7 lists the *shrinking percentage* from the initial X-region to the final X-region. Column 8 lists the minimum *fault-to-boundary distance* after applying the X-region shrinking, which is defined as the minimum netlist distance from an injected fault to the boundary of the final shrunk X-region. This minimum fault-to-boundary distance represents how effectively the X-region-shrinking procedure can reduce the X-region. The lower the distance, the better performance of the X-region-shrinking procedure. Last, Column 9 lists the runtime of the X-region shrinking in seconds.

circuit	test	failing	fault	size of X-region			min. flt. to	runtime
	pttns.	pttn. %	diameter	initial	final	shrink %	boundary	(sec)
s1196	201	40.67	3.45	267.50	91.45	65.81	0.20	0.60
s1423	82	63.90	4.35	270.65	174.25	35.62	0.25	0.32
s713	73	67.05	3.95	224.70	106.55	52.58	0.60	0.19
s5378	317	55.79	4.25	474.15	269.35	43.19	1.50	2.47
s13207	604	55.57	3.50	294.10	170.30	42.09	2.00	6.62
s35932	77	59.74	3.60	392.25	247.55	36.89	0.30	7.91
s38584	889	58.50	3.80	200.75	72.60	63.84	0.25	38.78
s38417	1371	54.03	3.35	622.90	358.15	42.50	2.05	79.58
average		56.91	3.78			47.81	0.89	

TABLE II
Experimental results for 100 3-stuck-fault CUDs.

As the result shows, the average shrinking percentage and minimal fault-to-boundary distance achieved by the X-region shrinking are 47.81% and 0.89, respectively. This result first demonstrates that the X-region shrinking can effectively eliminate the false candidate signals and minimize the suspect region such that the resulting X-region's boundary is very close to real faults. Once the X-region's boundary is reaching the real faults, it is difficult to further move the X-region's boundary inward and the fan-in cones connecting to the real faults still remain in the X-region. Also, this result shows the efficiency of the X-region shrinking. The longest runtime among all benchmark circuits is 79.58 seconds (including the runtime of fault-candidate ranking).

Table III shows the results of the fault-candidate ranking. Column 2, 3, and 4 list the number of repair trails required until the first, second, and third successful repair occurs, respectively. After a real fault is detected, we run the X-region shrinking again based on the new CUD, which removes the detected fault already. The runtime reported in the Column 5 is the sum of the runtime of each individual X-region shrinking and fault-candidate ranking. As the result shows, it requires in average 5.11, 9.85, and 15.32 repair trails to hit the first, second, and third successful repair, respectively. It implies that almost 5 trials are required to hit a successful repair. Note that the equivalent faults are viewed as individual ones in our diagnosis framework and hence the reported results may not look as promising as it could be.

circuit	1st successful repair	2nd successful repair	3rd successful repair	runtime (sec)
s1196	2.20	5.15	7.00	0.84
s1423	2.90	7.80	15.45	0.50
s713	6.80	12.35	15.40	0.24
s5378	6.50	13.10	27.75	3.29
s13207	8.60	11.15	16.40	11.33
s35932	2.50	6.80	8.45	13.99
s38584	4.00	5.85	6.90	57.59
s38417	7.35	16.60	25.25	117.09
average	5.11	9.85	15.32	

TABLE III
The number of repair trails required until the first, second, or third successful repair occurs for 3-stuck-at-fault CUDs.

Table IV shows the average results of the first X-region shrinking and fault-candidate ranking for 100 5-stuck-at-fault CUDs. A similar trend can be observed as shown in Table II and III. The shrinking percentage is 43.97% and the minimal fault-to-boundary distance is 0.54 in average, which demonstrates that the X-region shrinking can also effectively shrink the X-region's boundary toward real faults when more faults are in the CUD. The average numbers of repair trials required to hit the first and fifth successful repair are 4.31 and 23.64, respectively. In average, less than 5 repair trails are required to hit 1 successful repair. In addition, the longest runtime among all benchmark circuits is 191.12 seconds, showing that the proposed diagnosis framework is scalable when the number of real faults increases.

Note that the average fault diameter is 4.35 in Table IV, which is larger than that of the modeled region used in any published result of the region-based diagnosis for ISCAS benchmark circuits, such as [9] [10] [11]. Unlike the region-based diagnosis whose runtime increases exponentially with the increase of the modeled region's size, the proposed diagnosis framework requires no assumption on the size of the suspect region and its runtime is indifferent to the fault diameter. This further demonstrates the flexibility as well as the scalability of proposed diagnosis framework.

Compared with the SLAT-based methods, such as [12] [13] [14] [15], the advantage of the proposed diagnosis framework is that its performance does not rely on the number of existing SLAT patterns. The number of SLAT patterns can be small especially when the erroneous signals of the multiple faults interact with one another. In order to create such scenarios, we selectively sample the injected faults for 3-stuck-at-fault CUDs whose percentage of the resulting SLAT patterns is low. Table V shows the average results over 100 such 3-stuck-at-fault CUDs.

In Table V, Column 2 lists the average number of existing SLAT patterns, which is 7.56. Note that even a pattern is identified as a SLAT pattern, its erroneous response explained by a single fault may not result from a single real fault. It could result from multiple real faults. Also, the percentage of failing patterns is only 9.44% in Table V, which is almost one sixth of the failing-pattern percentage shown in Table II (56.91%). The above two numbers show that

circuit	failing pttn. %	fault diameter	first X-region shrinking				fault-candidate ranking		total runtime (sec)
			size of X-region			min. flt. to boundary	1st successful repair	5th successful repair	
			initial	final	shrink %				
s1196	63.46	4.05	351.10	190.30	45.80	0.10	2.05	10.60	1.47
s1423	74.45	4.25	265.85	159.20	40.12	0.35	3.15	24.80	0.73
s713	78.97	4.90	270.00	150.70	44.19	0.55	6.70	22.05	0.47
s5378	48.60	4.85	491.45	300.20	38.92	0.80	7.85	41.05	4.97
s13207	72.62	4.60	238.60	139.50	41.53	1.00	3.70	19.85	23.22
s35932	75.91	2.95	278.45	159.75	42.63	0.25	2.05	13.00	10.32
s38584	77.04	4.20	409.60	162.60	60.30	0.30	2.75	18.80	94.43
s38417	73.32	5.00	816.45	503.70	38.31	1.00	6.25	38.95	191.12
average	70.55	4.35			43.97	0.54	4.31	23.64	

TABLE IV
The experimental results of X-region shrinking and fault-candidate ranking for 5-stuck-at-fault CUDs.

circuit	# of SLAT pttn.	failing pttn.(%)	fault diameter	first X-region shrinking				fault-candidate ranking			total runtime (sec)
				size of X-region			min. flt. to boundary	1st successful repair	2nd successful repair	3rd successful repair	
				initial	final	shrink %					
s1196	7.53	3.78	3.51	191.57	37.31	80.52	0.27	2.26	4.33	6.61	0.61
s1423	4.12	5.15	3.14	144.33	74.21	48.58	0.90	5.13	10.31	13.52	0.39
s713	4.37	10.84	3.19	100.74	33.43	66.82	0.52	4.66	8.53	10.56	0.15
s5378	5.70	3.74	3.73	159.27	79.52	50.07	1.74	7.70	13.76	18.13	2.59
s13207	8.59	9.71	3.53	366.70	221.81	39.51	2.64	6.53	14.19	19.28	8.13
s35932	7.92	25.94	3.56	1498.74	979.00	34.68	1.42	2.87	5.83	9.48	9.98
s38584	12.70	7.13	3.54	419.33	196.90	53.04	1.39	5.70	14.24	18.10	35.13
s38417	9.56	9.24	3.64	335.59	148.82	55.65	2.11	6.68	18.65	25.77	76.36
average	7.56	9.44	3.48			53.61	1.37	5.19	11.23	15.18	

TABLE V
The experimental results of X-region shrinking and fault-candidate ranking for 3-stuck-at-fault CUDs with a small number of SLAT patterns.

the sampled combinations of the 3 stuck-at faults are hard to be identified with the SLAT-based methods or any diagnosis method using only the failing patterns.

As the result shows in Table V, our diagnosis framework can achieve an average 53.61% shrinking percentage but its average minimal fault-to-boundary distance is 1.37, which is 0.48 higher than that shown in Table II. It implies that the faults interacting with another may prevent the identification of the NSAv signals. However, the numbers of repair trails required to hit the first, second, and third successful repair are almost the same as that in Table II, which demonstrates that our diagnosis framework can still effectively diagnose those hard-to-identified multiple faults even though the number of SLAT patterns and the failing-pattern percentage are both low.

VI. Conclusions

In this paper, we proposed a diagnosis framework targeting multiple stuck-at faults. This framework utilizes the faulty-region identification to obtain a minimal suspect region covering all real faults, and the proposed ranking method ranks the fault candidates within the suspect region using the information obtained during the faulty-region identification. The experimental results demonstrate that the proposed diagnosis framework can effectively minimize the suspect region and its runtime is scalable to the circuit and the number of existing faults. In addition, the proposed framework requires no assumption on the number of existing faults as well as the size of the possible faulty region, which greatly increases the flexibility of its application.

References

[1] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A Reconfigurable Design-for-Debug Infrastructure for SoCs," *Design Automation Conference*, pp. 7–12, 2006.

[2] K.-H. Chang, I. L. Markov, and V. Bertacco, "Automating Post-Silicon Debugging and Repair," *International Conference on Computer-Aided Design*, pp. 91–98, 2007.

[3] P. Bernardi, M. Grosso, M. Rebaudengo, and M. Sonza Reorda, "A Pattern Ordering Algorithm for Reducing The Size of Fault Dictionaries," *VLSI Test Symposium*, pp. 386–391, 2006.

[4] I. Pomeranz and S. M. Reddy, "A Same/Different Fault Dictionary: An Extended Pass/Fail Fault Dictionary with Improved Diagnostic Resolution," *Design, Automation and Test in Europe*, pp. 1474–1479, 2008.

[5] B. Chess and T. Larrabee, "Creating Small Fault Dictionaries," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 3, pp. 346–356, 1999.

[6] B. Arslan and A. Orailoglu, "Fault Dictionary Size Reduction Through Test Response Superposition," *International Conference on Computer Design*, pp. 480–485, 2002.

[7] W. Zou, W.-T. Cheng, S. Reddy, and H. Tang, "Speeding Up Effect-Cause Defect Diagnosis Using a Small Dictionary," *VLSI Test Symposium*, pp. 225–230, 2007.

[8] Z. Wang, K.-H. Tsai, M. Marek-Sadowska, and J. Rajski, "An Efficient and Effective Methodology on The Multiple Fault Diagnosis," *International Test Conference*, vol. 1, pp. 329–338, 2003.

[9] V. Boppana, R. Mukherjee, J. Jain, M. Fujita, and P. Bollineni, "Multiple Error Diagnosis Based on XLISTs," *Design Automation Conference*, pp. 660–665, 1999.

[10] A. L. D'Souza and M. S. Hsiao, "Error Diagnosis of Sequential Circuits Using Region-Based Model," *Journal of Electronic Testing*, vol. 21, no. 2, pp. 115–126, 2005.

[11] N. Sridhar and M. S. Hsiao, "On Efficient Error Diagnosis of Digital Circuits," *International Test Conference*, pp. 678–687, 2001.

[12] L. M. Huisman, "Diagnosing Arbitrary Defects in Logic Designs Using Single Location at A Time (SLAT)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 1, pp. 91–101, 2004.

[13] X. Wen, S. Kajihara, K. Miyase, Y. Yamato, K. K. Saluja, L.-T. Wang, and K. Kinoshita, "A Per-Test Fault Diagnosis Method Based on The X-Fault Model," *IEICE Transactions on Information and Systems*, vol. E89-D, no. 11, pp. 2756–2765, 2006.

[14] R. Desineni and R. Blanton, "Diagnosis of Arbitrary Defects Using Neighborhood Function Extraction," *VLSI Test Symposium*, pp. 366–373, 2005.

[15] R. Desineni, O. Poku, and R. D. Blanton, "A Logic Diagnosis Methodology for Improved Localization and Extraction of Accurate Defect Behavior," *International Test Conference*, pp. 1–10, 2006.

[16] Y.-C. Lin, F. Lu, and K.-T. Cheng, "Multiple-Fault Diagnosis Based on Adaptive Diagnostic Test Pattern Generation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 5, pp. 932–942, 2007.

[17] M. Abramovici and M. A. Breuer, "Fault Diagnosis Based on Effect-Cause Analysis: An Introduction," *Design Automation Conference*, pp. 69–76, 1980.

[18] M. Abramovici and M. A. Breuer, "Multiple Fault Diagnosis in Combinational Circuits Based on An Effect-Cause Analysis," *IEEE Transactions on Computers*, vol. C-29, no. 6, pp. 451–460, 1980.

[19] C.-C. Yen, T. Lin, H. Lin, K. Yang, T. Liu, and Y.-C. Hsu, "Diagnosing Silicon Failures Based on Functional Test Patterns," *International Workshop on Microprocessor Test and Verification*, pp. 94–98, 2006.

[20] S.-Y. Huang, "On Improving The Accuracy of Multiple Defect Diagnosis," *VLSI Test Symposium*, pp. 34–39, 2001.

[21] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Transactions on Computers*, vol. C-30, no. 3, pp. 215–222, 1981.