SUMMARY  In general, we do not know which fault model can explain the cause of the faulty values at the primary outputs in a circuit under test before starting diagnosis. Moreover, under Built-In Self Test (BIST) environment, it is difficult to know which primary output has a faulty value on the application of a failing test pattern. In this paper, we propose an effective diagnosis method on multiple fault models, based on only pass/fail information on the applied test patterns. The proposed method deduces both the fault model and the fault location based on the number of detections for the single stuck-at fault at each line, by performing single stuck-at fault simulation with both passing and failing test patterns. To improve the ability of fault diagnosis, our method uses the logic values of lines and the condition whether the stuck-at faults at the lines are detected or not by passing and failing test patterns. Experimental results show that our method can accurately identify the fault models (stuck-at fault model, AND/OR bridging fault model, dominance bridging fault model, or open fault model) for 90% faulty circuits and that the faulty sites are located within two candidate faults.

key words: diagnosis, fault model, fault location, fault simulation, combinational circuits, pass/fail information

1. Introduction

Failure analysis has been an important step in manufacturing to maintain quality of VLSI. The failure analysis consists of locating the fault and characterizing the defect. With the scaling of VLSI feature and circuit size, we have to consider various fault models in addition to the single stuck-at fault model to improve the quality of the failure analysis. Thus, diagnosis methods for the multiple stuck-at fault model, the bridging fault model, and the open fault model have been proposed in previous papers [3]–[6]. In general, test engineers do not know what kind of fault model is able to explain the cause of the faulty values at the primary outputs in a circuit under test (CUT) before starting the diagnosis. However, an assumption of the previous diagnosis methods in [4]–[6] is that the fault model is known before starting diagnosis. Therefore, the methods have been done well premised on the assumed fault model.

The failure analysis under Built-In Self Test (BIST) environment is demanded because BIST is as effective way in testing. Under BIST environment, it is difficult to know which primary output has the faulty value on the application of a failing test pattern [1].

However, the previous methods for locating fault sites presented in [3]–[6] have positively used the locations of the primary outputs having the faulty values to reduce the number of candidate faults. Therefore, the previous methods are not able to apply the diagnosis under BIST environment.

Recently, a per-test fault diagnosis method by using an X-fault model has been proposed in [7]. The X-fault model represents all possible behaviors of a defect in a gate and/or on its fan-out branches. The proposed method in [7] does not use the passing test patterns to diagnose the faulty circuits. In our opinion, the method in [7] is able to diagnose the faulty circuits with short processing times, because the number of failing test patterns used in the diagnosis is relatively small. However, it is not easy for the diagnosis method without using the passing test patterns to distinguish between the actual fault and the easy-to-detect faults that do not exist in CUT, because the easy-to-detect faults are detected by many failing test patterns. At worst, the easy-to-detect faults that do not exist in CUT still remain in the set of candidate faults after performing the diagnosis using the failing test patterns. On the other hand, the diagnosis method using the passing test patterns is easy to identify the inconsistency caused by the easy-to-detect faults that do not exist in the CUT, because the easy-to-detect faults can be detected by many passing test patterns for CUT.

We believe that the fault diagnosis that identifies the fault model in the faulty circuit from the existing fault models such as stuck-at fault model, AND/OR bridging fault model, dominance bridging fault model, and open fault model is needed to diagnose the faulty circuits with accuracy. To the authors’ knowledge, there is no report for the method that identifies both the candidate fault models and the candidate faults by using only pass/fail information.

Previously, we proposed a diagnosis method on multiple fault models [8]. The study in [8] is the preliminary study for developing a diagnosis method presented in this paper.

Under the single fault assumption, we propose the method that identifies both the fault model and the fault location in the faulty circuit based on only pass/fail information on the applied test patterns. On the diagnosis method presented in this paper, we newly consider an open fault model [10] as the fault model.
The fundamental contributions of this study are
1) to propose a method that deduces a candidate fault model and candidate faults based on the number of detections for the single stuck-at fault at each line, by performing single stuck-at fault simulation with both passing and failing test patterns,
2) to propose the method that uses the logic values of lines and the condition whether the stuck-at faults at the lines are detected or not by passing and failing test patterns to improve the ability for identifying fault models,
3) to examine carefully the order of the diagnosis procedures of each fault model for reducing the CPU times, and
4) to validate experimentally that our method is effective in deducing both the fault model and the fault location.

From experimental results we show that the proposed method can accurately identify the fault models for 90% faulty circuits in ISCAS’85 and full-scan version of ISCAS’89 benchmark circuits and that can locate faulty sites within two candidate faults.

The rest of the paper is organized as follows. In Sect. 2, we define some terminologies to explain the proposed method. Also we describe the fault models which are considered in this paper.

We assume that either a single stuck-at fault, an AND/OR bridging fault, a dominance bridging fault or an open fault exists in the faulty circuit. We do not know what kind of fault model can explain the faulty responses of the faulty circuit before starting the fault diagnosis. Note that this assumption is different from those of the previous research efforts [3]–[6].

Next we explain those fault models briefly.

The stuck-at 0 (1) fault model assumes that a value at a line has 0 (1), permanently.

The bridging fault model is distinguished between an AND/OR bridging fault and a dominance bridging fault. Figure 1 shows three types of the bridging fault models that are considered in this paper. In the AND/OR bridging fault, the line is driven to the logic value 0/1 when the bridging fault is activated. In this paper, the bridging fault between lines $L_i$ and $L_j$ is denoted by $<L_i,L_j>$. In a dominance bridging fault, the logic value at one of the two lines (the dominated line) is always dominated by the logic value at the other one (the dominating line). Also the dominating line and the dominated line are denoted by $L_{in}$ and $L_{red}$, respectively.

Figure 2 shows the open fault model is considered in this paper. In this paper we assume the open fault model with considering the affects of its adjacent lines [10]. The voltage of a faulty line is determined by the linear combination of voltages at adjacent lines and the source line.

The open fault model, the line $v$ is completely open. There are coupling capacitors between the line $v$ and its adjacent lines $a_1$, $a_2$, and $a_3$. From the observation in the Ref. [9], we suppose that the adjacent lines have large influence on the voltage of the faulty line. The meaning is that the voltage level at $v'$ in Fig 2 will be at a high level or a...
low level according to the voltage level of \( n \) adjacent lines.

In other words, the logic value at the faulty line \( v \) with the open fault will be at 1 or 0 according to the logic values at the adjacent lines depending on the test pattern.

Now let \( v \) be a line with open defect and \( Val_v \) be the logic value on \( v \) in a faulty circuit. Also let \( a_i (i = 1, 2, \ldots, n) \) be adjacent line of \( v \), and \( Val_{ai} \) be the logic value on \( a_i \). We define the value \( Val_v \) of the faulty line \( v' \) by the following equation to propose the open fault model with considering the affects of \( n \) adjacent lines.

\[
Val_v = \text{majority}_n(Val_1, \ldots, Val_i, \ldots, Val_n) \tag{1}
\]

In this paper, we assume that the faulty line has the faulty value (the complemented value for the fault-free value) when the number of adjacent lines with the value 0 and the number of adjacent lines with the value 1 is even.

For example, three lines \( a_1, a_2 \) and \( a_3 \) adjacent exist on faulty line \( v \), as shown in Figure 2. When \( Val_1 = 0, Val_2 = 1 \) and \( Val_3 = 1 \), \( Val_v \) takes 1 according to Eq. (1). If the value of the source line of \( v \) takes 0, then the fault is excited. On the other hand, when \( Val_1 = 0, Val_2 = 0 \) and \( Val_3 = 1 \), \( Val_v \) takes 0. If the value of the source line of \( v \) takes 0, then the fault is not excited.

3. Method for Deducing Candidate Fault Models and Candidate Faults

3.1 Outline of Proposed Method

The proposed diagnosis method proceeds in the following two phases. Figure 3 shows an outline of the proposed method.

**Phase 1:** We perform single stuck-at fault simulation with the set of failing test patterns to count 0(1) detection times for the single stuck-at 0(1) fault at each line.

**Phase 2:** We use the following conditions to derive the diagnostic rules for each fault model,

1. 0(1) detection times for the lines that are obtained by Phase 1,
2. the condition whether the single stuck-at fault at the line is detected or not by a test pattern,
3. the logic values at the lines under a test pattern,
4. the bridging fault list that has been obtained by the layout extraction method [2],
5. the set of adjacent lines for each gate output, and
6. the fault excitation conditions and the fault propagation conditions on test generation for each fault model.

3.2 Phase 2 of the Proposed Method

Next, we explain the procedures of phase 2 which can identify the candidate fault model and can locate the candidate faults.

We use the codes shown in Table 1 to explain the diagnostic rules in phase 2. Also we use the codes shown in

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Definition for encoding the condition of the line.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value at line</td>
<td>0</td>
</tr>
<tr>
<td>detected</td>
<td>stuck-at 1 fault</td>
</tr>
<tr>
<td>code</td>
<td>e0</td>
</tr>
</tbody>
</table>

To implement the proposed method in the programming language.

**Code for diagnostic rules**

According to Table 1, we encode the values at the lines under the failing test patterns and the condition whether the single stuck-at 0 or 1 fault at the line is detected or not by the failing test pattern. \( e0(e1) \) represents that the value at the line is "0(1)" and the stuck-at 1(0) fault at its line is detected by the test pattern. On the other hand, \( u0(u1) \) represents that the value at the line is "0(1)" and the stuck-at 1(0) fault at its line is not detected by the test pattern.

In order to reduce the CPU times we examine carefully the order of the diagnostic procedures for each fault model. On the proposed method, the order of the diagnosis procedures for each fault model is changed from that of the method in [8]. To identify AND/OR bridging faults needs to deal with the pairs of lines. For example, we have to check the sum of detection times for stuck-at 0 at the lines on the diagnosis for AND bridging fault. The diagnosis procedures for AND/OR bridging faults are postponed. Because long processing time is needed to treat the pairs of the lines, to check the conditions for AND/OR bridging faults needs more CPU times.

Figure 4 shows a flowchart of the phase 2.

Next, we show the procedures of phase 2.

**Procedures of phase 2**

1. Encoding the status of the lines derived from the results of Phase 1.
2. Perform the diagnosis based on diagnostic rules for the stuck-at fault. If the candidate faults for the single stuck-at fault have been obtained, then we determine that the candidate fault model of the CUT will be the stuck-at fault and go to step 8.
3. Perform the diagnosis based on diagnostic rules for dominance bridging faults. If the candidate faults for dominance bridging faults have been obtained, then we determine that the candidate fault model of the CUT will be the dominance bridging fault and go to step 8.
4. Perform the diagnosis based on diagnostic rules for open faults. If the candidate faults for open faults have been obtained, then we determine that the candidate fault model of the CUT will be the open fault and go to step 8.

5. Perform the diagnosis based on diagnostic rules for AND bridging faults. If the candidate faults for AND bridging faults have been obtained, then we determine that the candidate fault model of the CUT will be the AND bridging fault and go to step 8.

6. Perform the diagnosis based on diagnostic rules for OR bridging faults. If the candidate faults for OR bridging faults have been obtained, then we determine that the candidate fault model of the CUT will be the OR bridging fault and go to step 8.

7. Determine the existing fault in CUT as the unknown fault model.

8. Report the candidate fault model and the candidate faults. □

3.3 Diagnostic Rules for Deducing Fault Models and Candidate Faults

We describe diagnostic rules for deducing the candidate fault model and the candidate faults that are used in phase 2.

3.3.1 Diagnostic Rules for Stuck-at Faults

Firstly, we describe the diagnostic rules for stuck-at faults.

**Diagnostic rules for stuck-at faults**

If the following conditions both (stuck-at fault-i) and (stuck-at fault-ii) are satisfied, we determine that the fault model of CUT will be the stuck-at fault.

- **(stuck-at fault-i):** The detection times for the single stuck-at 0 or 1 fault at line $L$ is equal to the total number of failing test patterns for CUT.
- **(stuck-at fault-ii):** The single stuck-at fault at line $L$ is not detected by all passing test patterns for CUT.

3.3.2 Diagnostic Rules for Dominance Bridging Faults

Next we describe the method for identifying dominance bridging faults.

**Diagnostic rules for dominance bridging faults**

If all conditions from (dominance bridging fault-i) to (dominance bridging fault-iv) are satisfied under the set of failing test patterns, then we determine that the fault model of $<L_{\text{ing}}, L_{\text{ted}}>$ will be the dominance bridging fault. By using the conditions (dominance bridging fault-i) and (dominance bridging fault-ii), we deduce the candidates of the dominated lines. And we deduce the candidates of the dominating lines based on condition (dominance bridging fault-iii).

In order to check whether the condition (dominance bridging fault-v) is satisfied or not, we use single stuck-at fault simulation with the set of passing test patterns.

- **(dominance bridging fault-i):** The sum of detection times of the single stuck-at 0 fault at $L$ and detection times of the single stuck-at 1 fault at $L$ is equal to the total number of failing test patterns for CUT. The lines that satisfy with the condition (dominance bridging fault-i) are the candidates of $L_{\text{ted}}$.
- **(dominance bridging fault-ii):** The candidates for $L_{\text{ted}}$ belong to the fault list of bridging faults obtained by the fault extraction method [2].
- **(dominance bridging fault-iii):** Let line $i$ and $j$ be an element of the set of candidates of $L_{\text{ing}}$ and $L_{\text{ted}}$, respectively. Under all failing test patterns, the value $v (v=0$ or $1)$ at line $i$ is always as same as stuck-at $v$ fault at line $j$ that is detected by the failing test pattern.
- **(dominance bridging fault-iv):** $<L_{\text{ing}}, L_{\text{ted}}>$ belongs to the list of bridging faults obtained by the fault extraction method [2].

Furthermore, we use the conditions based on the set of passing test patterns to reduce the number of candidates.

- **(dominance bridging fault-v):** If stuck-at $v$ fault at
Let \( L_{\text{ed}} \) be detected by a passing test pattern and if the value at \( L_{\text{ing}} \) is as same as the faulty value \( v \) for \( L_{\text{ed}} \) under the same passing test pattern, then \(< L_{\text{ing}}, L_{\text{ed}} >\) is excluded from the set of candidate faults. □

Next we describe that the method for determining whether the above conditions are satisfied or not.

In order to check the condition (dominance bridging fault-i), we use the sum of the detection times for the stuck-at 0 fault and the detection times for the stuck-at 1 fault, and the total number of failing test patterns for CUT.

In order to check the condition (dominance bridging fault-iii), we use the codes as shown in Table 1 to identify the candidates of the dominating line \( L_{\text{ing}} \). If the following two conditions are satisfied, then we deduce that the fault model of \(< L_{\text{ing}}, L_{\text{ed}} >\) will be the dominance bridging fault.

1. \((\text{dominance bridging fault-iii-a:})\) Under the failing test patterns in which the candidate dominated line \( L_{\text{ed}} \) has the code \( e_0 \), and \( L_{\text{ing}} \) has the code \( e_1 \) or \( u_1 \).
2. \((\text{dominance bridging fault-iii-b:})\) Under the failing test patterns in which the candidate dominated line \( L_{\text{ed}} \) has the code \( e_1 \), and \( L_{\text{ing}} \) has the code \( e_0 \) or \( u_0 \).

### 3.3.3 Diagnostic Rules for Open Faults

Next we describe the method for identifying open faults. Firstly, we describe the conditions for detecting the open faults with considering adjacent lines [10].

**Conditions for detecting the open fault at \( v \):**

Let \( v \) be a faulty line and \( a_1, a_2, \) and \( a_3 \) be its adjacent lines. If a test pattern \( t \) satisfies the following conditions, then the test pattern \( t \) can detect the open fault at \( v \).

**Condition for open fault detection:**

- When a test pattern \( t \) is applied to CUT, the value on line \( v \) determined by the Eq. (1) is different from the value on \( v \) in a fault free circuit.

**Condition for open fault propagation:**

- When a test pattern \( t \) is a passing test pattern, the value on \( v \) is propagated to one or more primary outputs of CUT.

If a test pattern \( t \) satisfies the above conditions, \( t \) can detect the open fault on line \( v \). □

**Diagnostic rules for open faults**

If all conditions from (open fault-i) and (open fault-ii) are satisfied under the set of failing test patterns, then we determine that the fault model of \( L \) will be the open fault. If condition (open fault-iii) is satisfied under the set of passing test patterns, then we determine that the fault model of \( L \) will be the open fault. Where line \( L \) has \( N \) adjacent lines. Let \( a_i \) \((i = 1, 2, \ldots N)\) be adjacent line of \( L \).

1. **(open fault-i):** Under each failing test pattern, the number of the adjacent lines that have the opposite value of the value at \( L \) is larger than the number of the adjacent lines.
2. **(open fault-ii):** The sum of detection times of the single stuck-at 0 fault at \( L \) and detection times of the single stuck-at 1 fault at \( L \) is equal to the total number of failing test patterns for CUT.
3. **(open fault-iii):** Even though the stuck-at fault at \( L \) is detected by the passing test pattern, the conditions for the excitation of the open fault at \( L \) is not satisfied with the passing test pattern. □

Next we describe that the method for determining whether the above conditions are satisfied or not.

In order to check the condition (open fault-i), we use the codes as shown in Table 1 to identify the candidates of line \( L \) and its adjacent lines.

In order to check the condition (open fault-ii), we use the sum of the detection times for the stuck-at 0 fault and the detection times for the stuck-at 1 fault, and the total number of failing test patterns for CUT.

In order to check whether the condition (open fault-iii) is satisfied or not, we use single stuck-at fault simulation with the set of passing test patterns.

### 3.3.4 Diagnostic Rules for AND(OR) Bridging Faults

Next we describe the method for identifying AND(OR) bridging faults.

**Diagnostic rules for AND(OR) bridging faults**

If all conditions (AND(OR) bridging fault-i), (AND(OR) bridging fault-ii), and (AND(OR) bridging fault-iii) are satisfied under the set of failing test patterns and if the condition (AND(OR) bridging fault-iv) or (AND(OR) bridging fault-v) is satisfied under the set of passing test patterns, then the fault model of \(< L_i, L_j >\) will be the AND(OR) bridging fault. To check the conditions from (AND(OR) bridging fault-i) to (AND(OR) bridging fault-v), we always consider the pair of lines.

1. **(AND(OR) bridging fault-i):** The sum of detection times for stuck-at 0(1) fault at \( L_i \) and detection times for stuck-at 0(1) fault at \( L_j \) is equal to the total number of failing test patterns for CUT.
2. **(AND(OR) bridging fault-ii):** \(< L_i, L_j >\) is belonged to the bridging fault list obtained by the fault extraction method in [2].
3. **(AND(OR) bridging fault-iii):** Under the failing test pattern, \( L_i \) and \( L_j \) have opposite values and the stuck-at 0(1) fault at one line \( L_i \) or \( L_j \) is detected.
4. **(AND(OR) bridging fault-iv):** Under all passing test patterns, \( L_i \) and \( L_j \) have the same values.
5. **(AND(OR) bridging fault-v):** Under all passing test patterns, both stuck-at 0(1) faults at \( L_i \) and \( L_j \) are not detected. □

Next we describe that the method for determining whether the above condition is satisfied or not.

In order to check the condition (AND(OR) bridging fault-i), we use the detection times for the stuck-at 0(1) fault obtained by phase 1 and the number of failing test patterns for CUT.

In order to check the condition (AND(OR) bridging fault-iii), we use the codes as shown in Table 1. If the status of \(< L_i, L_j >\) is the pair of codes \((e_0, e_0)\) or \((e_1, u_0)\), then the fault model of \(< L_i, L_j >\) is deduced as the AND bridging fault. If the status of \(< L_i, L_j >\) is the pair of codes \((e_1, e_0)\) or \((e_0, u_1)\), then the fault model of \(< L_i, L_j >\) is deduced as
the OR bridging fault.

In order to check the condition (AND(OR) bridging fault-iv), we use logic simulation with passing test patterns. In order to check the condition (AND(OR) bridging fault-v), we use fault simulation with passing test patterns to determine whether single stuck-at faults at \( <L_i, L_j> \) are detected by the passing test pattern or not.

4. Example of the Proposed Method

Phase 1: Consider the example of Table 2 which shows the 0(1) value times and the 0(1) detection times of lines \( A, B, \ldots, \text{and} \ F \) under all failing test patterns. Here the number of failing test patterns is 10.

Phase 2:

According to the procedures of phase 2, we perform the diagnosis based on diagnostic rules for the single stuck-at fault. We do not obtain the candidate fault that satisfy with the condition (stuck-at fault-i) for the single stuck-at fault.

Then we proceed with the diagnosis based on diagnostic rules for dominance bridging faults. Line \( F \) is deduced as a candidate for \( L_{\text{fed}} \) by the condition (dominance bridging fault-i) for the dominance bridging fault, since the sum of the 0 and 1 detection times is equal to the number of failing test patterns (10).

Next we check the condition (dominance bridging fault-iii) for the pair \( <A_{\text{ing}}, F_{\text{fed}}> \). The pair satisfies with the condition (dominance bridging fault-iii) for the dominance bridging fault, because the value 0(1) at line \( A \) is always as same as stuck-at 0(1) fault at line \( F \). If the pair \( <A_{\text{ing}}, F_{\text{fed}}> \) is not excluded by the conditions (dominance bridging fault-iv) and (dominance bridging fault-v), then we determine that the candidate fault model will be the dominance bridging fault and deduce that the pair \( <A_{\text{ing}}, F_{\text{fed}}> \) will be the candidate fault.

5. Experimental Results

Experiments were performed for ISCAS’85 and full-scan version of ISCAS’89 benchmark circuits. We used 1024 random patterns as the diagnostic test pattern set. The program was run on a computer with Pentium 3.4 GHz and 3 GB memory.

In the experiments, we prepared 20 faulty circuits for each fault model. Where the number of failing test patterns in the faulty circuit is less than 100. We injected each fault model randomly. Though bridging fault model is classified into AND, OR, and dominance bridging faults, we inject the three bridging fault models into the same location in CUT.

Table 3 shows the experimental results for ISCAS’85 benchmark circuits and ISCAS’89 benchmark circuits. Table 3 shows a hit ratio for the candidate fault model in CUT, a success ratio, and an average number of candidate faults for each fault model. Note that the candidate faults of the bridging fault models (dominance, AND, and OR) are the pairs of two lines.

The “hit ratio for fault model” contains the hit ratio that is defined as the following equation.

\[
\text{hit ratio for candidate fault model(\%) = } \frac{\text{# of CFM}}{\text{# of faulty circuits}} \times 100
\]

100 % in the hit ratio of candidate fault models means that the proposed method is able to accurately identify the injected fault model for all faulty circuits.

In this experiment, we call the diagnosis result that satisfies the following condition 1 or condition 2 the successful case.

Condition 1 for the successful case is that the candidate fault model corresponds to an injected fault model into CUT.

Condition 2 for the successful case is that the candidate fault corresponds to an injected fault location into CUT.

The "success ratio" contains the success ratio is defined as the following equation.

\[
\text{success ratio(\%) = } \frac{\text{# of successful cases}}{\text{# of faulty circuits}} \times 100
\]

The "candidate faults" row contains the average number of candidate faults deduced by the proposed method.

<table>
<thead>
<tr>
<th>line name</th>
<th>0 times</th>
<th>0 detection times</th>
<th>1 times</th>
<th>1 detection times</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>( B )</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>( C )</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>( D )</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>( E )</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>( F )</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 3 Experimental results of the method.

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Stuck-at</th>
<th>Dominance</th>
<th>Open</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>hit ratio for candidate fault model(%)</td>
<td>90.4</td>
<td>100</td>
<td>90.9</td>
<td>96.6</td>
<td>83.4</td>
<td>80.9</td>
</tr>
<tr>
<td>success ratio(%)</td>
<td>96.1</td>
<td>100</td>
<td>94.4</td>
<td>100</td>
<td>94.1</td>
<td>91.9</td>
</tr>
<tr>
<td>candidate faults</td>
<td>1.2</td>
<td>1.7</td>
<td>1.2</td>
<td>1.02</td>
<td>1.11</td>
<td>1.12</td>
</tr>
</tbody>
</table>
The “Stuck-at” column contains the diagnosis results for the faulty circuits with stuck-at faults. Also the “Dominance”, “Open”, “AND”, and “OR” columns contain the results for the faulty circuits with each fault model.

In order to show the hit ratio for the candidate fault model for each circuit, we show the results in Table 4. The “CPU time” column contains the average CPU time (sec.) consumed by the proposed method.

Experimental results show that our method is capable of deducing the fault model correctly for all most all faulty circuits. Also experimental results show that our method is capable of deducing the faulty lines within 2 candidates for all most all faulty circuits.

In the proposed method, we examine carefully the order of the diagnosis procedures of each fault model for reducing the CPU times. We compare the CPU times consumed by the proposed method and the CPU times consumed by the method [8]. The order of the method [8] is from the procedure for stuck-at faults, the procedure for AND bridging faults, the procedure for OR bridging faults, to the procedure for dominance bridging faults. In this experiment, we add the procedure for open faults after the procedure for dominance bridging faults.

Table 5 shows the CPU times consumed by the proposed method and the CPU times consumed by the method [8]. The “M [8]” columns contain the average CPU times (sec.) consumed by the method [8]. The “PM” columns contain the average CPU times (sec.) consumed by the method proposed in this paper. From the experimental results we show that the proposed method is able to reduce the CPU times for diagnosing the faulty circuits with dominance bridging faults and open faults.

For example of cs38584 circuits with the dominance bridging fault model, the average CPU times consumed by the method proposed in this paper is about $\frac{1}{25}$ of the CPU times consumed by the method proposed in [8]. Also the average CPU times for diagnosing the faulty circuits with the open faults is about $\frac{1}{8}$ of the CPU times consumed by the method proposed in [8].

6. Conclusion

We proposed the method that identifies one fault model and faulty sites based on only pass/fail information on the applied test patterns. Experimental results show that the proposed method can accurately identify the fault models for 90% faulty circuits, and can locate the faulty sites within two candidate faults.

When the proposed method is applied to the faulty circuit with the multiple fault sites (multiple defects), the candidate fault model deduced by the proposed method will be the unknown fault model. In the future study, we have to develop the method that is able to diagnose the faulty circuit with the multiple fault sites (multiple defects).

Acknowledgement

This work was supported in part by Semiconductor Tech-
ology Academic Research Center (STAR) under the Research Project and in part by Japan Society for the Promotion of Science (JSPS) under the Grant-in-Aid for Scientific Research C (2) (No.1550043).

The authors would like to thank Tetsuya Seiyama, Akane Yamasaki, and Yoshitaka Hayashi for implementing the proposed method.

References


Yuzo Takamatsu received the B.E. degree in Electrical Engineering from Ehime University, Japan, in 1966, and the Ph.D. degree from Osaka University, Japan, in 1976. From 1967 to 1987, he was on the faculty of Science and Engineering at Saga University, Japan. From 1975 to 1987 he was an Associate Professor in the Department of Electronic Engineering, Saga University. Since 1987 he has been a Professor in the Department of Computer Science, Ehime University. Currently he is also the dean of the Faculty of Engineering, Ehime University. His research interests include design for testability, and fault diagnosis. Dr. Takamatsu is a senior member of the IEEE and a member of IPS Japan. He was the Technical Program Chair of the IEEE Third Asian Test Symposium (ATS), 1994, the General Chair of ATS, 1997, and the Vice General Chair of ATS, 2002. He received the Certificate of Appreciation Award from IEEE Computer Society in 1997. He received the IEICE Paper Award in 2005.

Hiroshi Takahashi received the B.E. and M.E. degrees in electronic engineering from Saga University, Saga, Japan in 1988 and 1990, respectively. He received the Dr. degree from Ehime University, Japan in 1996. From 1997 to 1999 he was a Lecturer in the Department of Computer Science, Ehime University. Since 2000, he has been an associate professor at Ehime University. From May 2001 to March 2001, he was a research fellow in University of Wisconsin-Madison, USA. His research interests are test generation and fault diagnosis for digital systems. Dr. Takahashi is a member of the IEEE and a member of IPS Japan. He served as the Program committee member of 2003, 2004, 2005, 2007 IEEE Asian Test Symposium and as the Program Vis-Chair of the 2008 IEEE Asian Test Symposium.

Yoshinobu Higami received his B.E., M.E., and D.E. degrees from Osaka University in 1991, 1993 and 1996, respectively. After serving as a research fellow of the Japan Society for the Promotion of Science and a visiting fellow at University of Wisconsin-Madison, he joined Department of Computer Science, Ehime University in 1998. Currently he is an associate professor at Graduate School of Science and Engineering, Ehime University. He received the IEICE Best Paper Award in 2005. His research interests include test generation, design for testability and fault diagnosis of logic circuits. He is a member of the IEEE and IPSJ.

Takashi Aiko received the B.E. degree in electronics from Shibaura Institute of Technology, in 1978, and M.E. degree in electronic and communications engineering from Waseda University, in 1980. He joined Fujitsu Limited in 1980, and he was working on development of computer-aided design tools in LSI testing area. He is currently a senior manager at the Semiconductor Technology Academic Reserch Center, and he is also a Ph.D. student at the Graduate School of Science and Engineering, Ehime University. His reserch interests include design for testability, and fault diagnosis. He is an IEEE Computer society member.

Koji Yamazaki received the B.S., M.S. and Ph.D degrees from Meiji University, Japan, in 1989, 1991 and 1994, respectively. In 1994, he joined the Department of Computer Science, Meiji University, Japan, as a lecturer. Since 2004 he has been an associated professor in the School of Information and Communication, Meiji University, Japan. His research interests are fault diagnosis and test generation. He is a member of IEEE.