IPSJ Transactions on System LSI Design Methodology Vol.9 13–20 (Feb. 2016)

[DOI: 10.2197/ipsjtsldm.9.13]

Regular Paper

Diagnosis Methods for Gate Delay Faults with Various Amounts of Delays

Yoshinobu Higami[1,a)] Senling Wang[1] Hiroshi Takahashi[1] Shin-ya Kobayashi[1]
Kewal K. Saluja[2]

Received: June 2, 2015, Revised: September 3, 2015,
Accepted: October 21, 2015

Abstract: For the purpose of analyzing the cause of delay in modern digital circuits, efficient diagnosis methods for delay faults need to be developed. This paper presents diagnosis methods for gate delay faults by using a fault dictionary approach. Although a fault dictionary is created by fault simulation and for a specific amount of delay, the proposed method using it can deduce candidate faults successfully even when the amount of delay in a circuit under diagnosis is different from that of the delay assumed during the fault simulation. In this paper, we target diagnosing the presence of single gate delay faults and double gate delay faults. Experimental results for benchmark circuits demonstrate the effectiveness of the proposed methods.

Keywords: fault diagnosis, gate delay faults, fault dictionary

1. Introduction

In modern LSIs, physical defects or manufacturing variations may cause timing failures. The faster the clock speed of LSIs is getting, the more important becoming are delay fault testing and diagnosis. Delay fault is a fault model that represents excessive signal propagation delay, and it is divided into a number of sub-models, such as transition fault, gate delay fault, path delay fault and segment delay fault. Diagnosis of various delay fault models has been studied actively so far. Path delay faults, which involve accumulated delays along a target path, represent real faulty phenomenon. However, it is less suitable for diagnosis, particularly for aiming to find a defect site which causes the delay [1]. Even if a diagnosis method deduces only one candidate path, it still contains a number of gates and signal lines, which will be analyzed by physical analysis after the logic diagnosis. Gate delay faults represent an excessive delay on the output of the target gate. Therefore, if a physical defect within a gate causes an excessive delay, such a defect is likely to be deduced by a diagnosis method for gate delay faults.

Previously a number of research papers on diagnosis for delay faults have been presented. In Ref. [3], critical path tracing and six-valued simulation approach was used. In Ref. [2], diagnosis is performed using adjacency tests, where only one input value made a transition. In Ref. [12], a critical path tracing and a timing reasoning approach were used to diagnose delay faults. The timing reasoning approach identifies invalid candidate faults for fault exclusion using timing information. In Ref. [11], a delay diagnosis method has been proposed based on timing simulation. The timing simulation calculates propagation of delay window, which represents the smallest and the largest possible delay size. The delay diagnosis method [6] enhanced the resolution by using passing patterns and estimated delay-defect-size. In Ref. [7], a multiple delay diagnosis method was proposed, where a single delay fault simulation is performed and failure logs obtained at slower-than-nominal clock frequencies are used.

In this paper, we propose diagnosis methods for single gate delay faults and double gate delay faults. The methods use a fault dictionary which contains output responses as well as the latest transition times for each candidate fault. Since we do not know exactly how large is the amount of delay in a circuit under diagnosis (CUD), the output responses are not always consistent between the CUD and the fault dictionary. Our methods introduce set of rules that express the conditions to be satisfied by candidate faults or the contradictions of the output responses between the CUD and non-candidate faults.

Main contributions of this paper are as follows.

(1) We propose diagnosis methods for delay faults using a fault dictionary which contains the information of output responses and the latest transition times for each candidate fault.
(2) The methods target single gate delay faults and double gate delay faults.
(3) The methods are applicable for arbitrary amounts of delays.

The method for single gate delay faults has been presented in Ref. [4], but the method for double gate delay faults has not been presented, and it is newly proposed in this paper. Although the paper [4] has also considered the presence of clock delays, this paper never considers it, and focuses on diagnosis for gate delay faults in the absence of clock delays. The rest of the paper is

[1] Graduate School of Science and Engineering, Ehime University, Matsuyama, Ehime 790–8577, Japan
[2] University of Wisconsin - Madison, Madison, WI 53706–1691, U.S.A.
[a)] higami@cs.ehime-u.ac.jp

Table 1 An example of a fault dictionary.

		Dictionary			
		f_1	f_2	f_3	CUD
t_1	FF_1	B 30	B 35	B 30	Fail
	FF_2	$\overline{B}$ 35	$\overline{B}$ 35	$\overline{B}$ 35	Fail
	PO_1	-	-	B 20	Pass
t_2	FF_1	-	B 30	B 25	Fail
	FF_2	-	-	-	Pass
	PO_1	B 25	B 35	-	Pass

Symbol '-' means no erroneous value is propagated.

organized as follows. Section 2 describes fault simulation used in our methods. Section 3 gives some definitions. Section 4 and Section 5 explain the diagnosis method for single delay faults and for double delay faults, respectively. Section 6 gives experimental results for ISCAS'89 benchmark circuits. Section 7 concludes this paper.

2. Fault Simulation

The proposed methods use a fault dictionary which contains not only logic values on primary outputs (POs) and flip-flops (FFs) but also information on the latest transition times for every pattern and every candidate fault. For the purpose of creating the fault dictionary, delay fault simulation with 7 logic values proposed in Ref. [8] is performed. In this simulation, 7 logic values (0, 1, X, R, F, B, $\overline{B}$) are used and the latest transition times are also calculated. The values 0, 1, X, R and F denote stable 0, stable 1, unknown value, a rising transition and a falling transition, respectively. (In the paper [8], symbols D and $\overline{D}$ were used instead of R and F, respectively.) The values B and $\overline{B}$ denote a rising transition and a falling transition that are propagated from a fault site, respectively. We use the unit delay model, where one unit time is added on signal delay whenever it passes through a gate. (Note that it is straightforward to extend the simulation into the general delay model.)

An example of a fault dictionary is shown in **Table 1**. Consider a circuit with two FFs and one PO, and suppose that two test patterns t_1 and t_2 are applied and faults f_1, f_2 and f_3 are candidate faults. When t_1 is applied in the presence of fault f_1, erroneous transitions (B and $\overline{B}$) are propagated to FF_1 and FF_2 with the latest transition time 30 and 35, respectively. This means transitions are propagated to FF_1 and FF_2 at 30 and 35 units time. If test cycle is less than 35, an erroneous value will be observed at FF_2, and if it is less than 30, an erroneous value will be also observed at FF_1 and FF_2. It is noted that no erroneous transition is propagated to PO_1 for t_1.

3. Definitions

In the following definitions, z denotes a PO or a FF, f denotes a gate delay fault, and t denotes a test pattern.

Definition 1 $Tr(t, z, f)$ is the latest transition time that is denoted with respect to test pattern t, output z and fault f in the fault dictionary.

Definition 2 An output z is called a **DP_output** when a *pass* response (i.e. neither B nor $\overline{B}$) is observed at z for f and t in the fault dictionary.

Definition 3 An output z is called a **DF_output** when a *fail* response (i.e., either B or $\overline{B}$) is observed at z for f and t in the fault dictionary.

Definition 4 An output z is called a **CP_output** when a *pass* response (i.e., no erroneous value) is observed at z for t in the responses of a CUD.

Definition 5 An output z is called a **CF_output** when a *fail* response (i.e., erroneous value) is observed at z for t in the response of a CUD.

Definition 6 Variable Min_Tr is defined as follows.

$$Min\_Tr(f) = \min\{Tr(t, z_{cf}, f)\} \text{ for } {}^{\forall}t \text{ and } {}^{\forall}z_{cf}, \quad (1)$$

where z_{cf} is a CF_output.

Definition 7 $Num\_CFDP(t, f)$ is the number of outputs that are a CF_output as well as a DP_output with respect to t and f.

Definition 8 A set of CF_outputs and corresponding test patterns is defined as follows. Z_{CF} is a set of 2-tuples (t, z) such that z is a CF_output for test pattern t.

Definition 9 A set of DF_outputs and corresponding test patterns is defined as follows. $Z_{DF}(f)$ is a set of 2-tuples (t, z) such that z is a DF_output for test pattern t with respect to fault f.

4. Diagnosis for Single Delay Faults

In this section, we explain a diagnosis method for single gate delay faults [4]. The method deduces candidate faults by comparing the output responses and the latest transition times between a CUD and a fault dictionary. Some rules are introduced to check whether a fault can be a candidate fault or not. While Rule 1 is based on a similar idea that is used in dictionary based diagnosis [5], [10], Rule 2 is based on our original idea.

Rule 1 If the following expression is satisfied for fault f, then f is removed from the set of candidate faults. Here t is a test pattern.

$$\sum_t Num\_CFDP(t, f) > 0 \quad (2)$$

Rule 1 is explained using the example of Table 1. Now suppose that a CUD produces the responses of (FF_1, FF_2, PO_1) = (Fail, Fail, Pass) for test pattern t_1, and (Fail, Pass, Pass) for test pattern t_2. Consider whether fault f_1 remains as a candidate fault or not. The fault dictionary shows that no erroneous value from fault f_1 is propagated to FF_1 for t_2. Since FF_1 is a CF_output as well as a DP_output with respect to f_1 and t_2, f_1 is removed from the set of candidate faults according to Rule 1. If f_1 with larger amount of delay exists in a CUD, then f_1 may produce a fail response at FF_1 for t_2. However, the preliminary experiments in Ref. [4] have demonstrated that such a case rarely happens. Therefore, if f_1 had existed in the CUD, no erroneous value would be observed at FF_1 for t_2 in most cases. The probability that f_1 exists in the CUD is very low, and f_1 cannot be a candidate fault.

Rule 2 If the following expression is satisfied, then f is removed from the set of candidate faults.

$$Min\_Tr(f) \leq Tr(t, z, f) \text{ for a CP\_output } z \quad (3)$$

Rule 2 checks the latest transition times to deduce candidate faults. Example of Table 1 is used again to explain Rule 2.

Diagnosis for single delay faults
/* F_{cand}: A set of candidate faults */
1 : **for** (every fail pattern t)
2 : **for** (every fault f in F_{cand})
3 : **for** (every output z)
4 : **if** (z is a CF_output and a DP_output)
5 : Remove f from F_{cand}; /* Rule 1 */
6 : **else if** (z is a CF_output)
7 : Update $Min\_Tr(f)$;
8 : **end for**
9 : **end for**
10 : **end for**
11 : **for** (every pattern t)
12 : **for** (every fault f in F_{cand})
13 : **for** (every output z)
14 : **if** (z is a CP_output and a DF_output, and
$Min\_Tr(f) \leq Tr(t, z, f)$)
15 : Remove f from F_{cand}; /* Rule 2 */
16 : **end for**
17 : **end for**
18 : **end for**

Fig. 1 Diagnosis flow for single gate delay faults.

Now consider the case of fault f_2. FF_1 is a CF_output for t_1 and t_2, and $Min\_Tr(f_2) = 30$. With respect to PO_1 and t_2, $Tr(t_2, PO_1, f_2) = 35$, and thus expression (3) is satisfied. Consequently, f_2 is removed from the set of candidate faults. This is due to the following observation. The fault dictionary indicates that erroneous value B is propagated to PO_1 later than B is propagated to FF_1. If f_2 had existed in the CUD, PO_1 would produce a fail response. Some readers may think that the above observation is true only when the amount of delay is same between the fault dictionary and the CUD. However, preliminary experiments in Ref. [4] show that expression (3) is satisfied in most cases even when the amount of delay is different between the fault dictionary and the CUD. With respect to fault f_3, since expressions (2) and (3) are not satisfied, it remains as a candidate fault. The overall flow of the diagnosis method is shown in **Fig. 1**.

5. Diagnosis for Double Delay Faults

In this section, a diagnosis method for double delay faults is explained, which is newly proposed. The method uses a fault dictionary which is created assuming every single delay fault, and it deduces some single delay faults as candidates. Similarly in the method for single delay faults, Rule 2 is applied to deduce candidate faults. On the other hand, Rule 1 is modified to apply. In the proposed method, if the number of outputs which are CF_outputs as well as DP_outputs is relatively large, then corresponding faults are removed from the set of candidate faults. This is because the probability that such faults exist in a CUD is low. Although the fault existing in the CUD may be removed from the set of candidate faults, the method aims to reduce the set of candidate faults as much as possible and to deduce at least one of the faults existing in the CUD as a candidate. In the following rule, N_{TH} is introduced which is a threshold value and it is predetermined.

Rule 3 If the following expression is satisfied for fault f, then f is removed from the candidate faults. Here t is a test pattern.

Table 2 The second example of a fault dictionary.

		Dictionary		
		f_4	f_5	CUD
t_1	FF_1	B 20	B 25	Fail
	FF_2	-	-	Fail
	PO_1	B 25	-	Pass
t_2	FF_1	B 25	-	Fail
	FF_2	$\overline{B}$ 30	-	Fail
	PO_1	B 30	B 30	Pass

Symbol '-' means no erroneous value is propagated.

Table 3 The third example of a fault dictionary.

		dictionary			
pattern	output	f_6	f_7	f_8	CUD
t_1	FF_1	B	-	-	Fail
	FF_2	B	-	-	Fail
	PO_1	-	B	-	Fail
t_2	FF_1	B	-	-	Fail
	FF_2	-	$\overline{B}$	$\overline{B}$	Fail
	PO_1	B	-	B	Fail

Symbol '-' means no erroneous value is propagated.

$$\sum_t Num\_CFDP(t, f) > N_{TH} \quad (4)$$

Rule 3 is explained using an example of **Table 2**. Suppose that the responses of a CUD are (FF_1, FF_2, PO_1) = (Fail, Fail, Pass) for test pattern t_1, and (Fail, Fail, Pass) for test pattern t_2. We calculate $\sum_{t\in\{t_1,t_2\}} Num\_CFDP(t, f_4)$ and $\sum_{t\in\{t_1,t_2\}} Num\_CFDP(t, f_5)$. In this case, $Num\_CFDP(t_1, f_4) = 1$ and $Num\_CFDP(t_2, f_4) = 0$. As a result, $\sum_{t\in\{t_1,t_2\}} Num\_CFDP(t, f_4) = 1$. From the similar calculation, we have $\sum_{t\in\{t_1,t_2\}} Num\_CFDP(t, f_5) = 3$. If N_{TH} is set to 2, then expression (4) for f_5 is satisfied and f_5 is removed from the set of candidate faults. Fault f_4 remains as a candidate. Some readers may think that fault f_5 should remain as candidate because the combination of f_4 and f_5 explains the output responses of the CUD. Our method aims to make the number of final candidate faults as small as possible, and it is considered successful that either one of the double faults existing in the CUD remains as a final candidate fault. Therefore, Rule 3 removes the faults for which $\sum_t Num\_CFDP(t, f)$ is large.

In this paper, we do not propose a deterministic algorithm for determining optimum N_{TH}. N_{TH} is investigated in the experiments to determine an optimum or a near optimum value. In the later section, we will investigate the number of fail outputs in fault dictionaries and in CUDs, and discuss a method for determining optimum N_{TH} from these numbers.

After reducing candidate faults by Rule 2 and Rule 3, another rule is applied to reduce candidate faults further. The rule checks whether any combination of single candidate faults explains the output responses in the CUD.

Rule 4 If the following expression is satisfied, then fault f is removed from the candidate faults, where F_{cand} is a candidate fault set, and f and g are single delay faults.

$$Z_{CF} \not\subseteq Z_{DF}(f) \cup Z_{DF}(g) \text{ for } \forall g \in F_{cand}\ (f \neq g) \quad (5)$$

Now suppose that a fault dictionary shown in **Table 3** is given, and that FF_1, FF_2 and PO_1 produce fail responses for test patterns t_1 and t_2. Consider the case of faults f_6 and f_7. In this case,

Z_{CF}
$= \{(t_1, FF_1), (t_1, FF_2), (t_1, PO_1), (t_2, FF_1), (t_2, FF_2), (t_2, PO_1)\}$
$Z_{DF}(f_6) = \{(t_1, FF_1), (t_1, FF_2), (t_2, FF_1), (t_2, PO_1)\}$
$Z_{DF}(f_7) = \{(t_1, PO_1), (t_2, FF_2)\}$

are obtained. Therefore,

$$Z_{CF} \subseteq Z_{DF}(f_6) \cup Z_{DF}(f_7)$$

is satisfied, and fault f_6 and f_7 remain as candidates.

Next, consider the case of fault f_8. In this case,

$$Z_{DF}(f_8) = \{(t_2, FF_2), (t_2, PO_1)\}$$

is obtained.

$$Z_{CF} \not\subseteq Z_{DF}(f_8) \cup Z_{DF}(f_6)$$
$$Z_{CF} \not\subseteq Z_{DF}(f_8) \cup Z_{DF}(f_7)$$

is satisfied, and thus f_8 is removed from the candidate fault set. Although the combination of f_6 and f_7 explains the output responses of the CUD, any combinations of f_8 do not explain them sufficiently. As a result, only f_6 and f_7 remain as candidates.

The flow of the diagnosis method for double delay faults is shown in **Fig. 2**. Although Rule 4 works effectively in most cases, in few cases all the candidate faults are removed by the rule. Therefore, when the final candidate fault set becomes empty by Rule 4, all the candidate faults removed by Rule 4 are recovered.

Similar ideas to Rule 3 and Rule 4 have been introduced in the diagnosis method presented in Ref. [9], where incomplete intersection between CUD responses and simulation results has been considered and a covering problem has been considered for fail responses by multiple faults. The diagnosis method for multiple gate delay faults that considers latest transition times on signal values by Rule 2 has not been proposed so far.

Diagnosis flow of double delay faults

```
/* F_cand: A set of candidate faults */
1 : for ( every fail pattern t )
2 :   for ( every fault f in F_cand )
3 :     for ( every output z )
4 :       if ( z is a CF_output and a DP_output )
5 :         Increment Num_CFDP(t, f);
6 :         if ( Num_CFDP(t, f) > N_TH )
7 :           Remove f from F_cand; /* Rule 3 */
8 :       else if ( z is a CF_output )
9 :         Update Min_Tr(f);
10 :    end for
11 :  end for
12 : end for
13 : for ( every pattern t )
14 :  for ( every fault f in F_cand )
15 :    for ( every output z )
16 :      if ( z is a CP_output and a DF_output, and
             Min_Tr(f) ≤ Tr(t, z, f) )
17 :        Remove f from F_cand; /* Rule 2 */
18 :    end for
19 :  end for
20 : end for
21 : F_tmp = F_cand;
22 : for ( every delay fault f_i in F_cand )
23 :  for ( every delay fault f_j ( f_i ≠ f_j ) in F_cand )
24 :    if ( Expression (5) is unsatisfied )
25 :      Go to Line 22;
26 :  end for
27 :  Remove f_i from F_cand; /* Rule 4 */
28 : end for
29 : if ( F_cand = φ )
30 :  F_cand = F_tmp;
```

Fig. 2 Diagnosis flow for double delay faults.

6. Experimental Results

6.1 Diagnosis for Single Delay Faults

We carried out experiments for ISCAS'89 benchmark circuits in order to confirm the effectiveness of the proposed method. As test patterns, we used transition fault test patterns, which were generated by our in-house tool. **Table 4** shows fault coverage by the test patterns, where column "patterns" and "faults" show the number of test patterns, the number of target faults, respectively. Columns under "fault coverage" show fault coverage when the amounts of delay are 30%, 50%, 80% and 100% for the test cycle. The test cycle is one unit time plus unit time equivalent to the number of gates along the longest sensitized path.

Table 5 shows results for s9234 and s13207 circuits with various delay amounts. We generated 100 CUDs for each benchmark circuit by inserting one fault randomly selected among the single delay faults which have two or more fail patterns. The table shows, from left to right, circuit name, the number of CUDs which have two or more fail patterns, the average number of fail patterns, the average number of final candidate faults, the maximum number of final candidate faults, the number of CUDs for which the number of final candidate faults was one, the number of CUDs for which the final candidate faults set did not include a fault existing in the CUD, the amount of delay of injected faults

Table 4 Fault coverage by the transition faults test sets.

Circuit	patterns	faults	fault coverage (%)			
			30	50	80	100
s9234	194	4,054	7.3	16.5	78.0	80.5
s13207	127	5,146	12.9	16.4	54.3	80.4
s15850	144	6,896	3.1	12.6	60.3	73.1
s35932	54	24,408	58.7	59.3	66.5	86.6
s38417	139	17,418	27.8	56.2	97.0	97.2
s38584	258	22,896	2.2	2.8	34.4	86.3

Table 5 Experimental results for single delay faults with various delay amounts.

Circuit	CUD	pat	cand	max	sgl	not	delay	
							CUD	dict
s9234	100	5.9	5.4	18	14	0	30	30
s9234	100	5.9	5.4	18	14	0	30	50
s9234	100	5.9	5.4	18	14	0	30	80
s9234	100	7.5	2.2	9	33	11	50	30
s9234	100	7.5	2.4	9	37	0	50	50
s9234	100	7.5	2.4	9	37	0	50	80
s9234	100	12.6	2.2	18	54	3	80	30
s9234	100	12.6	2.3	18	55	1	80	50
s9234	100	12.6	2.3	18	54	0	80	80
s13207	100	7.0	5.4	31	35	0	30	30
s13207	100	7.0	5.4	31	35	0	30	50
s13207	100	7.0	5.4	31	35	0	30	80
s13207	100	9.6	1.6	6	61	0	50	30
s13207	100	9.6	1.6	6	61	0	50	50
s13207	100	9.6	1.6	6	61	0	50	80
s13207	100	11.5	2.4	8	46	0	80	30
s13207	100	11.5	2.4	8	45	0	80	50
s13207	100	11.5	2.4	8	45	0	80	80

and the amount of delay in the fault simulation for creating a fault dictionary, respectively. The amount of delay is represented in percentage for the test cycle. It is found that similar or same diagnosis results were obtained even when the amount of delay is different in creating a fault dictionary.

Table 6 shows experimental results for other benchmark circuits by the diagnosis method for single delay faults. In these experiments, 50% delay was used to create fault dictionaries. In the table, each column has the same meaning except for column "[11]", which denotes the number of candidate faults reported in Ref. [11]. In the experiments reported in Ref. [11], random size of delays were injected. It is noted that for s15850 only 74 faults which have 30% delay amount were detected by two or more transition test patterns. The results show that the average numbers of candidate faults were less than 7 for every circuit, and that they are smaller than those in Ref. [11]. We have not found any other papers that have the same purpose as this paper and that provide experimental results on ISCAS'89 benchmark circuits. The paper [11] presented the best results for diagnosing single delay faults and double delay faults. This is the reason we compare our results with those by the paper [11].

Next we selected single delay faults that are detected by only one test pattern, and carried out the experiments. **Table 7** shows the results for single delay faults with only one fail pattern. Each column of the table has the same meaning as Table 6. It is found that the numbers of final candidate faults in Table 7 were larger than those in Table 6. In particular, the numbers of final candidate faults for 30% delay were large. This is because faulty effects were propagated to limited outputs when 30% delays were injected.

In order to improve the diagnosis results for single delay faults with one fail pattern, we carried out additional experiments where fail test patterns were found among 10,000 random patterns and they are added as diagnosis patterns. It is noted that at most 20 fail patterns were added even when more fail patterns were found. **Table 8** shows the results. Each column in the table has same meaning as in Table 6 except for column "cand1" which denotes the number of candidate faults obtained by using only one fail pattern. It is found that the numbers of final candidate faults were increased. We, therefore, conclude that the proposed method is effective for faults which have more than one or several fail patterns.

Table 9 shows the number of candidate faults in order to see how each rule is contributed. In Table 9, column "Rule 1" and "Rule 1 & 2" denote the numbers of candidate faults after Rule 1 is applied and after Rule 1 and Rule 2 are applied, respectively. Column "delay" denotes the percentage of the amount of delay which is injected. From the results it is found that candidate faults were mainly reduced by Rule 1.

Table 10 shows the size of the fault dictionary and the computing time. Column "size", "dictionary" and "diagnosis" denote the size of the fault dictionaries, the computing time for creating a fault dictionary and the average computing time for diagnosing one CUD with a single delay fault. The program was run on Intel Xeon 3.5 GHz processor with 32 GB memory.

Table 6 Experimental results for single delay faults.

Circuit	CUD	pat	cand	max	sgl	not	delay	[11]
s9234	100	5.9	5.4	18	14	0	30	
s9234	100	7.5	2.4	9	37	0	50	7.2
s9234	100	12.6	2.3	18	55	1	80	
s13207	100	7.0	5.4	31	35	0	30	
s13207	100	9.6	1.6	6	61	0	50	7.8
s13207	100	11.5	2.4	8	45	0	80	
s15850	74	2.9	6.8	13	5	0	30	
s15850	100	6.5	2.6	9	39	0	50	NA
s15850	100	11.4	2.3	10	44	0	80	
s35932	100	6.4	2.6	10	24	0	30	
s35932	100	6.2	3.4	17	26	0	50	8.8
s35932	100	7.5	2.4	12	30	0	80	
s38417	100	7.4	3.1	15	36	0	30	
s38417	100	12.3	1.8	14	54	0	50	9.2
s38417	100	14.5	1.3	6	78	3	80	
s38584	100	11.1	2.3	10	50	0	30	
s38584	100	15.7	1.9	11	58	0	50	7.2
s38584	100	10.9	2.9	6	33	0	80	

Table 7 Experimental results for single delay faults with only one fail patterns.

Circuit	CUD	pat	cand	max	sgl	not	delay
s9234	100	1.0	12.2	29	8	0	30
s9234	100	1.0	4.8	29	26	0	50
s9234	100	1.0	3.4	11	30	0	80
s13207	100	1.0	16.3	56	3	0	30
s13207	100	1.0	4.2	23	23	0	50
s13207	100	1.0	4.0	17	20	0	80
s15850	100	1.0	27.4	81	3	0	30
s15850	100	1.0	7.2	81	10	0	50
s15850	100	1.0	4.4	19	26	0	80
s35932	100	1.0	6.3	17	17	0	30
s35932	100	1.0	7.0	17	15	0	50
s35932	100	1.0	6.8	17	30	0	80
s38417	100	1.0	10.6	58	9	0	30
s38417	100	1.0	3.9	23	30	0	50
s38417	100	1.0	3.5	23	35	0	80
s38584	90	1.0	5.5	18	17	0	30
s38584	100	1.0	2.2	8	31	0	50
s38584	100	1.0	3.8	14	16	0	80

Table 8 Experimental results for single delay faults with only one fail patterns by adding test patterns.

Circuit	CUD	pat	cand1	cand	max	sgl	not	delay
s9234	28	7.8	7.9	2.7	6	6	0	30
s9234	30	15.9	1.9	1.5	4	20	0	50
s9234	73	10.3	3.6	2.7	9	29	0	80
s13207	74	6.5	17.3	10.4	56	13	0	30
s13207	76	6.1	3.4	2.5	6	29	0	50
s13207	83	11.1	4.0	3.1	9	21	0	80
s15850	9	2.4	35.9	5.4	11	0	0	30
s15850	62	10.7	8.3	3.4	11	10	0	50
s15850	78	12.6	4.1	2.4	8	34	0	80
s35932	100	21.0	6.3	2.6	11	37	0	30
s35932	100	21.0	7.0	2.6	5	39	0	50
s35932	100	21.0	6.8	2.6	5	45	0	80
s38417	71	18.0	9.9	4.0	14	33	0	30
s38417	58	16.8	3.8	2.0	12	36	0	50
s38417	56	17.6	2.3	1.4	6	41	0	80
s38584	33	2.9	5.2	2.1	7	18	0	30
s38584	50	7.0	2.4	2.2	7	24	0	50
s38584	63	14.0	3.7	3.1	6	15	0	80

Table 9 Candidate faults by Rule1 and Rule 2.

Circuit	Rule 1	Rule 1 & 2	delay
s9234	9.8	2.4	50
s13207	4.5	1.6	50
s15850	7.6	2.6	50
s35932	5.5	3.4	50
s38417	4.6	1.8	50
s38584	3.6	1.9	50

Table 10 Results on computing time and dictionary size.

Circuit	size (Mbyte)	time(s)	
		dictionary	diagnosis
s9234	1.50	2.55	0.09
s13207	2.10	3.62	0.11
s15850	2.80	6.33	0.16
s35932	6.98	27.56	0.61
s38417	12.01	52.31	0.56
s38584	16.15	127.64	0.61

Table 11 Experimental results for double delay faults for s9234 and s13207.

Circuit	CUD	pat	cand	both	sgl	not	delay	N_{TH}
s9234	100	18.9	8.9	91	8	1	50	10
s9234	100	18.9	7.2	94	5	1	50	20
s9234	100	18.9	7.2	94	5	1	50	30
s9234	100	18.9	7.2	94	5	1	50	40
s9234	100	18.9	7.2	94	5	1	50	50
s13207	100	19.2	6.7	88	3	9	50	10
s13207	100	19.2	6.4	89	4	7	50	20
s13207	100	19.2	6.3	91	2	7	50	30
s13207	100	19.2	6.3	91	2	7	50	40
s13207	100	19.2	6.3	91	2	7	50	50

Table 12 Experimental results for double delay faults for s15850, s35932, s38417 and s38584.

Circuit	CUD	pat	cand	both	sgl	not	delay	N_{TH}
s15850	100	13.4	42.8	86	10	4	50	10
s15850	100	13.4	44.1	86	10	4	50	30
s15850	100	13.4	44.1	86	10	4	50	50
s35932	100	11.7	4.4	55	36	9	50	10
s35932	100	11.7	4.4	55	36	9	50	30
s35932	100	11.7	4.4	55	36	9	50	50
s38417	100	21.1	5.2	92	8	0	50	10
s38417	100	21.1	4.2	97	3	0	50	30
s38417	100	21.1	4.2	97	3	0	50	50
s38584	100	26.6	6.7	84	11	5	50	50
s38584	100	26.6	6.2	87	8	5	50	80
s38584	100	26.6	6.2	87	8	5	50	100

6.2 Diagnosis for Double Delay Faults

In this subsection, we show experimental results for double delay faults. Two delay faults were randomly selected among the detected single delay faults to form double delay faults which were injected in CUDs. One hundred CUDs for each benchmark circuit were created for diagnosis experiments. First, we carried out experiments for s9234 and s13207 benchmark circuits to find optimal value of N_{TH}, which is used in Rule 3. **Table 11** shows the results with several values of N_{TH}. In the table, from left to right, circuit name, the number of CUDs, the average number of fail patterns, the average number of final candidate faults, the number of CUDs for which the final candidate faults included both faults existing in the CUD, the number of CUDs for which the final candidate faults set included either one fault existing in the CUD, the number of CUDs for which the final candidate faults set included neither faults existing in the CUD, the amount of delay and value of N_{TH} are shown. When $N_{TH} = 10$, the number of CUDs for which the final candidate faults set included neither faults existing in the CUD was a little large. With $N_{TH} = 30$ and more, no difference in the results was found. **Table 12** shows the results for s15850, s35932 and s38417 when $N_{TH} = 10, 30$ and 50, and the results for s38584 when $N_{TH} = 50, 80$ and 100. Each column has the same meanings as in Table 11. For s15850, s38584 and the other circuits, the smallest number of candidate faults were obtained when $N_{TH} = 10, 80$ and 30, respectively. Therefore, we set $N_{TH} = 10, 80$ and 30 for s15850, s38584 and the other circuits, respectively, in the following experiments.

Table 13 Experimental results for double delay faults.

Circuit	CUD	pat	s-cand	cand	both	sgl	not	delay
s9234	100	6.0	4.9	20.9	38	52	10	30
s9234	100	18.9	0.1	7.2	94	5	1	50
s9234	100	31.2	0.1	5.4	99	1	0	80
s13207	99	15.0	0.9	38.2	58	32	9	30
s13207	100	19.2	0.1	6.3	91	2	7	50
s13207	100	24.3	0	5.2	96	1	3	80
s15850	100	3.0	15.5	42.2	6	68	26	30
s15850	100	13.4	0.1	42.8	86	10	4	50
s15850	100	16.4	0.1	44.2	90	7	3	80
s35932	100	10.2	0.1	5.0	53	38	9	30
s35932	100	11.7	0	4.4	55	36	9	50
s35932	100	13.8	0	4.5	56	35	9	80
s38417	100	11.5	1.4	17.4	51	48	1	30
s38417	100	21.1	0	4.2	97	3	0	50
s38417	100	34.9	0	4.1	95	5	0	80
s38584	100	23.1	0.3	15.7	63	26	11	30
s38584	100	26.6	0	6.2	87	8	5	50
s38584	100	29.3	0	5.9	90	5	5	80

Table 14 Comparison of the results.

Circuit	Our method			[11]	
	cand	DA	delay	cand	DA
s9234	20.9	0.64	30		
s9234	7.2	0.97	50	27.0	0.90
s9234	5.4	1.00	80		
s13207	38.2	0.75	30		
s13207	6.3	0.92	50	39.0	0.92
s13207	5.2	0.97	80		
s15850	42.2	0.40	30		
s15850	42.8	0.91	50	NA	NA
s15850	44.2	0.94	80		
s35932	5.0	0.72	30		
s35932	4.4	0.73	50	30.0	0.92
s35932	4.5	0.74	80		
s38417	17.4	0.75	30		
s38417	4.2	0.99	50	25.0	0.91
s38417	4.1	0.98	80		
s38584	15.7	0.76	30		
s38584	6.2	0.91	50	23.0	0.92
s38584	5.9	0.93	80		

Table 13 shows the results by the diagnosis method for double delay faults. Each column has the same meaning as in Table 11 except for column "s-cand", which denotes the average number of candidate faults which were obtained by the diagnosis method for single delay faults. It is found that the diagnosis method for single delay faults deduced zero candidate faults for most of the CUDs. This fact implies the diagnosis method for single delay faults is ineffective when double delay faults occur in a CUD. The average number of candidate faults were small when the amount of delay was 50% and 80% of the clock cycle except for s15850. It is noted that for s13207, 100 CUDs were created, but no fail patterns existed in one CUD.

Table 14 compares the results by our method with those reported in Ref. [11]. Column "DA" denotes diagnosability, which is defined as a ratio of the number of faults correctly deduced over the total number of injected faults. It is found that the numbers of candidate faults by our method were smaller than those presented in Ref. [11] for most of the circuits. In terms of the diagnosability, our method achieved higher diagnosability for many circuits than the paper [11].

Table 15 shows the number of candidate faults in order to see how each rule is contributed. In Table 15, column "Rule3",

Table 15 Candidate faults by Rule 2, Rule3 and Rule 4.

Circuit	Rule3	Rule2 & 3	Rule2 & 3 & 4	delay
s9234	1359.1	39.1	7.2	50
s13207	2411.7	50.4	6.3	50
s15850	2506.6	54.8	42.8	50
s35932	17904.5	23.2	4.4	50
s38417	11459.8	38.9	4.2	50
s38584	12227.1	19.8	6.2	50

Table 16 Computing time for diagnosis for double delay faults.

Circuit	time (s)
s9234	0.11
s13207	0.17
s15850	0.22
s35932	1.06
s38417	0.79
s38584	1.17

Table 17 Results on the number of fail outputs in the fault dictionaries.

Circuit	total_bits	ave	min	max	std
s9234	48500	16.7	1	322	28.5
s13207	100330	17.6	1	474	25.7
s15850	98496	20.9	1	1687	49.2
s35932	110592	9.0	1	36	5.4
s38417	242138	28.2	1	1421	43.4
s38584	446340	24.3	1	639	31.3

Table 18 Results on the number of fail outputs in CUDs.

Circuit	total_bits	ave	min	max	std
s9234	48500	34.6	4	136	77.4
s13207	100330	45.1	6	173	79.9
s15850	98496	23.5	4	64	13.5
s35932	110592	13.4	6	37	0.4
s38417	242138	62.8	6	397	50.8
s38584	446340	71.6	11	233	97.4

"Rule2 & 3" and "Rule2 & 3 & 4" show the the numbers of candidate faults after Rule 3 is applied, after Rule 2 and Rule 3 are applied and after Rule2, Rule3 and Rule 4 are applied, respectively. From the results, it is found that the number of candidate faults were mainly reduced by Rule 2. **Table 16** shows the average computing time for diagnosing one CUD with a double delay fault.

Additional experimental results are shown in order to see whether the number of fail outputs in the fault dictionaries or in the CUD responses is useful for finding optimum N_{TH}. **Table 17** shows results on the number of fail outputs in the fault dictionaries. Column "total_bits" denotes the product of the number of test patterns and the number of outputs, which is sum of the number of primary outputs and the number of FFs. Column "ave", "min" and "max" denote the average number, the maximum number and the minimum number of fail outputs for one candidate fault in the fault dictionaries, respectively. Column "std" denotes the standard deviation of the number of fail outputs for one candidate faults in the fault dictionaries. **Table 18** shows results on the number of fail outputs in the CUDs. Column "total_bits" denotes the same number as in Table 17. Column "ave", "min" and "max" denote the average number, the maximum number and the minimum number of fail outputs for one CUD, respectively. Column "std" denotes the standard deviation of the number of fail outputs for one CUD. In most of the circuits, there is a wide range of the number of fail outputs for each candidate fault and for each CUD. We investigated the relations between the number of fail outputs, N_{TH}, and the number of final candidate faults, but we did not find any meaningful relations between them. Therefore we do not think that it is easy to obtain optimum N_{TH} by investigating the number of fail outputs in the fault dictionaries or in the CUD responses.

7. Conclusions

In this paper, we proposed diagnosis methods for single delay faults and double delay faults. The methods use a fault dictionary that is created using seven logic value fault simulation where output responses and the latest transition times are calculated. In order to deduce candidate faults, certain rules are introduced which describe conditions for inclusion and exclusion of faults in the candidate faults. The experimental results for benchmark circuits demonstrated the effectiveness of the proposed methods. In our future work, we will develop a diagnostic test generation method with the objective to reduce candidate faults further. Also we will develop efficient method(s) for determining optimal N_{TH}. Moreover, the extension of the proposed methods to diagnosis for faults with more multiplicity remains as a future work. The proposed methods have not applied any circuit structure analysis like cone intersection pruning [10]. We will investigate how effective introducing such technique to the proposed methods is.

References

[1] Chen, Y., Kuo, M. and Liou, J.: Diagnosis Framework for Locating Failed Segments of Path Delay Faults, *Proc. Int. Test Conf.*, pp.1–8 (2005).

[2] Dastidar, J.G. and Touba, N.A.: Adaptive Techniques for Improving Delay Fault Diagnosis, *Proc. VLSI Test Symp.*, pp.168–172 (1999).

[3] Girard, P., Landrault, C. and Pravossoudovitch, S.: A Novel Approach to Delay-Fault Diagnosis, *Proc. Design Automation Conf.*, pp.357–360 (1992).

[4] Higami, Y., Takahashi, H., Kobayashi, S. and Saluja, K.K.: Diagnosis of Gate Delay Faults in the Presence of Clock Delay Faults, *IEEE Computer Society Annual Symp. on VLSI*, pp.320–325 (2014).

[5] Jha, N. and Gupta, S.: *Testing of Digital Systems*, Cambridge University Press (2003).

[6] Mehta, V., Marek-Sadowska, M., Tsai, K.-H. and Rajski, J.: Improving the Resolution of Single-Delay-Fault Diagnosis, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.27, pp.932–945 (2008).

[7] Mehta, V., Marek-Sadowska, M., Tsai, K.-H. and Rajski, J.: Timing-Aware Multiple-Delay-Fault Diagnosis, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.28, pp.245–258 (2009).

[8] Takahashi, H., Watanabe, T. and Takamatsu, Y.: Generation of Tenacious Tests for Small Gate Delay Faults in Combinational Circuits, *Proc. Asian Test Symp.*, pp.332–338 (1995).

[9] Venkataraman, S. and Drummonds, S.: POIROT: A Logic Fault Diagnosis Tool and Its Applications, *Proc. Int. Test Conf.*, pp.253–262 (2000).

[10] Wang, L.-T., Wu, C.-W. and Wen ed., X.: *VLSI Test Principles and Architectures*, Morgan Kaufmann Publishers (2006).

[11] Wang, Z., Marek-Sadowska, M.M., Tsai, K.-H. and Rajski, J.: Delay-Fault Diagnosis Using Timing Information, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.24, pp.1315–1325 (2005).

[12] Yang, K. and Cheng, K.-T.: Timing-Reasoning-Based Delay Fault Diagnosis, *Proc. Design Automation and Test in Europe*, pp.1–6 (2006).

Yoshinobu Higami received his B.E., M.E., and D.E. degrees from Osaka University in 1991, 1993 and 1996, respectively. Currently he is an associate professor at Graduate School of Science and Engineering, Ehime University. In 1998 and 2006, he was also an honorary fellow at University of Wisconsin-Madison, U.S.A. He received the IEICE Best Paper Award in 2005 and 2012. His research interests include test generation, design for testability and fault diagnosis of logic circuits. He is a senior member of the IEEE and a member of IEICE.

Senling Wang received his B.E. degree in the college of Electrical and information Engineering from Beihua University, China, in 2008, and the M.E. and the Ph.D. degree from the Department of Computer Science and Electronics of Kyushu Institute of Technology, Japan, in 2011 and 2014, respectively. Currently, he serves as an Assistant Professor in Ehime University, Japan. His research interest includes Field testing, Low power testing, Delay testing, Fault Diagnosis for Digital Systems, and Design for Testability. He is a member of the IEEE.

Hiroshi Takahashi received his Dr. degree from Ehime University, Japan in 1996. Since 2010, he has been a Full Professor at Ehime University. From May 2000 to March 2001, he was a research fellow at the University of Wisconsin-Madison, USA. He received the IEICE Best Paper Award in 2012. His research interests are test generation and fault diagnosis for digital systems. Dr. Takahashi is a senior member of the IEEE and IEICE. He served as the Program Chair of the 2012 IEEE Asian Test Symposium. He also served as the General Co-Chair of the 2016 IEEE Asian Test Symposium.

Shin-ya Kobayashi received his B.E. degree, M.E. degree, and Dr.E. degree in Communication Engineering from Osaka University in 1985, 1988, and 1991 respectively. He is a Professor at Graduate School of Science and Engineering, Ehime University. His research interests include distributed processing, and parallel processing. He is a senior member of the Information Processing Society of Japan, and a member of the Institute of Electrical Engineers of Japan, IEEE, and ACM.

Kewal K. Saluja obtained his Bachelor of Engineering (B.E.) degree in Electrical Engineering from the University of Roorkee (now IIT-Roorkee), India in 1967, M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Iowa, Iowa City in 1972 and 1973 respectively. He is at present an Emeritus Professor with the Department of Electrical and Computer Engineering at the University of Wisconsin-Madison. Prior to being Emeritus, as a Professor at University of Wisconsin-Madison from 1986 to 2015, he taught courses in logic design, computer architecture, microprocessor based systems, VLSI design and testing, and fault-tolerant computing. Before that he was at the University of Newcastle, Australia. Professor Saluja has held visiting and consulting positions at various national and international institutions including University of Southern California, Hiroshima University, Nara Institute of Science and Technology, the University of Roorkee, and Ehime University. He has also served as a consultant to the United Nations Development Program. He was the general chair of the 29th FTCS and he served as an Editor of the IEEE Transactions on Computers (1997–2001). He is currently the Associate Editor for the letters section of the Journal of Electronic Testing: Theory and Applications (JETTA). His research focus is in the areas of Digital Systems Testing, Fault-Tolerant Computing, and Sensor Networks. Professor Saluja has authored and co-authored over 300 technical papers that have appeared in conference proceedings and journals. Professor Saluja is a member of Eta Kappa Nu, Tau Beta Pi, a fellow of the JSPS and a Fellow of the IEEE.

(Recommended by Associate Editor: *Takeshi Matsumoto*)