LETTER

# A Method for Diagnosing Bridging Fault between a Gate Signal Line and a Clock Line

**Yoshinobu HIGAMI**[†a)], **Senling WANG**[†], ***Members***, **Hiroshi TAKAHASHI**[†], ***Senior Member***, **Shin-ya KOBAYASHI**[†], ***Member***, ***and*** **Kewal K. SALUJA**[††], ***Nonmember***

**SUMMARY** In this paper, we propose a method to diagnose a bridging fault between a clock line and a gate signal line. Assuming that scan based flush tests are applied, we perform fault simulation to deduce candidate faults. By analyzing fault behavior, it is revealed that faulty clock waveforms depend on the timing of the signal transition on a gate signal line which is bridged. In the fault simulation, a backward sensitized path tracing approach is introduced to calculate the timing of signal transitions. Experimental results show that the proposed method deduces candidate faults more accurately than our previous method.
***key words:*** *fault diagnosis, bridging faults, clock lines*

## 1. Introduction

Bridge is one of the defects that is most likely to occur, and thus an effective diagnosis method for bridging faults is needed. Although many methods for diagnosing bridging faults were proposed previously, few of them focused on bridging faults at clock lines. As the scale of VLSIs becomes large, the number of flip-flops (FFs) also increases and the clock line network becomes complicated. Therefore a diagnosis method for bridging faults on clock lines needs to be developed.

In this paper, we propose a diagnosis method for bridging faults between a gate signal line and a clock line. We analyze fault behavior of such bridging faults, and develop a fault simulator. The diagnosis is performed using the fault simulator to deduce candidate faults. In our previous paper [1], we proposed a diagnosis method for bridging faults on clock lines, where it is revealed that the faulty value of a clock line depends on not only the value of a gate signal line which is bridged but also the timing of the signal transition. However, the timing of signal transitions were determined by static information based on the circuit level, which shows how far the gate signal line is from primary inputs (PIs) and FFs. In this paper, the proposed method calculates the timing of a signal transition by a backward sensitized path tracing approach. Along sensitized paths, signal transitions were traced to obtain the signal transition arrival time at a target gate signal line. Experimental results for benchmark circuits show that the proposed method can deduce candidate faults more accurately than our previous method.

The rest of the paper is organized as follows. Section 2 describes a fault behavior of a bridging fault which is considered in our method. Section 3 explains the proposed diagnosis method. Section 4 gives experimental results for ISCAS'89 benchmark circuits. Section 5 concludes this paper.

Manuscript received October 20, 2016.
Manuscript revised April 3, 2017.
Manuscript publicized June 12, 2017.
[†]The authors are with Graduate School of Science and Engineering, Ehime University, Matsuyama-shi, 790–8577 Japan.
[††]The author is with the University of Wisconsin - Madison, Madison, WI, 53706–1691, U.S.A.
a) E-mail: higami@cs.ehime-u.ac.jp
DOI: 10.1587/transinf.2016EDL8210

## 2. Fault Behavior

In this section, we explain the behavior of a bridging fault between a gate signal line and a clock line [1]. First, we discuss the case of AND bridging faults while considering signal propagation delay. Figure 1 shows waveforms, where "clk" and "$v_g$" denote fault-free waveforms on a clock line and a gate signal line $g$, respectively, and "clk_g" denotes a faulty waveform when the two lines form an AND bridging fault. In the case of Fig. 1 (a), the propagation delay on $g$, referred to $d_g$, is assumed to be smaller than $W_p$, which is a half duration of the test cycle. Figure 1 (b) shows waveforms when $d_g$ is larger than $W_p$. We can see the difference on "clk_g" between Fig. 1 (a) and (b) depending on $d_g$. When a rising transition on $g$ occurs in $c_3$ cycle, a pulse still remains on "clk_g" in $c_3$ cycle in Fig. 1 (a), but the pulse disappears in $c_3$ cycle in Fig. 1 (b).

Below we summarize the behavior of an AND bridging fault, where $v_g$ denotes a signal value on gate signal line $g$, which is bridged, $d_g$ denotes the signal propagation delay on $g$, $W_p$ denotes a half duration of the test cycle, and $clk\_g$ denotes the faulty value of the two bridging lines.

[**AND bridging fault behavior**]

- **Case of** $d_g < W_p$: When $v_g$ takes static 0 at $c_i$ cycle, the positive edge disappears on the faulty value $clk\_g$ at $c_i$ cycle. When $v_g$ takes a rising transition at $c_i$ cycle, the positive edge on the faulty value $clk\_g$ still remains but delays at $c_i$ cycle.
- **Case of** $d_g \geq W_p$: When $v_g$ takes static 0 or a rising transition at $c_i$ cycle, the positive edge disappears on the faulty value $clk\_g$ at $c_i$ cycle.

Similar discussion arises for OR bridging faults, and their behavior is summarized as follows.

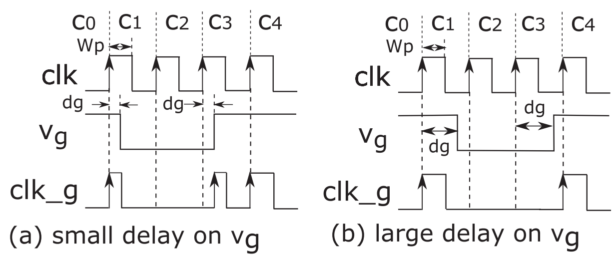


**Fig. 1** Effect by an AND bridging fault

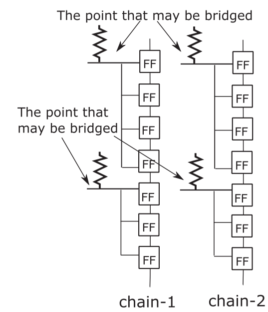


**Fig. 2** Circuit model

[**OR bridging fault behavior**]

- **Case of** $d_g \leq W_p$: When $v_g$ takes static 1 or a falling transition at $c_i$ cycle, the positive edge disappears on the faulty value *clk_g* at $c_i$ cycle.
- **Case of** $d_g > W_p$: When $v_g$ takes static 1 or a falling transition at $c_i$ cycle, the positive edge disappears on the faulty value *clk_g* at $c_i$ cycle. When $v_g$ takes a rising transition at $c_i$ cycle, the positive edge on the faulty value *clk_g* arrives earlier at $c_{i+1}$ cycle.

In the context of the above fault model, let us explain the difference between the flush test cycle and the system clock cycle. The fault behavior described in Sect. 2 underlines the relation between the test cycle $W_p$ and signal propagation delay $d_g$. Therefore, the proposed method is applicable when the amount of $d_g$ is calculated as above. When the amount of $d_g$ is extremely small compared with $W_p$, setup time and hold time of the FF need to be considered carefully. However, discussion of such a case is beyond the scope of this paper. Further, it is noted that accurate calculation of $d_g$ is also out of the scope of this paper.

## 3. Diagnosis Method

The overview of the proposed method is described below.

[**Proposed diagnosis method for bridging faults**]

*Inputs*:

- Test patterns
- Responses for CUD (Circuit Under Diagnosis)
- Initial candidate faults

*Outputs*:

- Deduced candidate faults

**Step 1:** Identify faulty scan chain
**Step 2:** Estimate propagation delay
**Step 3:** Fault simulation

As inputs of the method, test patterns, responses of a CUD and an initial candidate fault set are given. We assume that the initial candidate faults are extracted from layout data using a certain previously proposed method like [2]–[4]. Also a scan based flush test application technique is assumed to be used, where scan FFs are always in the scan shift mode, and test data are applied from the scan input and shifted out of the scan output [5]. FFs are assumed to be positive edge trigger types.

Next, we describe the circuit model. We assume a circuit has multiple scan chains, and a fanout branch of a clock line is bridged, which synchronizes a group of FFs existing on the same scan chain, as shown in Fig. 2. Since we assume the existence of a single bridging fault in a CUD, erroneous values are observed at only one scan chain output. A FF which is synchronized by a clock line with a bridging fault is called a faulty FF, and a scan chain on which faulty FFs exist is called a faulty scan chain. Therefore, the number of faulty scan chains is one, and by observing output responses at each scan chain, the faulty scan chain can be identified in a straight forward manner.

In **Step 1**, we identify a faulty scan chain among multiple scan chains. Only the responses scanned out from the faulty scan chain are compared with those of the CUD. In **Step 2**, for each gate signal line in the initial candidate fault set, we estimate propagation delay statically. We calculate maximum propagation delay (referred to *Max_delay*) and minimum propagation delay (referred to *Min_delay*), and classify the gate signal lines into three categories as follows. The static calculation of propagation delay is based on the number of gates which exists on the longest path and the shortest path from PIs and FFs to a target signal line.

[**Categories of signal lines**]

- **Signal line with large delay**: $Max\_delay \geq W_p$ and $Min\_delay \geq W_p$
- **Signal line with small delay**: $Max\_delay < W_p$ and $Min\_delay < W_p$
- **Signal line with variable delay**: Neither of the above conditions are satisfied

In **Step 3**, we perform fault simulation to obtain output responses in the faulty circuit where each candidate fault is injected. In the fault simulation, the value of the gate signal line which is bridged with a clock line needs to be calculated in order to check whether clock pulses disappear or not. When the clock pulses disappear due to the bridging fault, the affected FFs do not capture scan shift values. As explained in Sect. 2, faulty clock waveforms depend on the timing of signal transitions. Propagation delay on the signal

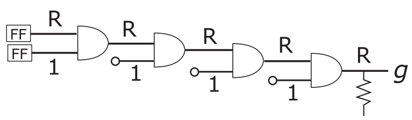


**Fig. 3** Calculation of propagation delay

lines with variable delay needs to be calculated. Note that the timing of a signal transition on such lines depends on the values of PIs and FFs. The proposed method calculates the propagation delay by a backward sensitized path tracing approach. A sensitized path is traced backward from the gate signal line of a bridging fault to PIs and FFs. For example, in the circuit shown in Fig. 3, it is found that the rising transition arrives at gate signal line $g$ with 4 units delay, where R denotes a rising transition. In our previous method [1], regardless of the values of PIs and FFs, the line $g$ is categorized as line with either large delay or small delay.

The details of the fault simulation for AND bridging faults are shown in Fig. 4. In this procedure, a test pattern means a set of values at PIs and FFs (pseudo-PIs). From line 16th to 31st, the propagation delay on gate signal line $g$, which is bridged, is calculated. Variable *delay_flag* is determined by the amount of the propagation delay on $g$. When *delay_flag* = SML (LRG), this means that the propagation delay on $g$ is smaller (larger) than $W_p$. From line 32nd to 42nd, it is checked whether a pulse signal on the faulty clock line disappears or not. If OR bridging faults are target, these lines should be replaced with the corresponding statements. Variable *capture_flag* is determined by the above calculation. When *capture_flag* = 0, the faulty clock pulse disappears and scan shift values are not captured on the FFs which are affected by the faulty clock line.

**Procedure: Fault simulation**

```
// g: a gate signal line of fault f
// v_g(p_i): the value on signal line g for i-th test pattern p_i
1: for each candidate fault f do
2:   Set 0 on all the FFs
3:   capture_flag = 1
4:   for each test pattern p_i do
5:     Apply p_i at primary inputs
6:     for j = 0 to j < scan_length do
7:       Scan shift by 1 bit on every scan chains
         and apply j-th bit value at every scan input
8:       if capture_flag = 0 then
           faulty FFs do not capture scan shift values
9:       end if
10:      Calculate value on the signal line g that forms
           the bridging fault f
11:      Compare the scan out values with those for the CUD
12:      if at least one bit of the scan-out values is different from
           that of the CUD then
13:        Exclude fault f from the candidate list
14:        go to Line 1
15:      end if
16:      switch ( category of gate signal line g )
17:        case Signal line with small delay:
18:          delay_flag = SML
19:        end case
20:        case Signal line with large delay:
21:          delay_flag = LRG
22:        end case
23:        case Signal line with variable delay:
24:          Calculate propagation delay by the backward
               sensitized path trancing
25:          if propagation delay ≥ W_p then
26:            delay_flag = LRG
27:          else then
28:            delay_flag = SML
29:          end if
30:        end case
31:      end switch
32:      if v_g(p_i) = 0 then
33:        capture_flag = 0
34:      else if delay_flag = SML &&
35:        v_g(p_{i-1}) = 0 and v_g(p_i) = 1 then
36:        Capture a scan-shift value at faulty FFs and
             calculate a value on the signal line g
37:        if v_g(p_i) = 0 then
38:          capture_flag = 0
39:        else
40:          capture_flag = 1
41:        end if
42:      end if
43:    end for
44:  end for
45: end for
```

**Fig. 4** The flow of the fault simulation for AND bridging faults

## 4. Experimental Results

Table 1 shows the experimental results for ISCAS'89 benchmark circuits. For each benchmark circuit, 50 CUDs were created by injecting a randomly selected bridging fault between a clock line and a gate signal line. Since we don't have any real faulty circuits, faulty responses were created virtually by the fault simulation which is same as that used in the proposed diagnosis method. Similar experimental setups have been also used by other researches [6]–[9]. We must point out that our proposed method guarantees that a fault existing in a CUD is always included in the candidate fault list obtained by the proposed method. For each candidate gate, AND bridge and OR bridge were included in an initial candidate fault set. As test patterns, 50 random patterns were applied. In the table, column "ave", "max", "min" and "sgl" mean the average number of candidate faults, the maximum number of candidate faults, the minimum number of candidate faults, and the number of CUDs for which only one candidate fault was deduced. Parameters $n_g, n_v$ and $n_c$ denote the following numbers.

- $n_g$: the number of gates which may bridge a candidate clock fanout branch
- $n_v$: the number of FFs which are driven by a candidate clock fanout branch
- $n_c$: the number of scan chains

We notice that in a large majority of the cases in every benchmark circuit only one candidate fault was deduced. Next we carried out experiments in order to see the difference between the proposed method and our previous method [1]. We generated responses of CUDs by the pro-

**Table 1** Experimental results by the proposed method

| circuit | ave | max | min | sgl | $n_g$ | $n_v$ | $n_c$ |
|---|---|---|---|---|---|---|---|
| s9234 | 2.3 | 19 | 1 | 44 | 20 | 7 or 8 | 8 |
| s13207 | 3.7 | 77 | 1 | 46 | 20 | 7 or 8 | 8 |
| s15850 | 3.1 | 33 | 1 | 44 | 20 | 7 or 8 | 8 |
| s35932 | 1.1 | 6 | 1 | 49 | 20 | 15 or 16 | 16 |
| s38417 | 1.4 | 14 | 1 | 48 | 20 | 15 or 16 | 16 |
| s38584 | 2.5 | 22 | 1 | 45 | 20 | 15 or 16 | 16 |

**Table 2** Experimental results by our previous method [1]

| circuit | ave | max | min | sgl | zero |
|---|---|---|---|---|---|
| s9234 | 2.5 | 18 | 0 | 34 | 9 |
| s13207 | 3.9 | 77 | 0 | 45 | 1 |
| s15850 | 3.4 | 33 | 0 | 38 | 6 |
| s35932 | 1.1 | 6 | 0 | 31 | 18 |
| s38417 | 1.4 | 14 | 0 | 45 | 3 |
| s38584 | 2.5 | 22 | 1 | 45 | 0 |

posed fault simulation, and applied the previous method [1] to deduce candidate faults. Table 2 shows the experimental results. Each column has the same meaning as in Table 1, except for column "zero", which denotes the number of CUDs for which no candidate faults were deduced. It is found that the previous method could not deduce any candidate faults for some cases. Also the average number of candidate faults is a little larger than the proposed method.

## 5. Conclusions

In this paper, we proposed a diagnosis method for bridging faults between a clock line and a gate signal line. Since a faulty clock waveform depends on the signal transition timing on a gate signal line which is bridged, the proposed method introduced the backward sensitized path tracing approach. The proposed method can deduce candidate faults more accurately than our previous method. In this paper, we focused mainly on the effect of the backward sensitized path tracing approach, and we dealt with resettable FFs and one circuit model, as shown in Fig. 2. In our future research, we will consider other circuit models like in our previous paper [1], where non-resettable FFs and a different type of circuit model were considered.

## Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP16K00075.

### References

[1] Y. Higami, H. Takahashi, S. Kobayashi, and K.K. Saluja, "Diagnosis for Bridging Faults on Clock Lines," Proc. Pacific Rim Int. Conf. on Dependable Computing, pp.135–144, 2012.

[2] A. Jee and F.J. Ferguson, "Carafe: An Inductive Fault Analysis Tool for CMOS VLSI Circuits," Proc. VLSI Test Symp., pp.92–98, 1993.

[3] S.T. Zachariah, S. Chakravarty, and C.D. Roth, "A Novel Algorithm to Extract Two-Node Bridges," Proc. Design Automation Conf., pp.790–793, 2000.

[4] K. Suemitsu, T. Ito, T. Kanamoto, M. Terai, S. Kotani, and S. Sawada, "A Parallel Method to Extract Critical Areas of Net Pairs for Diagnosing Bridge Faults," IEICE Trans. on Fundamentals, vol.E91-A, no.12, pp.3524–3530, 2008.

[5] F. Yang, S. Chakravarty, N. Devta-Prasanna, S.M. Reddy, and I. Pomeranz, "Improving the Detectability of Resistive Open Faults in Scan Cells," Proc. Int. Sympo. on Defect and Fault Tolerace in VLSI Systems, pp.383–391, 2009.

[6] S. Venkataraman and W. Fuchs, "A deductive technique for diagnosis of bridging faults," Dig. Int. Conf. on Computer-Aided Design, pp.562–567, 1997.

[7] Y. Gong and S. Chakravarty, "Locating bridging faults using dynamically computed stuck-at fault dictionaries," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.17, no.9, pp.876–887, 1998.

[8] S.-Y. Huang, "On improving the accuracy of multiple defect diagnosis," Proc. VLSI Test Symp., pp.34–39, 2001.

[9] 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 Trans. on Information and Systems, vol.E89-D, no.11, pp.2756–2765, 2006.