IPSJ Transactions on System LSI Design Methodology Vol. 2 250–262 (Aug. 2009)

Regular Paper

An Algorithm for Diagnosing Transistor Shorts Using Gate-level Simulation

Yoshinobu Higami,[†1] Kewal K. Saluja,[†2]
Hiroshi Takahashi,[†1] Sin-ya Kobayashi[†1]
and Yuzo Takamatsu [†1]

Conventional stuck-at fault model is no longer sufficient to deal with the problems of nanometer geometries in modern Large Scale Integrated Circuits (LSIs). Test and diagnosis for transistor defects are required. In this paper we propose a fault diagnosis method for transistor shorts in combinational and full-scan circuits that are described at gale level design. Since it is difficult to describe the precise behavior of faulty transistors, we define two types of transistor short models by focusing on the output values of the corresponding faulty gate. Some of the salient features of the proposed diagnosis method are 1) it uses only gate-level simulation and does not use transistor-level simulation like SPICE, 2) it uses conventional stuck-at fault simulator yet it is able to handle transistor shorts, thus suitable for large circuits, and 3) it is efficient and accurate. We apply our method to ISCAS benchmark circuits to demonstrate the effectiveness of our method.

1. Introduction

Fault diagnosis, which locates fault and defect sites in faulty LSIs, is very important, challenging, and costly step for improving design and manufacturing process and the corresponding yield. Various diagnosis methods have been proposed that target stuck-at faults or two-line bridging faults [4),9),11),13),15)]. However, such conventional gate-level fault models are not sufficient for modern and future LSIs that are designed using nanometer geometries. It is becoming important that other fault models, like transistor-level faults, must be considered. Diagnosis of transistor faults has been studied under IDDQ test environment. Although IDDQ test has a large potential for detecting and diagnosing transistor faults, it is difficult to apply to future LSIs because of its inherent limitation in distinguishing faulty IDDQ from fault free leakage current. Further, IDDQ test needs substantial longer test time than logic test, because IDDQ must be measured after a circuit settles down in a quiescent state. Hence, the logic test methods are more desirable for future large scaled LSIs. Diagnosis methods for transistor stuck-opens and intra-gate shorts under logic test environment have been proposed in Refs. 5) and 6), respectively. In these methods, transistor-level description is transformed into a gate-level description, and gate-level diagnosis tools are used. In Ref. 3), both logic-level and transistor-level simulations have been used for diagnosing transistor defects.

In this paper, we propose a diagnosis method for transistor shorts while using logic test environment. Target faults considered here are shorts between two nodes of a transistor. Nodes include source, drain, gate of transistors. Since the behavior of a transistor short depends on many physical parameters, it is difficult to describe it precisely. We define two types of transistor short models by focusing on the output values of the corresponding faulty gate. These two models are *strong short model* and *weak short model*. In the strong short model whenever the two shorted nodes have opposite logic values and there is conduction path from the fault site to the output of the gate, the corresponding gate produces an incorrect output. In the weak short model, the output of the faulty gate does not always produce an erroneous value when the two nodes that are shorted take opposite values and a conduction path exists from the faulty transistor to the output of the gate. This may happen when two nodes are weakly shorted and their values take fault free values for a particular input vector that sets the two nodes to opposite values. No previous research has considered such a fault model in fault diagnosis methods, except for the paper [7)]. In addition to the contents of paper [7)], this paper uses an enhanced diagnostic procedure to reduce candidate shorts by identifying equivalent shorts.

The proposed diagnosis algorithm uses only conventional gate-level simulation tools. Since a gate-level simulation is generally orders of magnitude faster than a transistor-level simulation, like SPICE, the proposed algorithm is applicable to large circuits. In our method, first the candidate gates are deduced by performing

†1 Department of Computer Science, Ehime University
†2 Department of Electrical and Computer Engineering, University of Wisconsin - Madison

stuck-at fault simulation with failing patterns, next candidate shorts are deduced among the shorts in the candidate faulty gates, and then candidate shorts are reduced by performing stuck-at fault simulation with passing patterns. Thus, at the end of this diagnosis we obtain candidate shorts, but we do not terminate the process yet, instead we collapse the faults set by identifying equivalent faults to quantify the diagnostic resolution of our method.

The rest of this paper is organized as follows. In Section 2, we describe the fault models and in Section 3 we propose diagnostic algorithms associated with the two models. In Section 4, experimental results for benchmark circuits are given. The paper finally concludes in Section 5 with summary of the results and remarks for future directions.

2. Fault Models

2.1 Basic Fault Behavior

The transistor shorts considered in this paper are shorts between two nodes of a transistor in a primitive gate. A transistor is represented by three nodes, namely source, drain and gate. The shorts between source and drain, gate and source, and gate and drain are denoted by **sd-short**, **gs-short** and **gd-short**, respectively. In general the behavior of a shorted transistor can be complex because various physical parameters affect it, and they vary considerably from defect to defect. As a result it is difficult to predict precisely what happens in a faulty circuit. Since we assume logic test environment, we do not have to pay a whole lot of attention to the electrical phenomenon, instead we focus on the values of the output of a faulty gate. Further, in our discussion we assume the followings:

- there is only a single fault in the circuit under diagnosis,
- the defect is assumed to be an intra-gate short,
- the defect does not cause a timing failure, and
- a short does not fail intermittently.

With respect to short defects, detailed discussions about their faulty behavior and fault modeling can be found in much of the existing literature [1),10),12),14)]. The focus of this paper is on diagnosis algorithm and not on fault modeling. Therefore we use the fault model previously proposed in Ref. 2), and extend it for fault diagnosis.

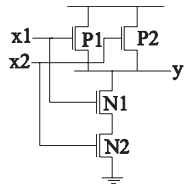

Fig. 1 two-input NAND gate.

Table 1 Values at transistor nodes.

in		out	P_1			P_2			N_1			N_2		
x_1	x_2	y	s	g	d	s	g	d	s	g	d	s	g	d
0	0	1	1	0	1	1	0	1	f	0	1	0	0	f
1	0	1	1	1	1	1	0	1	1	1	1	0	0	1
0	1	1	1	0	1	1	1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1	0	0	1	0	0	1	0

s: source, g: gate, d: drain

Table 2 Faulty effects.

in		P_1			P_2			N_1			N_2		
x_1	x_2	sd	gs	gd	sd	gs	gd	sd	gs	gd	sd	gs	gd
0	0	S	E	E	S	E	E	S	S	E	S	S	S
1	0	S	S	S	S	E	E	S	S	S	E	S	E
0	1	S	E	E	S	S	S	E	S	E	S	N	N
1	1	E	S	E	E	S	E	S	E	E	S	E	E

sd: source and drain, gs: gate and source gd: gate and drain

Table 1 shows the values of all transistor nodes in a fault-free two-input NAND gate. Symbol 'f' denotes a floating state. P_1, P_2, N_1 and N_2 are the transistors in a two-input NAND gate shown in **Fig. 1**. Clearly, the output of a faulty gate can never produce an erroneous value when a gate-input pattern is such that the two shorted nodes have identical values (both are logic 1 or both are logic 0). The gate in question may produce an erroneous output when a gate-input pattern sets opposite values at the two shorted nodes and a conduction path exists from one of the faulty nodes to the output of the gate. **Table 2** shows gate-input patterns that may produce erroneous values at the output of the gate. This table is similar to the table given in Ref. 2), in which diagnosis of faults was considered in IDDQ test environment. Our focus is on diagnosing faults in logic test environment as

a result we provide the symbolic signal values at the gate output. The choice of symbols and their meanings are explained below. The symbol 'S' means that the gate output does not produce an erroneous value because the gate-input pattern sets the two shorted nodes to same logic values. For example, when $x_1x_2 = 00, 10$ or 01 are applied to a two-input NAND gate and an sd-short at P_1 exists, the two nodes, source and drain of the transistor P_1, will have same values, therefore the gate output is logic 1 and it does not produce an erroneous value. Hence, in Table 2 at the intersections of rows $x_1x_2 = 00, 10$ and 01 in column "sd" under "P_1", the symbol 'S' are placed. The symbol 'E' means that the gate output may produce an erroneous value. Again, continuing with our example of a two-input NAND gate, when in input $x_1x_2 = 11$ is applied to the gate, the source and drain of the transistor P_1 take opposite values in the fault-free circuit in this gate, therefore in the presence of an sd-short at P_1 the gate output may produce an erroneous value. Hence at the intersection of row $x_1x_2 = 11$ and "sd" under "P_1", the symbol 'E' is assigned. The symbol 'N' means that the gate output does not produce an erroneous value even though the shorted nodes have opposite values because a conduction path does not exist from any one of the nodes to the gate output. In our running example, when $x_1x_2 = 01$ is applied to the NAND gate, the gate and the drain of the transistor N_2 have opposite values, yet the NAND gate with gd-short at N_2 does not produce an incorrect value because the transistor N_1 is in off-state, and the faulty nodes at N_2 do not affect the output of the gate. Similar to Table 2, we can construct tables for various gate types with 2, 3 or more inputs.

Before resorting to the development of an algorithm to diagnose transistor short faults, we define a test pattern that can potentially be used to detect a short fault in a gate as follows.

[**Definition 1**] Consider a short fault sh between a pair of nodes a and b of a transistor in a gate. An input pattern p is called an **e-pattern** of the transistor short fault sh provided the pattern p satisfies the following two conditions:

- nodes a and b take the opposite values in the fault free gate, and
- there is a conduction path from either a or b to the output of the gate.

For the two-input NAND gate of Fig. 1 the gate-input patterns with symbol 'E' in Table 2 are e-patterns for the corresponding shorts. For example, the inputs x_1x_2=00 and 01 are the e-patterns of gs-short at P_1. It should be noted that Table 2 does not specify the output values of the faulty gate, i.e., it only provides the information as to which gate-input patterns may produce erroneous values at the gate output. Thus, from this table we can know all e-patterns that may possibly produce erroneous values at the output, and non-e-patterns that will never produce erroneous values at the output. In a faulty LSI, all or some of the e-patterns may produce erroneous values at the output of a gate. However it is difficult, if not impossible, to know the exact behavior of a faulty transistor because physical parameters vary considerably in a faulty LSI. Hence, in the following subsections, we define fault models which are suitable for fault diagnosis assuming the logic test environment.

Table 3 values at the gate output when a strong short exists.

in		out	P_1			P_2			N_1			N_2		
x_1	x_2	y	sd	gs	gd	sd	gs	gd	sd	gs	gd	sd	gs	gd
0	0	1	-	0	0	-	0	0	-	-	0	-	-	-
1	0	1	-	-	-	-	0	0	-	-	-	0	-	0
0	1	1	-	0	0	-	-	-	0	-	0	-	-	-
1	1	0	1	-	1	1	-	1	-	1	1	-	1	1

2.2 Strong Short Model

In this subsection, we define a fault model, called **strong short model**, which specifies the faulty value at the output of faulty gate irrespective of the electrical parameters of the defects. In this fault model, the output of the faulty gate produces an erroneous value whenever an e-pattern of the fault is applied as an input to the faulty gate. **Table 3**, reproduced from Ref. 2) in the present notation, shows values at the output of a two-input NAND gate for all possible input conditions when a strong short exists. In this table, column "y" shows the fault free outputs for all possible inputs, and the columns "P_1","P_2", "N_1" and "N_2" show the actual erroneous outputs when each short exists in P_1, P_2, N_1 and N_2 transistor, respectively. A '-' indicates that the output y takes the fault free value even when the corresponding short exists.

Following interesting observation is made about the strong short faults and it is reported in Ref. 8). Test patterns that detect all stuck-at faults on the inputs and outputs of a gate g in a circuit, can also detect all strong short faults in

g. However, in a circuit if there is a redundant stuck-at fault on the input of some gate g, then the test patterns that detect all detectable stuck-at faults in g do not guarantee to detect all strong short faults in g. A consequence of this observation is that the test patterns that achieve 100% stuck-at fault efficiency do not necessarily achieve 100% strong short fault efficiency.

We must emphasize that Table 3 was originally given in Ref. 2) and is not a result of transistor or circuit level simulation performed by us. An important question one can ask is how well the strong short model covers the real defects. To answer this question we introduce a more realistic model in the following subsection where we also give the reasons for introducing the alternative model and its strength.

2.3 Weak Short Model

Behavior of a logic circuit in the presence of a fault often depends on various physical parameters of the fault. As a result, it is possible for the two nodes, that are shorted, to manifest a fault free value at the output of the gate or the circuit even when the signal values at the two shorted nodes are actually different. To allow for this we define another type of fault model, called **weak short model**, which covers a wider range of faulty behaviors than the strong short model. In this fault model, the output of a faulty gate produces an erroneous value for a subset of e-patterns. We believe that the weak short model is very powerful and it covers all transistor shorts that satisfy the following conditions:

(i) When an input pattern that sets the same value on the two shorted nodes, the faulty gate output never produces an erroneous value.

(ii) When there is no conduction path from either node to the gate output, the faulty gate output never produces an erroneous value.

Although we did not analyze real defects nor performed a detailed transistor-level simulation, we believe that more types of transistor short defects are covered by this model than the strong short model This is so, because the weak short model is such that a short defect that behaves like a strong short model will always be detected by the test patterns for weak shorts.

Some types of short defects do not behave as the weak short model. A short defect that does not satisfy either one of the above conditions (i) or (ii), is not covered by the weak short model. If such a short defect occurs in a circuit under diagnosis, the proposed algorithm may deduce incorrect candidate faults.

Table 4 Possible gate output values for a gd-weak-short at N_2.

x_1	x_2	y	case 1	case 2	case 3
0	0	1	-	-	-
1	0	1	0	0	-
0	1	1	-	-	-
1	1	0	1	-	1

We explain the bahavior of a weak short by using an example. For example, in the presence of the gd-short at N_2 in the two-input NAND gate of Fig. 1, the fault behavior can be any one of the three possible cases shown in **Table 4**, with respect to the output of the faulty gate. As before, in this table, column "y" shows the fault free outputs, and the columns "case 1", "case 2" and "case 3" show erroneous outputs when gd-weak short exists at N_2. A '-' indicates that the output takes the fault free value even when gd-weak short exists at N_2. Thus, for a weak short fault we assume that the output falls in one of the three possible cases, but we do not know which case actually applies. As a result, we must take care of all the possibilities during weak short fault diagnosis. The reason for introducing both these models is to compare them and also to show that candidate number of faults in the diagnosis list is small while using our algorithm even for the weak short model. Thus demonstrating the strength of the algorithm is resolving diagnostic ambiguity.

2.4 Fault Collapsing and Equivalent Shorts

It is important to reduce the candidate fault list of short faults that will help reduce the diagnosis runtime as well as help realize the potential of the diagnosis algorithm. In this subsection we explain fault collapsing in a gate and define equivalent shorts. Generally, if the output responses of a circuit with fault f_1 and that with a fault f_2 are identical then f_1 and f_2 are said to be equivalent faults. Similar to stuck-at faults, it is time-consuming to find all the equivalent shorts in a large circuit exactly. Therefore, we identify equivalent shorts in a gate to make an initial target fault list. (We explain how to find equivalent shorts between different gates in Section 3.6.)

In the strong short model, if for two distinct shorts the output responses of the faulty gates are identical for every gate-input pattern then the two shorts are

said to be equivalent shorts. For example, in the two-input NAND gate of Fig. 1, the sd-short at P_1 transistor and the sd-short at P_2 transistor produce the same output responses for all possible inputs to the gate, therefore they are equivalent shorts. With this method, in a two-input NAND gate, the 12 possible shorts can be reduced to 8 representative shorts and this is evident from Table 2.

In the weak short model, it is difficult to identify equivalent shorts exactly, because for a given input multiple faulty behaviors of this model are possible. For example, precisely speaking, in the weak short model, output responses by gd-short of P_1 and gd-short of N_1 may not be identical. However, since we do not know their behavior exactly, we cannot distinguish them. As a result, in this paper we use the same representative shorts as the strong short model.

3. Diagnostic Algorithm

3.1 Overview

In the section above we established the fault models by only focusing on the behavior of the logic values of a faulty gate, now we give a diagnosis algorithm that requires gate-level simulation and does not require transistor-level simulation. We assume the following inputs to the algorithm: a set of failing patterns, a set of passing patterns, and the information about the location of primary outputs where the erroneous values were observed for each failing pattern. As an output the algorithm provides the sets of candidate shorts and gates. The flow of the algorithm is shown in **Fig. 2** and it consists of four steps. In the first step, stuck-at fault simulation is performed with failing patterns to deduce candidate gates. In the second step, from all the shorts in the candidate gates, candidate shorts are deduced. In the third step, once again stuck-at fault simulation is performed but this time we target candidate gates with passing patterns. The gates that do not contain any candidate shorts are removed in each step. In the fourth step, the candidate shorts are further reduced by identifying equivalent shorts between different gates, inter-gate equivalent shorts as discussed in Section 3.6. All the steps, except for step 3, are used for both the strong short model and for the weak short model. Different procedures are applied in the third step for each fault model. In the following subsections we describe the procedures for each step of the algorithm to diagnose a single transistor short.

Input:
$V_f = \{v_1, v_2, ...\}$: a set of failing patterns
$V_p = \{u_1, u_2, ...\}$: a set of passing patterns
$PO(v_1), PO(v_2), ...$: information on location of primary outputs where erroneous values are observed for a faulty LSI with application of failing patterns $v_1, v_2, ...$
Output:
Sets of candidate shorts and gates
Algorithm:
Step 1: Deduce candidate gates by performing stuck-at fault simulation with V_f;
Step 2: Deduce candidate shorts by using the results of Step 1 and a prepared fault extraction table;
Step 3: Reduce candidate shorts by performing stuck-at fault simulation with V_p;
Step 4: Reduce candidate shorts by identifying equivalent shorts;

Fig. 2 Diagnostic algorithm.

3.2 Deduce Candidate Gates

In the first step of the diagnosis algorithm stuck-at fault simulation is performed with failing test patterns. The target faults are stuck-at 0 and 1 on the output of every gate. Clearly for a failing pattern there must a sensitized path from an actual faulty gate to a primary output. Also, a test pattern that detects a transistor short must also detect a stuck-at fault on the output of the corresponding gate. Therefore, if neither stuck-at 0 nor stuck-at 1 fault on the output of a gate g is detected by one or more failing patterns, then we deduce that there must not be a sensitized path from g to any of the primary outputs and hence there can be no actual short within the gate g. Also if a faulty effect of a gate g is propagated to a primary output which is different from the output where en erroneous value was observed, then g cannot be a candidate. Formally, we regard gate g as a candidate if g satisfies the following conditions.

[Condition 1: Keep g]
V_f: a set of failing patterns
$PO(v_i)$: a set of primary outputs where erroneous values are observed with

application of a $v_i \in V_f$

- Every test pattern $v_i \in V_f$ detects either stuck-at 0 or 1 fault on the output of gate g.
- Faulty effects are propagated to all the primary outputs included in $PO(v_i)$, but must not be propagated to a primary output which is not in $PO(v_i)$ when $v_i \in V_f$ is applied.

3.3 Deduce Candidate Shorts

In the second step of the algorithm candidate shorts are deduced from the list of all target shorts in the candidate gates obtained in step 1. We determine the patterns that appear at the inputs of each candidate gate when the failing patterns are applied. If all gate-input patterns that appear at the input of a candidate gate during simulation of failing patterns are e-patterns of a short, then that short is kept as a candidate short. In other words even if one input pattern that appears at the gate input is not an e-pattern of the short, then the short is removed from the set of candidate shorts. This is because a short can be excited only by e-patterns.

For efficient implementation of this step we prepare tables in advance that contain the relation between gate input patterns and possible shorts. Such a table for a two-input NAND gate is shown in **Table 5**, which contains only the collapsed faults i.e., representative faults. Thus, the faults sd at P_2, gs and gd at N_1 and gs at N_2, are not included in this table. The candidate shorts are easily deduced using this table. For example, consider the appearance of the patterns x_1x_2=00 and 10 at the two inputs of a NAND gate g when all the failing patterns are applied. In this case, gs-short and gd-short at P_2 transistor in g are the only two deduced candidate shorts. Similarly, when x_1x_2=00, 10 and 01 appear at inputs of the NAND gate g, no shorts are deduced and therefore the gate g is removed from the set of the candidate gates. This is because there is no short that has x_1x_2=00, 10 and 01 as e-patterns.

Table 5 is based on the fact that in the strong short model and the weak short model, only e-patterns produce erroneous values. Therefore, fail patterns are presumed to produce e-patterns on the inputs of candidate faults. If an actual defect in a circuit under diagnosis does not behave as a strong short model or a weak short model then the final candidate faults may not represent the actual defect. This is a basic limitation of the model-based fault diagnosis algorithms like the proposed algorithm.

Table 5 Shorts deduced by gate-input patterns.

gate-input x_1x_2 00	10	01	11	deduced shorts
√				gs and gd at P_1, gs and gd at P_2
	√			gs and gd at P_2, sd and gd at N_2
√	√			gs and gd at P_2
		√		gs and gd at P_1, sd at N_1
√		√		gs and gd at P_1
	√	√		ϕ
√	√	√		ϕ
			√	sd and gd at P_1, gd at P_2, gd at N_2
√			√	gd at P_1, gd at P_2
	√		√	gd at P_2, gd at N_2
√	√		√	gd at P_2
		√	√	gd at P_1
√		√	√	gd at P_1
	√	√	√	ϕ
√	√	√	√	ϕ

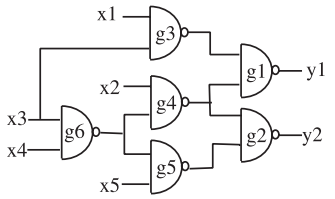

Fig. 3 c17 benchmark circuit.

The following example shows the application of step 1 and step 2 for an example circuit.

Example: Consider the diagnosis of c17 benchmark circuit, shown in **Fig. 3**. Suppose that failing patterns $x_1x_2x_3x_4x_5$ = 10101 and 10010 are given, and that the erroneous values are observed at y_1 with application of these two test patterns. In step 1, as a result of stuck-at fault simulation, g_1 and g_3 are deduced to be candidate gates, because stuck-at faults on the outputs of g_1 and g_3 are detected by both the test patterns on y_1, but the stuck-at faults on the other gate outputs are not detected on y_1 by either one or both of the patterns. In step

2, we investigate gate-input patterns appearing at inputs of gate g_1 and g_3 for these failing patterns. When the two test patterns are applied, pattern 01 and 11 appear at inputs of g_1. Therefore gd-short at P_1 in g_1 is deduced using Table 5. With respect to g_3, since pattern 10 and 11 appear at inputs of g_3, gd-shorts at P_2 and N_2 transistors in g_3 are deduced as candidate shorts. Thus at the end of these two steps there are only 3 short-faults in the list of candidate faults.

3.4 Reduce Candidate Shorts in the Strong Short Model

In step 3, candidate shorts are reduced further by performing stuck-at fault simulation for passing patterns. The methods used in this step for the strong and the weak short models are different. In this subsection, we explain the case of the strong short model.

When a passing pattern is applied during the fault simulation, either the output of the actual faulty gate must take a fault free value, or its effect must not be propagated to any primary output(s). We check for each candidate short whether an e-pattern appears at inputs of the gates or not. If no e-pattern appears at the inputs of the gate for all the passing patterns, then the gate as well as the shorts in the gate remain as candidates. If there is an e-pattern appearing at the inputs of the gate, we check whether the stuck-at fault on the output of the gate is detected by the passing pattern in question or not. If it can be detected, then corresponding short is removed from the set of candidate shorts. This is because the actual short is never detected by passing patterns in the fault simulation.

We provide a formalism of this in the following. Let sh be a candidate short in a gate g, u be a passing pattern, and p be a gate-input pattern appearing at inputs of g. If there is at least one passing pattern u for which both requirements listed in Condition 2 below are satisfied, then the short sh is removed from the set of candidate shorts in the gate g.

[Condition 2: Remove *sh* strong]

sh: a short g: the gate in which sh exists

u: a passing pattern

p: a gate-input pattern appearing at inputs of g with application of u

- Gate-input pattern p is an e-pattern of sh.
- The stuck-at fault on the output of g is detected by u.

3.5 Reduce Candidate Shorts in the Weak Short Model

In the weak short model, unlike the strong short model, the output values at the faulty gate are not specified for every e-pattern. When an e-pattern appears at the inputs of a candidate gate on application of a passing pattern, we must consider two possibilities for the output of the gate: 1) the gate produces an erroneous value at its output, and 2) it produces a fault free value at its output. Note that both these conditions need to be considered because in the case of week short some e-patterns can produce fault free values even if the fault is present. (Each input pattern can be either a failing pattern or a passing pattern. We do not assume that an input pattern produces an erroneous output or a fault-free output depending on the factors such as the strength of the driver of the faulty gate. Instead, in the weak fault model, we assume that we simply do not know whether each pattern produces an erroneous value or not.) Thus, even if a short satisfies Condition 2, it cannot be removed from the set of candidate shorts. However, if an e-pattern p appears at inputs of the gate for a failing pattern, and if Condition 2 is satisfied with p for a passing pattern, then the corresponding short can be removed. This is because of the fact that the appearance of an e-pattern p at a gate g with application of a failing pattern implies that the output of g produces an erroneous value for p, provided g includes an actual fault.

Again, formally speaking let sh be a candidate short in gate g. If there is at least one passing pattern u on which all the conditions in Condition 3 are satisfied with respect to sh, then short sh is removed from the set of candidate shorts.

[Condition 3: Remove *sh* weak]

sh: a short g: the gate in which sh exists

u: a passing pattern

p: a gate-input pattern appearing at inputs of g with application of u

- Gate-input pattern p is an e-pattern of sh.
- The stuck-at fault on the output of g is detected by u
- Gate-input pattern p also appears at g with application of a failing pattern.

3.6 Reduce Candidate Shorts by Equivalent Fault Identification

When more than one short remain in the candidate set, there is a possibility that equivalent shorts remain. As described in Section 2.4, equivalent shorts in a gate (intra-gate shorts) are collapsed, therefore we must find equivalent shorts

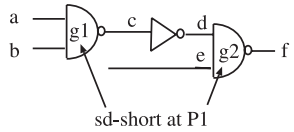

Fig. 4 Example of equivalent faults.

between different gates among the obtained candidate shorts. To do this, we find stuck-at faults which are equivalent to candidate shorts, and find equivalent stuck-at faults between different gates. It is relatively easy to find equivalent stuck-at faults between different gates by performing gate-level logic simulation. Also it is easy to find stuck-at faults equivalent to transistor shorts in a same gate. For example, consider the case where sd-shorts at P_1 transistor in gates g_1 and g_2 remain as candidates after the diagnosis algorithm is applied for a circuit shown in **Fig. 4**. An sd-short at P_1 is equivalent to the stuck-at 1 fault on the output of the gate, because only gate input pattern 11 produces an erroneous value on the output of the gate for both the faults. Therefore, sd-shorts at P_1 in g_1 and g_2 are equivalent to stuck-at 1 on line c and f, respectively. Also stuck-at 1 on c is equivalent to stuck-at 1 on f. This can be found by performing gate-level logic simulation. As a result, it is found that sd-shorts at P_1 in g_1 and g_2 are equivalent.

For formalizing the concept explained above, we need the following definition.

[**Definition 2**] A fault (short or stuck-at) is said to be **g-excited**, if the output of g takes an erroneous value when a test pattern is applied.

The concept of "g-excited" is not a new concept, but it is introduced here for the ease of understanding the explanation.

We now state a theorem on equivalent short faults. The theorem describes the condition of equivalence between a stuck-at fault and a short fault in a gate.

[**Theorem 1**] Let sa be a stuck-at fault on an input or the output of a gate g, and sh be a short in g. A short sh is equivalent to a stuck-at fault sa, provided the fault sh is g-excited by any gate input pattern that g-excites sa, and sa is g-excited by any gate input pattern that g-excites sh.

(Proof) Consider a circuit C formed of primitive gates. Let C_{sa} and C_{sh} be the two copies of the circuit C with a stuck-at fault sa and a short sh in a gate g, respectively. Also, let $gp_1, gp_2, \ldots$ be the gate-input patterns that g-excite sa and sh. Consider the case where a test pattern t_d detects sa. In this case t_d definitely produces gp_i, and t_d also g-excites sh. Since the erroneous value is propagated from g to a primary output, sh is also detected by t_d. Next consider the case where a test pattern t_u does not detect sa. If t_u does not produce gp_i, then t_u does not g-excite sh, either. If t_u produces gp_i but does not propagate the erroneous value to any primary output(s), then t_u will also not detect sh. This is because g takes the same erroneous value in C_{sa} and C_{sh}, and it is not propagated to any primary output(s) in C_{sa}. Therefore sh is detected whenever sa is detected, and it is not detected whenever sa is not detected. Hence, sa is equivalent to sh. ∎

The procedure for finding equivalent faults in step 4 is formally described as follows. Let sh_{ij} be a candidate shorts which is in gate g_i. First a stuck-at fault which is equivalent to $sh_{ij}(i, j = 1, 2, \ldots)$ is found among the stuck-at faults on the inputs and the output of g_i. Let sa_{ij} be a stuck-at α_{ij} ($\alpha_{ij} \in \{0, 1\}$) fault on line l_{ij}, which is equivalent to sh_{ij}. Next, logic simulation is performed by setting α_{ij} on line l_{ij}. If line l_{pq} takes on a value α_{pq} ($\alpha_{pq} \in \{0, 1\}$), and if no fanout branch exists between l_{ij} and l_{pq}, then it is deduced that sa_{ij} is equivalent to stuck-at α_{pq} fault on l_{qp}, which is denoted by sa_{pq}. Together the conditions imply that sh_{ij} is equivalent to a transistor short sh_{pq}, which is equivalent to sa_{pq}.

4. Experimental results

We implemented the proposed diagnostic algorithm using C language, and experimented with ISCAS'85 combinational circuits and ISCAS'89 sequential circuits with scan designs. First, we used test patterns that achieved 100% stuck-at fault efficiencies. **Table 6** shows fault coverage for the strong short model by the test patterns. The number of test patterns, the total number of shorts, the number of detected shorts and fault coverage are shown in columns "pattern", "total", "detect" and "cov(%)", respectively.

Next we show experimental results of fault diagnosis for the benchmark circuits. The benchmark circuits were converted so that every gate has 4 or fewer inputs. In the experiments, we generated ten different faulty circuits for each benchmark

Table 6 Fault coverage in the strong short model.

circuit	pattern	total	detect	cov(%)
c1355	92	4492	4484	99.82
c1908	145	6717	6700	99.75
c2670	79	9303	9022	96.98
c3540	137	13108	12787	97.55
c5315	88	19717	19623	99.52
c6288	33	19680	19580	99.49
c7552	108	27480	27155	98.82
s5378	137	15551	15414	99.12
s9234	178	31086	29553	95.07
s13207	285	42534	42193	99.20
s15850	172	52914	52021	98.31
s35932	67	119583	107775	90.13
s38417	179	123402	123120	99.77
s38584	208	139493	134836	96.66

Table 7 Results for strong short model.

	Average		Maximum		Average
circuit	gate	short	gate	short	time(s)
c1355	1.8	2.8	2	4	4.6
c1908	5.4	5.6	25	25	13.1
c2670	4.4	6.0	8	13	19.1
c3540	3.0	5.1	10	18	31.4
c5315	1.2	1.9	2	3	44.7
c6288	1.7	2.7	5	8	54.6
c7552	3.3	4.4	19	19	85.7
s5378	1.2	1.2	2	2	37.0
s9234	1.7	1.9	4	4	85.4
s13207	1.8	2.7	4	7	201.5
s15850	3.3	4.2	7	10	307.9
s35932	2.0	3.2	3	5	559.3
s38417	1.6	2.4	2	4	464.7
s38584	1.3	1.6	2	3	909.3

Table 8 Average numbers of faults obtained after each step.

	step1	step 2	step 3		step 4	
ckt	gate	short	gate	short	gate	short
c1355	5.3	11.4	2.3	3.3	1.8	2.8
c1908	34.2	56.2	5.6	5.8	5.4	5.6
c2670	54.7	157.0	6.2	8.8	4.4	6.0
c3540	58.8	158.7	4.6	7.5	3.0	5.1
c5315	16.5	37.1	2.2	2.9	1.2	1.9
c6288	11.3	24.4	1.9	2.9	1.7	2.7
c7552	26.5	46.2	4.1	5.2	3.3	4.4
s5378	12.7	33.6	2.3	2.3	1.2	1.2
s9234	24.2	60.1	1.8	2.0	1.7	1.9
s13207	68.4	315.9	2.2	3.2	1.8	2.7
s15850	22.0	56.4	5.8	7.0	3.3	4.2
s35932	5.1	13.8	2.3	3.5	2.0	3.2
s38417	8.6	28.8	1.8	2.1	1.6	2.4
s38584	12.2	29.6	2.3	3.2	1.3	1.6

Table 9 Results for weak short model.

	Average		Maximum		Average
circuit	gate	short	gate	short	time(s)
c1355	2.8	9.4	3	10	4.9
c1908	2.5	7.2	4	18	13.6
c2670	6.8	19	18	55	19.8
c3540	4.9	10.7	14	26	32.5
c5315	3.0	9.1	6	18	48.1
c6288	1.8	6.6	6	21	58.0
c7552	2.8	6.8	5	10	87.4
s5378	2.5	6.2	9	11	50.1
s9234	6.0	27.5	11	58	101.8
s13207	3.1	13.6	7	35	221.6
s15850	5.5	11	14	27	332.8
s35932	2.7	9.5	3	10	601.6
s38417	3.0	8.9	4	12	491.3
s38584	2.3	7.2	4	14	974.2

circuit, and each faulty circuit had one short injected, which was randomly selected. The diagnostic program was run on SUN Ultra10 workstation (440 MHz). **Table 7** includes the number of gates and the number of shorts obtained by the proposed algorithm in columns "gate" and "short", respectively. Columns "Average" and "Maximum" show average numbers and maximum numbers for ten faulty circuits, respectively. The last column shows the average runtimes in seconds. In all these experiments, the injected fault was a fault based on the strong short model, therefore, as expected the injected fault was always in the candidate fault list, thus validating the algorithm implementation.

In order to show the effectiveness of each step we also show the average number of faults obtained after each step in **Table 8**. Data in columns in "step 4" are same as those in columns "average gate" and "average short" of Table 7. From this table, it is found that passing patterns, which are used in step 3, were useful for reducing the number of faults. Also identification of equivalent faults in step

Table 10 Fault coverage in the strong short model with N-detection patterns.

circuit	pattern	total	detect	cov(%)
s5378	1033	15551	15414	99.12
s9234	1109	31086	29558	95.08
s13207	2489	42534	42193	99.20
s15850	1031	52914	52024	98.32
s35932	173	119583	107707	90.07
s38417	1008	123402	123127	99.78
s38584	1347	139493	134838	96.66

4 help reduce the number of faults further, albeit only a little.

Table 9 shows experimental results for the weak short model. Each column has the same meaning as labels in Table 7. For most of the circuits, the numbers of candidate gates and shorts are larger for the weak short model compared to the strong short model. However, for c1908 and c7552, the results for the weak short model have fewer candidate gates and shorts than the strong short model in the Maximum numbers. This is due to the difference of the number of failing patters and passing patterns. For example, when a short at a certain NOT gate was injected in c7552, 41 failing patterns and 67 passing patterns were obtained for the strong short model, while 10 failing patterns and 98 passing patterns were obtained for the weak short model. The larger number of passing patterns could remove more shorts from the candidate set in the weak short model.

The results of diagnosis depends on the quality of a diagnosis algorithm as well as the quality of diagnostic test patterns. In order to show the effectiveness of the proposed algorithm fairly by applying the different quality of test patterns, we also conducted experiments for N-detection test patterns, for $N = 5$, for ISCAS'89 benchmark circuits with scan designs. N-detection test patterns are generated by an ATPG that tries to detect each fault by N-different test patterns. **Table 10** shows the profiles of the test patterns for each circuit. **Tables 11** and **12** show results for the strong short model and the weak short model, respectively. Each column in Table 10, 11 and 12 has the same meanings as in Table 6, 7 and 9, respectively. For most circuits, results in Table 11 and 12 show the same or slightly smaller numbers than the results in Table 7 and 9, respectively. For s13207, larger numbers of gates and shorts were obtained with N-detection tests. This is because a particular gate-input pattern did not appear

Table 11 Results for strong short model with N-detection patterns.

	Average		Maximum		Average
circuit	gate	short	gate	short	time(s)
s5378	1.1	1.1	2	2	126.2
s9234	1.7	1.9	4	4	316.1
s13207	3.5	5.2	17	25	957.8
s15850	3.3	4.1	7	9	1313.4
s35932	2.0	3.2	3	5	1682.3
s38417	1.5	2.3	2	4	1229.9
s38584	1.3	1.6	2	3	2803

Table 12 Results for weak short model with N-detection patterns.

	Average		Maximum		Average
circuit	gate	short	gate	short	time(s)
s5378	2.4	6.0	9	11	128.1
s9234	5.2	23.4	8	45	333.3
s13207	5.7	25.8	17	85	1081.6
s15850	5.5	11.2	14	27	1480.6
s35932	2.7	9.5	3	10	1887.4
s38417	2.9	8.7	4	12	1330.6
s38584	2.5	7.7	4	14	3056.8

at a certain gate with application of N-detection patterns, and as a result some shorts in the gate remained as candidates.

Finally, the use of equivalent fault reduction is important because it reduces the simulation time and results into a fair evaluation metric. However, with respect to physical analysis, the count of number of candidate faults must include all faults that are equivalent to the fault in the diagnosis list. This is because all the faults equivalent to the candidate faults must be analyzed during the physical analysis process. To address this concern of the reader, the next two tables show the number of candidate faults including equivalent faults. **Table 13** and **Table 14** show the results for the strong short model and the weak short model, respectively. Column "Average" and "Maximum" show the average number of shorts and the maximum number of shorts, respectively. Column "with" and "without" show the number of shorts with collapsing and without collapsing, respectively. The numbers with collapsing are same as in Table 7 and 9. The average number of shorts without collapsing was from 2 to 19 for strong short

Table 13 Effect on collapsing for strong shorts with stuck-at patterns.

	Average		Maximum	
circuit	with	without	with	without
c1355	2.8	5.2	4	8
c1908	5.6	11.3	25	49
c2670	6.0	18.4	13	66
c3540	5.1	14.1	18	48
c5315	1.9	6.3	3	10
c6288	2.7	5.0	8	15
c7552	4.4	8.3	19	38
s1196	1.6	3.8	3	10
s1238	1.7	3.8	3	12
s1423	2.3	4.2	6	12
s1488	1.5	5.0	4	11
s5378	1.2	4.3	2	18
s9234	1.9	2.7	4	5
s13207	2.7	4.8	7	18
s15850	4.2	13.1	10	38
s35932	3.2	5.3	5	8
s38417	2.4	4.0	4	14
s38584	1.6	6.7	3	16

Table 14 Effect on collapsing for weak shorts with stuck-at patterns.

	Average		Maximum	
circuit	with	without	with	without
c1355	9.4	15.2	10	8
c1908	7.2	13.1	18	49
c2670	19.0	39.6	55	66
c3540	10.7	24.4	26	48
c5315	9.1	20.0	18	10
c6288	6.6	9.2	21	15
c7552	6.8	12.0	10	38
s1196	8.9	17.5	14	10
s1238	8.1	15.8	16	12
s1423	9.5	16.4	20	12
s1488	10.1	18.5	20	11
s5378	6.2	11.8	11	18
s9234	27.5	33.7	58	5
s13207	13.6	22.4	35	18
s15850	11.0	25.9	27	38
s35932	9.5	14.4	10	8
s38417	8.9	13.4	12	14
s38584	7.2	19.6	14	16

model, and from 11 to 40 for weak short model.

5. Conclusion

In this paper, we proposed a diagnosis algorithm for transistor shorts in logic test environment. Our fault model was established by focusing on values of the output of faulty gates. Thus, the diagnosis algorithm used gate-level simulation and not transistor-level simulation. Also it reduced candidate faults by identifying equivalent faults. In experimental results for the benchmark circuits, from 1.2 to 6 and 6.2 to 27.5 candidate shorts were obtained as average number for the strong short model and the weak short model faults, respectively. For reducing the number of candidate faults further, we need to apply the test patterns that distinguish them further. Moreover, identifying indistinguishable pairs is also important for evaluating whether the number of candidate faults is sufficiently small or not. In the future we will develop a diagnostic test generation method for transistor shorts. Also, development of an algorithm to handle multiple faults is a challenging future work.

Acknowledgments The authors would like to thank Prof. Kajihara who provided us the test patterns used in the experiments.

References

1) Abramovici, M., Breuer, M.A. and Friedman, A.D.: *Digital Systems Testing and Testable Design*, Computer Science Press (1990).
2) Aitken, R.C.: Fault location with current monitoring, *Proc. Int. Test Conf.*, pp.623–632 (1991).
3) Amyeen, M., Nayak, D. and Venkataraman, S.: Improving Precision Using Mixed-Level Fault Diagnosis, *Proc. Int. Test Conf.*, Paper 22.3 (2006).
4) Chakravarty, S. and Gong, Y.: An algorithm for diagnosing two-line bridging faults in cmos combinational circuits, *Proc. Design Automation Conf.*, pp.520–524 (1993).
5) Fan, X., Moore, W., Hora, C. and Gronthoud, G.: A novel stuck-at based method for transistor stuck-open fault diagnosis, *Proc. Int. Test Conf.*, pp.378–386 (2005).
6) Fan, X., Moore, W., Hora, C., Konjinenburg, M. and Gronthoud, G.: A gate-level method for transistor-level bridging faults diagnosis, *Proc. VLSI Test Symp.*, pp.266–271 (2006).
7) Higami, Y., Saluja, K.K., Takahashi, H., Kobayashi, S. and Takamatsu, Y.: Diagnosis of Transistor Shorts in Logic Test Environment, *Proc. Asian Test Symp.*, pp.354–359 (2006).

8) Higami, Y., Saluja, K.K., Takahashi, H. and Takamatsu, Y.: Fault Coverage and Fault Efficiency of Transistor Shorts using Gate-level Simulation and Test Genration, *Proc. Int. Conf. on VLSI Design*, pp.781–786 (2007).
9) Lavo, D.B., Chess, B., Larrabee, T. and Ferguson, F.J.: Diagnosing realistic bridging faults with single stuck-at information, *IEEE Trans. on Computer-Aided Design*, Vol.17, pp.255–268 (1998).
10) Malaiya, Y.K. and Rajsuman, R.: *Bridging Faults and IDDQ Testing*, IEEE Computer Society Press (1992).
11) Millman, S., McCluskey, E. and Acken, J.: Diagnosing cmos bridging faults with stuck-at fault dictionaries, *Proc. Int. Test Conf.*, pp.860–870 (1990).
12) Sachdev, M. and deGyvez, J.P.: *Defect-Oriented Testing for Nano-Metric CMOS VLSI Circuits*, Springer (2007).
13) Venkataraman, S. and Fuchs, W.: Diagnosis of bridging faults in sequential circuits using adaptive simulation, state storage, and path-tracing, *Proc. Int. Test Conf.*, pp.878–886 (1997).
14) Wang, L.-T., Wu, C.-W. and Wen, X. (ed.): *VLSI Test Principles and Architectures*, Morgan Kaufmann Publishers (2006).
15) Wu, J. and Rudnick, E.M.: Bridge fault diagnosis using stuck-at fault simulation, *IEEE Trans. on Computer-Aided Design*, Vol.19, pp.489–495 (2000).

(Received November 17, 2008)
(Revised February 20, 2009)
(Accepted April 13, 2009)
(Released August 14, 2009)

(Recommended by Associate Editor: *Xiaoqing Wen*)

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, 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. 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. 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 the IEICE.

Kewal K. Saluja obtained his Bachelor of Engineering (BE) degree in Electrical Engineering from the University of 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 currently with the Department of Electrical and Computer Engineering at the University of Wisconsin-Madison as a Professor, where he teaches courses in logic design, computer architecture, microprocessor based systems, VLSI design and testing, and fault-tolerant computing. Prior to this 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, and the University of Roorkee. 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.

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 2000 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 senior 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 Vice-Chair of the 2008 IEEE Asian Test Symposium.

Shin-ya Kobayashi received the 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 member of the Information Processing Society of Japan, the Institute of Electrical Engineers of Japan, IEEE, and ACM.

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. His research interests include test pattern generation, fault simulation, and fault diagnosis. Dr. Takamatsu is a senior member of the IEEE and a member of IPSJ. 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.