Diagnosis of Tunneling Opens

James C.-M. Li and E. J. McCluskey

Center for Reliable Computing, Stanford University

cmli,ejm@crc.Stanford.edu

Abstract

This paper resolves two issues regarding diagnosis of tunneling opens: efficient screening and accurate localization. In the first part, a test pattern selection and sorting algorithm is presented. It is shown that the presented algorithm saves $I_{DDQ}(t)$ test time without impacting on its effectiveness. The second part of this paper presents a locating algorithm which combines both VLV and $I_{DDQ}(t)$ test results. This technique is shown to be able to accurately locate the tunneling opens with higher resolution than a commercial single stuck-at fault diagnosis tool.

1. Introduction

A tunneling open is a very thin layer of oxide located in a via or a contact. The tunneling open is so thin that it allows electrons or holes to tunnel through it [Li 00]. Diagnosis of tunneling opens is important to improve the process problems and hence increase the yield. (In this paper, the term "diagnosis" includes both identifying the defective chips as well as locating the defects within the chips.) However, chips with tunneling opens are difficult to identify because they pass thorough at-speed tests at nominal voltage. Furthermore, once we identify a tunneling open suspect chip, it is also difficult to perform failure analysis because the tunneling open is small and hidden. This paper provides solutions for these diagnosis issues: 1. how to efficiently identify chips with tunneling opens and 2. how to accurately locate tunneling opens within chips.

Figure 1 illustrates the diagnosis flow of tunneling opens. Those chips that failed Very-low-voltage (VLV) testing are further tested by *$I_{DDQ}(t)$ testing* which is defined as taking multiple continuous I_{DDQ} measurements in a single pause. *$I_{DDQ}(t)$ drift over time* is defined as the $I_{DDQ}(t)$ values decrease significantly during the duration of measurement. Those chips that have $I_{DDQ}(t)$ drift over time are possible tunneling open suspects [Li 00]. They go on to the next stage, locating the failures. Finally, a culprit list is obtained. A *culprit list* is a collection of circuit gates that can cause the observed faulty behavior.

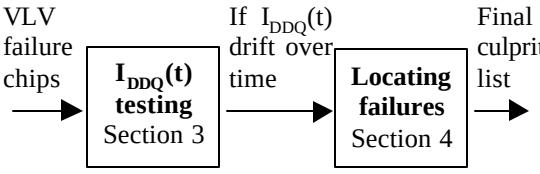

Figure 1. Diagnosis flow of tunneling opens

The first portion of this paper aims at efficient screening for tunneling opens. Since $I_{DDQ}(t)$ testing is a time consuming test, short test patterns that detect the tunneling opens as soon as possible are needed. Instead of building an ATPG tool to generate test patterns, a simple but efficient test pattern selection and sorting algorithm is presented. Just as an I_{DDQ} pattern selector selects test pattern for I_{DDQ} testing [Mao 92], the presented $I_{DDQ}(t)$ pattern selector selects (and sorts) test pattern for $I_{DDQ}(t)$ testing. This selector takes any available test patterns generated by the designers or by ATPG tools. It adopts a "greedy" algorithm which sorts test patterns to maximize the probability of detecting tunneling opens as early as possible. The presented $I_{DDQ}(t)$ pattern selector is economic both in terms of test pattern selection process as well as test time.

After identifying the tunneling open suspects, the second issue is to locate the tunneling opens. (In this paper, we assume that tunneling opens are caused by random defects. They do not occur in every via or contact.) Traditionally, people use single stuck-at fault (SSF) diagnosis tools to locate failures. For tunneling opens, the existing single stuck-at fault diagnosis tools may not be accurate enough because a circuit node

with a tunneling open not necessarily stuck at a logic zero or one. On the other hand, I_{DDQ} diagnosis algorithms which have been previously presented are based on bridging fault or transistor leakage fault model [Aitken 91] [Chakravarity 92] [Nigh 97] [Thibeault 97]. These fault models are not accurate for tunneling opens. So far, there is no diagnosis algorithm that is suitable for locating a tunneling open.

In this paper, a two-step algorithm is presented for locating the tunneling opens with high resolution. The first step uses VLV test results to generate a small initial culprit list. The second step further reduces the number of culprits by utilizing the $I_{DDQ}(t)$ drift over time information. It will be shown that the presented algorithm gives correct results with higher resolution than a commercial single stuck-at fault diagnosis tool. A diagnosis is *correct* if the actual defect location is included in the reported culprit list. A diagnosis has *high resolution* if the number of the reported culprits is small.

This paper is organized as follows. The second section shows two example circuits and defines the terminology. The third section describes the test pattern selection and sorting algorithm. The fourth section describes the locating algorithm. In both Sections 3 and 4, simulation and experimental results are shown to validate the presented algorithm. Then the fifth section summarizes this paper.

2. Examples and Definitions

2.1. An Inverter Example

Figure 2(a) shows an example inverter with a tunneling open located at its input. Suppose the metal voltage (V_m) rises, the Fowler-Nordheim tunneling current (J_{tunnel}) charges the gate capacitor (C_{gate}). The voltage on the poly (V_p) therefore follows. While the V_p is at an intermediate value, high I_{DDQ} is observed. As trap assisted tunneling current continues to charge C_{gate}, V_p rises at a slow speed. $I_{DDQ}(t)$ therefore decreases gradually with time. Figure 2(b) shows an example $I_{DDQ}(t)$ drift over time which is measured from one of the Murphy test chips [Li 00][McCluskey 00]. Similar things happen when the input signal has a fall transition. The conclusion is that $I_{DDQ}(t)$ drift over time is observed when the input of inverter has high-to-low or low-to-high transition.

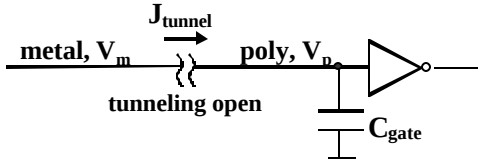

Figure 2(a) Inverter with tunneling open

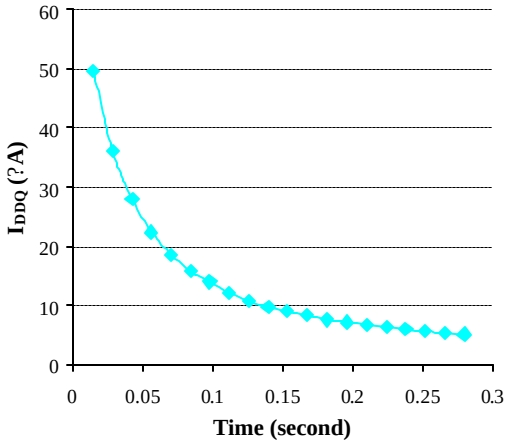

Figure 2(b) $I_{DDQ}(t)$ drift over time

2.2. Definitions

Some terms are defined here to facilitate further discussions. *Pseudo transition fault test patterns* are defined as test patterns that detect transition faults (slow-to-rise or slow-to-fall). The fault effect, in analogy to pseudo stuck-at fault test patterns, does not have to propagate to the primary outputs to be observed. The fault effect, however, can be observed by $I_{DDQ}(t)$ testing. A *pseudo transition fault test pattern pair*, denoted as {p1, p2}, is made up of two patterns. The first pattern, p1, initializes the state and the second pattern, p2, launches a transition. *Pseudo transition fault coverage* is defined as the number of transition faults which are detected by a set of pseudo transition fault test patterns over the number of total transition faults. A *pseudo transition fault dictionary* is a lookup table which specifies the pseudo transition fault test patterns and the transition faults they detect.

2.3. An NAND Gate Example

Figure 3 shows a NAND gate T that is part of a larger circuit. Two of the primary inputs (P.I.), A and B, are connected to gate T. The output of gate T (node Z) is connected to some other gates. Figure 4 shows the schematic of the NAND gate T implemented in static CMOS. All the fourteen via or contact locations where a tunneling open can occur are marked with d#.

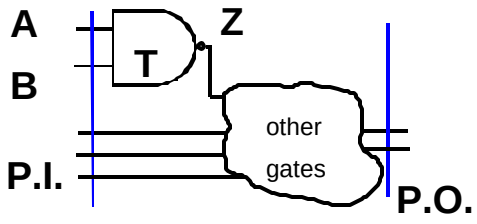

Figure 3. Example circuit with NAND gate T

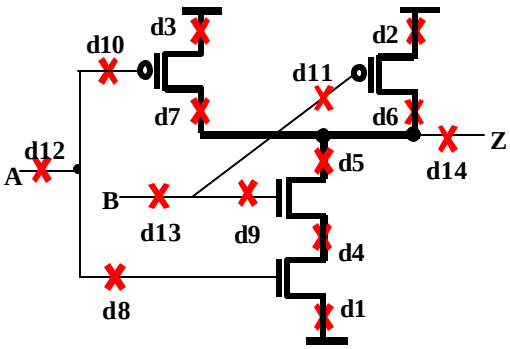

Figure 4. Fourteen defect locations in gate T

Table 1 shows a pseudo transition fault dictionary for the circuit in Figure 3. Every row represents a test pattern pair. For example, the first test pattern pair launches a falling transition at input A. Whether or not this transition fault propagates to the primary outputs, this test pattern is a valid pseudo transition fault test pattern. This pattern pair detects A slow-to-fall and Z slow-to-rise faults. These two columns, test patterns and their detected faults, constitute the pseudo transition fault dictionary.

Table 1. Pseudo transition fault dictionary

Test pattern			Detected Faults*	Detected Defects
A	B	Z		
Fall	1	Rise	Af,Zr	d3,d7,d10,d12,d14
Rise	1	Fall	Ar,Zf	d1,d4,d5,d8, d10,d12,d14
1	Fall	Rise	Bf,Zr	d2,d6,d11,d13,d14
1	Rise	Fall	Br,Zf	d1,d4,d5,d9, d11,d13,d14

* Af = A-slow-to-fall fault

There is an additional column showing the tunneling open defects that can be detected. For example, the first test pattern pair can detect five defects: d3,d7,d10,d12 and d14. It can be observed that, these four pseudo transition fault test patterns detect all transition faults in a NAND gate, they also detect all the tunneling opens. That is, 100% pseudo transition fault coverage of a NAND gate guarantees to detect all tunneling opens in it.

3. Test Pattern Selection and Sorting

In $I_{DDQ}(t)$ testing, the test time is proportional to the number of pauses needed (each pause consists of multiple continuous I_{DDQ} measurements). The *test length* of an $I_{DDQ}(t)$ testing is defined as the number of pauses needed. In this section, a test pattern selection algorithm is presented. Section 3.1 describes the algorithm. Section 3.2 and 3.3 shows simulation and experimental results respectively.

3.1. Algorithm

An initial fault list **F** is set to be all the transition faults in the circuit. The initial pattern list **P** is the collection of pattern pairs of every two neighboring test patterns. For example, if the original test pattern sequence is $\{p_1,p_2,p_3...p_n\}$, then $\{p_1,p_2\}$ is the first pattern pair and $\{p_2,p_3\}$ is the second pattern pair, *etc*.

The first step is to run a customized fault simulation to build a pseudo transition fault dictionary for every pattern pair in **P**. Based on this dictionary, the second step selects a pattern pair that detects the most transition faults. The third step updates the fault list **F** by deleting the detected faults. The pattern list **P** is also updated by deleting those pattern pairs that do not detect any undetected faults in **F**. If both **F** and **P** are not empty, the algorithm goes back to the second step. Otherwise it stops and the selected pattern pairs are reported (in the selection order). This algorithm maximizes the possibility to detect transition faults and hence maximizes the possibility to detect tunneling opens. Because the selected patterns have the same pseudo transition fault coverage as the original patterns, the selected patterns have the same effectiveness as the original ones.

In the case of combinational circuits, the selected test patterns can be applied in the sorted order. In the case of sequential circuits, the test patterns (assumed to be sequential patterns, not scan patterns) have to be applied in the original order. However, $I_{DDQ}(t)$ measurements are necessary only for those selected patterns.

3.2. Simulation Results

To prove the effectiveness of the pattern selection and sorting algorithm, a computer simulation is performed. Table 2 shows two circuits used in this simulation. They are also used in the Murphy chips. The first circuit is a six-bit squarer. It has 313 test patterns which are originally generated by an academic ATPG tool which detects every single stuck-at fault 15 times. If no pattern selection is performed, it would require pauses at all 313 patterns to make $I_{DDQ}(t)$ measurements. After selection, only 51 test patterns were selected. The test length is reduced by 83.7% while the same pseudo transition fault coverage is preserved. For the other circuit (m12, a multiplier), the algorithm also reduced the test length by 82.8%.

Table 2. Test length (original vs. selected)

Circuit	Original	Selected	Reduction
6sq	313	51	83.7%
m12	466	80	82.8%

Figure 5 compares the pseudo transition fault coverage of the original test patterns with the selected

(and sorted) patterns. The Y-axis is the pseudo transition fault coverage and the X-axis is the test pattern length. This figure is plotted using the data from 6sq. It can be seen that the curve of the sorted test patterns (thick dots) grows faster than the curve of the original test patterns (thin line). This figure shows that the sorted patterns have higher probability of detecting a tunneling open earlier than the original test patterns. Both curves have the same final values of almost 100% fault coverage. This shows that the selected and sorted test patterns have the same effectiveness as the original ones. The other circuit, m12, shows similar trend and is therefore not plotted.

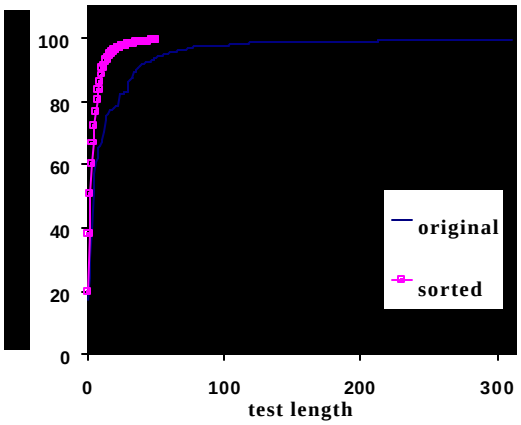

Figure 5. Growth of fault coverage (original vs. sorted)

3.3. Experimental Results

$I_{DDQ}(t)$ testing is performed on five of the Murphy *CUTs* (circuits under test) that are suspected to have tunneling opens. They are either 6sq or m12. In the $I_{DDQ}(t)$ experiment, after applying a specified test pattern, the tester paused and 20 continuous I_{DDQ} measurements are taken. Every single measurement takes between 7 and 21ms. Every I_{DDQ} measurement is recorded without setting any pass/fail threshold. If the last I_{DDQ} measurement is smaller than the first I_{DDQ} measurement by al least 10% and the difference is greater than 3?A, this test pattern is claimed to have $I_{DDQ}(t)$ drift over time. (The $I_{DDQ}(t)$ values of a good CUT remain low and constant below 3?A).

Both the original and sorted test patterns are applied in this experiment. Table 3 shows the first pattern that detects $I_{DDQ}(t)$ drift over time. Assuming the $I_{DDQ}(t)$ testing stops as soon as the first $I_{DDQ}(t)$ drift over time is detected, then these numbers represent the test time needed. Take the first CUT for example, if all the 313 original test patterns are applied, the thirty-second pattern is the first one to detect $I_{DDQ}(t)$ drift over time. If only the sorted 51 test patterns are applied, the $I_{DDQ}(t)$ drift over time can be detected as early as in the nineteenth pattern. The test time saved is 41%. Only in one case, CUT #4, the sorted version requires more patterns than the original version. For all these five CUTs, 55% of the test time is saved on the average. None of these five cases escapes the selected (and sorted) test patterns. The experimental results show that the test pattern selection and sorting algorithm saves $I_{DDQ}(t)$ test time without loosing effectiveness.

Table 3. First pattern to observe $I_{DDQ}(t)$ drift

CUT ID	Original	Sorted	Test time saved
1	32	19	41%
2	2	2	0%
3	5	1	80%
4	4	7	-
5	52	14	73%
Average	19	8.6	55%

4. Locating the Tunneling Open

The second portion of this paper addresses locating tunneling opens within the circuits. Section 4.1 first describes the algorithm. Section 4.2 and 4.3 show simulation and experimental results.

Notice that this localization technique is based on gates, not faults. From Table 1, it is observed that some defects can be modeled as more than one transition faults (e.g., d14 can be modeled as Zf and Zr). Therefore, this technique reports results in terms of gates instead of faults.

4.1. Two-step Localization Algorithm

Step 1 – Culprit List Generation

The first step utilizes the Boolean test results collected from the VLV testing. Traditionally, SSF diagnosis tools are used to diagnose Boolean failures. However, circuits with opens have sequential faulty behavior [Soden 89] which can confuse the SSF diagnosis tool and produce results containing a large number of innocent gates.

To understand the effect of the above problem, a computer simulation is performed. Table 4 shows the simulation results. The same Murphy circuits as section 3.2 are used again. The first circuit, 6sq, has 651 gates. The second circuit, m12, has 1677 gates. The test patterns applied are the same as that is used in section 3.

Six representative tunneling opens are injected by simulation. The first three rows show three different tunneling opens injected in a NOR gate in the 6sq circuit. These three tunneling opens locate in the same gate but their defect locations are different (similar to Fig. 4, there are also 14 different defect locations in a two-input NOR gate). The second three rows show

three tunneling opens injected in a two-input AND gate in the m12 circuit. A fault simulation is performed to produce the faulty behavior of these faulty circuits.

The first column of Table 4 shows the numbers of culprit gates reported by a commercial SSF diagnosis tool. A *culprit gate* is the gate that contains the diagnosed fault. Although SSF diagnosis tool is correct in all six cases, the numbers of culprit gates are large (i.e. low resolution). The second column will be used in section 4.2 later.

Step 2 – Culprit List Reduction

To solve the above resolution problem, $I_{DDQ}(t)$ drift over time information is used to reduce the number of culprits. From Table 1, we know that $I_{DDQ}(t)$ is observed when the defective NAND gate is tested by pseudo transition test patterns. However, we do not know which pair it is because we do not know the defect location. The conclusion is that the necessary condition for $I_{DDQ}(t)$ drift to occur is that the defective gate is tested by one pair of the pseudo transition test patterns.

Before we go into the details of the algorithm, three example cases are discussed in Table 5. Suppose five test patterns ($P_1 – P_5$) are applied in sequence and $I_{DDQ}(t)$ are measured after every pattern. Assume that $I_{DDQ}(t)$ drift over time is observed after applying patterns P_2 , P_4 and P_5. For patterns P_1 and P_3, their $I_{DDQ}(t)$ values remain low and constant (i.e., no drift). Every row represents a gate G. Letter D in column P_n row G denotes that at least one fault of gate G is detected by a pseudo transition fault test pattern pair $\{P_{n-1}, P_n\}$. Letter ND denotes that none of the faults in gate G is detected by the pseudo transition fault test pattern pair $\{P_{n-1}, P_n\}$. A dash means don't care (either D or ND).

In the first case, the culprit gate 1 is among one of the initial culprits reported by the step 1. Gate 1 was detected by three pattern pairs and they match the observed $I_{DDQ}(t)$ drifts. Gate 1 is called a *single culprit gate (SCG)* which is defined as a single gate that can explain all the $I_{DDQ}(t)$ drifts. In the second case, gate 2 and 3 are both reported by step 1. Although gate 2 or 3 alone does not match the $I_{DDQ}(t)$ drifts, the combination of them does. Therefore, gate 2 and 3 are called a pair of *multiple culprit gates (MCG)*. In case 3, suppose gate 4 and gate 5 are the only two gates reported by step 1. However, they can not explain all $I_{DDQ}(t)$ drifts. In this case, gate 4 and gate5 are only a portion of multiple culprit gates. Some warning message will be given saying that there should be some other undiagnosed faulty gates in additional to gates 4 and 5. The detailed algorithm of step 2 is as follows,

1. Construct a table similar to Table 5 with every column represents a test pattern with $I_{DDQ}(t)$ drift and every row represents a initial culprit gate obtained from step 1.
2. Check for single culprit gate (SCG)
3. If no SCG, check for multiple culprit gates (MCG)
4. If no MCG, check for incomplete multiple culprit gates (IMCG). Give warning messages.

4.2. Simulation Results

To compare with the commercial SSF diagnosis tool, the presented two-step algorithm is performed on the same six faulty circuits as in section 4.1. The $I_{DDQ}(t)$ test results are produced by computer simulations. The second column shows the numbers of culprit gates. The numbers are much smaller than the commercial SSF diagnosis tool gives. In all six cases, the presented algorithm gives single culprit gate (SCG) results and the injected faulty gate are located correctly. This shows that the presented algorithm successfully enhance the diagnosis resolutions without loosing correctness. On the average, the presented algorithm reports 3.7 culprit gates. Compared with the commercial SSF diagnosis tool, the presented algorithm reduces 94% of the culprits.

Table 4. Number of culprit gates (simulation)

Circuit	SSF	Presented	Reduction
6sq	21	4 (SCG)	81%
Total gate:651	12	4 (SCG)	67%
Test length:313	4	4 (SCG)	0%
M12	226	5 (SCG)	98%
Total gate: 1677	81	3 (SCG)	96%
Test length: 466	3	2 (SCG)	33%
Average	57.8	3.7	94%

Table 5. Three cases of step 2

Case #	Initial culprits	P_1 no drift	P_2 drift	P_3 no drift	P_4 drift	P_5 drift	Results
Case 1	Gate1	-	**D**	-	**D**	**D**	Single culprit gate (SCG)
Case 2	Gate 2	-	ND	-	ND	**D**	Multiple culprit gates (MCG)
	Gate 3	-	**D**	-	**D**	ND	
Case 3	Gate 4	-	ND	-	**D**	ND	Incomplete Multiple culprit gates (IMCG)
	Gate 5	-	**D**	-	ND	ND	

D = detected by $\{P_{n-1}$ to $P_n\}$, ND = not detected, - = don't care

4.3. Experimental Results

To prove the effectiveness of presented two-step algorithm, experimental data are collected from the same five Murphy CUTs. For the VLV testing, all CUTs are tested at 1.7V which is about two times the transistor's threshold voltage. The waiting time between applying the primary input and strobing the primary output is 1?s. The testing continues until the end without aborting on failures. Every failing output is recorded. For $I_{DDQ}(t)$ testing, experiment is performed in the same way as described in Section 3.3 except that $I_{DDQ}(t)$ data are measured for every test pattern to maximize the amount of information (i.e. no selection).

Table 6 compares the numbers of culprit gates obtained from the commercial SSF diagnosis tool with our presented algorithm. In the first two cases, the presented algorithm shows incomplete multiple culprit gates. This indicates that there are still some culprit gates undiagnosed by step 1. In the other cases, single culprit gates are reported. The presented algorithm reduces the number of culprits by 67% on the average.

Table 6. Number of culprit gates (experiment)

CUT ID	SSF	Presented	Reduction
1	1	1 (IMCG)	0%
2	3	3 (IMCG)	0%
3	10	7 (SCG)	30%
4	6	1 (SCG)	83%
5	25	3 (SCG)	88%
Average	9	3	67%

5. Summary

This paper provides solutions to two diagnosis issues of tunneling opens. In the first part of this paper, a test pattern selection and sorting algorithm is presented. This algorithm selects (and sorts) pseudo transition fault test patterns to efficiently identify tunneling open chips. Table 2 shows that the selected patterns are approximately 80% shorter than the original ones. The experimental results are summarized in the left portion of Table 7.

In the second part of this paper, a two-step algorithm is presented to locate the tunneling opens. The first step uses VLV testing results to generates a small initial culprit list. The second step further reduces the number of culprits by using $I_{DDQ}(t)$ drifts over time information. Experimental results show that the presented algorithm gives only one third as many culprit gates as a commercial SSF diagnosis tool. The results are summarized in right part of Table 7.

References

[Aitken 91] Aitken, R., " Fault Location with Current Monitoring," *Proceeding of International Test Conference*, pp.623-632, 1991.

[Chakravarity 92] Chakravarity, S. and M. Liu, "Algorithms for I_{DDQ} Measurement Based Diagnosis of Bridging Faults," *Journal of Electronic Testing: Theory and applications*, No.3, pp.91-99, 1992

[Li 00] Li, J. C.M and E.J. McCluskey, "Testing for Tunneling Opens," *Proceeding of International Test Conference*, pp. 85-94, 2000.

[Mao 92] Mao, W. and R. Gulati, "QUIESTEST: A methodology for selecting I_{DDQ} Test Patterns," Journal of Electronic Testing: Theory and Applications, 3, 349-357, pp.63-71, 1992.

[McCluskey 00] McCluskey, Edward .J. and C.W. Tseng, "Stuck-at Fault versus Actual Defects", *Proceeding of International Test Conference*, pp.336-343, 2000.

[Nigh 97] Nigh, P., D. Flolenza and F. Motika., "Application and Analysis of I_{DDQ} Diagnostic Software, " *Proceeding of International Test Conference*, pp.319-327, 1997.

[Soden 89] Soden, J., R. Treece, M. Taylor and C. Hawkins, "CMOS IC stuck-open Fault Electrical Effects and Design Considerations," *Proceedings of International Test Conference*, pp. 423-430, 1989.

[Thibeault 97] Thibeault, C., "A Novel Probabilistic Approach for IC Diagnosis Based on Differential Quiescent Current Signature," *IEEE VLSI Test Symposium*, pp.80-85, 1997.

Acknowledgement

This research is supported by Intel Corporation.

Table 7. Summary of the experimental results

CUT ID	Test pattern selection and sorting			Localization		
	Before sorting	After sorting	Time saved	SSF	Presented	Culprit reduction
1	32	19	41%	1	1	0%
2	2	2	0%	3	3	0%
3	5	1	80%	10	7	30%
4	4	7	-	6	1	83%
5	52	14	73%	25	3	88%
Average	19	8.6	55%	9	3	67%