Physically-Aware N-Detect Test Pattern Selection[*]

Yen-Tzu Lin, Osei Poku, Naresh K. Bhatti, and R. D. (Shawn) Blanton
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh PA 15213
{yentzul,blanton}@ece.cmu.edu

Abstract

N-detect test has been shown to have a higher likelihood for detecting defects. However, traditional definitions of N-detect test do not necessarily exploit the localized characteristics of defects. In physically-aware N-detect test, the objective is to ensure that the N tests establish N different logical states on the signal lines that are in the physical neighborhood surrounding the targeted fault site. We present a test selection procedure for creating a physically-aware N-detect test set that satisfies a user-provided constraint on test-set size. Results produced for an industrial test chip demonstrate the effectiveness and practicability of our pattern selection approach. Specifically, we show that we can virtually detect the same number of faults 10 or more times as a traditional 10-detect test set and increase the number of neighborhood states and the number of faults with 10 or more states by 18.0 and 4.7%, respectively, without increasing the number of tests over a traditional 10-detect test set.

1. Introduction

The single stuck line (SSL) fault model [1] assumes that a single signal line in a faulty circuit is permanently fixed to either logic 0 or 1. Because of its simplicity, the SSL fault model has been widely used as the basis of test generation and evaluation. Nevertheless, as the minimal feature size shrinks and circuit complexity increases, the behavior of even static defects[1] involves more complex mechanisms that can no longer be sufficiently dealt with using the SSL fault model alone. N-detect test was proposed to improve defect coverage without drastically increasing complexity in terms of modeling and test generation [2]. In N-detect test, each SSL fault is detected at least N times by N distinct test patterns. The hope is that the N different tests increase the likelihood of activating some arbitrary defect that affects the targeted line. Another advantage of N-detect test over other defect-oriented models is that existing approaches to automatic test pattern generation (ATPG) can be easily modified to generate N-detect test patterns.

The traditional requirement for N-detect test is to ensure every SSL fault is detected by at least N "different" patterns. Two test patterns are different if at least one input (primary or scan) has opposite logic values applied. Without physical information, however, a test set may satisfy the N-detect criteria but does not have the potential to improve defect coverage. Another issue with N-detect test is that the number of test patterns grows approximately linearly with N [3]. This may not only require test equipment to support a larger memory size, but also potentially lengthens testing time. So it is imperative that most effective tests be used when employing N-detect testing.

The objective of this work is to generate a compact but effective N-detect test set that has a higher capability to detect unmodeled defects [4]. We describe a physically-aware N-detect test selection approach for creating the objective test set which is guided by layout information. The main application of the proposed test selection approach is to generate a "physically-aware" N-detect test set from a large, traditional M-detect test set ($M \gg N$) in the absence of a physically-aware ATPG tool. Additionally, if a physically-aware N-detect test set is provided, the selection approach can be used as a static compaction technique since it is possible that tests generated later in the ATPG process may be more effective than those generated early in the process. Our physically-aware test selection approach is independent of any ATPG tool, and therefore is compatible with and easily integrated into existing testing flows.

The rest of this paper is organized as follows. Section 2 describes background relevant to this work. Our physically-aware test selection flow is described in Section 3, and experiment results based on this flow are presented in Section 4. Finally, in Section 5, conclusions are drawn.

2. Background

In the section we will introduce the physically-aware metric utilized in the physically-aware N-detect test, and provide an example to illustrate the effectiveness of physically-aware N-detect test. A comparison between

[*]This work is supported by the Semiconductor Research Corporation under Task no. 1246.001.

[1]Detection of static defects does not depend on the timing established by the test patterns or their sequence of application.

defect-oriented test and physically-aware test will be addressed, followed by the discussion of N-detect test set compaction.

2.1. Physically-aware metric

Various experiments with N-detect test have shown their ability to improve defect detection [2, 5, 6, 7]. We demonstrate that N-detect test can be further improved however by exploiting defect locality as measured by the physically-aware N-detect metric [4]. The metric requires each test pattern to establish a unique logical state for the physical "*neighborhood*" that surrounds the targeted line. The neighborhood[2] of a targeted line is defined to include all signal lines that are within some specified, physical distance from the target. The logic values established on the neighborhood lines by a test pattern that sensitizes the targeted line is defined as a *neighborhood state*. Neighborhood information can be obtained in a variety of ways. For example, one approach can be critical-area based, while another, less accurate approach can utilize parasitic extraction data. Note that some neighborhood states are not achievable due the the logic structure of the circuit. The number of achievable neighborhood states for a SSL fault provides a feasible upper bound of the value N, the targeted number of SSL fault detections. The analysis in [4] showed that the effort spent generating N-detect tests should be guided by the physically-aware metric, where the establishment of different neighborhood states would increase the probability of activating a static defect that affects the targeted line.

The original neighborhood state definition implicitly assumes that all possible neighborhood states are of equal importance [4]. Both failure analysis results and intuition reveal that certain neighborhood states are more likely to cause defect activation. If a list of "preferred" states is provided, the test generation/selection procedure can be biased towards achieving those states first. Here we define a class of neighborhood states, called *preferred states*, which are believed to have a greater likelihood of activating a defect affecting the targeted line. Ideally, an N-detect test set detects each SSL fault at least N times with N different neighborhood states. For a targeted stuck-at-0 (1) fault, the preferred states include those that maximize the number of logic zeroes (ones) achievable on the neighborhood lines. (Note we use maximum since the all-0 and all-1 states may not be possible.) Intuitively, these states produce a neighborhood environment that has a greater capability for activating a defect, and therefore are considered first before any other neighborhood states.

2.2. Effectiveness of physically-aware test

Here we use two examples to illustrate why physically-aware N-detect test is believed to be more effective than

[2]In our diagnosis work [8], the notion of a neighborhood is even more generalized to include the driving and receiving cells of the targeted line and the drivers of its neighbors.

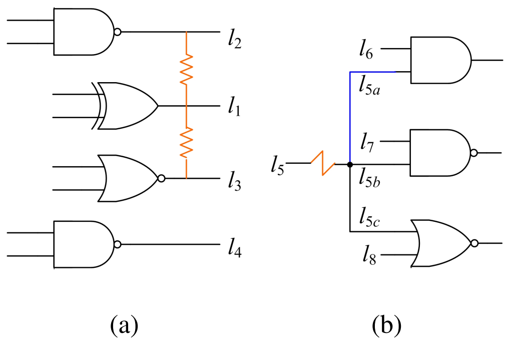


Figure 1. Example gate-level circuits with defective lines l_1 and l_5, where (a) l_1 is a bridged net, and (b) l_5 an opened net.

traditional N-detect. Figure 1 shows two gate-level circuits, both have a defective signal line l_1. In Figure 1(a), l_1 is a bridged net, and the logic value of l_1 is controlled by its neighbors, *i.e.*, l_2, and l_3. In physically-aware N-detect test, the logic values of l_2 and l_3 will be driven to the all-0 and all-1 states for at least two tests that sensitize l_1 for both stuck-at-0 and stuck-at-1. For this situation, the bridged net l_1 is more likely to be driven to the faulty value if indeed the neighborhood lines control activation. In Figure 1(b), the defective line l_5 is an opened net, and its floating fanout, l_{5a}, l_{5b} and l_{5c}, is assumed to behave like a multiple stuck line (MSL) fault [9]. Typically, the neighborhood of a targeted line includes the inputs of cells that drive the targeted line (called *drivers*) and receive the targeted line (called *receiver side-inputs*). When the targeted line has multiple fanout, the neighborhood includes the neighbors of the stem and all of its fanout. Thus, the neighborhood of l_5 likely includes l_6, l_7 and l_8. With this neighborhood, there is a test that sensitizes one of the fanout lines (*e.g.*, l_{5a}) that establishes a neighborhood state (*e.g.*, $l_6l_7l_8 = 101$) that transforms the MSL fault to an SSL fault that is detected.

Since traditional N-detect test is physically unaware, it is less likely to establish the neighborhood states necessary to activate a bridge or an open defect affecting the targeted line. Based on this intuition, we believe physically-aware N-detect test should be much more effective in detecting defects.

2.3. Defect-oriented test vs. physically-aware test

In defect-oriented test, defect behaviors are characterized as complex fault models, *e.g.*, bridge and open faults. However, the behaviors of bridges and opens are not known a priori, which means that a plethora of behaviors must be considered to ensure completeness. On the other hand, in physically-aware N-detect test, establishing all neighborhood states subsumes any activation conditions of any potential bridge or open behavior, and accomplishes this task concisely through the neighborhood concept.

Although physically-aware N-detect test and defect oriented test both utilize layout information, the layout usage of the former is very limited. Specifically, in physically-aware test, physical neighbors are required for each targeted line. This type of information is essentially already available in netlists generated during parasitic extraction. Thus, there is no additional cost related to the use of layout. Other neighborhood information involving driver gate inputs and receiver side inputs is easily obtained from the logical netlist, so layout is not required in those cases. On the contrary, defect-oriented models such as opens and shorts require precise analysis of the layout to identify their precise location. The complexity and cost associated with defect-oriented test is therefore higher than that of the physically-aware N-detect test.

2.4. N-detect test compaction

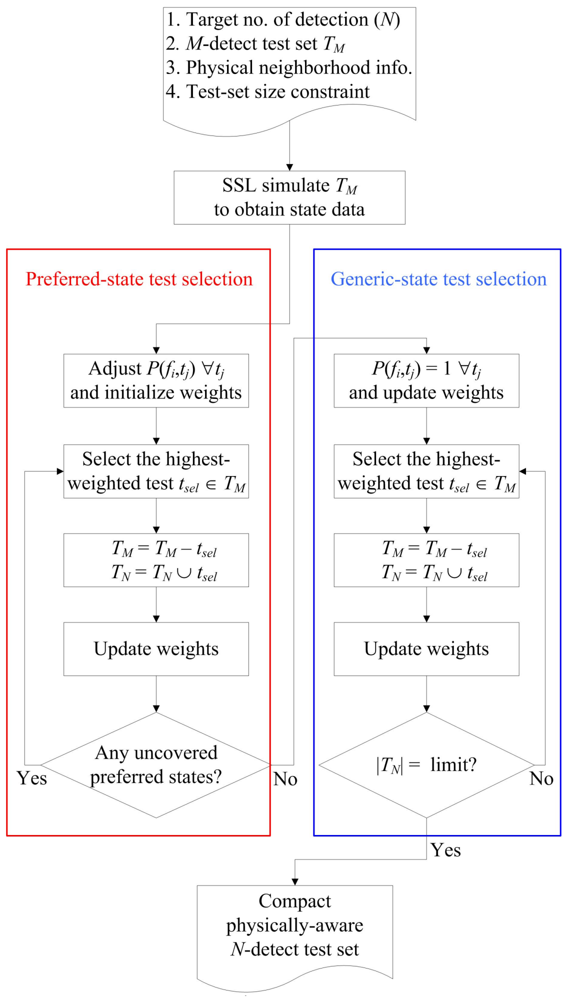
Recently, a physically-aware N-detect ATPG algorithm has been developed [10]. However, a significant issue that arises with N-detect is the expanded test-set size that results for both the traditional metric [2, 3, 5] and the physically-aware metric [10]. Various techniques were introduced to generate smaller N-detect test sets directly [3, 11, 12]. Nevertheless, these approaches do not consider physical information. Although the test-set size can be reduced by using these approaches, the test quality can be further enhanced by employing the physically-aware metric.

3. Physically-aware test selection

The physically-aware test selection (PATS) approach generates a compact, physically-aware N-detect test set by greedily selecting the most effective tests from a (preferably large) test pool. The objective of the procedure is to maximize neighborhood state coverage for a user-provided, test-set size constraint.

Figure 2 shows a flowchart describing PATS. The target number of detections N, a pre-generated M-detect test set T_M (where $M \gg N$), physical neighborhood information, and a test-set size constraint are all inputs to the main test selection procedure. The output is a compact, physically-aware N-detect test set, T_N, whose size satisfies the given size constraint.

The selection procedure starts with fault simulation of the test set T_M to obtain the list of neighborhood states established by T_M. Tests are weighted and selected from T_M based on their ability to detect faults that improve coverage as measured by the physically-aware metric. The main procedure of test selection consists of two phases: (1) preferred-state test selection and (2) generic-state test selection. Tests that establish the preferred states are given priority in the selection process to improve the detection capability of the selected test set. The generic-state test selection phase is similar to the preferred-state test selection phase, except that the bias towards preferred states is removed. Initially, the selected test set T_N is empty. Tests are then selected one at a time and removed from T_M and



1. Target no. of detection (N)
2. M-detect test set T_M
3. Physical neighborhood info.
4. Test-set size constraint
SSL simulate T_M to obtain state data
Preferred-state test selection
Generic-state test selection
Adjust $P(f_i,t_j)$ $\forall t_j$ and initialize weights
$P(f_i,t_j) = 1$ $\forall t_j$ and update weights
Select the highest-weighted test $t_{sel} \in T_M$
Select the highest-weighted test $t_{sel} \in T_M$
$T_M = T_M - t_{sel}$ $T_N = T_N \cup t_{sel}$
$T_M = T_M - t_{sel}$ $T_N = T_N \cup t_{sel}$
Update weights
Update weights
Any uncovered preferred states?
Yes
No
$|T_N| =$ limit?
No
Yes
Compact physically-aware N-detect test set

Figure 2. Physically-aware test selection (PATS) procedure.

added to T_N. The effectiveness of the remaining tests in T_M are then re-evaluated based on a weighting system. The one-at-a-time test pattern selection process repeats until the size of T_N reaches the constraint. The following sections describe the steps of PATS in greater detail.

3.1. Neighborhood state information

The first step of PATS is to obtain the neighborhood states established by the M-detect test set T_M given the neighbors of each line in the circuit under test. The neighborhoods are extracted from the layout. T_M is a pre-generated test set that provides a pool of tests that PATS uses to derive T_N. All stuck-at faults are simulated, without fault dropping, over the test set T_M. For each test t_j that detects a fault f_i, the neighborhood state established for f_i is recorded. Let $nbr(f_i, t_j)$ denote the neighborhood state of fault f_i under the application of test t_j. If t_j does not detect f_i, $nbr(f_i, t_j)$ is empty. The set of neighborhood states established for each SSL fault f_i by T_M, $S_{nbr}(f_i)$, is the union of the neighborhood states for f_i over each test $t_j \in T_M$:

$$S_{nbr}(f_i) = \bigcup_j nbr(f_i, t_j), \quad j = 1 \ldots |T_M|.$$

The size of $S_{nbr}(f_i)$ represents the number of times the fault f_i is detected with unique neighborhood states. Note that the selected N-detect test set can only establish up to $|S_{nbr}(f_i)|$ neighborhood states for a given fault f_i. The quality of the original test set therefore affects the quality of the selected test set T_N. A larger M-detect test set T_M (*i.e.*, a larger M) may have a higher $|S_{nbr}(f_i)|$ for each fault f_i, providing more tests and a larger search space for selection. Therefore the value of M should be sufficiently large to ensure high-quality results. The neighborhood state coverage of the 1-detect test set of the circuit under test can serve as the reference for determining the value M. In Section 4, we provide results for M=50 and N=10.

3.2. Test weighting

PATS greedily selects the most effective tests from the original test set T_M. To evaluate the effectiveness of a test in exercising the physical neighborhood of each detected fault, we define the weight of each fault and test as shown in equations (1) and (2), respectively.

$$FW(f_i) = (N - AS(f_i))^s \tag{1}$$

$$TW(t_j) = \sum_{i=1}^{F} FW(f_i) \cdot NS(f_i, t_j) \cdot P(f_i, t_j) \tag{2}$$

where

F = total number of SSL faults,

$$NS(f_i, t_j) = \begin{cases} 1 & \text{if test } t_j \text{ detects fault } f_i \text{ and the} \\ & \text{state has not yet been established by} \\ & \text{tests currently placed into } T_N, \\ 0 & \text{otherwise.} \end{cases}$$

$$P(f_i, t_j) = \begin{cases} 1 & \text{if } t_j \text{ establishes a preferred} \\ & \text{state of } f_i, \\ 0 & \text{otherwise.} \end{cases}$$

The weight of a fault f_i, denoted by $FW(f_i)$, is a dynamic function that changes as the tests are selected. It is an exponential whose base is the difference between N (the target number of detections, each with a unique state) and $AS(f_i)$ (the number of neighborhood states of f_i established by the current set of tests placed into T_N). T_N is initially empty and therefore $AS(f_i) = 0$ initially for all i. Both $AS(f_i)$ and $FW(f_i)$ are updated (if necessary) after each test is selected and added to T_N.

The exponent s in equation (1) is a constant used to spread the distribution of weights for faults that have a similar number of un-established states. For example, suppose at some time in the selection process, fault f_1 has two unique neighborhood states established by the tests in T_N, and fault f_2 has three. That is, $AS(f_1) = 2$ and $AS(f_2) = 3$. Let the target number of detections, N, be five. By setting s to one, fault weights for f_1 and f_2 become $(5 - 2)^1 = 3$ and $(5 - 3)^1 = 2$, respectively. The difference between $FW(f_1)$ and $FW(f_2)$ is therefore small at $3 - 2 = 1$. When s however is set to three, the difference becomes $(5 - 2)^3 - (5 - 2)^3 = 27 - 8 = 19$. With a larger s, a fault with fewer established neighborhood states will have a much higher weight than a fault with more established states. Since fault weights are used in the calculation of test weights, tests that detect faults with fewer established states will have much greater test weights and therefore be selected with higher priority. Tests selected into T_N will therefore establish approximately the same number of unique neighborhood states for each SSL fault (if structurally possible). In our experiments, $s = 3$ is empirically chosen as an appropriate spreading exponent.

The weight for a test t_j, denoted by $TW(t_j)$ in equation (2), is the sum of the weights of faults that are detected by t_j. The parameter $NS(f_i, t_j)$ indicates whether test t_j establishes a neighborhood state for fault f_i that has not yet been established by the tests already selected and placed into T_N. Specifically, if test t_j generates a duplicate neighborhood state for f_i, that is, the state has been established by some other test in T_N, then $NS(f_i, t_j) = 0$ resulting in the removal of $FW(f_i)$ from $TW(t_j)$. Conversely, $NS(f_i, t_j)$ is set to one if adding t_j to T_N will increase $AS(f_i)$. In the preferred-state test selection phase (see Figure 2), the function $P(f_i, t_j)$ represents whether the state established by t_j is a preferred state. In the generic-state test selection stage, $P(f_i, t_j)$ defaults to 1 for all i and j.

3.3. Test selection

Given the neighborhood states established by the test set T_M and the test weighting metrics described in Section 3.1 and 3.2, PATS performs preferred-state test selection followed by generic-state test selection. In the preferred-state test selection phase, only tests establishing the preferred states are considered using $P(f_i, t_j)$ as described in equation (2).

Initially, when $T_N = \phi$, the weights of all the faults are equal to N^s. The test selected, t_{sel}, is the one that detects the most faults with preferred states. Test t_{sel} is removed from T_M and added to T_N. Fault and test weights are then updated as described next. For each fault f_i detected by test t_{sel}, if t_{sel} establishes a new neighborhood state for f_i that has not been established by tests already selected into T_N, then the neighborhood state is marked as covered. The number of established neighborhood states $AS(f_i)$ is also incremented by one, thereby reducing the weight $FW(f_i)$ of fault f_i. The weights of detected faults are removed from the weights of tests that establish the same neighborhood

	1-detect	10-detect	PATS	50-detect	Percentage change (%) PATS versus 1-detect	Percentage change (%) PATS versus 10-detect
Test-set size	165	1,243	1,243	5,753	653.3	0
No. of faults detected $\geq$ 10 times	69,162	91,590	91,199	91,590	31.9	-0.4
No. of established states	1,205,116	3,852,769	4,547,812	9,012,492	277.4	18.0
No. of faults with $\geq$ 10 states	35,997	53,090	55,604	57,172	54.6	4.7

Table 1. Comparison of test-set sizes and effectiveness of traditional N-detect and the PATS.

states. For instance, suppose tests t_2 and t_{sel} ($t_2 \neq t_{sel}$) both detect f_1 with a neighborhood state not established by tests in T_N. When t_{sel} is chosen and added to T_N, the neighborhood state of f_1 is marked as covered. Test t_2 no longer establishes a new neighborhood state for f_1, so $FW(f_1)$ is removed from $TW(t_2)$ by setting $NS(f_1, t_2)$ to zero. Whenever a test is chosen from T_M and added to T_N, the fault and test weights are updated as just described. Ties among tests with the highest weight are broken by arbitrarily choosing one test. The preferred-state test selection procedure continues selecting tests with the highest weight in the test set T_M until all testable faults have preferred states established, which implies that each SSL fault is detected at least once.

After preferred-state test selection, PATS enters the generic-state test selection phase. This phase is similar to the preferred-state test selection phase, except the termination condition and the use of $P(f_i, t_j)$ are changed. Each $P(f_i, t_j)$ is set to one permanently in the generic-state test selection phase. That is, the weight of fault f_i is added to the weight of test t_j as long as t_j detects f_i and establishes a unique neighborhood state not yet established by tests in T_N. The test weights are updated before this phase begins due to the change in value of $P(f_i, t_j)$. As in the preferred-state test selection phase, the procedure continues by repeatedly selecting the highest-weighted test followed by an update of the weights of the remaining tests. The process stops however when the size of T_N reaches the user-provided constraint.

4. Silicon experiment

The physically-aware test selection procedure described in Section 3 is applied to an LSI test chip fabricated using 90nm technology. Specifically, PATS is used to select a physically-aware 10-detect test set (*i.e.*, $T_N = T_{10}$) from a 50-detect test set (*i.e.*, $T_M = T_{50}$) with a size no larger than a traditional 10-detect test set. The traditional 1-, 10- and 50-detect test sets, generated by the same commercial tool, provide test-set bounds for comparison purposes. Physical neighborhoods are extracted from the layout for a radius of 150nm, a radius assumed to completely contain any defect affecting the targeted line. Neighborhood state coverage established by the 50-detect test set is obtained by fault simulating the test set using the fault tuple [13] simulator FATSIM [14]. Preferred-state test selection is performed followed by generic-state test selection. In this experiment, neighborhood states that maximize the number of logic zeroes and logic ones are both used as preferred states for stuck-at-0 and stuck-at-1 faults.

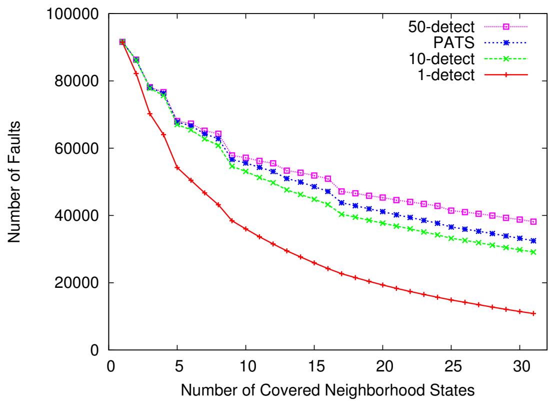


Figure 3. Comparison of the test sets based on the distribution of the number of faults against the number of established states.

Table 1 shows a comparison of the 1-detect, 10-detect, PATS, and 50-detect test sets. Although the PATS test set does not perform better than the traditional 10-detect test set based on the traditional N-detect metric (0.4% less faults are detected at least 10 times by the PATS test set), it significantly outperforms the 10-detect test set based on the physically-aware metric. The number of neighborhood states established by the 10-detect test set is 3.85 million, while the PATS test set covers 4.54 million states, an 18.0% increase. Our test set also detects 4.7% more faults that satisfy the 10-detect requirement based on the physically-aware metric. These results show that neighborhood state coverage can be significantly increased over traditional N-detect with no increase in test execution cost (*i.e.*, test-set size is maintained).

Figure 3 plots the number of faults versus the number of neighborhood states established for the test sets. Specifically, the chart shows the number of faults (y-axis) that satisfy the physically-aware N-detect metric for various values of N (x-axis). The numbers from the 1-detect test set form the lower bound, and the upper bound stems from the 50-detect test set. Comparing the PATS test set with the 10-detect, we see that for every established neighborhood state count (*i.e.*, for every possible value of N), the PATS test set has an equal or greater number of faults. This implies that the PATS test set detects more faults that satisfy the physically-aware metric for each N, and the average number of established neighborhood states is increased. The result also shows that the overall neighborhood state count is not increased by reducing the states achieved for individual faults. Consequently, the PATS test set is more effective than the traditional 10-detect test set when measured by the physically-aware metric.

We fault simulated the test sets and computed the defect level (defect parts per million, DPM) using bridge and open faults (see Table 2). The defect level (DL) is calculated using the formula $DL = 1 - Y^{1-FC}$, where Y is the yield and FC is the fault coverage [15]. Table 2 presents DL for $Y = 75\%$, 90%, and 99%. For bridge faults, the 4-way bridge fault model is utilized, along with two other cases where both bridged lines are faulty concurrently. For opens, we assume the net model [9] and randomly select a neighborhood state as the activation condition for each open. Table 2 shows that the PATS test set achieves lower defect levels for both fault models when compared to the traditional 10-detect test set of the same size, thus demonstrating the ability of PATS to detect realistic fault models.

		1-det	10-det	PATS	50-det
Bridge	FC (%)	87.22	88.15	88.19	88.20
	DPM @ Y=75%	36089.9	33504.4	33413.2	33376.8
	DPM @ Y=90%	13371.8	12403.3	12369.2	12355.6
	DPM @ Y=99%	1283.3	1189.8	1186.6	1185.2
Open	FC (%)	56.72	58.08	58.17	58.32
	DPM @ Y=75%	117062.7	113603.1	113382.4	113004.1
	DPM @ Y=90%	44573.3	43203.9	43116.7	42967.2
	DPM @ Y=99%	4340.1	4204.0	4195.4	4180.5

Table 2. Comparison of DPM ($10^6 \times DL$) of traditional N-detect and PATS.

The PATS test set is used for package test of the LSI test chips. Scan-chain flush, traditional 1-detect and PATS test sets are applied to every chip. Among the 551 chips going through this test flow, none uniquely fails the PATS test set. Due to the small sample size, the PATS test set has not been able to show any advantage over the 1-detect test set. We are however currently conducting a larger experiment to further investigate the effectiveness of physically-aware tests compared to traditional tests.

5. Summary

A test selection approach was described for generating compact N-detect test sets based on a physically-aware metric. Experiment results showed that for a given traditional N-detect test set, PATS can generate a physically-aware test set that is more effective in exercising physical neighborhoods that surround targeted lines. PATS provides an efficient alternative for improving test quality when physically-aware ATPG tools are not available, and can be integrated into existing testing flow easily. Compared to traditional N-detect test sets, the quality of PATS test set is enhanced without any increase in test execution cost.

6. Acknowledgement

The authors would like to thank Chris Schuermyer and Brady Benware of Mentor Graphics (formerly LSI) for their collaboration on the silicon experiment. The authors would also like to thank CMU Ph.D. candidate Jason Brown for his help with extracting physical neighborhood information.

References

[1] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Boston MA: Kluwer Academic Publishers, 2000.

[2] S. C. Ma, P. Franco, and E. J. McCluskey, "An experimental chip to evaluate test techniques experiment results," *Proc. International Test Conference*, pp. 663–672, Oct. 1995.

[3] S. M. Reddy, I. Pomeranz, and S. Kajihara, "On the effects of test compaction on defect coverage," *Proc. VLSI Test Symposium*, pp. 430–435, Apr.-May 1996.

[4] R. D. Blanton, K. N. Dwarakanath, and A. B. Shah, "Analyzing the effectiveness of multiple-detect test sets," *Proc. International Test Conference*, pp. 876–885, Sep.-Oct. 2003.

[5] I. Pomeranz and S. M. Reddy, "A measure of quality for N-detection test sets," *IEEE Transactions on Computers*, vol. 53, pp. 1497–1503, Nov. 2004.

[6] B. Benware, C. Schuermyer, N. Tamarapalli, K.-H. Tsai, S. Ranganathan, R. Madge, J. Rajski, and P. Krishnamurthy, "Impact of multiple-detect test patterns on product quality," *International Test Conference*, pp. 1031–1040, Sep.-Oct. 2003.

[7] M. E. Amyeen, S. Venkataraman, A. Ojha, and S. Lee, "Evaluation of the quality of N-detect scan ATPG patterns on a processor," *Proc. International Test Conference*, pp. 669–678, Oct. 2004.

[8] R. Desineni, O. Poku, and R. D. Blanton, "A logic diagnosis methodology for improved localization and extraction of accurate defect behavior," *Proc. International Test Conference*, 2006.

[9] S. Venkataraman and S. Drummonds, "A technique for logic fault diagnosis of interconnect open defects," *Proc. VLSI Test Symposium*, pp. 313–318, Apr.-May 2000.

[10] J. E. Nelson, J. G. Brown, R. Desineni, and R. D. Blanton, "Multiple-detect ATPG based on physical neighborhoods," *Proc. Design Automation Conference*, pp. 1099–1102, July 2006.

[11] S. Lee, B. Cobb, J. Dworak, M. R. Grimaila, and M. R. Mercer, "A new ATPG algorithm to limit test set size and achieve multiple detections of all faults," *Proc. Design, Automation and Test in Europe*, pp. 92–99, March 2002.

[12] K. R. Kantipudi and V. D. Agrawal, "On the size and generation of minimal N-detection tests," *Proc. the 19th International Conference on VLSI Design*, pp. 425–430, Jan. 2006.

[13] R. D. Blanton, K. N. Dwarakanath, and R. Desineni, "Defect modeling using fault tuples," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 2450–2464, Nov. 2006.

[14] K. N. Dwarkanath and R. D. Blanton, "Universal fault simulation using fault tuples," *Proc. Design Automation Conference*, pp. 786–789, Jun. 2000.

[15] T. W. Williams and N. C. Brown, "Defect level as a function of fault coverage," *IEEE Transactions on Computers*, vol. C-30, pp. 987–988, Dec. 1981.