An Optimal Test Pattern Selection Method to Improve the Defect Coverage

Yuxin Tian, Michael R. Grimaila†, Weiping Shi and M. Ray Mercer
Department of Electrical Engineering
Texas A&M University, College Station, Texas 77843, USA
Fax: 979-845-2630 E-mail: {ytian, wshi, mercer}@ee.tamu.edu

† Department of Systems and Engineering Management
Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433, USA
E-mail: Michael.Grimaila@a t.edu

Abstract

It is well known that n-detection test sets are effective to detect unmodeled defects and improve the defect coverage. However, in these sets, each of the n-detection test patterns has the same importance on the overall test set performance. In other words, the test pattern that detects a fault for the rst time plays the same important role as the test pattern that detects that fault for the (n)-th time. In this paper, we propose a linear programming-based optimal test pattern selection method which aims at reducing the overall defect part level (DPL). Using resistive bridge faults as surrogates, our experimental results on ISCAS85 circuits demonstrate the proposed test pattern selection method achieves a higher defect coverage than traditional n-detection method.

1 Introduction

Semiconductor manufacturers strive to attain a high yield (ideally 100%) when fabricating integrated circuits. Unfortunately, numerous factors can lead to a variety of manufacturing defects which may reduce the overall yield. The purpose of testing is to identify and eliminate any defective chips after the chips are manufactured. However, it is currently impractical to test exhaustively for all possible defects. This is a result of the computational infeasibility of accurately modeling defects, limitations imposed by existing manufacturing test equipment and time/economic constraints imposed by the test engineers. For these reasons, the stuck-at-fault (SAF) model has been accepted as the standard model to generate test patterns. Most of the existing commercial ATPG tools use the SAF coverage as a metric of the quality of a test set and terminate test generation when a high SAF fault coverage is attained.

Although single stuck-at fault detection is widely ac-cepted in industry, it can not detect many other types of defects [19]. Especially as the widespread usage of deep submicron technology, the probability of development of transient and pattern sensitive faults is rising tremendously. To address the deficiency of stuck-at fault detection, one approach is to build more sophisticated fault models, such as the bridge fault model [15][12] and the delay fault model [10]. However, since most existing ATPG tools are designed for single stuck-at faults, considerable effort must be made to extend these tools to the new fault models. In this paper, we follow another approach: n-detection, where each single stuck-at fault is detected multiple times in order to catch other types of defects [18] [8]. In the n-detection approach, the existing ATPG tools can be used with minimum efforts to achieve the best results.

The first work on n-detection was done by Ma, Franco and McCluskey [14]. In an experiment, they showed that as the number of unique detections for each fault increases, the defect coverage always improves compared with other test generation schemes. When n is sufficiently large, the n-detection stuck-at test pattern sets lead to the identification of almost all defective chips. Subsequent works [16][20] have also provided the analysis of the effectiveness of n-detection test pattern sets.

However, the set of test patterns that detect each stuck-at fault n times is often too large, resulting in both high tester memory usage and long testing time. This is because the n-detection method treats all test patterns that detect a stuck-at fault equally important when the fault is detected less than n times. Previous research has proved that not all the patterns in the test sets are equally efficient in reducing the defect coverage [9]. Thus, n-detection on each single stuck-at fault would somehow waste some test patterns and reduce the performance of the test set. Therefore, a post test generation compaction

or optimization is necessary.

In this paper, we propose a linear programming-based new test pattern generation strategy which selects the optimal subset of test patterns from the n-detection test set based on a weighted defect part level estimation model – MPG-D model [4]. Instead of generating the test patterns directly, our method aims at selecting an optimal pattern set from a big n-detection set. We use weighted MPG-D model as our objective function to select the patterns, which accurately reflect the contributions of each pattern to detect unmodeled defects. Since the accurate defect coverage is almost impossible to get [1], we use the surrogate detection approach to compare the fault coverage of the test pattern set with the n-detection test pattern set. (Here, surrogates mean the fault models that are not targeted directly during the test pattern generation but are used to evaluate the quality of the test pattern sets). Under this approach, the test patterns are generated for a specific target fault model (single stuck-at fault) and then simulated on the surrogate fault model (bridge fault).

The rest of this paper is organized as follows: In section 2, we introduce the weighted MPG-D defect part level estimation model. Section 3 introduces the application of linear programming to select optimal test patterns from an n-detection test set. Section 4 summarizes the experimental results and compares the surrogate fault coverage of our method and that of the n-detection method on ISCAS85 benchmark circuits. Finally, in section 5, we conclude our research results.

2 Defect Part Level Estimation Model

To generate a test set, test generators always use some objective functions to determine if the final objective is achieved. For the single stuck-at fault oriented ATPG tools, the objective function is the single stuck-at fault coverage, and the objective is to maximize, to 100% if possible, the fault coverage.

For the n-detection methods, the objective is to detect each stuck-at fault by different patterns at least n times. As long as n-detection on a fault is achieved, this fault is dropped from the fault list and no longer targeted under additional tests. This process can be explained by the following model: Given a set of test patterns $T = \{t_1, t_2, \ldots, t_n\}$ and fault set $F = \{f_1, f_2, \ldots, f_m\}$, define the objective function as:

$$M(F,T) = \sum_{i=1}^{m} M(f_i, T)$$

where

$$M(f_i, T) = \begin{cases} 1 & \text{if } f_i \text{ is detected less than } n \text{ times by } T \\ 0 & \text{otherwise.} \end{cases}$$

The objective of the n-detection methods is to generate a set T so that the objective function $M(F, T)$ is minimized.

However, this step objective function of n-detection is not optimal because as the number of detections increases, the objective function should gradually decrease since as more defects are detected at each site, the probability of exciting one of the remaining defects becomes smaller.

Consider Figure 1, the entire test space available at a test site in the circuit is represented as a box in the Venn diagram. Each oval represents the portion of that test space which will excite a particular defect at that test site (given that the site is observed). Thus, a defect is detected when a test is chosen so that it falls within that defect's corresponding oval (Multiple defects may in some cases be detected by the same test pattern). Once a defect is detected, detecting it again will not further reduce the defect part level, therefore, defect's oval no longer appears among those undetected defects. At any particular point in time, the probability of exciting an as yet undetected defect given that the site is observed is simply the ratio of the area of the test space covered by the ovals to the total area. Thus, the probability of exciting an as yet undetected defect tends to decrease as test patterns are applied and defects are detected.

Based on this observation, a defect part level estimation model, MPG-D model, was proposed [3]. Unlike other defect part level estimation models which depend upon fault coverage, it does not go to zero when fault coverage reaches 100%. In addition, it represents the fact that the probability of excitation decreases as the number of site observations increases and that more than one defect may be present at any fault site.

The original MPG-D model defines the relationship between the number of times a site is observed and its contribution to the *DPL*. First of all, this model assumes the uniform distribution of all the defects across the fault sites. Thus, for each fault site j, an initial contribution to the overall *DPL* is defined as:

$$DPL_j(0) = \frac{1 - Y}{\text{No. of Fault Sites}} \quad (1)$$

where Y is the manufacturing yield. After the application of a test pattern sequence $T = \{p_1, p_2, \ldots, p_k\}$, where $k \geq 1$, some fault sites have been observed, some have been excited, and some defects have been detected. Thus, refer to Figure 1, the probability of exciting an undetected defect at a site given that site is observed would be reduced as more test patterns are applied. This probability has been studied and shown to be a decaying exponential function of the number of times that site has been observed previously and a time constant ϕ [3]:
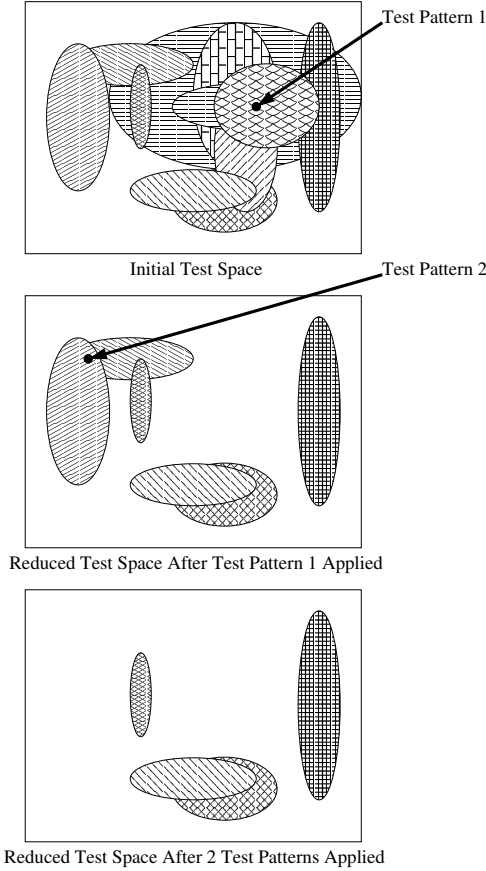
Figure 1. The probability of excitation decreases as testing progresses

Figure 2. Weight Function Generation

$$P_{excite} = e^{-\frac{obs_j}{\phi}} \qquad (2)$$

where obs_j means the total number of observations of test site j.

Since the individual probabilities are disjoint, the overall *DPL* is merely the sum of the contribution from each individual site. Therefore,

$$DPL(T) = \sum_{j=1}^{\text{No. of Fault Sites}} DPL_j(k) \qquad (3)$$

And for this test set T with k patterns ($k \geq 1$), each site's defect part level contribution is given in the following equation [3]:

$$DPL_j(k) = DPL_j(k-1)(1 - A * e^{-\frac{obs_j}{\phi}})^{obs_{jk}} \quad (4)$$

where A and ϕ are two constants related to the manufacturer's process and circuits, and $DPL_j(k-1)$ is the defect part level contribution when the first $k-1$ test patterns are applied to the circuit under test. The value obs_j means the total number of observations of test site j so far. The value of obs_{jk} is 1 if fault j is observed by vector k, otherwise it is 0.
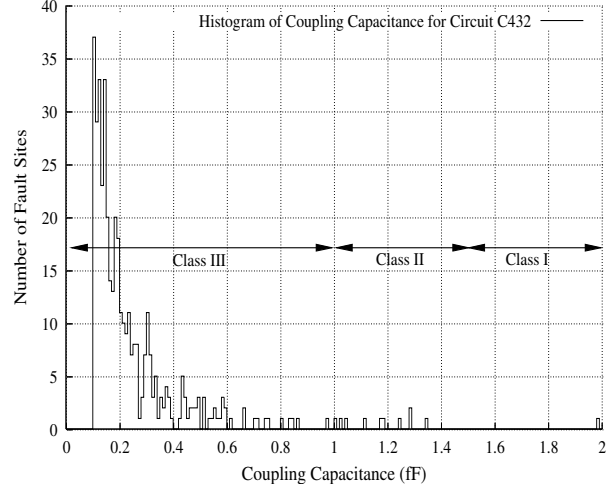
Although the original MPG-D model can be used to predict the *DPL*, it suffers from the limitation assumption that all sites in the circuit are equally sensitive to defects. In reality, we find that each site in the circuit has different physical and geometric characteristics that make it more or less susceptible to a uniform distribution of defects. Therefore, each site's contribution to the overall *DPL* is not uniform but instead is a strong function of its physical layout. Larger sites with more die area are more likely to suffer from defects than smaller sites. In order to create a more accurate model, we must account for the physical characteristics as represented in the layout of the circuit. In this paper, we will use a refined MPG-D model to estimate the *DPL* resulting from the application of a given test set. We refer to this model as the weighted MPG-D since we will assign different weight for each site in the circuit based upon the physical and/or electrochemical characteristics of each site. In general, the weights may be a function of all the characteristic such as (but not limited to) the physical area of the site, proximity to adjacent sites, proximity to active areas, ion implantation energy, diffusion gradient, etc.

As recent studies have shown [2][13], in the deep submicron technology era, the problems due to increasing coupling capacitance have a significant adverse effect on the proper function and performance of VLSI system. Therefore, for simplicity, in this paper we will employ the coupling capacitance extracted from the layout as an indicator to generate the weight function and refine the original MPG-D model. First, the circuit layout was generated with Cadence Silicon Ensemble in TSMC 250nm 3V 3-metal technology. Commercial parasitic extraction tools are used to extract the coupling capacitance of each site. To simplify the problem without losing generality, we classify the fault sites into several classes based on their greatest coupling capacitance, for the classes with bigger coupling capacitance, we assign

a bigger weight, for the classes with smaller coupling capacitance, we assign a smaller wight. To show these steps in detail, the histogram of coupling capacitance for circuit C432 fault sites is shown in Figure 2. For this circuit, we set up two thresholds for to separate the fault sites into three classes, if the coupling capacitance of a fault site is bigger than the first threshold (1.5fF), we put them into class I; if the coupling capacitance is between the second threshold (1fF) and the first threshold (1.5fF), we put them into class II; For all other sites, we put them into class III. Therefore the weight function is defined in the following step function:

$$ w_j = \begin{cases} \beta_1 & \text{if fault site } j \text{ is in class I} \\ \beta_2 & \text{if fault site } j \text{ is in class II} \\ \beta_3 & \text{otherwise.} \end{cases} $$

where $\beta_1 \geq \beta_2 \geq \beta_3$. In our experiments, we choose the weights between 1.0 and 0.85.

Now we have the weighted MPG-D model defined in equation (5):

$$ DPL(T) = \sum_{j=1}^{\text{No. of Fault Sites}} w_j * DPL_j(k) \qquad (5) $$

The solid line in Figure 3 shows an example of *DPL* contribution estimated by equation (4) with the weight of 1.0. The curve shows that as the number of observations on test site j increases, the contribution to the overall defect part level decreases as an exponential function.

Now we have derived the *DPL* estimation function shown in equation (5), our objective is to generate a test pattern set so that the *DPL* is minimized. Instead of generating the test patterns directly, our method selects an optimal pattern set from a big n-detection superset. The n-detection superset are generated by running the standard ATPG tools repeatedly until every fault is detected n times.

As we discussed before, although n-detection test set could effectively reduce the defect part level, the number of generated patterns is much larger than conventional methods. Limitations on tester memory size and the high cost of testing time severely limits the number of test patterns that can be applied to a device in a commercial setting. As a result, a post processing optimization process is required to select the best subset of patterns from the n-detection superset to use as the actual test set. In the next section, we are going to introduce a linear programming based method to select the patterns from this big n-detection test pattern set.

3 Optimal Test Pattern Selection

In this section, we develop a linear programming method to solve the optimal test pattern selection prob-
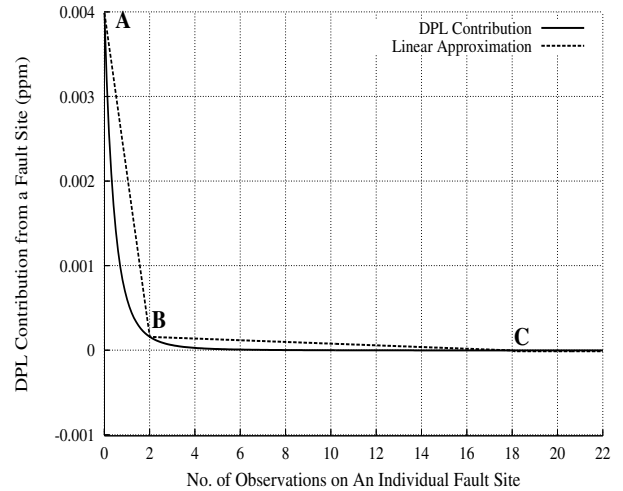

Figure 3. Linearizing the *DPL* Reduction Curve

lem discussed in section 2.

Linear programming is being widely used today in a wide variety of optimization problems, including the VLSI design and testing problems [5][7]. The final goal of our linear programming model is to select an optimal or near-optimal set of test patterns from a superset with the objective that the objective function (5) is minimized.

Since the *DPL* function defined in (5) is not a linear one, it is not easy to minimize this objective function directly. Therefore, we first linearize the weighted MPG-D model in terms of equations and constraints. Consider the *DPL* contribution curve shown in Figure 3 which shows the relationship between the number of detections for each fault and the expected value of the objective function for a specific fault, if we want to generate the optimal test pattern set which results in the minimum expected value, we can linearize the weighted MPG-D curve with the three linear segments shown in Figure 3 to approximate the real objective function, and this approach will significantly improve the solvability of the original problem without sacrificing much accuracy.

From the previous discussion, we can define the problem as follows: Given a fault dictionary $\Pi \subseteq \mathcal{T} \times F$, where $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ is the big n-detection set generated by running ATPG tools repeatedly and $F = \{f_1, f_2, \ldots, f_m\}$ is the fault set. If $(t_i, f_j) \in \Pi$, pattern t_i detects fault f_j. The problem is how to select a set $T \subseteq \mathcal{T}$ so that $|T| \leq u$, where u is the upper bound allowed for set T, and at the same time, the defect part level is minimized. Therefore, the objective function of

the problem is defined as:

$$Min : DPL(T) = \sum_{j=1}^{m} w_j * DPL_j(v) \qquad (6)$$

where m is the number of stuck-at faults in the circuit, w_j is the weight assigned to fault j, v is the number of detections on fault f_j and $DPL_j(v)$ is the individual defect part level contribution from fault site j.

Since $DPL(T)$ is not a linear function, it is not so easy to find the optimal solution of objective function (6) directly. Therefore, we can approximate $DPL_j(v)$ to a linear function $\overline{DPL}_j(v)$ so that we can use the standard linear programming method to solve it. Then the objective function becomes:

$$Min : DPL(T) = \sum_{j=1}^{m} w_j * \overline{DPL}_j(v) \qquad (7)$$

where $\overline{DPL}_j(v)$ includes the three linearized segments shown in Figure 3. These three lines correspond to three linear functions (here, they are named as M_1, M_2 and M_3, respectively) as follows:

$$\overline{DPL}_j(v) = \begin{cases} M_1 = \frac{DPL_j(b) - DPL_j(0)}{b} v + DPL_j(0) \\ \qquad \text{(If } 0 \leq v < b) \\ M_2 = \frac{DPL_j(c) - DPL_j(b)}{c-b}(v-b) \\ \qquad + DPL_j(b) \quad \text{(If } b \leq v < c) \\ M_3 = 0 \quad \text{(If } v \geq c) \end{cases} \qquad (8)$$

where v is the number of detections on a fault, b and c are the x-coordinate values of the points B and C in Figure 3.

We can also find that in each segment, (e.g., $0 \leq v < b$, $b \leq v < c$ and $v \geq c$), $\overline{DPL}_j(v)$ is always the maximum value of the three functions defined in (8). Therefore, for all $v \geq 0$, the three equations in (8) are equivalent to the following objective function:

$$\overline{DPL}_j(v) = \text{Maximum}\{M_1, M_2, M_3\} \qquad (9)$$

In the fault dictionary, for any $t_i \in \mathcal{T}$, we define an indicator variable x_i such that if t_i is in T, $x_i = 1$; otherwise, $x_i = 0$. Therefore,

$$x_i = \begin{cases} 1 & \text{If} \quad t_i \in T \\ 0 & \text{If} \quad t_i \notin T \end{cases} \qquad (10)$$

Since the total number of patterns in T is limited to u, we have

$$\sum_{i=1}^{n} x_i \leq u \qquad (11)$$

By using the indicator variable, we can easily find that v, which is the number of detections on a fault can be expressed as:

$$v = \sum_{(t_i, f_j) \in \Pi} x_i \qquad (12)$$

Here, in equations from (6) to (12), $1 \leq i \leq n$ and $1 \leq j \leq m$.

Combining equations (7), (8), (9), (10), (11) and (12) together, we can transform the original optimization problem into an integer linear programming problem since x_i must be an integer.

This problem can be solved by integer linear programming algorithm. However, integer linear programming problem is known an NP-complete problem without any efficient solutions. There are several ways to transfer them into a non-integer problem, such as interior point method, differentiable method and Taylor approximation method [17][21]. In this paper, we use the relaxation and rounding method to obtain a non-integer linear programming model.

First of all, the integer indicators $x_1, x_2, \ldots, x_n$ in the original problem are relaxed and redefined. After the problem is solved, we will round these variables back to integers. So, we first convert the constraint (10) and change it to a problem that can be solved more easily. We redefine x_i as the probability of test pattern t_i to be selected from superset $\mathcal{T}$. Thus, for every $1 \leq i \leq n$, we form the following constraint:

$$0 \leq x_i \leq 1 \text{ ,where } 1 \leq i \leq n \qquad (13)$$

With this transform, we can finalize our optimal pattern selection problem into the following linear programming model:

Min: $DPL(T) = \sum_{j=1}^{m} w_j * \overline{DPL}_j(v)$

Subject to:

$$\overline{DPL}_j(v) \geq \frac{DPL_j(b) - DPL_j(0)}{b} \sum_{(t_i, f_j) \in \Pi} x_i$$
$$+ DPL_j(0)$$

$$\overline{DPL}_j(v) \geq \frac{DPL_j(c) - DPL_j(b)}{c-b} [\sum_{(t_i, f_j) \in \Pi} x_i - b]$$
$$+ DPL_j(b)$$

$$\overline{DPL}_j(v) \geq 0$$

$$\sum_{i=1}^{n} x_i \leq u$$

$$0 \leq x_i \leq 1 \qquad (14)$$

where $1 \leq i \leq n$, $1 \leq j \leq m$.

Compared with the original integer linear programming one, this linear programming problem is much easier to solve. After this problem is solved, we need to round the solution of x_i back to 0 or 1 since we assume it is the probability of selecting test pattern t_i from the superset $\mathcal{T}$.

Since the definition of x_i is changed, we need to verify that the final result of the x_i satisfies the original conditions (10) and (11). Suppose the solutions of the linear programming model above are sets $\{\hat{x}_i | 1 \leq i \leq n\}$ and $\{\widehat{DPL}_j(v) | 1 \leq j \leq m\}$.

Then based on the definition of x_i, we can round x_i back to 0 or 1 by following the distributions:

$$\begin{aligned} \text{Probability}[x_i = 1] &= \hat{x}_i \\ \text{Probability}[x_i = 0] &= 1 - \hat{x}_i \end{aligned} \quad (15)$$

This distribution can be accomplished in the following way: 1. Generate a random number r; 2. Compare r and $\hat{x}_i$; 3. If $r \leq \hat{x}_i$, then x_i is set to 1, otherwise, x_i is set to 0. By rounding the result set $\hat{x}_i$ in this way, the original constraint equation (10) is satisfied.

Now, we need to consider the original constraint (11). It is possible that this constraint is violated; However, based on the probability theory [6], if we consider the expected value of the results, we can get the following equation:

$$\begin{aligned} E[\sum_{i=1}^{n} x_i] &= \sum_{i=1}^{n} E[x_i] \\ &= \sum_{i=1}^{n} P[x_i = 1] \\ &= \sum_{i=1}^{n} \hat{x}_i \leq u \end{aligned} \quad (16)$$

In the derivation here, the first equality is because the operator is linear, even for random variables that are dependent, and the third one is from equation (15). Therefore, from equation (16), we can see that constraint (11) is satisfied on the "average". That means the expected value of the result is limited to the upper bound of the set T, which satisfies the constraint (11) of the original problem.

From the discussion in this section, we have set up a linear programming model which attempts to obtain the minimum objective function defined in section 2 for a test pattern set selected from a superset. The whole optimization method can be summarized as the follows:

First, an n-detection test pattern set as the test pattern superset $\mathcal{T}$ and the corresponding fault dictionary $\Pi = \mathcal{T} \times F$ are generated, then the linear programming model which is discussed in this section is formed and

solved. After the linear programming problem is solved, the result is rounded to select the set T from the original superset $\mathcal{T}$. Finally, the resistive bridge fault is used as the surrogates and the set selected is input to the surrogate fault simulator PROBE [11] to get the resistive bridge fault coverage as the indicator of the test quality.

4   Experimental Results

The linear programming problem presented in this paper were solved using an AMD Athlon XP 2800 PC running the Redhat Linux operating system; The bridge fault simulator PROBE are working on a SUN Ultra-4 workstation running Solaris 5.7 operating system. Standard n-detection test patterns were generated by running Mentor FastScan^{TM}, a commercial ATPG tool, repeatedly. The linear programming problem is solved by So-Plex 1.2.1, a linear programming solver based on sequential object-oriented simplex algorithm written in the C language [22].

4.1   Experiment Design

We recognize that it is virtually impossible to get the actual defect coverage of a test set without intentionally inducing defects in each device and conducting a detailed deconstruction of all devices. However, to show the effectiveness of linear programming based pattern selection methodology, we designed an experiment that would compare the surrogate fault coverage between n-detection method and pattern selection method we proposed. In the experiment, there are two independent variables (circuit and number of patterns generated) and the resulting surrogate fault coverage is compared. Our experiments use the ten non-trivial ISCAS85 benchmark circuits. For each circuit, we generate a collapsed stuck-at fault list and run standard ATPG tool repeatedly to get an n-detection superset (here in the superset, n is set to be 15). Meanwhile, 2-detection set to 7-detection test sets are also generated by the same method. Then, we use the pattern selection methodology discussed before to select the test patterns from this 15-detection superset and generate the set of test patterns, whose sizes are limited to the 2-detection to 7-detection test sets, respectively. When all the test sets are ready, we use PROBE simulator to do the resistive bridge fault simulation and get the fault coverage as the final results.

We are also interested in varying the number of test patterns to reveal the trade-offs between test set size and the resulting surrogate fault coverage. Here, we generate a test pattern set which has 100% single stuck-at fault coverage and use the number of test patterns generated as the base number k. Specifically, the test set size limits are varied from one times (k) to nine times $(9k)$ the number of test patterns that would be applied using
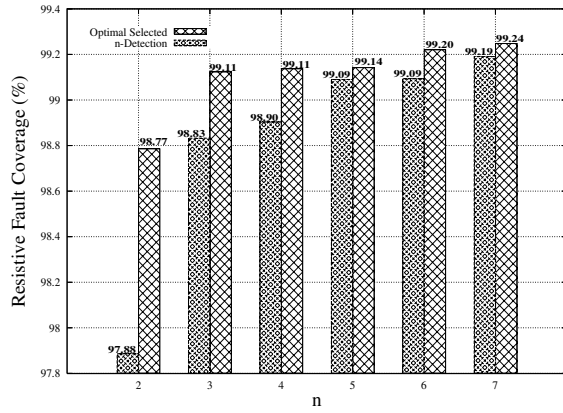
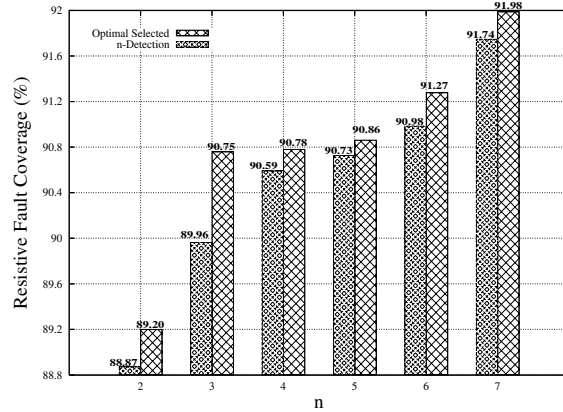**Figure 4. Bridge fault coverage vs. test set size for circuit c432**

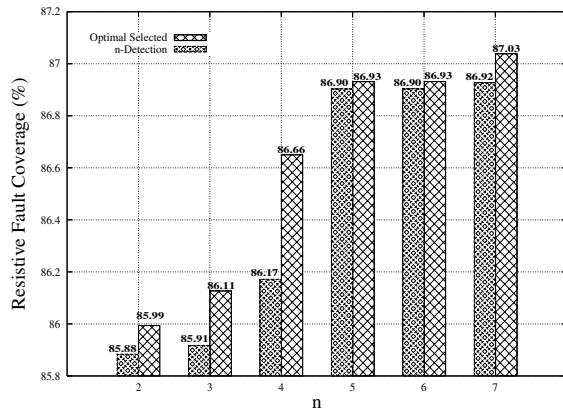**Figure 5. Bridge fault coverage vs. test set size for circuit c2670**

Figure 6. Bridge fault coverage vs. test set size for circuit c3540

Figure 7. Bridge fault coverage vs. test set size for circuit c5315
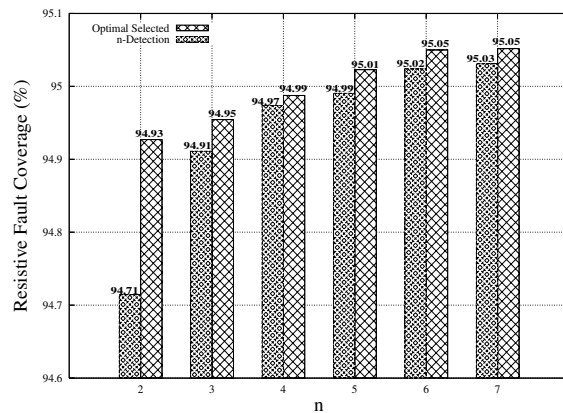

Figure 8. Bridge fault coverage vs. test set size for circuit c7552

standard best practice ATPG tool. Subsequently, we use the ATPG tool to generate a 15-detection test pattern superset. From this superset, subsets of various sizes were selected for each circuit. Then the bridge fault simulation is performed to get the final bridge fault coverage.

For each of the circuits, test generation methodologies, and number of patterns generated, we used PROBE resistive bridge fault simulator to get the bridge fault coverage as the indicator of the defect coverage.

4.2 Experiment Results

Figures 4 to 8 shows the resistive fault coverage of the different test pattern size on different circuits and methods. n is the number we used in n-detection method, in these figures, n changes from 2 to 7, which means each fault is detected 2 to 7 times. The numbers on the top of the columns indicate the resistive fault coverage for two different methods.
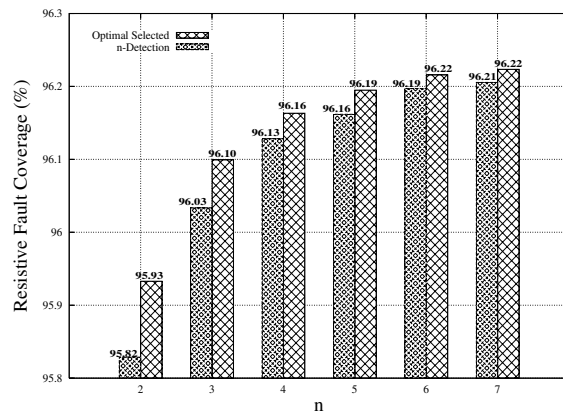
Table 1. CPU time to generate the 15-detection superset and to select optimal test patterns

Circuit	CPU_{15-det}	CPU_{opt}	$T_{overhead}$
c432	112.6s	5.5s	4.9%
c499	122.7s	6.7s	5.5%
c880	223.6s	12.3s	5.5%
c1355	456.8s	29.6s	6.5%
c1908	1092.5s	103.3s	9.5%
c2670	1403.4s	198.5s	14.1%
c3540	1982.5s	567.3s	28.6%
c5315	2010.5s	579.3s	28.8%
c6288	1490.3s	386.9s	25.9%
c7552	3503.5s	1109.5s	31.7%

Figure 9. Resistive bridge fault coverage vs. test set size for circuit c880

Based on the superset test patterns number of patterns generated by n-detection method, an optimal test pattern set is selected such that it has the same number of patterns as n-detection. The other circuits have the similar results as these figures show. The results shown here verify that optimal pattern selection method always get a higher bridge fault coverage than the n-detection methodology as the size of the test set is same. These results also confirm the correctness of weighted MPG-D model.

When the number of patterns is smaller, the surrogate fault coverage difference between the two methods is bigger, while with the number of patterns increase, the difference is smaller. This is because with the increase of the number of patterns in the test set, the improvement on the resistive fault coverage is becoming harder and harder to get, also the overlap between the two test sets is bigger since our superset is a 15-detection set generated by the same n-detection methodology, thus, the difference is smaller for bigger n in these figures.

Table 1 shows the time required for test pattern generation and optimization using the proposed algorithm for each of the ten ISCAS85 circuits. Column CPU_{15-det} indicates the CPU time used to generate the 15-detection superset, CPU_{opt} shows the CPU time required to select the optimal test pattern sets and column $T_{overhead}$ shows the CPU time overhead which is calculated as the ratio of optimization time to generation time. We can observe from the table that depending on the circuit size, the CPU time overhead varies from 4.9% to 31.7%.

4.3 Economic Implications

Management relies upon the expertise and wisdom of test engineers to define the test regimes for mass manufactured integrated circuits. Test engineers are often faced with the critical question: How many test patterns should be applied to the device under test?". Their decision has an enormous impact on the quality and prof-
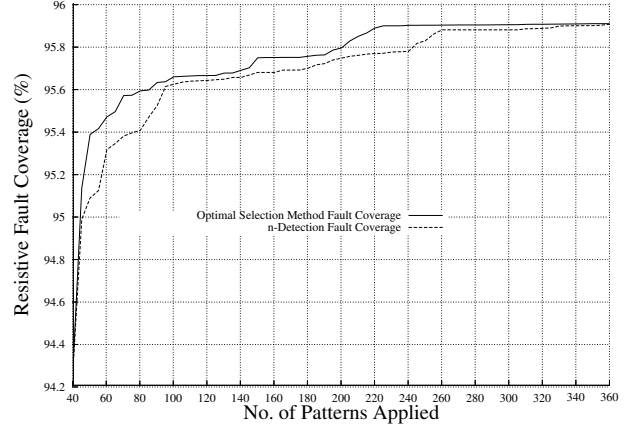
itability of the product. Current industry testing regimes typically consist of a combination of DC parametric, IDDQ, functional, and scan stuck-at fault based tests. Our results and comments are focused upon the scan stuck-at fault based tests. Applying too many test patterns will increase the test time, increase the required tester memory, and increase the cost of testing. In contrast, if the test pattern set is too small, unmodeled defects may be missed resulting in increased costs due to product returns and a loss of confidence in the manufacturer. In order to make a rational decision, one must take into account many factors including the cost of testing time, the desired defect coverage, the size of available tester memory, the expected volume of production, and contractual obligations. The analysis presented in this paper can be used as a decision support aid for test engineers. For example, in Figure 9, we use circuit c880 as an example to show a graphical representation of the trade-offs in test quality in terms of surrogate fault coverage and the test set size. For circuit c880, a 1-detection test set includes 40 patterns, we vary the number of patterns applied from k=40 to 9k=360. Here, both optimal pattern selection method and n-detection method are used and the results are compared, it is obvious that optimal pattern selection method always results in a higher fault coverage than n-detection. Graphs such as this provide a visual guide to aid the test engineer in making trade-off decisions. Similar experiments are also carried on the other ISCAS85 circuits. This graphical method can also be used to determine the number of patterns generated by two different methods when the same bridge fault coverage is achieved.

5 Conclusions

In this paper, we presented an optimal test pattern selection methodology based on the weighted MPG-D

model. The coupling capacitance value extracted from the benchmark circuit layout is used as the indicator to select the weight for each fault site to estimate the *DPL*. The objective of the methodology is to select a subset of test patterns from a superset that results in the optimal defect coverage for a given, fixed test size. Comparison of the method to n-detection showing the effectiveness of optimal pattern selection at increasing defect coverage. The primary result of this research is the increase of the resistive bridge fault coverage. We achieved this result by the formulation of a linear programming model which was then solved based on appropriate constraints for the given circuit. The experimental results proved that this method is effective in detecting the unmodeled defects and thus improve the defect coverage. We believe that the increase in the surrogate fault coverage will allow the methodology to be incorporated into commercial practice. Finally, a graphical representation of the trade-off between the test quality and test size is presented to aid the test engineer in making decisions regarding trade-offs between test set quality and test set size.

6 Acknowledgment

References

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

[2] Kwang-Ting Cheng, S. Dey, M. Rodgers, K. Roy, "Test challenges for deep sub-micron technologies," *Proc. Design Automation Conference*, pp.142-149, 2000.

[3] J. Dworak, M. R. Grimaila, S. Lee, L. C. Wang and M. R. Mercer, "Enhanced DO-RE-ME Based Defect Level Prediction Using Defect Site Aggregation - MPG-D," *Proc. International Test Conference*, pp.930-939, 2000.

[4] J.Dworak, J.D.Wicker, S.Lee, M.R.Grimaila, K.M.Bulter, B.Stewart, L.C.Wang and M.R.Mercer. Defect-Oriented Testing and Defective Part Level Prediction. *IEEE Design and Test of Computers*, 31-41, Jan.-Feb. 2001.

[5] F. Fallah, S. Devadas and K. Keutzer, "Functional vector generation for HDL models using linear programming and boolean satisfiability," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 20(8), pp.994-1002, Aug. 2001.

[6] W. Feller, *An Introduction to Probability Theory and Its Applications, 3rd Edition*, John Wiley and Sons, New York, 1968.

[7] P. F. Flores, H. C. Neto and J. P. Marques-Silva, "An exact solution to the minimum size test pattern problem," *ACM Trans. on Design Automation of Electronic Systems*, 16(4), pp.629-644, Oct. 2001.

[8] M. R. Grimaila, S. Lee, J. Dworak, K. M. Bulter, B. Stewart, H. Balachandran, B. Houchins, V. Mathur, J. Park, L. C. Wang and M. R. Mercer, "RE-DO - random excitation and deterministic observation - first commercial experiment," *Proc. VLSI Test Symposium*, pp.268-274, 1999.

[9] R. Kapur, J. Park and M. R. Mercer, "All tests for A fault are not equally valuable for defect detection," *Proc. International Test Conference*, pp.762-769, 1992.

[10] A. Krstic and K.-T. Cheng, *Delay Fault Testing for VLSI Circuits*, Kluwer Academic Publishers, 1998.

[11] C. Y. Lee and D. M. H. Walker, "PROBE: A PPSFP simulator for resistive bridging faults *Proc. VLSI Test Symposium*, pp.105-110, 2000.

[12] Z. Li, X. Lu, W. Qiu, W. Shi and H. Walker, "A circuit level fault model for resistive opens and bridges," *ACM Trans. on Design Automation of Electronic Systems*, Vol. 8, No. 4, pp. 546-559, 2003.

[13] Jing-Jia Liou, A. Krstic, Yi-Ming Jiang, Kwang-Ting Cheng "Modeling, testing, and analysis for delay defects and noise effects in deep submicron devices" *IEEE Trans. on CAD of Integrated Circuits and Systems*, Vol. 22, No. 6, pp. 756-769, 2003.

[14] 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, 1995.

[15] S. D. Millman and J. P. Garvery, Sr., "An accurate bridging fault test pattern generator," *Proc. International Test Conference*, pp.411-418, 1991.

[16] I. Polian, I. Pomeranz, and B. Becker, "Exact computation of maximally dominating faults and its application to n-detection tests," *Proc. Asian Test Symposium*, pp.9-14, 2002.

[17] P. Raghavan, "Probabilistic construction of deterministic algorithms: approximating packing integer programs," *J. of Computer and System Science*, Vol. 37, pp.130-143, 1988.

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

[19] T. M. Storey, W.Maly, J.Andrews, and M. Miske, "Stuck fault and current testing comparison using CMOS chip test," *Proc. International Test Conference*, pp.311-318, 1991.

[20] C. Tseng, S. Mitra, S. Davidson and E. J. McCluskey, "An evaluation of pseudo random testing for detecting real defects," *Proc. VLSI Test Symposium*, pp.404-409, 2001.

[21] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*, Kluwer Academic Publishers, Jun. 2001.

[22] R. Wunderling, *Paralleler und Objektorientierter Simplex-Algorithmus*, ZIB technical report TR 96-09, Berlin, Germany, 1996.