Test-Data Volume Optimization for Diagnosis

Hongfei Wang, Osei Poku, Xiaochun Yu, Sizhe Liu,
Ibrahima Komara and R. D. (Shawn) Blanton
ECE Dept., Carnegie Mellon University
Email: {hongfeiw, blanton}@ece.cmu.edu

ABSTRACT

Test data collection for a failing integrated circuit (IC) can be very expensive and time consuming. Many companies now collect a fix amount of test data regardless of the failure characteristics. As a result, limited data collection could lead to inaccurate diagnosis, while an excessive amount increases the cost not only in terms of unnecessary test data collection but also increased cost for test execution and data-storage. In this work, the objective is to develop a method for predicting the precise amount of test data necessary to produce an accurate diagnosis. By analyzing the failing outputs of an IC during its actual test, the developed method dynamically determines which failing test pattern to terminate testing, producing an amount of test data that is sufficient for an accurate diagnosis analysis. The method leverages several statistical learning techniques, and is evaluated using actual data from a population of failing chips and five standard benchmarks. Experiments demonstrate that test-data collection can be reduced by > 30% (as compared to collecting the full-failure response) while at the same time ensuring >90% diagnosis accuracy. Prematurely terminating test-data collection at fixed levels (e.g., 100 failing bits) is also shown to negatively impact diagnosis accuracy.

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Testing

General Terms

Algorithms, Design, Reliability

Keywords

Test Cost; Diagnosis; Test Data Collection; Statistical Learning

1. INTRODUCTION

Given the presence of process variations and defects during production, integrated circuits (ICs) are typically tested to verify they satisfy performance specifications. Test patterns are generated and applied to the IC for this purpose. The output responses of failing ICs are recorded into failure-log files in case diagnosis is later performed. Generally, failure diagnosis achieves its best possible outcome when a large amount of test measurement data from a failing IC is available for analysis.

Different test-data collection strategies are adopted in industry. Some companies collect no test data, whereas others collect a finite amount of test data, irrespective of the characteristics of the failing IC under testing. Other companies even attempt to collect data for the entire production test set, despite the number of erroneous bits exceeding 1M. However, the cost of collecting test-measurement data can be significant due to the extra test time incurred for going beyond the first failing pattern. The data-storage cost can also be significant especially for high-volume products. Also, even when it is desirable to collect ample amounts of test data, it may not be possible due to the limited memory that modern ATE (automatic test equipment) has for storing test-response data.

DAC 2012, June 3-7, 2012, San Francisco, California, USA.

An adjacent area of research focuses on reducing test-execution cost. One related approach adopted in practice is multi-site testing, where several ICs are tested simultaneously. Multi-site testing improves test throughput and thus saves expensive ATE time, although it does not necessarily reduce test-data volume. The other commonly used and effective approach for reducing test-execution cost is test compaction [1-4]. This area is related since if fewer tests are performed, then less test data is produced. Instead of applying the entire set of generated tests, test compaction identifies a subset of tests and then uses them to determine the overall pass/fail status of an IC. Since a smaller number of tests are applied when compaction is employed, the total volume of the test data that can be collected is also reduced.

Reducing test data via test compaction may not be beneficial for diagnosis however since the objective is to determine chip pass-fail status at minimum cost. To understand the incongruence between test-execution cost reduction and diagnosis further, consider a "stop-on-first-fail" test strategy. For this case, test data is minimized but the negative impact on diagnosis is likely quite significant since there is minimal information for distinguishing the various sources that could be responsible for failure. Moreover, once a subset of tests is selected from the original set, it is fixed and little consideration is given to chip to chip variation. This means that different failing chips may require varying amounts of test-data for an accurate diagnostic analysis. A dynamic method can therefore be quite useful and could counter any negative effects on diagnosis due to test-cost reduction.

The focus of this work is on reducing the cost of test-data collection from a different perspective. In our analyses, we have observed for a substantial number of failing ICs that their diagnosis results do not change with an increasing amount of test data. Although it may seem intuitive that more test data would improve diagnostic resolution and accuracy, it turns out that it may not, or even worse, it may degrade. It is even possible that test data from the first few failing test patterns is sufficient for obtaining an accurate diagnosis result with optimal resolution. If the testing procedure can be terminated after a sufficient amount of data is collected for diagnosis, test-data collection can be reduced.

The histogram in Fig. 1 further illustrates the relation between diagnostic accuracy (defined in Section 2.4) and the amount of test data collected from an actual fabricated IC. Accuracy improves with more test data collected as evidenced by the overall increasing trend of peak heights in the histogram, though the slope of this trend decreases. About 78% of the total sample of 456 failing ICs examined required only 50 bits to perform an accurate diagnosis. On the other hand, if the test-data collection is terminated at a fixed level, e.g., 100 bits, insufficient information

is then gathered for an accurate diagnosis for some failing ICs, 51 for the data of Fig. 1.

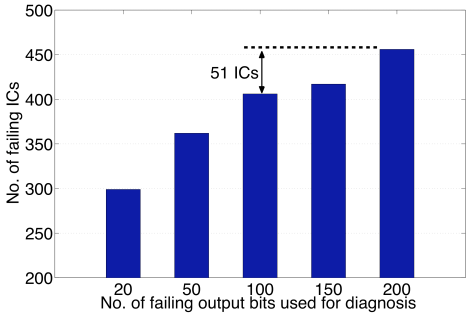

Fig. 1: Number of failing ICs that result in an accurate diagnosis when using the amount of test data shown on the x-axis.

In this work, the objective is to predict the minimal amount of test data necessary to produce both a precise and accurate diagnosis. We propose a method that dynamically predicts test termination for producing an amount of test data that is sufficient for obtaining a quality diagnosis result. The prediction model is learned from the test data produced by a sample of fully-tested ICs that have failed. The learned model is then deployed in production to predict the termination point when testing future ICs of the same type. Specifically, when an IC is tested, for each new failing pattern observed, the prediction model is invoked to determine if sufficient data has been collected for producing a quality diagnosis. The method is dynamic since test termination can vary from one failing IC to next rather than being fixed. In this way, the variation in individual ICs is taken into account in order to produce an optimized test-data volume for each, ensuring that the overall cost from collecting test data is minimized without sacrificing diagnosis quality. The test-termination prediction model is based on analysis of the failing tests and outputs of the ICs, which exhibit certain patterns that can be interpreted as a signal for judging the termination point of testing. Several statistical learning techniques are employed to discover these patterns from full-fail test data.

The work in [5] has the same objective as the work presented here. Specifically, their work dynamically determines test termination by evaluating the benefits of executing a subsequent test, based on the additional fault model coverage it would achieve. This work therefore assumes there is a positive correlation between model coverage and diagnostic resolution/accuracy. In our work, test termination is based on a model that is formulated using characteristics from real defective circuits, thus removing the barrier of abstraction that exists between fault models and actual defects. Moreover, our work complements the test-set compaction [1-4] and test-response compaction [6] based approaches. In other words, our work can be applied after these procedures to further minimize total cost.

The rest of this paper is organized as follows: Section II describes how the raw test-data is pre-processed for further analysis. Section III introduces several statistical learning techniques explored in this work for deriving a test-termination prediction model. Experiment validation is presented in Section IV and Section V summarizes our contributions.

2. DATA EXTRACTION

We begin with a description of the various data sources used in this work, and then address the issue of extracting useful information from the raw test data collected. Different metrics for data preparation and processing are also presented.

2.1 Sources

The first data set stems from a fabricated test chip from the LSI Corporation which consists of 64 identical ALUs (arithmetic logic units). A total of 233 test patterns were used to test this chip, including its scan-chains. Test data and diagnosis results for 493 chips were collected. Some chips were tested more than once, resulting in multiple test-data files. The test outcomes and diagnosis results for a particular chip, however, do not greatly differ among the various test executions. Therefore we arbitrarily decide to use the test data generated by the first test execution.

Besides the above real-world data set, experiments that use five benchmark circuits are also performed to demonstrate the viability of our method for a variety of circuit types. The circuits include c499 and c7552 from the ISCAS'85 benchmarks, s5378 and s7552 from the ISCAS'89 sequential circuits, and b12 from the ITC'99 benchmark suite.

The defect simulation framework from [7] is used to create a population of realistic failures from various benchmark designs, which are the largest circuits available that have simulation responses from layout-injected defects. In [7], randomly-selected defects are generated for each possible defect type that includes open, bridge, polysilicon, transistor stuck-open, and transistor stuck-closed. Defects are injected, one at a time, into the layout of each benchmark. An extracted netlist of each defective circuit is then simulated at the circuit-level using 100% stuck-at test sets generated by a commercial ATPG tool. The resulting simulation responses form the virtual test data for the failing population of circuits created.

2.2 Data Set Overview

In a typical statistical learning scenario, the data set used to learn a prediction model can be represented as an $n \times p$ matrix (denoted by $\boldsymbol{X}$), where each row is an input instance described by p features. A $n \times 1$ vector (denoted by $\boldsymbol{Y}$) is also provided, which contains the labels (e.g., quality diagnosis versus non-quality diagnosis) for each corresponding row $\boldsymbol{X}(i,\cdot)$.

The remaining part of this section describes the data scrubbing process used for obtaining $\boldsymbol{X}$ and $\boldsymbol{Y}$ from the test data and diagnosis results, respectively.

2.3 Feature Extraction

A defect in an IC, once activated, produces one or more errors that may be propagated to one or more outputs. Since each output has a fan-in logic cone, back-cone tracing is employed during diagnosis to locate possible defect sites. Consequently, determining the test-termination point for a given IC under test is based on simple observations made from the failing outputs. It is difficult however to use this raw data directly to discover underlying patterns and trends that suggest a termination point for testing. Therefore, the raw test data is processed and then organized into a set of features that are more readily usable for forming a test-termination prediction model. Although "passing patterns" (test patterns that do not cause any outputs to fail) can be useful in some forms of diagnosis [8], they are less focused upon since they can only provide indirect information concerning the source of failure. Table 1 summarizes the features derived from test data.

2.4 Evaluation Metrics

To perform a diagnosis for a failing IC, the circuit design, the corresponding test patterns, and the measured test response are

provided to a software-based diagnostic tool [9]. In order to create training data for forming a test-termination prediction model, an IC that has failed N test patterns is used to produce N diagnostic results. Specifically, test patterns that include the first one (either passing or failing) through the first failing pattern are used to produce the first diagnostic result, test patterns that include the first through the second failing pattern are used to produce the second diagnostic result, and so on until N diagnostic results are created using all N failing test patterns. Failing patterns are focused upon since the impact of passing patterns (patterns applied to failing IC that produce expected results) on the diagnostic result is typically less significant than the failing patterns. Each of the N diagnoses produces a list possible solutions or candidates, each of which typically includes a fault type (stuck-at, bridge, etc.), the possible location, along with a percentage score, where 100% indicates a perfect match between the fault simulation response and the tester response. A *golden* diagnosis result for a failing IC refers to the one generated when using all the applied tests, that is, from the first test pattern through the last failing test. The golden result does not have to have exact one fault candidate however. An *intermediate* result refers to any outcome generated using a subset of the test patterns corresponding to diagnoses 1 through N-1. Terminating the test of an IC early may still produce a diagnosis result that is the "same" as the golden one.

Table 1. Seven features extracted from raw test data. Each feature becomes a column vector in the matrix X, denoted by $X(\cdot, j)$.

Feature	Feature description
$X(\cdot,1)$	Number of test patterns (both passing and failing) that have been applied thus far.
$X(\cdot,2)$	Number of failing test patterns that have been applied thus far.
$X(\cdot,3)$	Total number of erroneous output bits that have been accumulated thus far.
$X(\cdot,4)$	Number of erroneous output bits for the current test pattern.
$X(\cdot,5)$	Total number of different erroneous output bits that have been accumulated thus far.
$X(\cdot,6)$	Number of unique erroneous output bits produced by the current test pattern.
$X(\cdot,7)$	Indicates the first failing test pattern.

Our objective is to predict the termination point during the test of an IC that will produce the golden result. In other words, for each failing pattern, we want to predict whether or not enough test data has been obtained to achieve an acceptable diagnosis result. To accomplish this objective, it is desirable to quantitively measure how close an intermediate result matches the golden result. To accomplish this, we examine two metrics:

$$m_A = \frac{\text{No.(intermediate candidates} \cap \text{golden candidates)}^2}{\text{No.(intermediate candidates)} \times \text{No.(golden candidates)}}$$

and

$$m_B = \begin{cases} 0, \text{ intermediate candidates} \cap \text{golden candidates} = \varnothing \\ 1, \text{ otherwise} \end{cases}.$$

The numerator in the equation for m_A is the square of the number of common candidates from the golden result and an intermediate result. Each candidate resulting from the diagnosis is associated with a rank (i.e., a percentage between 0~100%), representing the confidence in the result produced by the diagnosis tool. The highest ranked candidates are selected and used for set intersection. It should also be noted that for a specific common candidate, its rank may differ in the golden and intermediate results. The value range for metric m_A is [0,1], with 1 denoting a perfect match among the candidates from the intermediate diagnosis result and the golden one. For such a case, the ideal test-termination point is the failing test pattern that produces this intermediate diagnosis result.

Metric m_B takes binary values only: 0 for an intermediate diagnosis result that is distinctly different from the golden, and 1 if any candidates are in common. m_B is essentially a relaxed formed of m_A since

$$m_B = \begin{cases} 0, m_A = 0 \\ 1, m_A \neq 0 \end{cases}.$$

m_B may also be useful since any highly-ranked candidate may be sufficient for a follow-on application. For example, physical failure analysis of a failing IC typically only needs one highly-ranked candidate. In summary, m_A looks for an exact match of the golden result, whereas m_B captures a more relaxed notion of correctness. The metric values for intermediate diagnoses, obtained by employing either m_A or m_B are stored in $\boldsymbol{Y}$, sorted in the same order of the IC indices in $\boldsymbol{X}$.

3. LEARNING TECHNIQUES

Predicting test termination can be formulated as a classification task in statistical learning: for each failing test pattern, a decision is made about whether to terminate testing, or to continue applying further test patterns, by observing and analyzing the test data thus far collected. To this end, an adopted learning technique is expected to produce a model that is capable of classifying a new data instance by $\boldsymbol{X}_i^*$, representing a failure response for the most-recent test applied to an IC, with an estimated binary value $\hat{Y}_i^*$, denoting whether to terminate or continue testing. The asterisk notation (*) means the data instances are from an IC that is currently being tested. The learning process is considered a success if $\hat{Y}_i^*$ precisely matches the true Y_i^*, which could be obtained from using the entire test response for diagnosis once the testing of this IC is finished. A binary classification problem of this nature can be handled by various statistical learning techniques, one of which is chosen based on effectiveness.

3.1 Linear Regression

The most widely and frequently used statistical learning technique is linear regression [10]. In particular, linear regression with least-squares estimation builds a model that represents how the response vector $\boldsymbol{Y}$ depends on the feature matrix $\boldsymbol{X}$. This dependence is studied, and then generalized to predict future instances of $\boldsymbol{X}^*$ whose $\boldsymbol{Y}^*$ is unknown. The learned model can be expressed by

$$Y = X \cdot \beta + \varepsilon,\ \beta = \left[\beta_1, \beta_2, \cdots, \beta_p\right]^{\mathrm{T}},\ \varepsilon = \left[\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_p\right]^{\mathrm{T}},$$

where $\boldsymbol{\beta}$ is a $p \times 1$ vector of regression coefficients and $\boldsymbol{\varepsilon}$ is the error to be minimized by the least-squares estimation. $\boldsymbol{\beta}_i$ weighs the impact of the corresponding feature on the value of $\boldsymbol{Y}$. It allows hypothesis testing of the correlation between selected features and the response vector, which will be investigated in Section IV. For classification problems, a threshold is usually pre-specified to discretize the output values.

The specific framework for linear regression employed here is LASSO (Least Absolute Shrinkage and Selection) [11]; it minimizes the objective function

$$\left\| X \cdot \beta - Y \right\|_2^2 + \lambda \left\| \beta \right\|_1.$$

The first term in the above expression minimizes the sum-of-square errors. $\boldsymbol{\lambda}$ in the second term is a tuning parameter that controls the weight of penalty, formulated in the L1 norm for the sum of the absolute values of the coefficients. A large $\boldsymbol{\lambda}$ decreases the absolute value of $\boldsymbol{\beta}$'s, and may eventually forces some to 0. This beneficial property helps to yield a parsimonious model, in addition to achieving high accuracy.

It may seem that the use of a linear regression technique such as LASSO is misguided since we have not shown that the test-termination problem is inherently linear. But many linear regression approaches weakly depend on linearity, and employ techniques such as regularization to produce models that effectively deal with the relationships between X and Y. Evidence of this being the case here is demonstrated by the high accuracy produced by the LASSO model presented in Section 4.3.

3.2 *k*NN

Unlike other statistical learning techniques, *k*NN (*k*-nearest neighbor) [12] does not require a separate training process. It searches $\boldsymbol{X}$ and finds k instances that have the nearest distance from $\boldsymbol{X}_i^*$. These k instances are the k-nearest neighbors of $\boldsymbol{X}_i^*$ in this distance-based approach. Using their corresponding values from the response vector $\boldsymbol{Y}$, *k*NN applies majority voting to compute the $\hat{Y}_i^*$. *k*NN is simple but can be very accurate in many cases [13].

There are two factors concerning the implementation of *k*NN. One is the choice of "*k*", the number of nearest neighbors to be included when making classification. For the application considered here, $k = 1$ is found to produce the best results for an analysis that considered all integers $k < 16$. The other factor is the distance measurement. One particular metric is based on cosine similarity between two instances (e.g., $\boldsymbol{X}_p$ and $\boldsymbol{X}_q$), which can be expressed by

$$\theta = \cos^{-1}\frac{X_p \cdot X_q}{\|X_p\| \|X_q\|}.$$

According to [14], the cosine-based distance metric is more appropriate than Euclidean or Mahalanobis distances in many applications. It is thereby employed here given the task of test data collection.

3.3 Support Vector Machines

SVM is becoming more and more popular in recent years, due to its robust and accurate performance [13], and is favored by many researchers from disciplines such as electrical engineering and computational biology. We therefore consider SVM for this application as well.

For binary classification, SVM (support vector machines) aim to find a boundary in the hyperspace defined by the features of the data (column vectors in $\boldsymbol{X}$), according to their response vector $\boldsymbol{Y}$. The boundary creates a margin between the two classes of instances so that the minimal distance is maximized. In some cases, instances of the opposite classes are allowed to be mistakenly separated by the boundary. Tolerance of such errors produces a less accurate boundary describing $\boldsymbol{X}$ and its associated $\boldsymbol{Y}$, but may provide a better classification strategy for a future instance $\boldsymbol{X}_i^*$. The search for the aforementioned boundary can be formulated as an optimization problem of

$$\min_{w,\xi} \quad \frac{1}{2}w^T w + \lambda \sum \xi_i,$$

where w defines the direction of the boundary, and minimizing $\frac{1}{2}w^T w$ is equivalent to maximizing the margin. The parameter $\boldsymbol{\lambda}$ (>0) is a tuning parameter that penalizes the error ξ_i, i.e., the distance associated with misclassification. Solving for the vector w can be achieved by employing kernel functions, as explained in [13]. Typical kernels include linear, polynomial, radial, and sigmoid. In our experiments, the linear kernel produced the best tradeoff between accuracy and efficiency.

3.4 Decision Tree

Decision-tree models are straightforward to create and can be easily interpreted by users with even limit statistical background. In addition, they perform well as compared to other statistical learning techniques for classification tasks.

In this work, we apply CART (Classification And Regression Trees) [15] to build a decision-tree prediction model. Unlike other tree-constructed algorithms, the splitting rules in CART are guided by the Gini measure of impurity [13,15]. CART recursively partitions the data, described by $\boldsymbol{X}$ and its associated response $\boldsymbol{Y}$, until the stopping criteria is satisfied which includes, for example, a minimal number of instances in the resulting partitions. The decision-tree paths to leaf nodes, representing the binary classification decision (terminate or continue testing), are used to predict the failing pattern for test termination.

A large number of experiments are conducted for each aforementioned statistical learning technique in order to improve their classification accuracy, reduce run-time and memory space required. In addition, we scrutinize the parameter settings, such as the choice of kernel for SVM, the minimum node size to terminate partitioning for a decision tree, and the distance-measuring metric for *k*NN to maximize accuracy. Details of these analyses are omitted however due to limited space.

4. EXPERIMENT RESULTS

In this section, correlations among the output responses of failing ICs, the diagnosis results that identify possible failure locations, and the amount of test data needed to produce an ideal/acceptable diagnosis results are explored and studied.

4.1 Data Sets

Each failing test pattern is transformed into a data instance, based on the feature extraction procedure described in Section 2. In other words, multiple data instances are derived for each failing chip. The total number of chips for the LSI test chip and the benchmarks are 493 and 1,745, respectively, which produces 35,829 and 57,981 data instances, respectively. The raw test data are further processed before being given as input to the statistical learning techniques.

4.2 Feature Effectiveness

In Section II, we described the seven features extracted from the raw data. However, naively and intuitively, one could perform a coin toss to decide the test-termination point after observing a failing test. Thus it would be tempting to know if these features are chosen reasonably and are effectively correlated with the label vector storing test-termination points (denoted by $\boldsymbol{Y}$). Therefore, a hypothesis testing is employed to examine correlation. Specifically, two models are compared: one with all seven features, and the other one with a random variable drawn from a binomial distribution (with $p = 0.5$) to simulate a coin-tossing process. Both models are constructed using linear regression. An F-test is conducted under the null hypothesis that the coin-tossing model fits the data well. The test statistic is

$$F = \frac{(RSS_{ct} - RSS_{full})/(df_{ct} - df_{full})}{RSS_{full}/df_{full}}$$

where *RSS* is the residual sum of square errors, and *df* is the degrees of freedom. Subscripts *ct* and *full* denote the "coin-tossing" model, and the model using all the features, respectively. The resultant *p*-value is quite small (< 0.05) for this test, suggesting strong evidence against the null hypothesis. In other words, the seven features extracted from the raw test data are proved to be closely correlated with test-termination points that produce accurate diagnosis results, thereby providing more useful information for predicting the $\boldsymbol{Y}$ labels than a random coin toss.

4.3 Accuracy and Data Volume Reduction

For each data set, 90% of the IC population is randomly chosen for learning a statistical model. The failure responses, together with the diagnosis results, are processed into $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle$. The remaining 10% of ICs, whose failure output behaviors are denoted by $\boldsymbol{X}^*$, are reserved and not included in the learning process. Their response vector $\boldsymbol{Y}$ is assumed unknown and are used to gauge the effectiveness of the learned model. In other words, $\langle \boldsymbol{X}_i^*, \hat{\boldsymbol{Y}}_i^* \rangle$ denotes a failure response for any of the ICs in the reserved 10% population, and its corresponding action of whether to continue or terminate testing suggested by the learned model base on $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle$. We repeat this procedure 10 times as to perform a 10-fold cross-validation for each experiment to evaluate the average performance.

Test-termination prediction is inaccurate if the resulting diagnosis result does not match the full-data one as measured by one of the metrics m_A or m_B. Inaccuracy is typically due to early termination, which simply means insufficient test data has been collected to produce a quality diagnosis. Some inaccuracy is expected from statistical learning techniques since they produce data-driven, nondeterministic solutions. To prevent early test termination, a guard-band is imposed. Different from the guard-banding approach employed in [1,16], where multiple models are required to make identical decisions, the guard band invoked here uses a single model. Specifically, testing is only terminated after the model predicts test termination for *M* successive failing patterns, where *M* is an integer that can be 0 (no guard banding), 1, 2, etc. A sequence of consistently identical test-termination decisions increases the confidence that a learned model will produce a quality diagnosis result. Guard banding is expected to reduce the chances of early termination but the trade off is that test data-volume reduction (DVR) will be reduced.

Of course, besides accuracy, the other key factor affecting the performance of the model is the level of DVR, which can be measured by

$$DVR = \frac{1}{n}\sum\left(1 - \frac{|t_i^T|}{|T_i|}\right)$$

where $|T_i|$ is the test-data volume for the i^{th} IC analyzed. This is obtained by accumulating the number of failing output bits for all failing test patterns, from the first to the last. $|t_i^T|$ is the optimized volume calculated in a similar way, from the first pattern to the test-termination point predicted for this IC. The DVR for one IC is not statistically significant; thus it is averaged over the entire population of *n* failing chips per design.

All four statistical learning techniques, *k*NN, linear regression (LASSO), SVM, and decision tree, are applied to the industrial data set of a test chip from the LSI Corporation to measure both the accuracy and DVR. Metric m_A is used to measure the diagnosis results. The results are presented in Fig. 2 which shows that a model produced by any of the statistical-learning techniques investigated is able to effectively predict test termination with high accuracy, which can be further improved with guard banding. For example, in Fig. 2(a), with a guard band >16, the accuracy of *k*NN is increased by over 30% comparing to a guard band of two. On the other hand, although guard banding proves to increase the accuracy in most of the experiments, it is not always guaranteed that using a large guard band will necessarily improve accuracy of a learned statistical model. For example, in Fig. 2(c), when using SVM with guard bands from 16 to 20, the accuracy is slightly degraded. This non-monotonic behavior of accuracy is caused by using "too much" collected data, as opposed to early test-termination. Choosing a proper guard band is therefore critical to further improve the accuracy of a statistical model.

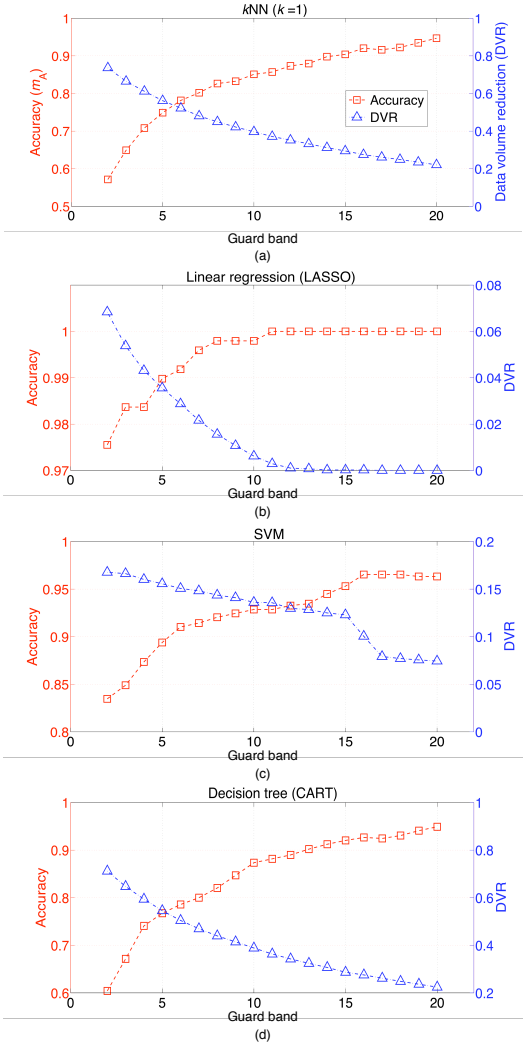

Fig. 2: Experiment results that compare test-termination prediction for an industrial test chip using: (a) *k*NN, (b) linear regression (LASSO), (c) SVM and (d) decision tree. Dashed lines with squares illustrate the accuracy of the models based on the metric m_A; dash-dot lines with triangles are the levels of DVR achieved. Different guard bands are applied, ranging from 2 to 20.

Unlike accuracy, DVR drops monotonically with the increase of guard band. There is a trade-off between accuracy and DVR. LASSO achieves the best accuracy. With a guard band greater than 11, it reaches 100%. The cost for this perfect accuracy

however is that there is little to no DVR benefit. SVM has lower accuracy than LASSO, but is better than *k*NN and a decision tree. However, the improved accuracy of SVM over *k*NN and a decision tree becomes negligible when the guard band is large. Moreover, although significantly greater than LASSO, the DVR achieved by SVM cannot compete with either *k*NN or a decision tree. The performance of a decision tree and *k*NN are very similar, with a decision-tree model being slightly better in terms of accuracy. For *k*NN, it is reported to lack robustness when a small *k* is adopted [13]. Consequently, considering the balance between accuracy and DVR, the best learning technique among those examined is the decision tree. Specifically, a decision tree exhibits, on average, an accuracy of over 90% when achieving a DVR of 32.4%. Tree models are also more easily constructed than SVM and LASSO, both of which require data scaling. The remaining experiments and discussion therefore focus on using decision-tree models.

It should be noted that, once a statistical model is constructed, it takes little time (less than a millisecond) to predict the test-termination point, especially for a decision-tree model. They therefore can easily be incorporated into a production test flow where prompt termination decisions are desired in the course of testing ICs to produce an optimized test-data volume.

4.4 Viability

To demonstrate the general applicability of this methodology, standard benchmarks are analyzed in addition to the test chip from the LSI Corporation. The results are summarized in Table 2.

Table 2. Decision-tree results using 10-fold cross-validation to obtain average accuracy and DVR.

Design	Performances (%) and guard bands (GB) used					
	Accuracy	DVR	GB	Accuracy	DVR	GB
c499	82.9	26.0	1	90.6	18.3	3
c7552	84.7	33.0	3	91.7	21.3	6
s5378	83.1	64.5	2	90.2	50.9	5
s9234	87.0	54.0	2	91.4	46.4	4
b12	83.7	21.5	1	90.1	13.1	3

Guard banding (GB) is expected to prevent early termination but the tradeoff is that DVR will degrade as evidenced by Fig. 2. Determining a GB level for test termination is a function of the design, the tests applied, the defects occurring, the acceptable accuracy, and the desired level of DVR. These areas of uncertainty are naturally handled however by employing a well-known statistical estimation method called cross-validation (Section 4.3). It is therefore straightforward to use cross-validation to identify the optimal level of GB value for a given design, test, and fabrication process. For the designs considered here, GB ≤ 6 ensures accuracy is greater than 80%. A GB > 20 is not required since a lower value has equivalent or superior results.

Including the experiment results from the LSI chip, 38.9% of the test-data volume can be reduced on average with an accuracy that exceeds 85.3%. For an average accuracy greater than 90%, the DVR decreases to 30.4%, which is still significant. The experiment results for standard benchmarks also indicate our method can be effectively applied to a variety of designs.

Generally, a higher accuracy level results in a smaller DVR, which means test cost is not minimized. In practice, the acceptable accuracy level should be determined considering both the test economics and the desired diagnosis accuracy.

4.5 Evaluation Using Different Metrics

As a final experiment, we repeat the analysis of the LSI test chip but instead employ metric m_B. The results are illustrated in Fig. 3.

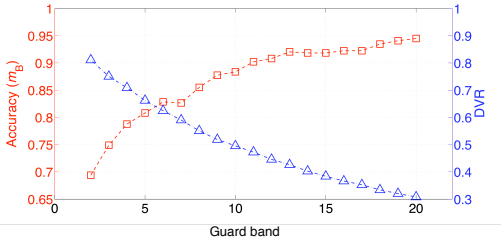

Fig. 3: Decision tree outcome using metric m_B for the LSI test chip.

Comparing with Fig. 2(d), both the accuracy and DVR are increased, as expected, due to using a less stringent metric m_B. The experiment is provided to illustrate that: 1) the performance of a learned model is subject to change when a different metric is used to evaluate diagnosis results and 2) an increase in DVR and accuracy can be expected if partially correct diagnosis results are deemed acceptable.

5. SUMMARY

This paper explored various aspects of using statistical-learning techniques to optimize the cost of collecting test data for producing a high-quality diagnostic result. Test data collected from full-tested ICs that fail are used to learn a model that allows the test-termination points to be accurately predicted. The learned model uses easily obtainable data (number of erroneous outputs, identification of unique erroneous outputs, number of failing test patterns, etc.) and thus can be used in real-time to decide when testing should be terminated. Experiment results from both industrial and simulated data sets demonstrate that the amount of test data needed to obtain an accurate diagnosis can vary from failing IC to failing IC.

6. REFERENCES

[1] S. Biswas and R. D. Blanton, "Statistical Test Compaction Using Binary Decision Trees,"*IEEE Des. Test. Comput.*, vol. 23, no. 6, pp. 452-462, 2006.

[2] L. Amati, C. Bolchini, and F. Salice, "Optimal Test Set Selection for Fault Diagnosis Improvement," *IEEE DFT*, pp.93-99, 2011.

[3] M. Shukoor and V. Agrawal, "A Two Phase Approach for Minimal Diagnostic Test Set Generation," *IEEE ETS*, pp. 115-120, 2009.

[4] Y. Higami et al., "Compaction of pass/fail-based diagnostic test vectors for combinational and sequential circuits," *IEEE ASPDAC*, 2006.

[5] L. Amati et al., "A Formal Condition to Stop an Incremental Automatic Functional Diagnosis," *IEEE DSD*, pp. 637-643, 2010.

[6] W. Cheng, K. Tsai, Y. Huang, N. Tamarapalli, J. Rajski, "Compactor independent direct diagnosis," *IEEE ATS*, pp. 204-209, 2004.

[7] W. C. Tam and R. D. Blanton, "SLIDER: A Fast and Accurate Defect Simulation Framework," *IEEE VLSI*, pp. 172-177, 2011.

[8] R. Desineni, O. Poku, and R. D. Blanton, "A Logic Diagnosis Methodology for Improved Localization and Extraction of Accurate Defect Behavior," *IEEE ITC*, 2006.

[9] TetraMAX ATPG [Online]. Available: http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Pages/TetraMAXATPG.aspx.

[10] S. Weisberg, *Applied Linear Regression,* 3rd edition. New York: Wiley, 2005.

[11] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," Technical report, University of Toronto, 1994.

[12] T. Cover and P. Hart, "Nearest Neighbor Pattern Classification," *IEEE Trans. Inf. Theory* , IT-11, 21–27, 1967.

[13] X. Wu and V. Kumar, *The Top Ten Algorithms in Data Mining*. Chapman & Hall/CRC Press, 2009.

[14] A.M. Qamar et al., "Similarity Learning for Nearest Neighbor Classification," *IEEE ICDM*, pp.983-988, 2008.

[15] L. Breiman et al., *Classification and Regression Trees*. Wadsworth, Belmont, 1984.

[16] S. Biswas and R. D. Blanton, "Specification Test Compaction for Analog Circuits and MEMS," *IEEE DATE*, pp. 164-169, 2005.