PADRE: Physically-Aware Diagnostic Resolution Enhancement

Yang Xue, Osei Poku, Xin Li and R. D. (Shawn) Blanton
Advanced Chip Testing Laboratory
www.ece.cmu.edu/~actl
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh PA 15213
{yxue, xinli, blanton}@ece.cmu.edu

Abstract

Diagnosis is the first step of IC failure analysis. The conventional objective of identifying the failure locations has been augmented with various physically-aware techniques that are intended to improve both diagnostic resolution and accuracy. Despite these advances, it is often the case however that resolution, i.e., the number of locations or candidates reported by diagnosis, exceeds the number of actual failing locations. Imperfect resolution greatly hinders any follow-on, information-extraction analyses (e.g., physical failure analysis, volume diagnosis, etc.) due to the resulting ambiguity. To address this major challenge, a novel, unsupervised learning methodology that uses ordinarily-available tester and simulation data is described that significantly improves resolution with virtually no negative impact on accuracy. Simulation experiments using a variety of fault types (SSL, MSL, bridges, opens and cell-level input-pattern faults) reveal that the number of failed ICs that have perfect resolution can be more than doubled, and overall resolution is improved by 22%. Application to silicon data also demonstrates significant improvement in resolution (38% overall and the number of chips with ideal resolution is nearly tripled) and verification using PFA demonstrates that accuracy is maintained.

1. Introduction

Diagnosis is a fast and non-destructive approach to preliminarily identify and locate possible defects in a failing IC [1]. It is a software-based method that analyzes the applied tests, the failed IC tester response, and its netlist/layout to produce a list of diagnostic candidates that represent the locations and sometimes behaviors/types of defects in the chip. Diagnosis can be then followed by physical failure analysis (PFA), a time-consuming and destructive approach for exposing the defect in order to characterize the failure mechanism [2]. Due to the high cost and destructive nature of PFA, the accuracy and resolution of diagnosis is of critical importance.

In addition to being an integral part of PFA, diagnosis results from a population of failed chips also serve as input for a number of applications in a variety of other areas. For instance, the diagnostic results can reveal both important statistics including the defect distribution or the primary yield detractors [3, 4], and useful feedback for evaluating and improving the quality of manufacturing test [4, 5, 6].

In practice, diagnosis tends to be non-ideal for a variety of reasons. Two such reasons include the limitation on test-set size, and the equivalent circuit I/O behavior that inherently exists among candidates. Because there is a trade-off between the time needed to both create and apply tests and the cost of test, it is always the case that not all possible defects are fully exposed when they are detected by the production test set. Even if a comprehensive test is economically viable, there still can be candidates that have equivalent logical behavior among the many locations within the IC that are specific to the standard cells used and their interconnections. Also the fault models employed for both test and the diagnosis are not perfect either, meaning it is quite likely that the actual defective behavior cannot be fully explained by the fault model(s) selected [7]. The overall result is an imperfect diagnosis that typically produces an accurate result but a non-ideal resolution. Specifically, more than one candidate is often reported, where one or more may correspond to the actual failing locations while many others do not. Figure 1 illustrates this point by showing the cumulative diagnosis resolution of an in-production commercial chip. It can be easily seen that less than 10% of the diagnosed chips exhibit ideal resolution.

However, it is possible to improve the resolution with add-on techniques that rely on existing diagnosis results. Improving diagnostic resolution requires the derivation of certain characteristics that enable good candidates to be distinguished from bad ones. For instance, there have been a number of proven heuristics that allow candidates to be effectively ranked. In [8, 9], it is suggested that candidates detected by more tester-passing patterns are less likely to capture the actual defective location(s). Other work reveals that the same neighborhood state of a good candidate should not be observed in both the Tester-Pass-Simulation-Fail (TPSF) and the Tester-Fail-Simulation-Fail (TFSF) patterns [1, 4, 10, 11, 12]. If a neighborhood state appears in both TPSF and TFSF patterns, the candidate is said to be inconsistent and is likely incorrect [1, 4, 10, 11, 12]. While these techniques are effective (e.g., the work in [10] reports a resolution improvement of 67% for 2,293 chips), they only

978-1-4799-0859-2/13/$31.00 ©2013 IEEE

utilize a limited amount of the tester and design data available for comparing and contrasting candidates. Our work here explores the use of additional information derived from fault simulation and ordinarily-available tester data for further improving diagnostic resolution. Specifically, chip- and candidate-specific features are created to both characterize and distinguish the diagnostic candidates. Some of the features are well known and involve the comparisons between the observed tester response and the fault-simulation response of a candidate. Other new features are also established in this work and are combined with existing features in order to characterize each chip failure and its corresponding candidates. The feature data from a population of candidates are then supplied to a classifier for learning a model that separates good candidates from bad ones.

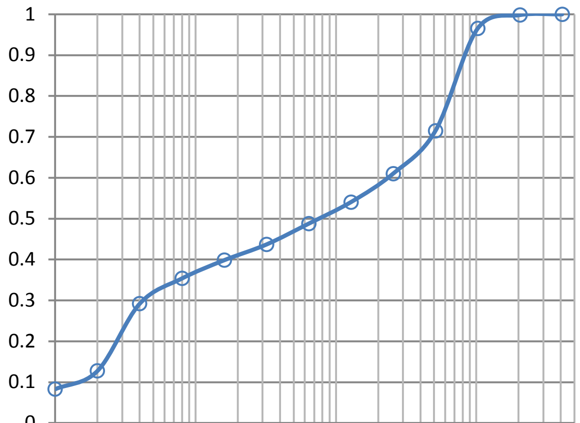

Figure 1: The cumulative diagnostic resolution distribution of a commercial chip shows that less than 10% of a population of 1,202 failed chips has an ideal resolution of one. The cumulative diagnostic resolution plot sorts the chips by the number of candidates for each chip. Each point on the plot shows the proportion of chips in the entire population that have candidates equal or below certain number, ranging from 1 to the maximum number of candidates any chip has.

One issue with creating a supervised classifier is the need for "training data" [13]. For the diagnosis application, this means we need a population of failed chips with candidates that are all correctly labeled as "good" or "bad". One obvious choice for deriving training data is through PFA of actual diagnosed chips. But because PFA is both costly and time consuming, it is very unlikely that (1) it will be explicitly used to identify bad candidates and (2) it will result in a training-set size that is statistically significant. **In this work, we have virtually eliminated this problem by a novel technique that derives labeled candidates (i.e., training data) from unlabeled candidates. Specifically, we use intuitive heuristics to identify failed chips that allow their corresponding diagnosis candidates to be correctly labeled, for the most part, as either good or bad.**

In past work [1, 4, 10, 11, 12], it has been shown that the consistency check was very adept at identifying bad candidates. We therefore employ a two-level classifier, where the first-level is a simple rule-base check of the neighborhood consistency of each candidate. The second-level classifier is learned from those candidates that pass through the first-level classifier. SVM (support vector machine) [13] is used in our work for learning the second-level classifier. It should be noted however that other classifiers (KNN [13], decision trees [13], etc.) can also be likely used as well.

SVM is a versatile and robust machine learning framework for performing classification based on training-data features [13]. Machine learning techniques such as SVM have been shown to be effective in various tester-time reduction methods [14-17], and in a variety of diagnosis applications [18, 19]. In [18], the authors use decision trees to identify subtle bridge defects, and in [19] the authors use SVM to correlate the board-level diagnosis results with the root-causes of failure. Nevertheless, these learning techniques are often limited by low-quality training sets. (Both [18] and [19] derive training data from simulation[1].) But it is well known that simulation of fault models rarely results in behaviors that match those exhibited by real defects [20]. An alternative for obtaining realistic training data is to use historical data, i.e., existing diagnosis results from other designs that have been labeled through PFA or other means. There are two major limitations in using historical data however. The first limitation has already been mentioned and again is the scarcity of labeled data from PFA. For a reliable performance, a classifier requires an adequate number of both good and bad candidates for training. A second limitation lies in the relevance of the previous diagnosis data. Using "old chip data" to train a classifier for a new design likely introduces error that substantially undermines the performance of the classifier.

As already mentioned, we solve this problem by deriving training data (i.e., labeled candidates) from a population of candidates using intuitive heuristics. This means that an abundant amount of training data specific to the actual chip under diagnosis will always be available for use.

In the remaining sections of this paper, we describe the details of the PADRE methodology in Section 2; Section 3 demonstrates the applicability of PADRE in experiments that use both virtual and real failed ICs. Finally, Section 4 discusses the experiment results and provides conclusions and directions for future work.

2. PADRE

Our approach for PADRE (Physically-Aware Diagnostic Resolution Enhancement) involves a two-level classifier that identifies bad candidates in the first level and good candidates in the second. PADRE takes as input the diagnosis results for a set of M failed chips $C =$

[1]Although simulation data is used in this work, it should be noted that it is not integral and is only used in order to verify accuracy.

$\{c_1, c_2, \ldots, c_M\}$. Each chip c_i has N ($N \geq 1$) diagnosis candidates $D_i = d_{i,1}, d_{i,2}, \ldots, d_{i,N}$, where each $d_{i,j}$ has P feature values $F_{i,j} = f_{i,j,1}, f_{i,j,2}, \ldots, f_{i,j,P}$. Each feature value $f_{i,j,k}$ is a particular characteristic of the candidate $d_{i,j}$ that includes data from the design (the logical netlist and/or the physical layout) or from the test data (the simulation and/or the tester).

PADRE consists of three major steps, namely candidate-feature construction, first-level identification of bad candidates, and second-level identification of good candidates. Each step is discussed in detail in sub-sections 2.1-2.3.

2.1 Candidate Features

Candidate features are specific design and test characteristics that differentiate good diagnosis candidates from bad ones. The candidate features now considered in PADRE are summarized in Table 1.

Feature	Description
no_nbrs	No. of neighbors for a given candidate
no_TPSFs	No. of Tester-Pass-Simulation-Fail (TPSF) patterns associated with a given candidate
no_TFSFs	No. of Tester-Fail-Simulation-Fail (TFSF) patterns associated with a given candidate
no_TFSPs	No. of Tester-Fail-Simulation-Pass (TFSP) patterns associated with a given candidate
no_TPSF_outputs	No. of Tester-Pass-Simulation-Fail (TPSF) outputs associated with a given candidate
no_TFSF_outputs	No. of Tester-Fail-Simulation-Fail (TFSF) outputs associated with a given candidate
no_TFSP_outputs	No. of Tester-Fail-Simulation-Pass (TFSP) outputs associated with a given candidate
tot_no_fail_outputs	Total no. of failing outputs of the Simulation-Fail patterns associated with a given candidate
min_no_fail_outputs	No. of failing outputs of the Simulation-Fail patterns that exhibit the smallest number of failing outputs
max_no_fail_outputs	No. of failing outputs of the Simulation-Fail patterns that exhibit the largest number of failing outputs
mean_no_fail_outputs	Average number of failing outputs of all the Simulation-Fail patterns
unique_no_fail_outputs	No. of unique failing outputs in the Simulation-Fail patterns among all chip candidates
no_pass_states	No. of different neighborhood states in the TPSF patterns
no_fail_states	No. of different neighborhood states in the TFSF patterns
no_incst_states	No. of neighborhood states exhibited by both the TPSF and TFSF patterns
candidate_entropy	The uncertainty level of a candidate's logic value as a function of its possible neighborhood states

Table 1: Diagnostic-candidate features that are based on various test and design characteristics.

The candidate features no_pass_states, no_fail_states, no_incst_states, and candidate_entropy characterize the physical characteristics of a candidate when it is both activated and sensitized. Specifically, the neighborhood state of a candidate is defined to be the logic values driven on lines that are in physical proximity of the candidate for tests that detect the candidate (i.e., TPSF and TFSP patterns) [1]. The neighborhood of a candidate, as illustrated in Figure 2, includes:

1. **Physical neighbors**: nets that are in close proximity of the candidate as determined by the design layout.
2. **Drivers**: inputs of the cell that drives the candidate.
3. **Side inputs**: side inputs of cells driven by the candidate.

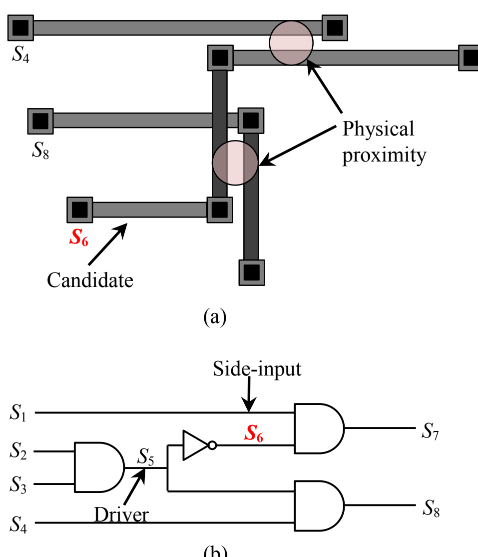

Figure 2: Example of a neighborhood for a candidate associated with net S_6: (a) the physical neighbors ≡ nets in physical proximity and (b) the logical neighbors ≡ driver and receiving-cell side inputs.

Despite the size of the circuit, the logic value of a candidate, whether faulty or fault-free, is assumed to be largely determined by the neighborhood state, i.e., the logic values of its neighbors [1]. The characteristics of the neighborhood may also provide an indication of the authenticity of a candidate. The heuristic is that if a candidate is indeed a site of failure, its failing behavior should be a consistent function of its neighborhood failing and passing states. More detailed discussions of the neighborhood-related features and other major features employed in PADRE are described next.

Entropy. The notion of neighborhood entropy [12] is used as a candidate feature (candidate_entropy). In [12], the

authors describe the entropy of a candidate as a measure of the level of uncertainty of its logic value with relation to its neighbors. As already mentioned, the logic value of a candidate is assumed to be correlated to its neighborhood site. Moreover, for a particular set of neighborhood states, the lower the candidate_entropy, the greater the correlation between the candidate and its neighborhood.

The feature candidate_entropy is the weighted average of the entropy of the observed neighborhood states. It is calculated as $H(X_d|X_n, G_d = g_d^k)$ by the following equation:

$$H(X_d|X_n, G_d = g_d^k) = -\sum_{j=1}^{2}[P(X_n = x_n^j|G_d = g_d^k) \times \sum_{i=1}^{2} P(X_d = x_d^i|X_n = x_n^j, G_d = g_d^k) \times \log_2 P(X_d = x_d^i|X_n = x_n^j, G_d = g_d^k)]$$

where X_d is the random variable that represents the actual value of the candidate d, x_f is a possible value of X_d, X_n is the random variable that represents the value of neighbor n of d, x_n is a possible value of X_n, G_d is the random variable that represents the fault-free value of d, and g_d is a possible value of G_d. $P(X_n = x_n^j|G_d = g_d^k)$ is the conditional probability of observing a neighborhood state x_n^j given the fault-free value of candidate g_f^k. $P(X_d = x_d^i|X_n = x_n^j, G_d = g_d^k)$ is the conditional probability of observing a candidate value x_d^i given the candidate fault-free value g_d^k and neighborhood state x_n^j.

If the correlation between X_n and X_d is significant, or the actual value of d is entirely controlled by the neighborhood state, then $H(X_d|X_n, G_d = g_d^k) = 0$. If the correlation is insignificant, in the worst case, for the given condition, X_d has equal chance to be 0 and 1, then the entropy is the maximum, i.e., $H(X_d|X_n, G_d = g_d^k) = 1$.

Unique failing outputs. The number of unique chip outputs (unique_no_fail_outputs) that fail as predicted by fault simulation of a candidate is potentially a good feature for characterizing the correctness of a candidate. If the failing output of a failed chip is explained[2] by only one particular candidate, while other outputs are explained by many candidates, it may be likely that this particular candidate is correct. It should be noted however that the feature unique_no_fail_outputs may be zero for all of the candidates of a given chip.

Passing/failing patterns and outputs. In addition to comparing/contrasting the pass-fail status of the test patterns on the tester and in simulation, we also compare/contrast the failing and passing outputs measured on the tester with those predicted by fault simulating the candidates. The intuition for performing this bit-level analysis in addition to the pattern-level analysis is that it accounts for the differences that seemingly do not exist among conventionally-ranked candidates.

Consider, for example, a three-output failed chip with test set $T = \{t_1, t_2, t_3, t_4\}$ that has the associated candidate simulation (S) and failed-chip tester (T) responses shown in Table 2. Table entries with a "P" indicate that the failed chip (candidate) did not cause the corresponding output to fail when the test was applied (fault simulated). A table entry of "F", on the other hand, has the opposite meaning.

Test	O_1		O_2		O_3	
	S	T	S	T	S	T
t_1	P	P	P	P	P	P
t_2	P	F	P	F	F	F
t_3	F	P	P	F	F	P
t_4	P	F	P	F	P	F

Table 2: Output-level comparison of an example failed-chip tester response and a candidate-simulation response for a three-output circuit.

Comparing and contrasting the tester response with the candidate fault simulation response at the pattern level leads to the feature values shown in Table 3.

Test	Response
t_1	TPSP
t_2	TFSF
t_3	TFSF
t_4	TFSP

Feature	Value
no_TPSPs	1
no_TFSPs	1
no_TPSFs	0
no_TFSFs	2

Table 3: Pattern-level feature values derived from the example responses from Table 2.

Comparing and contrasting the tester response with the candidate fault simulation response at the output level leads to the feature values shown in Table 4.

Test	O_1	O_2	O_3
t_1	TPSP	TPSP	TPSP
t_2	TFSP	TFSP	TFSF
t_3	TPSF	TFSP	TPSF
t_4	TFSP	TFSP	TFSP

Feature	Value
no_TPSP_outputs	3
no_TFSP_outputs	6
no_TPSF_outputs	2
no_TFSF_outputs	1

Table 4: Output-level feature values derived from the example responses from Table 2.

It can be observed that the pattern-level features (i.e., no_TFSPs, no_TPSFs, and no_TFSFs) and the output-level features (i.e., no_TFSP_outputs, no_TPSF_outputs, and no_TFSF_outputs) have very different values. It can be easily argued that the output-level features subsume the pattern-level features, and as a result, the pattern-level features should be simply eliminated. In the end, the pattern-level features may indeed be ignored by the classifier but we do not a priori impose that decision on the classifier because all commercial diagnosis tools find the pattern-level features useful for distinguishing candidates through

[2]A candidate is said to "explain" a failed output o_j observed on the tester for a given test pattern t_k if its fault simulation response for t_k predicts the failure of o_j.

ranking. For example, Synopsys TetraMAX® [21] uses the following pattern-level feature formulation to rank candidates:

$$score = \frac{\text{no_TFSPs}}{\text{no_TFSFs} + \text{no_TPSFs} + \text{no_TPSFs}}$$

2.2 First-level Classifier

The first-level classifier of PADRE is a one-rule discriminator that is based on the inconsistent-state feature (no_incst_states). As described earlier, this feature counts the number of unique neighbor states that both appear in at least one TPSF pattern and one TFSF pattern. The existence of an inconsistent state is likely an indication that the candidate is not actually a good candidate. While it is possible that a defective location can behave inconsistently, past work has shown that this feature is an excellent discriminator [1, 4, 10, 11, 12].

In the first-level classifier, any candidate with a non-zero no_incst_states is labeled as a bad candidate. For all the remaining candidates with no inconsistent state, their labels remain unknown.

$$label\ (c_i) = \begin{cases} \text{bad} & (\text{no_incst_states}\ (c_i) > 0) \\ \text{unknown} & (\text{no_incst_states}\ (c_i) = 0) \end{cases}$$

2.3 Second-level Classifier

Although the first-level classifier is able to accurately identify a good number of the bad candidates. Many of the candidates still remain unlabeled. We introduce a second-level classifier to further process the remaining unlabeled candidates, with a particular focus to identify the good candidates.

All unlabeled candidates from the first-level classifier are processed by an SVM-based second-level classifier. SVM requires a training set to derive (i.e., learn) a classifier. In our work, the training sets are constructed using novel and reasonable heuristics that identify the likely good and bad candidates from unlabeled candidates.

The construction of the training sets from unlabeled candidates follows a three-step workflow as follows:

- **Initial sets**: Initial good training set consists of all the chips with a single candidate; and the initial bad training set consists of all chips with more than Q candidates, where Q is a user-set threshold. In this work, we use $Q = 20$.
- **Refined sets**: Interquartile range (IQR) [22] is used to remove any outliers from the initial sets. Specifically, any candidate with a feature value that is 3 × IQR away from the feature mean is eliminated from the training sets.
- **Balanced sets**: Oversampling the smaller data set is performed to balance the sizes of the good and bad data sets.

A detailed discussion of each step is given next.

Initial set. Unlike conventional supervised-learning methods, the second-level classifier learned in PADRE derives training data from the pool of unlabeled candidates. Specifically, the initial good training set is obtained from all chips with a single candidate. Assuming that diagnosis is accurate, collecting a statistically-significant set of single-candidate chips ensures that the features of good diagnosis candidates are reasonably captured. On the other hand, the bad training set is obtained from all the chips with more than $Q = 20$ candidates. It is reasonable to assume that there is likely only one good candidate in a chip. Thus for a chip with multiple candidates, all but one will be bad. By taking all chips with more than $Q = 20$ candidates, the bad training set should mostly consist of bad candidates with an error bounded at 5%, i.e., at most 5% of candidates in the bad training set would be actually good candidates. This approach for identifying bad-candidate data introduces very little error. For example, for the simulation-based experiment discussed in Section 3.2, we found that fewer than 2.9% of the bad training-data candidates are actually good.

Refined set. To improve the quality of the training set, IQR (interquartile range) is employed to remove outliers from the training set. The outlier analysis removes candidates from the initial sets that are outside the 3 × IQR of the initial training set, which corresponds to 4.72 sigma (i.e., 99.9992% in a normal distribution). This ensures the quality of the training set by only including the most typical good and bad candidates.

Balanced set. Because the size of the good training set and the bad training set are likely different (the good candidates identified from the unlabeled pool of candidates are typically much fewer than the bad candidates), a biased classifier from SVM may result. Specifically, if the size of the bad training set overwhelms the size of the good training set, the classification margin obtained by SVM will be heavily biased towards the bad candidates, causing the good candidates to fall within the margin of bad candidates [23]. As a result, it would be more likely for the resulting classifier to incorrectly categorize an actual good candidate as bad. A straightforward solution for this dilemma involves balancing the size of the training data by employing simple sampling techniques. There are two ways to perform the sampling. One approach is to under-sample the bad training set, and the alternative is to oversample the good training set [23]. Under-sampling the bad training set has the benefit of lower computational cost, but depending on the sizes of the training sets, it may incur the risk of not properly characterizing both the good and bad candidates.

A major drawback of the oversampling method is over-fitting. The duplicates reinforce the distribution of the existing candidates in the training set, instead of the real distribution of the good training set. The latter can only be

obtained by using an increased number of real good candidates. However, in our case, the imbalance among the good and bad training sets is not too significant. For example, in our simulation data sets, the number of bad candidates is about 9× larger than the good candidates. Therefore, any over-fitting introduced is likely not to significantly affect the performance of the classifier. We therefore chose to balance the two training sets by oversampling the good training set. Specifically, good candidates are randomly duplicated until their number equals the number of bad candidates.

The SVM algorithm is chosen for PADRE due to its robust performance in classifying two different classes of data by error-margin maximization. Classification problems are essentially concerned with assigning a label y to a sample based on its feature set $\boldsymbol{x}$:

$$y = f(\boldsymbol{x})$$

where the scalar y is the label of the sample and the vector $\boldsymbol{x}$ is the set of features of the sample. The function $f(\boldsymbol{x})$ is the classifier that we want to learn for determining the labels. A linear classifier uses the following discriminant function:

$$f(\boldsymbol{x}) = \begin{cases} 0 & (\boldsymbol{w}^T\boldsymbol{x} + c \geq 0) \\ 1 & (\boldsymbol{w}^T\boldsymbol{x} + c < 0) \end{cases}$$

where $\boldsymbol{w}$ is the weight vector for each feature, and c is a constant.

SVM is robust because it attempts to distinguish different classes by maximizing the error margin. This will usually give a robust separation of the classes [13]. The margin-maximization problem associated with SVM can be formulated as:

$$\min_{\boldsymbol{w},c,\xi} \sum \xi_i + \lambda \boldsymbol{w}^T\boldsymbol{w}$$

$$\text{S.T. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i + c) \geq 1 - \xi_i;\ \xi_i \geq 0$$
$$(i = 1,2,\ldots,N)$$

where λ regularizes the penalty of misclassification, and ξ_i is the error of the i-th data point. To avoid over-fitting, cross validation is usually used to determine λ.

The aforementioned SVM formulation is referred to as the soft-margin SVM in the literature [13]. It is able to classify samples with allowance of classification error, such as the case illustrated in Figure 3. The soft-margin SVM appropriately meets the needs of our application because the training set constructed from unlabeled candidates is inherently imperfect. This is particularly true for the bad training set, where a small number of the good candidates are knowingly included within the larger set of bad candidates. Soft-margin SVM allows robust classification even with an imperfect training set.

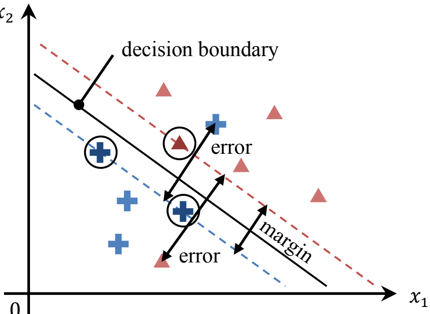

Figure 3: Two sets of data (blue crosses and red triangles) are plotted with respect to two features $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$. A soft-margin SVM allows classification errors and maximizes the margin between support vectors (circled crosses and triangle) to find a decision boundary.

3. Experiment

A comprehensive simulation experiment is performed to evaluate PADRE. In the experiment, fault-tuple macrofaults [24] are used to emulate real defects. Specifically, a large number of different fault types are injected into the benchmark circuit, and a typical testing procedure is employed to obtain the initial fault diagnosis results. PADRE is subsequently applied to refine the diagnostic resolution which is then evaluated for the level of improvement and accuracy achieved. The experiment is also repeated for actual failed chips from the LSI Corporation.

3.1 Setup

The circuit B12 from the ITC’99 benchmark suite [25] is used as the “chip under test”. B12 consists of 1,000 gates and 121 flip-flops. Five-hundred instances of six different types of faults are randomly injected into the circuit to emulate defective chips. The fault-tuple macrofaults used to emulate the defects are simulated using FATSIM [24]. The injected macrofaults include: and-, or-, and dominating-type bridge faults, cell-level input pattern faults [26], SSL faults, and MSL faults. The numbers of chips injected for each type of fault and detected by the applied test set are listed in Table 5.

Fault type	Number of chips
and-bridge	493
or-bridge	497
dominant-bridge	499
input pattern	487
SSL	490
MSL	421

Table 5: Fault-tuple macrofaults of various types are randomly injected into B12 circuit to construct a virtual set of defective chips for diagnosis.

The fault types listed in Table 5 are chosen to represent the large variety of actual defects that occur in real chips. The advantage of using virtual failed chips is that it provides a large number of failed chips with known defect types and locations, which is essential for verifying accuracy.

The test set used in this experiment is a 100% stuck-at test set generated by a commercial ATPG tool. Each failed chip is tested and analyzed using physically-aware diagnosis [1, 10, 12, 27]. Because each failed chip is virtual in nature (i.e., we know the locations of the "defective" lines), the accuracy of the diagnosis refinement is easy to verify. However, a list of golden answers is never available in practice.

The number of candidates represents the resolution of the initial diagnosis. The cumulative distribution of the initial resolution for the virtual population is illustrated in Figure 4. For every candidate of a failed chip, the previously described features in Section 2.1 are extracted for the classification process.

3.2 Results and Discussion

A total of 33,178 candidates result from diagnosing the virtual population of failed chips listed in Table 5. The number of bad candidates being labeled by the first-level classifier is 13,622. Comparing these results with the inject locations reveals that 99.85% of the predicted bad candidates are indeed the wrong candidates.

A total of 19,556 unlabeled candidates from the first-level classifier are processed by the second-level classifier. The construction of the training sets from unlabeled candidates follows the three-step workflow described in Section 2.3. The sizes of the training sets in each step are tabulated in Table 6a.

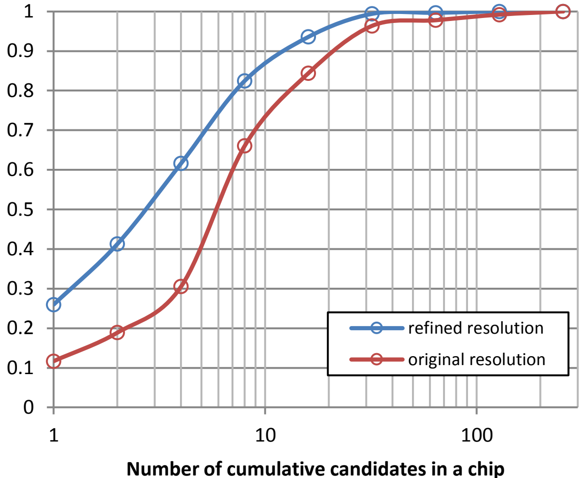

Figure 4: Resolution refinement through PADRE shows that resolution can be improved by 21.7% on average. Moreover, the number of chips that exhibit perfect resolution is more than doubled (i.e., 11% to 27%).

Chip-level diagnosis results of the total 2,887 chips are classified into four classes to understand the performance of PADRE. The classification examines the quality of the refined candidates sets of the chips, i.e., the sets of candidates produced by PADRE that have a reduced number of candidates compared to the original candidates of the respective chips. The description and number of chips that fall into each class are shown in Table 8.

	Good training set	Bad training set
Initial sets	468	4,409
Refined sets	324	3,283
Balanced sets	3,283	3,283

(a)

	Good training set	Bad training set
Initial sets	897	1,228
Refined sets	895	1,085
Balanced sets	1,085	1,085

(b)

Table 6: The sizes of training sets in each step of the training-set-construction workflow for (a) the virtual failed-chip population and (b) the actual failed-chip population from LSI.

The SVM module from MATLAB is used to perform second-level classification. The balanced training sets are fed into SVM to learn the classifier. The second-level classifier labels each of the unlabeled candidates as either good or bad. A total of 793 good candidates are labeled by the second-level classifier, which means that a total of 32,385 bad candidates are labeled by the PADRE. The candidate-level diagnosis results are summarized in Table 7. Comparing the predicted results with the actual locations reveals that 94.70% of the labeled good candidates are correct. For the labeled bad candidates, the first-level classifier has an accuracy of 99.85% while the second-level classifier only has an accuracy of 70.88%. Given that it is costly to mistake a good candidate as bad, we decide to only consider the classification results from the first-level classifier for bad candidates in PADRE. Although including the second-level classification for the bad candidates can aggressively improve the resolution, a considerable number of good candidates would be lost, which is undesirable.

Type of labeling result		Number of candidates
Label good	Correct	751
	Wrong	42
Label bad	Correct	28,610
	Wrong	3,775

Table 7: The second-level classifier of PADRE shows a high accuracy for good-candidate classification and a moderate accuracy for bad-candidate classification.

Overall, PADRE improves resolution for 1,959 of the 2,887 chips. For the remaining 928 chips, PADRE does not identify any good candidate with the second-level classifier or eliminate any bad candidates with the first-level classifier, which may be due to the result of limited training sets that do not comprehensively cover all the possible characteristics of the good and bad candidates. Therefore, no improvement of resolution is achieved for those chips. However, by comparing the refined resolution, i.e., the number of candidates in refined candidate sets, with the original resolution, it is clear that the resolution improvement for the refined chips is significant as shown in Figure 4. Specifically, the average per-chip resolution improvement is over 32.0% for the refined chips, or 21.7% for all the chips. Moreover, the number of chips that exhibit perfect resolution (i.e., only one candidate) is more than doubled (i.e., 11% to 27%).

Class	Description	Number of chips
1	All good candidates remain in the refined candidate set; no bad candidates exist in the refined candidate set.	814
2	All good candidates remain in the refined candidate set; some bad candidates also exist in the refined candidate set.	1,869
3	Not all good candidates remain in the refined candidate set; no bad candidates exist in the refined candidate set.	12
4	Not all good candidates remain in refined candidate set; some bad candidates exist in the refined candidate set.	192

Table 8: PADRE results viewed at the chip level reveal a high probability of including all the good candidates in the refined candidate set, and a relatively high probability that some of the bad candidates are also included in the refined candidate set.

The computational time of PADRE does not degrade with circuit size, but does with the number of chips in the population. A total of 3.9s was used by the classifier to process the feature data of 33,178 candidates of 2,887 chips, which is equivalent to 0.12ms per candidate or 1.4ms per chip, on average. It is important to note however that PADRE is applied after traditional diagnosis, so it will always add to the time overhead. Besides, the objective of PADRE is to enable better diagnostic resolution instead of faster diagnosis, so the optimization of time overhead is not emphasized in current development.

PADRE is also applied to actual failed test chips manufactured by the LSI Corporation that includes mostly 74X181 ALU circuits. The LSI ALU consists of 2,309 gates. Specifically, a total of 5,362 failed test chips are diagnosed using a commercial tool, leading to a total of 36,186 diagnosis candidates. Similar to the virtual failed-chip population, we provide the training set statistics for the LSI failed-chip population in Table 6b. The candidates of the LSI chips were processed by the first-level classifier and subsequently the second-level classifier. Figure 5 compares the (cumulative) distribution of diagnostic resolution produced by the commercial tool and the improved distribution produced by PADRE. Resolution is improved on average by 37.9% for 5,362 failed chips, and the number of chips that have ideal resolution is nearly tripled (i.e., 16% to 46%).

Verifying the accuracy of PADRE for the virtual failed-chip population is straightforward since the fault type and location are known for each chip. This is not the case however for the silicon failed-chip population. But we do have in hand five chips that have been PFA'ed, the details of which are given in Figure 6[3]. PADRE is deemed accurate if the actual failure locations are contained within the refined set of diagnostic candidates. As shown in Figure 6, for the five chips examined, diagnostic resolution is either dramatically improved or maintained, and in all cases, the correct candidate(s) are contained within the reduced candidate set.

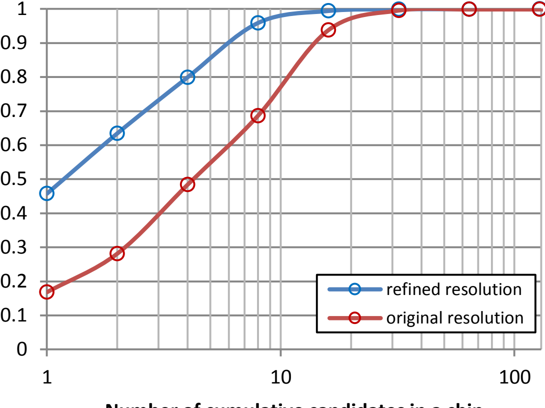

Figure 5: Resolution refinement through PADRE shows that resolution can be improved by 37.9% on average. Moreover, the number of chips that exhibit perfect resolution is nearly tripled (i.e., 16% to 46%).

4. Conclusions and Future Work

An improved diagnosis resolution reduces the time and cost of PFA and also benefits other applications that utilize the results of diagnosis for a population of chips (i.e., volume diagnosis). Despite the existence of various approaches for improving diagnostic resolution, there is still much room for improvement. In this work, we present a novel resolution refinement method that uses a two-level unsupervised learning classifier, combined with a series of existing and new heuristics to distinguish good and bad candidates. PADRE (Physically-Aware Diagnostic Resolution Enhancement) constructs good and bad training data from the originally available unlabeled candidates, instead of relying on the historical data as is usually accomplished in past work [18, 19]. PADRE is shown to improve average resolution over conventional results by 21.7%, and also more than doubles the number of chips that exhibit perfect resolution for a virtual population of failed chips. These numbers improve even further for actual silicon chips. Specifically, resolution is reduced for 38% of the nearly 3,000 failed chips examined, and the number of chips exhibiting ideal resolution is nearly tripled (i.e., 16% to 46%). Finally, for five of the silicon failed-chips, we demonstrated that PADRE does not at all degrade diagnostic accuracy.

PADRE demonstrates that it is possible to better understand the nature of a diagnostic candidate by exploiting the logical and physical information. Our future work will focus on exploring other effective chip- or candidate-related features to further improve the capability of PADRE.

[3]It should be noted that the resolution reported here systematically differs from reference [10] simply because candidates (i.e., signal lines) are here equated with signal-line locations, and in [10] candidates are equated with faults.

	Chip 1	Chip 2	Chip 3	Chip 4	Chip 5
Original resolution	16	1	4	11	2
New resolution	4	1	4	3	2
Accuracy maintained?	Yes	Yes	Yes	Yes	Yes
SEM of failure	4-line bridge	Gate-to-gnd short	Poly-to-active short	Open via	Open contact

Figure 6: For these five failed chips, diagnostic resolution is significantly improved or maintained without degrading accuracy.

References

[1] R. Desineni, O. Poku and R. D. Blanton, "A Logic Diagnosis Methodology of Accurate Defect Behavior," in *Proc. International Test Conference*, 2006.

[2] L. C. Wagner, "Chemical Analysis," in *Failure Analysis of Integrated Circuits Tools and Techniques*, Kluwer Academic Publishers, pp. 195-204, 1999.

[3] L. M. Huisman, M. Kassab and L. Pastel, "Data Mining Integrated Circuit Fails with Fail Commonalities," in *Proc. International Test Conference*, 2004.

[4] W. C. Tam, O. Poku and R. D. Blanton, "Precise Failure Localization using Automated Layout Analysis of Diagnosis Candidates," in *Proc. Design Automation Conference*, 2008.

[5] R. D. Blanton and Y. Lin, "Test Effectiveness Evaluation through Analysis of Readily-Available Tester Data," in *Proc. International Test Conference*, 2009.

[6] X. Yu and R. D. Blanton, "Estimating Defect-Type Distributions through Volume Diagnosis and Defect Behavior Attribution," in *Proc. International Test Conference*, 2010.

[7] R. C. Aitken, "Finding Defects with Fault Models," in *Proc. International Test Conference*, 1995.

[8] J. A. Waicukauski, "Failure Diagnosis of Structured VLSI," *Design & Test of Computer,* vol. 6, no. 4, pp. 49-60, 1989.

[9] D. B. Lavo, I. Hartanto and T. Larrabee, "Multiplets, Models, and the Search for Meaning: Improving Per-test Fault Diagnosis," in *Proc. International Test Conference*, 2002.

[10] R. D. Blanton, W. C. Tam, X. Yu, J. E. Nelson and O. Poku, "Yield Learning Through Physically Aware Diagnosis of IC-Failure Populations," *IEEE Design & Test of Computers,* vol. 29, no. 1, pp. 36 - 47, 2012.

[11] R. Desineni and R. D. Blanton, "Diagnosis of Arbitrary Defects using Neighborhood Function Extraction," in *Proc. VLSI Test Symposium*, 2005.

[12] X. Yu and R. D. Blanton, "Improving Diagnosis through Failing Behavior Identification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 31, no. 10, pp. 1614-1625, 2012.

[13] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2007.

[14] S. Biswas and R. D. Blanton, "Reducing Test Execution Cost of Integrated, Heterogeneous Systems Using Continuous Test Data," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 30, no. 1, pp. 148-158, 2011.

[15] E. Yilmaz, S. Ozev and K.M. Butler, "Adaptive Test Flow for Mixed-Signal/RF Circuits Using Learned Information from Device Under Test," in *Proc. International Test Conference*, 2010.

[16] S. Goyal, A. Chatterjee and B. Shenouda, "Test Time Reduction of Successive Approximation Register A/D Converter by Selective Code Measurement," in *Proc. International Test Conference*, 2005.

[17] N. Kupp, K. Huang, J. M. Carulli and Y. Makris, "Spatial Correlation Modeling for Probe Test Cost Reduction in RF Devices," in *International Conference on Computer-Aided Design*, 2012.

[18] J. E. Nelson, W. C. Tan and R. D. Blanton, "Automatic Classification of Bridge Defects," in *Proc. International Test Conference*, 2010.

[19] F. Ye, Z. Zhang, K. Chakrabarty and X. Gu, "Board-Level Functional Fault Diagnosis Using Learning Based on Incremental Support-Vector Machines," in *Proc. Asian Test Symposium*, 2012.

[20] Y. Lin and R. D. Blanton, "METER: Measuring Test Effectiveness Regionally," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 30, no. 7, pp. 1058-1071, 2011.

[21] Synopsys Inc., "TetraMAX ATPG User Guide," Version W-2004.12, 2005.

[22] G. Upton, Understanding Statistics, Oxford University Press, 1996.

[23] Y. Liu, A. An and X. Huang, "Boosting Prediction Accuracy on Imbalanced Datasets with SVM Ensembles," *Advances in Knowledge Discovery and Data Mining,* vol. 3918, pp. 107-118, 2006.

[24] K. N. Dwarakanath and R. D. Blanton, "Universal Fault Simulation Using Fault Tuples," in *Proc. Design Automation Conference*, 2000.

[25] F. Corno, M. S. Reorda and G. Squillero, "RT level ITC'99 Benchmarks and First ATPG Results," *IEEE Design and Test of Computers,* vol. 17, no. 3, pp. 44-53, 2000.

[26] . D. Blanton and J. P. Hayes, "Properties of the Input Pattern Fault Model," in *Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors*, 1997.

[27] Y. Lin, O. Poku, R. D. Blanton, P. Nigh, P. Lloyd and V. Iyengar, "Evaluating the Effectiveness of Physically-Aware N-Detect Test using Real Silicon," in *Proc. International Test Conference*, 2008.